HPI Future SOC Lab: Proceedings 2012

Christoph Meinel, Andreas Polze, Gerhard Oswald,
Rolf Strotmann, Ulrich Seibold, Bernhard Schulzki
(Hrsg.)

Technische Berichte Nr. 85

des Hasso-Plattner-Instituts für
Softwaresystemtechnik
an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 85

Christoph Meinel | Andreas Polze | Gerhard Oswald | Rolf Strotmann |
Ulrich Seibold | Bernhard Schulzki (Hrsg.)

HPI Future SOC Lab

Proceedings 2012

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2014
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Druck: docupoint GmbH Magdeburg

ISBN 978-3-86956-276-6

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
URL http://pub.ub.uni-potsdam.de/volltexte/2014/6899/
URN urn:nbn:de:kobv:517-opus-68991
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-68991

Contents

Fall 2012

Parallelizing H.264 Decoding with OpenMP Superscalar

Chi Ching Chi, Ben Juurlink
Embedded Systems Architecture
Einsteinufer 17
10551 Berlin
{chi.c.chi,b.juurlink}@tu-berlin.de

Abstract

Since the advent of multi-core processors and systems, programmers are faced with the challenge of exploiting thread-level parallelism (TLP). In the past years several parallel programming models have been introduced to simplify the development of parallel applications. OpenMP Superscalar is a novel task-based programming model, which incorporates advanced features such as automated runtime dependency resolution, while maintaining simple pragma-based programming for C/C++. We have parallelized H.264 decoding using OpenMP Superscalar to investigate its ease-of-use and performance.

1 Introduction

Because multi-core processors have become omnipresent, there is a lot of renewed interest in parallel programming models that are easy to use while being expressive, and allow to write performance-portable applications. An important question is, however, how to evaluate the ease of use, expressiveness, and performance of a programming model. In this work we try to answer this question by describing our experiences in parallelizing H.264 decoding using OmpSs, a novel task-based parallel programming model. To evaluate its performance, the performance of the OmpSs application is compared to a similary structured Pthreads applications. H.264 decoding is an excellent case study because it is highly irregular and dynamic, exhibits many different types of data dependencies as well as rather fine-grained tasks, and requires different types of parallelism to be exploited.

In OmpSs [3], in addition to OpenMP functionality, programmers can express parallelism by annotating certain code sections (typically functions) as *tasks*, as well as the *inputs*, *outputs*, and *inputs/outputs* of these tasks. When these functions are called, they are added to a task graph instead of being executed. The task dependencies are resolved at runtime, using the input/output specification of the function arguments. Once all input dependencies of a task are resolved, it is scheduled for execution.

A key difference between OmpSs and other task-based parallel programming models is the ability to add tasks before they are ready to execute. This is a powerful feature which allows more complex parallelization strategies that simultaneously exploit function-level and data-level parallelism. In this paper we show how these features can be used to implement a parallel implementation of H.264 decoding.

The structure of this paper is as follows: Sections 2 to 5 describe the implementation and optimization of parallel H.264 video decoding using OmpSs. In Section 6 the performance of the OmpSs version is evaluated and compared to an optimized Pthreads implementation. Conclusions are drawn in Section 7.

2 Pipelining H.264

The H.264 decoder pipeline consists in our design of 5 pipeline stages, shown in Figure 1. In the read stage the bitstream is read from the disk and parsed into separated frames. In the parse stage the headers of the frame are parsed and a picture info entry in the picture info buffer (PIB) is allocated. The entropy decode (ED) stage performs a lossless decompression by extracting the syntax elements for each macroblock in the frame. Some syntax elements are directly processed, e.g. the motion vectors differences are transformed into motion vectors. The macroblock reconstruction stage allocates a picture in the decoded picture buffer (DPB) and reconstructs the picture using the syntax elements and motion vectors. The output stage reorders and outputs the decoded pictures either to an output file or the display.

In contrast to other task-based programming models, like Cilk++ [7] and OpenMP [2], pipeline parallelism can be straightforwardly expressed in OmpSs, because OmpSs tasks can be spawned before its dependencies have been resolved [8, 6]. Listing 1 presents the simplified code of the pipelined main decoder loop using OmpSs pragmas. A task is created for each pipeline stage in each loop iteration. For correct pipelining of the tasks, it is required that all tasks in iteration *i* are

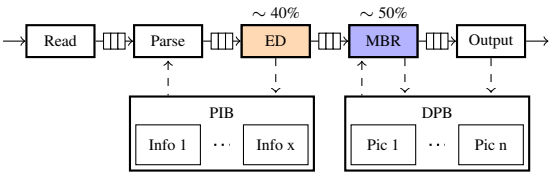


Figure 1. H.264 decoder pipeline stages in our design

executed in-order. To accomplish this, each task in the same iterations is linked to the previous task in the same iteration via one or more input and output/inout pairs. Additionally, task *T* of iteration *i* must be completed before the instance of the same task *T* in iteration *i+1* is started. To accomplish this, each task has a context structure that is annoted as inout, e.g., ReadContext *rc, NalContext *nc, EntropyContext *ec, etc. The pipeline parallelism is uncovered by using entry of a circularly buffer of size *N* for the task inputs and outputs. This eliminates the WAR and WAW hazards that would have occurred if the same entry is used in each iteration, which would eliminate all the parallelism.

Exploiting the FLP, however, is not enough to get scalable performance. The performance of the pipelined implementation is limited by the longest stage in the pipeline. The entropy decode and macroblock reconstruct stages take around 40% and 50% of the total execution time, respectively. The total speedup is in this case limited to a factor of two. To gain additional speedups both the entropy decode and macroblock reconstruct stages must be further parallelized. Both these stages exhibit DLP and how they can be exploited is discussed in the following sections.

3 Parallelizing Entropy Decoding

The ED stage performs entropy decoding using CABAC or CAVLC. In both these methods the interpretation of each bit in the stream depends on the previous bit. Therefore, no task parallelism exists inside the entropy decode of one frame. Multiple frames, however, can be decoded in parallel as they are separated by start codes. The frames, however, are not fully independent as illustrated in Figure 2.

In Figure 2, four frames are decoded in parallel. The hatched blocks represent the current MBs that are decoded in parallel and the blue blocks denote the already processed MBs in each frame. For blocks in the B-frames some blocks may need the motion vectors of the co-located block in the previous frame. To express this parallelism, the code segment in Listing 2 can be used replacing the entropy_decode_task in Listing 1.

The entropy decode task is split in three tasks. The init task initializes the context tables, the decode_entropy_line_task entropy decodes a macroblock line of the picture, and the release task releases one or more Picture Info entries, which are no longer referenced. To have multiple entropy decodes in flight, the EntropyContext is renamed in the same fashion as the pipeline buffers shown in Listing 1.

To maintain the dependencies shown in Figure 2, the entropy_decode_line_task has several annotated arguments. The EntropyContext is annotated with inout to enforce that the lines of each picture are decoded sequentially. H264Mb *mb_in is annotated as an input and H264Mb *mb_out is annotated as an output, to maintain the dependencies between frames. By passing the pointers to the first H264Mb of the co-located line in the previous entropy buffer and the

```
#pragma omp task inout(*rc) output(*frm)
void read_frame_task(ReadContext *rc, EncFrame *frm
    );

#pragma omp task inout(*nc, *frm) output(*s)
void parse_header_task(NalContext *nc, Slice *s,
  EncFrame *frm);

#pragma omp task inout(*ec, *s) input(*frm) output(
    *mbs)
void entropy_decode_task(EntropyContext *ec, Slice
    *s,
  EncFrame *frm, H264mb *mbs);

#pragma omp task inout(*rc) input(*s, *mbs) output(
    *pic)
void reconstruct_task(MBRecContext *rc, Slice *s,
  H264Mb *mbs, Picture *pic);

#pragma omp task inout(*oc) input(*pic)
void output_task(OutputContext *w, Picture *pic);

EncFrame frm[N]; Slice slice[N];
H264Mb *ed_bufs[N]; Picture pic[N];

int k=0;
while(!EOF){
  read_frame_task(rc, &frm[k%N]);
  parse_header_task(nc, &slice[k%N], &frm[k%N]);
  entropy_decode_task(ec, &slice[k%N], &frm[k%N],
    ed_bufs[k%N]);
  reconstruct_task (rc, &slice[k%N], ed_bufs[k%N],
    &pic[k%N]);
  output_task(oc, &pic[k%N]);
  k++;
#pragma omp taskwait on (*rc)
}
```

Listing 1. Pipelining the main decoder loop using OmpSs pragmas

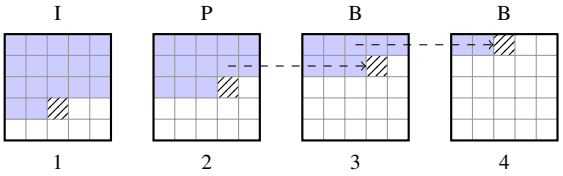


Figure 2. Parallelism in entropy decoding in multiple consecutive frames. Colored MBs have been entropy decoded. Hatched blocks are decoded in parallel.

```
#pragma omp task inout(*ec, *s, *mbs) input(*frm)
void init_entropy_task(EntropyContext *ec, Slice *s
    ,
  EncFrame *frm, H264Mb *mbs);

#pragma omp task inout(*ec, *s) input(*mb_in)
  output(mb_out[0;columns])
void entropy_decode_line_task(EntropyContext *ec,
  int line, Slice *s, H264Mb *mb_in, H264Mb *mb_out
      );

#pragma omp task inout(*s, *dummy)
void release_PI_task(Slice *s, int *dummy);

...
init_entropy_task(&ec[k%N], &slice[k%N], &frm[k%N],
  ed_bufs[k%N]);
for(int i=0; i<row; i++){
  entropy_decode__line_task(&ec[k%N], i, &slice[k%N
      ],
    &ed_bufs[(k+N-1)%N][i], &ed_bufs[k%N][i]);
}
release_PI_task(&slice[k%N], &k);
...
```

Listing 2. Code fragment replacing the entropy task to perform parallel entropy decoding.

first H264Mb of the current line in the current entropy buffer to mb_in and mb_out, respectively, it is ensured that each line *x* in frame *n* is decoded before starting to decode line *x* in frame *n+1*. A task for each macroblock line is created instead for each macroblock to increase the task granularity at expense of parallelism. The parallelism is still sufficient, however, with a maximum of 135 for QFHD resolution videos.

4 Parallelizing Macroblock Reconstruction

In the macroblock reconstruction stage the image is reconstructed using the syntax elements produced by the entropy decoding stage. To reconstruct a macroblock in H.264 several pixel areas from adjacent reconstructed macroblock are required. For each hatched macroblock in Figure 3, the adjacent red pixels are needed for the intra-prediction and the deblocking filter kernels. Therefore, only macroblocks on a wavefront are parallel. The wavefront parallelism is not massive, but sufficient with a maximum of 120 free macroblocks in $4k \times 2k$ resolution videos. The wavefront parallelism can be exploited using the code fragment in Listing 3.

The wavefront dependencies are static and are covered by the dependencies to the left macroblock and the upper right macroblock. The wavefront dependencies of the reconstruct_mb_task are maintained through its H264mb*arguments by annotating the left macroblock *ml* and upper right macroblock *mur* as inputs and the current macroblock *m* as inout. Since the reconstruct_mb_tasks are added in scan line order, this input and the output specification ensures that the wavefront dependencies are maintained. In addition to the pragmas, only the code that computes the left and upper right macroblock must be added to the sequential code. The ability to express dependencies between tasks makes the OmpSs implementation relatively clean and simple.

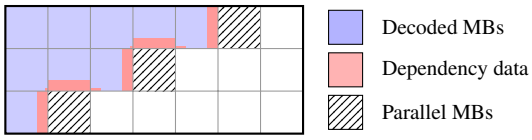


Figure 3. Wavefront parallelism in H.264 macroblock reconstruction.

```
#pragma omp task input(*rc, *s, *ml, *mur) inout(*m
    )
void reconstruct_mb_task(MBRecContext *rc, Slice *s
    ,
  H264mb *ml, H264mb *mur, H264mb *m);

#pragma omp task inout(*rc) input(*s, mbs[0;rows
    *cols])
  output(*pic)
void reconstruct_task(MBRecContext *rc, Slice *s,
  H264Mb *mbs, Picture *pic){
  init_ref_list(s);
  get_picture_buffer(rc, s);
  for(int i=0; i< rows; i++){
    for(int j=0; j< cols; j++){
      H264mb *m = &mbs[i*cols + j];
      H264mb *ml = m - ((j > 0) ? 1: 0);
      H264mb *mur = m - (((j < cols-1) && (i >0))
        ? cols-1: 0);
      reconstruct_mb_task(rc, s, ml, mur, m);
    }
  }
  H264mb *lastmb = &mbs[smb_width*smb_height -1];
#pragma omp taskwait on (*lastmb)
  release_ref(rc, s);
  *pic = s->pic;
}
```

Listing 3. Wavefront algorithm expressed in OmpSs.

5 Optimizing Task Granularity

The OmpSs implementation of parallel macroblock reconstruction shown in Listing 3 is a clean way to express the wavefront parallelism. The decode_mb_tasks, however, are fine-grained and have an average execution time of around 2μs on a commodity processor. The task management overhead of OmpSs does not allow such fine-grained tasks to perform well. A technique to overcome this is to coarsen the tasks by grouping several macroblocks. Due to the wavefront dependencies, however, macroblocks must be grouped in tetris-block shapes, as shown in Figure 4.

Expressing task dependencies between these tetris-shaped superblocks directly is not straightforward, especially when it is desired to support an arbitrary picture and superblock sizes. To overcome this the su-

perblocks are remapped to a regular structure. Independent of their shapes, all superblock variants still exhibit wavefront dependencies. The dependencies can be easily checked, in the same way as the regular ungrouped macroblocks when they are remapped to a regular matrix form as depicted in Figure 4.

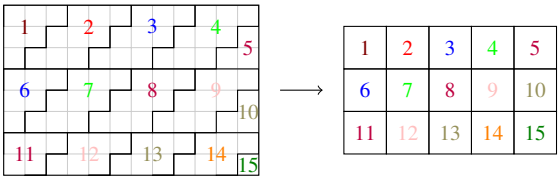


Figure 4. Remapping the irregular superblock shapes back to regular shapes simplifies the dependency expression for the programmer and task dependence checking for the runtime system.

The remapping does somewhat increase the code complexity. First, it must be calculated how many superblocks fit in the width and the height of a picture. Second, when decoding the superblock it must be checked which macroblocks belong to this particular superblock. This can be performed by checking if the macroblocks covered by the superblock shape are inside the picture.

If it is ensured that every first superblock in a line has the maximum number of macroblocks in its top row, as is the case in Figure 4, then the code fragment in Listing 4 can be used instead of the code fragment in Listing 3 to reconstruct coarsened superblocks instead of individual macroblocks, for an arbitrary block and picture size.

The code in Listing 4 resembles the code in Listing 3. The main differences are the reduced loop boundaries smb_rows and smb_columns, calling the reconstruct_super_mb_task, and checking for valid macroblocks covered by the superblock before calling the inner reconstruct_mb function. This method is generally applicable to specify coarsened wavefront parallelism in OmpSs for arbitrary superblock sizes and picture dimensions with minimal runtime dependency checking overhead.

6 Experimental Results

A 4-socket cc-NUMA machine with a total of 32 cores is used for the performance evaluation. The full hardware and software specification of our evaluation platform is listed in Table 1.

Figure 5 shows the effect of increasing the task granularity described in the previous section. From the figure we can see that the highest performance is achieved with a superblock size of 8×8. When the task size is too small the performance is bottlenecked

```
#pragma omp task input(*rc, *s, *ml, *mur) inout(*m
    )
void reconstruct_super_mb_task(MBRecContext *rc,
  Slice *s, Supermb *ml, Supermb *mur, Supermb *m){
  for (int k=0, i=mby; i< m->mby + sheight; i++, k
       ++)
    for (int j= m->mbx -k ; j< m->mbx- k + swidth;
         j++){
      // if (i,j) is a valid macroblock
      if (i< rows && j>=0 && j<columns)
        reconstruct_mb(rc, s, i, j);
    }
}

#pragma omp task inout(*rc) input(*s,
  smbs[0;srows*scols]) output(*pic)
void reconstruct_task(MBRecContext *rc, Slice *s,
  Supermb *smbs, Picture *pic){
  init_ref_list(s);
  get_picture_buffer(rc, s);
  for(int i=0; i< srows; i++)
    for(int j=0; j< scols; j++){
      Supermb *m = &smbs[i*scols + j];
      Supermb *ml = m - ((j > 0) ? 1: 0);
      Supermb *mur = m - (((j < scols-1) && (i >0))
          ?
        scols-1: 0);
      reconstruct_super_mb_task(rc, s, ml, mur, m);
    }
  Supermb *lastmb = &smbs[srows*scols -1];
#pragma omp taskwait on (*lastmb)
  release_ref(rc, s);
  *pic = s->pic;
}
```

Listing 4. Decoding tetris-shaped superblocks instead of single macroblocks.

Hardware		Software	
Processor	Xeon X7550	OS	Ubuntu 10.10
Cores	8	Kernel	2.6.35.10
Frequency	2.00 GHz	Compiler	GCC 4.4.5
Last level cache	18 MB	OmpSs compiler	Mercurium (git Aug'11)
Sockets	4		
Total cores	32	OmpSs runtime	Nanos++ (git Aug'11)
Total memory	1 TiB		
Total mem. BW	102.3 GB/s	Opt. level	-O2
SMT	Disabled		
Turbo mode	Disabled		

Table 1. Experimental setup.

by the runtime dependence checking. Increasing the block size from 2×2 to 4×4 improves the performance almost proportional to the increase in block size. Too large block sizes, on the other hand show again lower performance, because the parallelism is reduced too much.

The performance of the OmpSs H.264 decoder is comparing to an optimized Pthreads implementation [1] for up to 32 cores in Figure 6. For the OmpSs variant a block size of 8 × 8 is used. The figure shows that the performance results of the two are similar up to 8 cores, but are drifting further apart at higher core counts. By grouping the macroblocks, however, the parallelism is limited, which in turn limits the performance at higher core counts. In the Pthreads version of h264dec the synchronization is highly optimized us-

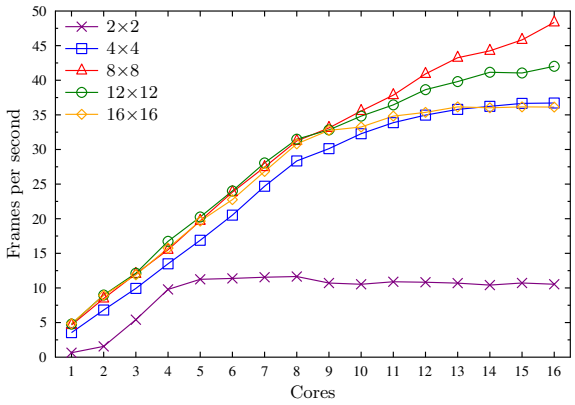


Figure 5. Performance impact of the different superblock sizes.

ing a line decoding strategy and, therefore, grouping of tasks is not necessary, resulting in a higher scalability.

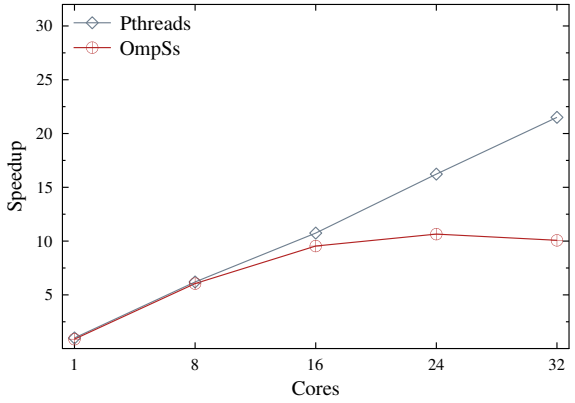


Figure 6. Speedup results for an optimized Pthreads and OmpSs variant of the full H.264 decoder

To improve the scalability more parallelism can be exploited by starting to decode the next frame before the current frame is completely decoded. Because the motion vectors have limited sizes some macroblocks of the next frame can already be decoded. The difficulty is that another dependency must be added between macroblocks in consecutive frames, which could increase the task dependence checking overhead more than the additional parallelism gains in performance. Another way to increase the scalability is to reduce the task dependency checking overhead. This can be achieved by changing the order tasks are submitted, in order to reduce the tasks-in-flight without reducing the parallel schedulable tasks. Less tasks-in-flight lead to a lowere cost for task dependency checking.

7 Conclusions

In this report we have discussed how H.264 decoding can be parallelized in OmpSs to capture both the function-level and data-level parallelism. In OmpSs this can be done intuitively in contrast to Pthreads and other task-based programming models such as Cilk++ and OpenMP. The runtime task overhead, however, forces that the granularity of the tasks is increased, which reduces the scalability at higher core counts. In future work we will investigate how parallelism can be exploited in consecutive frames. Also we will try to reduce the task dependence checking overhead by reducing the task window size for a given amount of parallelism.

8 Acknowledgements

This research has been supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under the ENCORE Project (www.encore-project.eu), grant agreement n° 248647 [4]. Computational support has been provided by the Future SOC Lab of Hasso-Plattner-Institute Potsdam [5].

References

[1] C. C. Chi and B. Juurlink. A QHD-Capable Parallel H.264 Decoder. In *Proc. 25th Int. Conf. on Supercomputing*, 2011.

[2] L. Dagum and R. Menon. OpenMP: A Proposed Industry Standard API for Shared Memory Programming. *IEEE Computing in Science and Engineering*, 1997.

[3] A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, and J. Planas. OmpSs: A Proposal for Programming Heterogeneous Multi-Core Architectures. *Parallel Processing Letters*, 21, 2011.

[4] Encore Project. ENabling technologies for a future many-CORE.

[5] Hasso-Plattner-Institut Potsdam. Future SOC Lab. http://www.hpi.uni-potsdam.de/forschung/future_soc_lab.html.

[6] A. Pop and A. Cohen. A Stream-Computing Extension to OpenMP. In *Proc. 6th Int. Conf. on High Performance and Embedded Architectures and Compilers*, 2011.

[7] K. H. Randall. *Cilk: Efficient Multithreaded Computing*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1998.

[8] H. Vandierendonck, P. Pratikakis, and D. Nikolopoulos. Parallel Programming of General-Purpose Programs Using Task-Based Programming Models. In *Proc. 3rd USENIX Workshop on Hot Topics in Parallelism*, 2011.

Service-Based 3D Rendering and Interactive 3D Visualization

Benjamin Hagedorn
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
benjamin.hagedorn@hpi.uni-potsdam.de

Jürgen Döllner
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
doellner@hpi.uni-potsdam.de

Abstract

This report describes the subject and preliminary results of our work in the context of the HPI Future SOC Lab, which generally aims on how to exploit high performance computing (HPC) capabilities for service-based 3D rendering and service-based, interactive 3D visualization. A major focus is on the application of HPC technologies for the creation, management, analysis, and visualization of and interaction with virtual 3D environments, especially with complex 3D city models.

1 Motivation

Virtual 3D city models represent a major type of virtual 3D environments. They can be defined as a digital, geo-referenced representation of spatial objects, structures and phenomena of a distinct geographical area; its components are specified by geometrical, topological, graphical and semantically data and in different levels of detail.

Virtual 3D city models are, e.g., composed of digital terrain models, aerial images, building models, vegetation models, and city furniture models. In general, virtual 3D city models serve as information models that can be used for 3D presentation, 3D analysis, and 3D simulation. Today, virtual 3D city models are used, e.g., for urban planning, mobile network planning, noise pollution mapping, disaster management, or 3D car and pedestrian navigation.

In general, virtual 3D city models represent prominent media for the communication of complex spatial data and situations, as they seamlessly integrate heterogeneous spatial information in a common reference frame and also serve as an innovative, effective user interface. Based on this, virtual 3D city models, as integration platforms for spatial information, represent essential building blocks of today's and future information infrastructures.

1.1 Complexity of 3D city models

Virtual 3D city models are inherently complex in multiple dimensions, e.g., semantics, geometry, appearance, and storage. Major complexities are described in the following:

Massive amounts of data: Virtual 3D city models typically include massive amounts of image data (e.g., aerial images and façade images) as well as massive amounts of geometry data (e.g., large number of simple building models, or smaller number of buildings modeled in high detail). Vegetation models represent another source of massive data size; a single tree model could contain, e.g., approximately 150,000 polygons).

Distributed resources: In today's so called geospatial data infrastructures (GDIs), the different components (i.e., base data) of virtual 3D city models as well as functionalities to access, and process (e.g., analyze) virtual 3D city models can be distributed over the Internet. In specific use cases such as in emergency response scenarios, they need to be identified, assembled, and accessed in an ad-hoc manner.

Heterogeneity: Virtual 3D city models are inherently heterogeneous, e.g., in syntax (file formats), schemas (description models), and semantics (conceptual models).

As an example, the virtual 3D city model of Berlin contains about 550,000 building models in moderate and/or high detail, textured with more than 3 million single (real-world) façade textures. The aerial image of Berlin (covering an area of around 850 km2) has a data size of 250 GB. Together with additional thematic data (public transport data, land value data, solar potential) the total size of the virtual 3D city model of Berlin is about 700 GB.

1.2 Service-based approach

The various complexities of virtual 3D city models have an impact on their creation, analysis, publishing, and usage. Our overall approach to tackle these complexities and to cope with these challenges is to design and develop a distributed 3D geovisualization system as a technical framework for 3D geodata integration, analysis, and usage. For this, we apply and combine principles from Service-Oriented Computing (SOC), general principles from 3D visualiza-

tion systems, and standards of the Open Geospatial Consortium (OGC).

Figure 1: 3D client for exploring the 3D city model of Berlin, running on an iPod.

To make complex 3D city models available even for small devices (e.g., smart phones, tablets), we have developed a client/server-system that is based on server-side management and 3D rendering [1]: A portrayal server is hosting a 3D city model in a preprocessed form that is optimized for rendering, synthesizes images of 3D views of this data, and transfers these images to a client, which (in the simplest case) only displays these images. By this, the 3D client is decoupled from the complexity of the underlying 3D geodata. Also, we can optimize data structures, algorithms and rendering techniques with respect to specialized software and hardware for 3D geodata management and 3D rendering at the server-side. – Figure 1 shows our 3D client running on an iPod; it allows a user to interactively explore the virtual 3D city model of Berlin.

Our project in the context of the HPI Future SOC Lab aims on research and development of how to exploit its capabilities for such a distributed 3D visualization system, especially for 3D geodata preprocessing, analysis, and visualization. The capabilities of interest include the availability of many cores, large main-memory, GPGPU-based computing, and parallel rendering.

2 Processing massive 3D city models

As raw geodata cannot be used directly for visualization and rendering purposes, this data needs to be transformed into graphics representations that can be rendered by a 3D rendering engine. Geodata preparation includes preprocessing of terrain data (geometry and textures, e.g., aerial images) as well as preprocessing of building data (geometry and façade textures). Building data are, e.g., originally provided in the CityGML format, an XML-based standard model and format for the specification of 3D city models, including semantics, geometry, attributes, and appearance information. Figure 2 illustrates the preprocessing of such CityGML building data, which includes the following three major stages and sub tasks:

Data extraction: building feature extraction, geometry extraction and triangulation, object-id assignment to each building object, spatial organization of all buildings in a quadtree structure.

Geometry optimization: geometry batching, data serialization.

Façade texture optimization: texture atlas computation, texture atlas tiling, texture coordinate adjustment.

Typically, the texture data of a large virtual 3D city model does not completely fit into a graphics card's texture memory. Thus, rendering large 3D city models requires selecting and loading the data (in appropriate level of detail) that is required for a specific camera position and view frustum. Texture preprocessing and optimization is a time-consuming stage

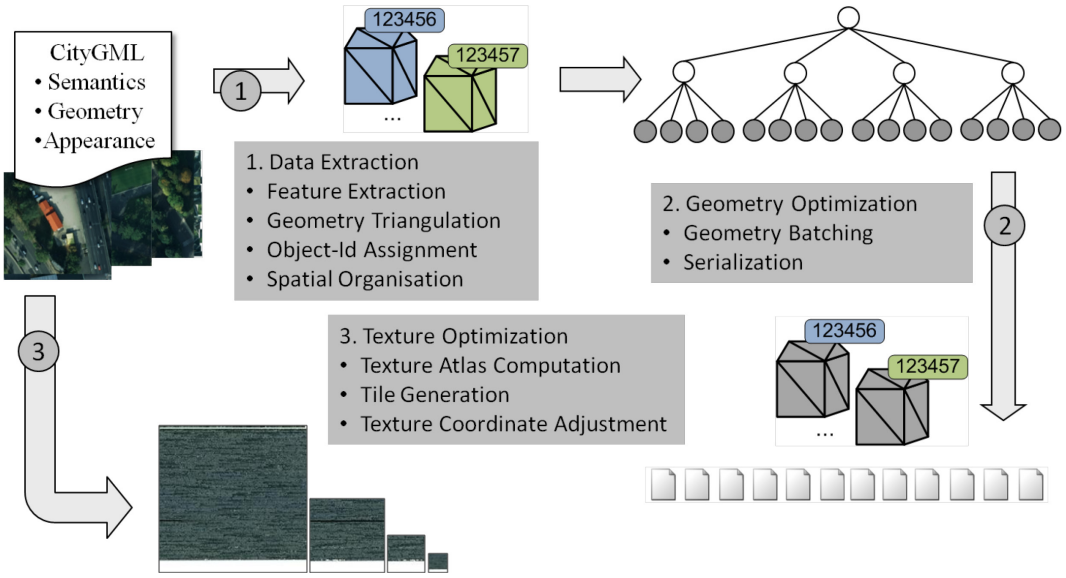


Figure 2: Preprocessing scheme for 3D geodata for 3D rendering.

in the preparation of massive 3D building data for efficient rendering and visualization. It includes a) arranging many single façade textures in texture atlases which represent parts of a very large virtual texture and b) cutting these texture atlases into even smaller parts that could be selected and handed over to the graphics card to render a specific view.

Our original implementation of this texture preprocessing stage was designed for being executed on standard desktop PCs. Due to relatively small main memory and disk space (compared to HPC servers), it had to encode and store intermediate texture data as image files on the hard disk. Starting with this implementation we experimented to take advantage of the HPC servers of the HPI Future SOC Lab to reduce the time for texture preprocessing.

First experiments included to reduce I/O-related overhead of this process. For this, we set up and used a 160 GB large virtual RAM drive in the server's main memory. Only through this, we extremely reduced the time to preprocess the massive 3D city model of Berlin (ca. 550.000 buildings including façade textures) from more than a week on a desktop PC (2.80 GHz, 8 logical cores; 6 GB main memory) to less than 15 hours on the Future SOC Lab's RX600-S5-1 (256 GB RAM; 48 logical cores).

Also, we redesigned the implementation of our preprocessing tools to take advantage of potentially very large main memory (for storing intermediate texture data instead of encoding it and writing to hard disk) and large number of available threads (for increasing the degree of parallel tiling of texture atlases).

3 Processing massive 3D point clouds

3D point clouds are another major source of 3D geodata, which are collected, e.g., via airborne or terrestrial laser scanning. 3D point clouds can represent a digital surface model of the earth's surface and are a starting point for deriving high-level data, e.g., based on classification, filtering, and reconstruction algorithms. For such algorithms it is crucial to be able to handle and manage the often very large 3D point data sets. 3D point clouds of a single city can easily contain several billion points. In the past, we had developed a set of algorithms and tools to cope with this challenge and to process, analyze, and visualize massive 3D point clouds [2, 3].

Spatial organization and rasterization are two major preprocessing tasks for 3D point clouds:

Spatial organization: To efficiently access and spatially analyze 3D points, they need to be ordered in a way that allows efficient access to the data; quadtrees and octrees represent common structures for their organization.

Rasterization: Rasterized 3D point clouds are a central component for visualization techniques and processing algorithms, as they allow efficient access to points within a specific bounding box. Rasterization transforms arbitrary distributed 3D points into a gridded, reduced, and consolidated representation; representative points are selected and missing points are computed and complemented. Rasterized point clouds are used, e.g., for the computation of triangulated surface models, for consistent level-of-detail techniques, and other efficient processing algorithms.

In the context of the Future SOC Lab we have started to research on how the HPC capabilities can help to improve speed and quality of these two tasks.

3.1 Spatially organizing 3D point clouds

To create a quadtree/octree structure, we have used the PARTREE algorithm, a parallel algorithm that creates several trees that are combined to a single one later (Figure 3).

The parallel quadtree/octree generation process for 3D point clouds was implemented based on OpenMP. We tested our implementation with data sets of up to 26.4 million 3D points. In this setting, the HPC system with 8 logical cores (FluiDyna Typhoon, 2.4 GHz, 24 GB RAM, 8 logical cores) was faster than a Desktop PC with 4 logical cores; however, the gain in time is not dramatically (Figure 4). The time required to merge the generated component trees into one tree is increasing with their number (i.e., the number of threads); in this step, the HPC server outperforms the desktop system.

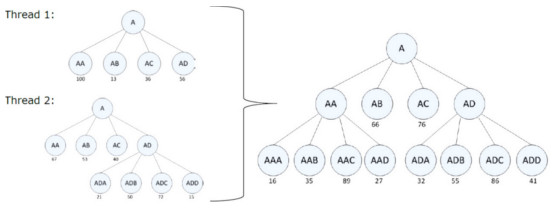


Figure 3: Functional scheme of PARTREE algorithm. Two threads create two local trees (here: quadtrees) that are merged then into a single one.

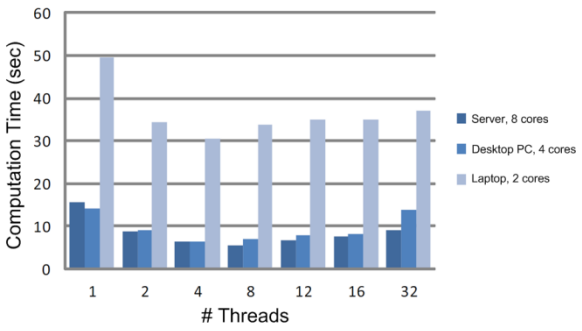


Figure 4: Time to compute an octree from 26.4 million 3D points by PARTREE method.

3.2 Rasterization of 3D point clouds

Rasterization is a multi-step process which 1) identifies for each point of an unordered 3D point cloud the corresponding raster cell and assigns the raster cell ID, 2) ordering the points according to their raster cell ID, 3) computing one characteristic 3D point for each raster cell (representing the relevant input points), and 4) interpolating cell points for empty raster cells. For sorting the points according to their raster cell ID, our implementation uses the Bitonicsort algorithm, which requires to establish a bitonic order before sorting.

We have implemented the rasterization algorithm in two versions: a) a version for multi-core processors (using OpenMP) and b) a data-parallel GPU-based version (using CUDA). The CUDA version has been tested with the Future SOC Lab's TESLA system. It can rasterize a point cloud of 30.5 million 3D points in only 22 minutes in contrast to more than 5 hours of a single threaded CPU-version.

4 Next Steps

The continuation of our work in the area of exploitation of HPC capabilities for service-based 3D rendering and 3D visualization systems and in the context of the HPI Future SOC Lab will include further improvement of our algorithms, processes, and tools. Also, we plan to extend our work to research and development on HPC-based analysis of massive 3D geodata (such as solar potential analysis) and on service-based technologies for assisted interaction and camera control in massive virtual 3D city models [4]. For this, we will exploit parallel and GPU-based algorithms to generate so called "best views" on virtual 3D city models based on visual, geometrical and usage-related characteristics.

5 Conclusions

This report briefly described the subject of our research and development in the context of the HPI future SOC Lab, preliminary results, as well as intended future work. Work and results were mainly in the areas of preprocessing massive 3D city model data and processing of massive 3D point clouds. Here, we could dramatically increase the time required to preprocess raw 3D geodata (CityGML data with geometry and textures; and massive 3D point clouds). Also, we identified additional opportunities for optimizing these algorithms. More generally, this work leads to new opportunities for research and development on advanced and innovative technologies for the exploitation (e.g., analysis and visualization) of massive spatial 3D data sets.

Acknowledgment

We would like to thank Markus Behrens for his help with the implementation and evaluation of 3D point cloud processing and its evaluation.

References

[1] D. Hildebrandt, J. Klimke, B. Hagedorn, J. Döllner: Service-oriented Interactive 3D Visualization of Massive 3D City Models on Thin Clients. *In: Proc. of 2nd Int. Conf. on Computing for Geospatial Research & Application COM.Geo 2011*, 2011.

[2] R. Richter, J. Döllner: Out-of-Core Real-Time Visualization of Massive 3D Point Clouds. *In: Proc. of 7th Int. Conf. on Virtual Reality, Computer Graphics, Visualisation and Interaction in Africa*, pp. 121-128, 2010.

[3] R. Richter, J. Döllner: Potentiale von massiven 3D Punktwolkendatenströmen. *In: Proc. of Geoinformatik 2012*, Braunschweig, 2012.

[4] J. Klimke, B. Hagedorn, J. Döllner: A Service-based Concept for Camera Control in 3D Geovirtual Environments. *In: Proc. of 7th Int. 3D GeoInfo Conference 2012*. (accepted)

Future SOC Lab Autumn Term Project Activities Report: Benchmarking and Tenant Placement for Efficient Cloud Operations

Multi-Tenancy Project Team
SAP Innovation Center Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany

Jan Schaffner
EPIC Chair, Hasso-Plattner-Institute
August-Bebel-Straße 88
14482 Potsdam, Germany

Abstract

Cloud computing mainly turns the overall IT costs of companies that move into the cloud into operational expenditures that they pay for the consumed services to a cloud provider. Consequently, the risk of correctly dimensioning the infrastructure as well as the need to keep capital expenditures as well as administrative costs at viable levels is transferred to the cloud provider. Multi-tenancy, i.e. consolidating several customers onto the same infrastructure, is one method to achieve higher utilization and therefore a more efficient cloud operation.

This project's main focus is on the efficiency and reliability of cloud services provided by server machines in virtualized, multi-tenant environments. By intelligently placing and later migrating tenants in this environment, we aim to achieve load balancing and a consistent service quality of the provided cloud services while keeping operational costs minimal. Before these investigations can take place, an evaluation and testing environment must be built up and suitable testing scenarios, e.g. in the form of benchmarks, need to be defined. This was the first major work stream that we followed out in the past lab term. The second work stream relates to tenant placement. Both are work-in-progress and are concisely described in the following. A proposal for continuation of this project in the coming lab term concludes this report.

1 Introduction

Having instantaneous access to relevant business information anywhere and anytime is becoming more and more important [1]. To realize this economically, it is necessary to deal with huge amounts of data and load at fewer costs. Big companies that felt this pressure like Amazon [2] and Google [3] developed approaches to handle these requirements that are now called Cloud Computing.

The idea behind the cloud is to offer computing as a service with seemingly endless capacity that can be added or removed on demand [4]. The user keeps the data inside the cloud infrastructure and has access to the performance of a data center to execute complex operations on it. Through the network, data can be accessed in an easy way with various devices.

One essential aspect to achieve the required cost savings is multi-tenancy [4], which describes consolidating multiple clients on a small hardware set. An additional advantage is that clients can not only share the server hardware, but also applications, content and data.

When thinking about multi-tenancy, several challenges become obvious [5, 6], such as tenant placement and tenant migration. Another very important topic is benchmarking. It is not enough to develop new technologies, but also to demonstrate where exactly the strengths of a system are and how it performs compared to others. Furthermore, there is a need to support the development process with tools that can help checking the quality of the implemented approach. To do this, benchmarks can be helpful tool.

2 Benchmarking

Current database benchmarks do not support cloud computing scenarios and multi-tenancy. To realize this, the following tasks have to be tackled:

- handling of data sizes and user scenarios that are typical for cloud computing
- supporting multi-tenant environments
- defining an industry benchmark to compare different approaches

The goal of this first work stream is to implement a first benchmark that fits the needs described above. Our approach is guided through the reuse of the already existing database benchmark called Composite Benchmark for Transactions and Reporting (CBTR) [7] developed by Anja Bog at the EPIC chair of HPI.

This system brings in several useful functionalities compared to other database benchmarks. It provides the possibility to mix analytical and transactional workloads and it operates on a dataset based on real customer data. Furthermore, it uses the Order-to-Cash scenario with a schema definition typical for SAP customers, namely implementing several tables with more than 100 columns.

The original implementation can handle one tenant that maintains the connection to the database. This tenant serves several clients that continuously query the database, to identify its properties under maximum load. This scenario had to be extended to a multi-tenant system with variable load profiles and think times.

Based on the original CBTR implementation we conducted an extension of the existing architecture as well as implemented new functionalities to support multiple tenants. We call the resulting Benchmark CBTRmt.

The following functionalities are part of CBTRmt:

- Maintaining multiple tenants at the same time, where each of them handles a number of different clients.
- Maintaining connection pooling for each of the tenants.
- Configurable think times between the queries submitted by each client.
- A load profile that specifies the workload of each tenant on the base of a 24h day.
- Exchangeable tenants.

While developing the described benchmark, the need to create an arbitrary number of tenants with different sizes, loads and datasets became apparent. The so-called Experiment Generator (ExpG) was therefore conceptualized to be able to create all necessary files and setting for a complete experiment setup.

The functionality covered by the ExpG includes the generation of realistic data sets for the database schema used in the CBTR benchmark. The implementation not only considers the data type, length, primary key and foreign key specifications, but also the size relation between the input tables. In parallel to the data generation, also query parameters are generated. This allows the querying of the generated data, just as it would be the original customer data. The ExpG is implemented in Java, based on a randomization algorithm. By specifying the seed of the Random object, one can influence the data generation directly. It is also possible to regenerate an old dataset by knowing its seed value.

One open issue for the ExpG is the support of different database schemas, which is already planned for the upcoming project phase. The output of the ExpG can then be directly loaded into the CBTRmt Benchmark and run as a whole experiment on the database.

This work stream heavily employed the SAP HANA machine in the Future SOC Lab as well as one to two high-performance servers to generate the tenant load on the SAP HANA machine as client machines.

3 Tenant Placement

The second work stream focuses on the Robust Tenant Placement and Migration Problem (RTP). The goal of this optimization problem is to assign $r \geq 2$ copies of a given number of tenants to a number of (cloud) servers such that

- no server is overloaded in terms of memory and CPU,
- no server contains more than one copy per tenant,
- the failure of a single server does not cause overloading any other server, and
- the number of active servers is minimal.

A tenant comprises a set of database tables as well as users issuing requests against those tables. Any tenant t is characterized by its size $\sigma(t)$ (i.e. the amount of main memory each replica of the tenant consumes) and its load $l(t)$. For in-memory column databases, the latter depends on the current request rates of the tenant's users as well as the size of a tenant. For more information about how to experimentally obtain $l(t)$ for a given set of database tables and workload, we refer to our previous work on performance prediction for in-memory column databases [9]. Note that we assume that $l(t)$ is additive across multiple tenants. We assume that queries are load balanced across tenant replicas in a round-robin fashion. Thus, a server only receives a fraction of the total load of a tenant, depending on the number of replicas.

Tenants are placed on servers. We call a server active if the server holds at least one tenant with non-zero size and non-zero load.

Example 1. Figure 1 (left) depicts an exemplary placement. Tenants A, B, C, D, E each have two copies and are distributed on four servers S1–S4. Each server has load capacity $lcap = 1.0$ and memory is limit-less ($\sigma cap = \infty$). The load consumption of the tenants are depicted as small numbers in the boxes. The total load of a tenant is split across all its replicas (e.g. the total load of tenant A is 0.2). The failure of a server, say server S2, means that server S4 must in

turn handle the load that was formerly shared between servers S2 and S4. This must not lead to the overload of S4. In this example, S4 gets a load of 0.3 after failure of S2, which does not lead to overloading server S4.

Figure 1: Left: Greedy-Placement and Mirroring. Right: Interleaving

At first glance, RTP resembles the two-dimensional bin- packing with conflicts, where the conflicts arise from the constraint that no server should hold more than one copy of the same tenant.

However, RTP is different from the two-dimensional bin-packing [8] with conflicts [9] problem because we are interested in finding an assignment of tenants to servers such that the assignment is robust towards server failures. A server failure causes a load increase on those servers that hold copies of tenants that were placed on the failed server. This excess load is shared among the remaining servers that hold replicas of those tenants. While it is common to handle replication using a static placement strategy (e.g. mirroring), we let the tenant placement be flexible within the following constraints: tenants can be placed on arbitrary servers, as long as no two copies of any particular tenant reside on the same server; and, each server must have enough spare capacity to deal with extra load coming from tenants on a failed servers. As a result of allowing a flexible placement strategy, assignments with fewer active servers can often be found, as we shall see in the following example.

Example 2. Fig. 1 (right) shows an example with five tenants, A, B, C, D, and E, as well as their load $l(t)$ (we neglect $\sigma(t)$ in this example). A conventional approach to assigning those tenants to servers would be to sort the tenants by load (in descending order), use a simple greedy algorithm assuming only one copy per tenant must be placed, and finally mirroring the resulting placement. This approach, shown on the left side of the figure, requires four servers. A flexible way to place the same tenants, which we shall call interleaving, shown on the right side of the figure, requires only three servers. Table 1 shows the total load on each server both for normal operations and with one other server failing. For example, in the mirrored case, a failure of server S1 doubles the load on S3, while the load on S2 and S4 remains constant. With interleaving, the excess load caused by a failure is distributed among multiple servers.

In addition to guaranteeing that placements are robust towards single-server failures, we acknowledge that dealing with a variable number of servers requires frequent migration of tenant replicas between servers. RTP is an incremental problem in the sense that it has to be solved periodically using an existing placement as a starting point. The frequency of the reassignment interval limits the amount of change that can be applied to the original placement. In previous work we have found that migrating tenants away from or onto an active server temporarily reduces the server's ability to serve requests [10]. In the case of in-memory column databases, we are able to precisely quantify the reduction of a servers query processing capacity *lcap* incurred by migration for a given set of database tables and workload. The size of a tenant does not impact how much the capacity of a server is reduced during migration but it affects how long it takes to migrate the tenant [10]. Thus, the amount of migration that is permissible in each step depends on the length of the reorganization interval and to what degree migrations can be performed in parallel.

Currently, we are conducting experiments on a variety of algorithmic approaches to solve the RTP as well as experiments for finding a good benchmark. Algorithms we evaluate on the Future SOC Lab computing resources range from special purpose greedy-heuristics to meta-heuristics and, finally, exact algorithms. Our experiments are work-in-progress.

4 Project Proposal for Next Lab Term

As both work streams are still work-in-progress we propose a continuation of the current project also in the Future SOC Lab spring term of 2012. We plan to apply current implementations and findings in order to achieve the defined goals. The focus remains on efficiency and reliability of cloud services provided by server machines in virtualized, multi-tenant environments. By migrating tenants in this environment, we aim to achieve load balancing and a consistent service quality of the provided cloud services.

A migration manager responsible for this process shall act as follows: First, a forecasting method estimates the load that tenants are likely to produce. This forecast could be based on load profiles of customers as well as business information such as a factory calendar. Based on this forecast, a multi-criteria optimization takes place. Examples for objectives in the optimization could be cost of the migration process

and possible re-migration, benefit of load balancing and expected quality of service.

Success will be evaluated based on fulfillment of given service level agreements in experiments based on real-world scenario as well as on other criteria that will be defined as part of this project. In a next step we plan to extend the approach to include failure prediction algorithms for migration decision-making once a potential failure has been identified. This aims at increasing the reliability of the overall cloud solution for cases in which server machines may fail. These activities would make use of the FutureSOC Lab computing resources in a similar way as in the first project phase.

5 Acknowledgements

The project members wish to thank Bernhard Rabe of Future SOC Lab/HPI for his invaluable administra-tive support to setup and run the numerous tests as part of the benchmarking efforts.

6 References

[1] Shuai Zhang, Shufen Zhang, Xuebin Chen, Xiuzhen Huo. *Cloud Computing Research and Development Trend.* Future Networks, International Conference on, pp. 93-97, 2010 Second International Conference on Future Networks, 2010.

[2] Amazon Elastic Compute Cloud [URL]. http://aws.amazon.com/ec2/, access on Feb. 2012.

[3] Google App Engine [URL]. http://code.google.com/appengine/, access on Feb. 2012.

[4] Peter Mell, Timothy Grance. *The NIST Definition of the Cloud Computing.* NIST Special Publication 800-145. 2011.

[5] Curino, Carlo et al. *Relational Cloud: A Database-as-a-Service for the Cloud.* 5th Biennial Conference on Innovative Data Systems Research, CIDR 2011, January 9-12, 2011 Asilomar, California.

[6] Dean Jacobs, Stefan Aulbach. *Ruminations on Multi-Tenant Databases*. BTW 2007: 514-521

[7] Anja Bog, Hasso Plattner, Alexander Zeier. *A mixed transaction processing and operational reporting benchmark.* Springer Science + Business Media, LLC 2010.

[8] W. Leinberger, G. Karypis, and V. Kumar. *Multi-Capacity Bin Packing Algorithms with Applications to Job Scheduling under Multiple Constraints*. In ICPP, pages 404–412, 1999.

[9] L. Epstein and A. Levin. *On Bin Packing with Conflicts*. SIAM Journal on Optimization, 19(3):1270–1298, 2008.

[10] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier. *Predicting in-memory database performance for automating cluster management tasks*. In Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany, pages 1264–1275, 2011

Towards Multi-Core and In-Memory for IDS Alert Correlation: Approaches and Capabilities

Sebastian Roschke, Seraj Fayyad, David Jaeger, Feng Cheng, Christoph Meinel
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
{sebastian.roschke, seraj.fayyad, feng.cheng, meinel}@hpi.uni-potsdam.de
{david.jaeger}@student.hpi.uni-potsdam.de

Abstract

Intrusion Detection Systems (IDS) have been widely deployed in practice for detecting malicious behavior on network communication and hosts. The problem of false-positive alerts is usually addressed by correlation and clustering of alerts. As real-time analysis is crucial for security operators, this process needs to be finished as fast as possible, which is a challenging task as the amount of alerts produced in large scale deployments of distributed IDS is significantly high. We identify the data storage and processing algorithms to be the most important factors influencing the performance of clustering and correlation. The Security Analytics Lab (SAL) is developed to make use of multi-core and in-memory processing. Using the SAL, a multitude of algorithms is implemented, such as Attack-Graph based correlation using HMMs, QROCK categorical clustering, and rule-based correlation using a knowledge base. The SAL is using the Common Event Expression (CEE) and supports generic flat log data.

1 Alert Correlation and its Performance

The alert correlation framework usually consists of several components [4]: *Normalization*, *Aggregation (Clustering)*, *Correlation*, *False Alert Reduction*, *Attack Strategy Analysis*, and *Prioritization*. Over the last years, alert correlation research focused on new methods and technologies for these components. IDMEF[5], CEE [7] and CVE [6] are important efforts in the field of *Normalization*. Approaches of aggregation are mostly based on similarity of alerts or generalization hierarchies. The correlation algorithms [4] can be classified as: *Scenario-based correlation*, *Rule-based correlation*, *Statistical correlation*, and *Temporal correlation*. The aspect of performance and quality is the focus of the "Security Analytics Lab" (SAL) [2]. The SAL is developed to provide an experiment environment for IDS correlation algorithms to support security operators with real-time monitoring and forensics. The SAL is designed with efficiency in mind, i.e., optimized quality and performance of the algorithms as well as the storage and organization of original alerts. The platform introduced in [2] considers different storage mechanisms and can handle massive amounts of data for specific algorithms that make heavy use of the caching mechanisms of the platform. For storage, a column-based database, an In-Memory alert storage, and memory-based index tables lead to significant improvements of the performance. Furthermore, the SAL supports the usage of GPU-based algorithms by providing necessary APIs.

We believe that research in the area of IDS and network security as application for multi-core and In-memory based platforms can provide new paradigms for conducting security. Correlation and clustering is currently only done in a limited way using filtered data sets. Using the multi-core and In-memory platforms, it might be possible to do correlation and clustering on an unfiltered data set. Thus, it might not be necessary to fine tune (e.g., exclude certain detection rules) the IDS sensors anymore, as the correlation and clustering can do meaningful reasoning on all alerts in a short time. Furthermore, we expect correlation and clustering services offered in the Cloud. A flexible and extensible correlation platform can provide the foundation work for a new paradigm in security.

2. Results and Achievements

During the last few month, we have been working on improving the implementation on the Common Event expression (CEE) [7]. Furthermore, we have been implementing several convenience features to improve the handling of the platform in case of multiple data sets. A data set switching mechanism is implemented that allows the switching of data bases from the web UI. To improve the Attack Graph (AG) based correlation, we optimized the modeling of the attack graph

and adjusted the corresponding AG correlation module. The AG-based algorithm was redesigned to fit additional attack scenarios. We further developed an algorithm for evaluating attack paths and attacks progress depending on multiple conditions, such as importance of the target, existence of a public exploit, etc.
Apart from the practical achievements, we have been able to publish a paper on the Attack Graph based algorithm [3], which is currently finalized for a Journal publication. A detailed description of the results is given in the following subsections.
We deployed the prototype of the correlation platform a FutureSOC VM (1 CPU, 4 GB Ram) and developed multiple features to improve performance and usability. Furthermore, we conducted some tests and experiments using the NVIDIA FluiDyna System as well as the Fujitsu RX600 S5 1.

3 Towards High-quality Attack-Graph-based Correlation

In this section, a modified AG based correlation algorithm is described which only creates explicit correlations. Implicit correlations, as described in [14], make it difficult to use the correlated alerts in the graph for forensic analysis of similar attack scenarios. Furthermore, the hardware environment used for the In-Memory databases provides machines with huge amounts of main memory which downgrades the priority of memory efficiency for this work. The algorithms consists of five steps, while each step can be parameterized to fine tune the results: 1) preparation, 2) alert mapping, 3) aggregation of alerts, 4) building of an alert dependency graph, and 5) searching for alert subsets that are related. In the preparation phase, all necessary information is loaded, i.e., the system and network information is gathered, the database with alert classifications is imported, and the AG for the network is loaded. The proposed algorithm works based on the exploit dependency graph created by MulVAL [15]. This type of attack graph is used simply because the tool is freely available. MulVAL is used to generate an AG which describes the corresponding system and network information for the target network. The output of MulVAL is a simple graph description which is not using a standardized format. Thus, the MulVAL output is interpreted and a corresponding graph structure is build in memory.
The algorithm is based on a set of basic definitions.

3.1 Definitions

Let $\mathcal{T}$ be the set of all timestamps, $\mathcal{H}$ be the set of possible hosts, and $\mathcal{C}$ be the set of classifications. $\mathcal{A}$ can be defined as:

$$\mathcal{A} = \mathcal{T} \times \mathcal{H} \times \mathcal{H} \times \mathcal{C} \tag{1}$$

Let a single alert $a \in \mathcal{A}$ be a tuple $a = (t, s, d, c)$ while the following functions are defined:

- $ts(a) = t$ - returns $t \in T$, the timestamp of the alert
- $src(a) = s$ - returns $s \in H$, the source host of the alerts
- $dst(a) = d$ - returns $d \in H$, the destination host of the alert
- $class(a) = c$ - returns $c \in C$, the classification of the alert

Let $\mathcal{I}$ be the set of impacts described by MulVAL [15] and $\mathcal{VR}$ be the set of known vulnerabilities. Let V be a set of vertices defined as:

$$V = \mathcal{I} \times \mathcal{H} \times \mathcal{VR} \tag{2}$$

For each triple $v = (im, h, r) \mid v \in V$, the following functions are defined:

- $imp(v) = im$ - returns $im \in \mathcal{I}$, the impact of the vertex
- $host(v) = h$ - returns $h \in \mathcal{H}$, the host of the vertex
- $ref(v) = r$ - returns $r \in \mathcal{VR}$, the vulnerability reference of the vertex

Let $AG = (V, E)$ be an AG with vertices V and edges E. An edge $e \in E \subseteq V^2$ is an ordered tuple of vertices (v, v') with $v \in V \wedge v' \in V$. PAG defines all the paths in the AG. The path $P \in PAG$ is defined as a set of edges $P = (v, v') \in E$. $ord(P)$ defines the number of edges in the path P. $in(v, P)$ depicts whether a vertex lies in the path:

$$in(v, P) := \exists (v, v') \in P \vee \exists (v', v) \in P \tag{3}$$

3.2 Mapping

The mapping function map_i maps matching alerts to specific nodes in the AG and is defined as:

$$map_i : a \mapsto \{v \in V \mid \Phi_i(a, v)\} \tag{4}$$

There are different kinds of $\Phi_i(a, v)$ defined in (5), (6), (7), (8), and (9) to parameterize the mapping function.

$$\begin{aligned} \Phi_1(a, v) := \quad & \exists v' \in V : (src(a) = host(v')) \\ & \wedge (dst(a) = host(v)) \\ & \wedge (class(a) = ref(v)) \end{aligned} \tag{5}$$

$$\Phi_2(a, v) := (dst(a) = host(v)) \wedge (class(a) = ref(v)) \tag{6}$$

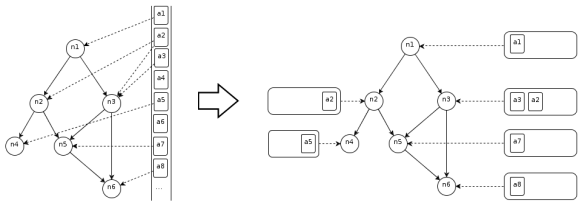


Figure 1. AG-based Correlation Algorithm - Mapping

$$\Phi_3(a, v) := (class(a) = ref(v)) \quad (7)$$

$$\begin{aligned}\Phi_4(a, v) := \quad & \exists v' \in V : (src(a) = host(v')) \\ & \wedge (dst(a) = host(v))\end{aligned} \quad (8)$$

$$\Phi_5(a, v) := (dst(a) = host(v)) \quad (9)$$

In this work, the modes use a specific $\Phi_i(a, v)$. The match modes are named as follows:

- $\Phi_1(a, v)$ - match mode *cvesrcdst*
- $\Phi_2(a, v)$ - match mode *cvedst*
- $\Phi_3(a, v)$ - match mode *cve*
- $\Phi_4(a, v)$ - match mode *srcdst*
- $\Phi_5(a, v)$ - match mode *dst*

Figure 1 shows the principle of the mapping step. The stream of alerts is mapped to the nodes of the AG, i.e., a filtering is done based on the match mode.

3.3 Aggregation

Let $A \subset \mathcal{A}$ be the set of alert that is supposed to be aggregated. Let th be a threshold and $x \in A, y \in A$ two alerts, then the relation R_A is defined as:

$$\begin{aligned}R_A \quad = \quad & \{(x, y) \in A^2 : \\ & (\mid ts(x) - ts(y) \mid < th) \wedge (src(x) = src(y)) \\ & \wedge (dst(x) = dst(y)) \\ & \wedge (class(x) = class(y))\}\end{aligned} \quad (10)$$

R_A^* defines an equivalence relation on the transitive closure of R_A. The alert aggregation combines alerts that are similar but where created together in a short time, i.e., the difference of the timestamps is below a certain threshold th. It defines a set of equivalence classes $A_{/R_A^*}$ over the equivalence relation R_A^*.

3.4 Alert Dependencies

Let $A_m \subset A$ be the set of alerts that have been matched to a node in an AG:

$$A_m = \{[a] \in A_{/R_A^*} \mid map_i(a) \neq \emptyset\} \quad (11)$$

The alert dependencies are represented by a graph $DG = (A_m, E_{m,k})$, with $E_{m,k}$ as defined in (12).

$$E_{m,k} = \{([x], [y]) \in (A_{/R_A^*})^2 \mid \Psi_k([x], [y])\} \quad (12)$$

The set $E_{m,k}$ can be parameterized by the functions Ψ_k as shown in (13), (14), and (15).

$$\begin{aligned}\Psi_1([x], [y]) \quad := \quad & (ts([x]) < ts([y])) \\ & \wedge (\exists (v, w) \in E : (v \in maps_i(x) \\ & \wedge w \in maps_i(y)))\end{aligned} \quad (13)$$

$$\begin{aligned}\Psi_2([x], [y]) \quad := \quad & (ts([x]) < ts([y])) \\ & \wedge (\exists P \in PAG : (ord(P) = n) \\ & \wedge (\exists v, w : \\ & (v \in maps_i(x) \wedge w \in maps_i(y) \\ & \wedge in(v, P) \wedge in(w, P))))\end{aligned} \quad (14)$$

$$\begin{aligned}\Psi_3([x], [y]) \quad := \quad & (ts([x]) < ts([y])) \\ & \wedge (\exists P \in PAG : \exists v, w : \\ & (v \in maps_i(x) \wedge w \in maps_i(y) \\ & \wedge in(v, P) \wedge in(w, P)))\end{aligned} \quad (15)$$

The dependency graph DG is defined by the matched and aggregated alerts A_m as vertices and the relations between these alerts as edges $E_{m,k}$. There are three possible ways to define these relations using Ψ_k. Ψ_1 defines two alerts as related, if they are mapped to neighboring vertices in AG. Ψ_2 defines two alerts as related, if they are mapped to two vertices in AG that are connected by the path P with the length of n. Ψ_3 defines two alerts as related, if they are mapped to two vertices in AG that are part of the same path P.
As shown in Figure 2, the alert dependency graph is created based on the mapping results.

3.5 Searching

Each path in the alert dependency graph DG identifies a subset of alerts that might be part of an attack scenario. DG is used in the last step to determine the most interesting subsets of alerts, respectively the most interesting path in the alert dependency graph. The last step of searching alert subsets is done by performing a Floyd Warshall algorithm [16, 17] to find all

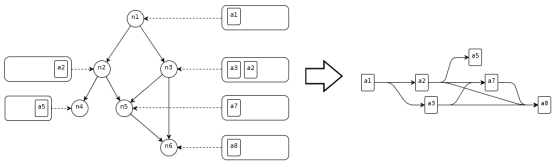


Figure 2. AG-based Correlation Algorithm - Creation

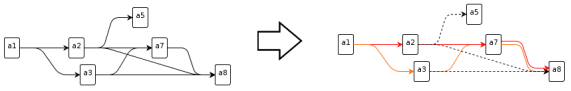


Figure 3. AG-based Correlation Algorithm - Finding

the shortest paths. Furthermore, the diameter dia (i.e. the value of the longest instance of the shortest paths) is determined and each path DP_i that has the length $ord(DP) = dia$ is converted in subsets of alerts. All the subsets S_x are defined as:

$$S_x = \{a \in A \mid in(a, DP_i)\} \tag{16}$$

Figure 3 shows the identified attack subsets in the dependency graph that correspond to the diameter.
With a simple optimization, the algorithm allows to identify multiple different attack scenarios of the same anatomy. By sorting the suspicious alert subsets according to the smallest difference between alert a_1 and a_n, the algorithm will identify the alerts that are near each other on the time-line as related to one attack scenario. Let $as = \{a_1, ..., a_n\}$ and $bs = \{b_1, ..., b_n\}$ be two alert sets where the only difference between a_i and b_i is the times-tamp $ts(a_i) \neq ts(b_i)$. There are three different combinations how these alerts can be located on a time-line:

1. as and bs are not overlapping at all, i.e., $ts(a_n) < ts(b_1)$
2. $as = \{a_1, a_2, ..., a_k, ...a_n\}$ and $bs = \{b_1, ..., b_n\}$ are partially overlapping, i.e., $\exists k \in \mathbb{N} \forall a_{i \in \{1,k\}} \mid ts(a_k) > ts(b_1)$
3. bs is completely overlapped by as, i.e., $(ts(a_1) < ts(b_1)) \wedge (ts(a_n) > ts(b_n))$

The modified algorithm can identify both suspicious alert sets in case 1. Due to the memory limitation, this algorithm only considers the last matching alert for each node in the AG. The cases 2 and 3 are difficult to be identified correctly.

4 Evaluation and Analysis of Attack Graphs

Attack Graphs describe a pre-defined known set of attack scenarios that are existing in the current network under surveillance. These scenarios can be used to match attack steps very closely and to follow an attacker through the network in real-time. Additionally, the different attack paths can be evaluated according to certain severity scores. Thus, having a low severity score on a currently exploited attack path, the security operators might be able to finish more urgent tasks before dealing with this specific attack. Whereas if an attacker is exploiting an attack path that covers very important systems and therefore having a high severity score, the security operators might need to deal with this attack urgently. One example application of the evaluation of the attack paths is the sensor placement. Before data can be monitored and be managed in the SAL, sensors have to be installed in the scenario. These sensors can come in two possible forms, i.e., as host-based IDS (HIDS) or as network-based IDS (NIDS). Each of these forms is suited for the detection of its specific kind of attack. Host-based sensors can be used to monitor unexpected behavior on a specific host while Network-based sensors can be used to monitor unexpected behavior on the network. Our goal is an effective monitoring infrastructure, which only employs a small number of sensors that can detect each possible attack in the scenario. With these conditions and the outlined preferences for the sensors, we can place sensors with the following strategy.
HIDS sensors are the only way to detect local exploits, thus installing a HIDS on all hosts that have at least one locally exploitable vulnerability might be necessary. NIDS on the other hand, can only detect remote exploits but can detect these for multiple hosts, depending on the host they are installed on. In this respect it is necessary to find all possible remote exploits and extract the hosts that are routing the network traffic for these exploits. The severity score can be considered to find an optimal solution according to the scenario. In order to do this, all single step attack paths can be extracted from the attack graph and the network route between the source and target of the attack path can be determined. Eventually, all hosts need to be listed in the network route, except the starting node (e.g. the external network connection), because this one can be considered to be under control of the attacker. Using the set cover algorithm described in [18], a minimal number of hosts can be found, which are able to intercept all remote exploits. Finally, these hosts are directly used for the placement of the NIDS.

5. Future Work

Within the next few months, we want to conduct a few more step to improve the quality of the correlation al-

gorithms deployed on this platform. We would like to work towards our vision with the following practical steps. Setting up a real deployment with multiple sensors to conduct practical experiments might be very useful. Testing the platform with a dataset of 1 TB as well as implementation of more algorithms with multi-core support will prove the applicability of the platform. The implementation of HANA support can provide a performance boost and improves the flexibility of the SAL. An interactive coding module for security operators would provide exploration functionality for data sets. Apart from the implementation requirements, the are multiple research topics, such as research on correlation algorithms that are using environment information and attack graphs, research on statistical correlation algorithms, research on visualization techniques for correlation results, research on prediction of alerts and events, research evaluation of attack paths and attackers, research on prevention techniques, simulation of attack traffic and alerts, as well as research on correlation of log events and corresponding requirements (in contrast to correlation of IDS alerts). We hope that we can open one or two of these new research areas by prolonging the project.

References

[1] S. Roschke, F. Cheng, Ch. Meinel: *Using Vulnerability Information and Attack Graphs for Intrusion Detection* In: Proceedings of 6th International Conference on Information Assurance and Security (IAS'10), IEEE Press, Atlanta, United States, pp. 104-109 (August 2010).

[2] Roschke, S., Cheng, F., Meinel, Ch.: An Alert Correlation Platform for Memory-Supported Techniques. In: Concurrency and Computation, Wiley Blackwell, 2011 (to appear).

[3] Roschke, S., Cheng, F., Meinel, Ch.: A New Correlation Algorithm based on Attack Graph. In: Proceedings of the 4th Conference on Computational Intelligence in Security for Information Systems (CISIS'11), Springer LNCS 6694, Torremolinos, Spain, pp. 58-67 (2011).

[4] R. Sadoddin, A. Ghorbani: *Alert Correlation Survey: Framework and Techniques*, In: Proceedings of the International Conference on Privacy, Security and Trust (PST'06), ACM Press, Markham, Ontario, Canada, pp. 1-10 (2006).

[5] Debar, H., Curry, D., Feinstein, B.: *The Intrusion Detection Message Exchange Format, Internet Draft*, Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004).

[6] Mitre Corporation: *Common vulnerabilities and exposures (CVE)*, WEBSITE: http://cve.mitre.org/ (accessed Mar 2012).

[7] Mitre Corporation: *Common Event Expression (CEE)*, WEBSITE: http://cee.mitre.org/ (accessed Apr 2011).

[8] H. Plattner: *A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database*, In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'09), ACM Press, Providence, Rhode Island, USA, pp. 1-2 (2009).

[9] S. Roschke, F. Cheng, Ch. Meinel: *An Extensible and Virtualization-Compatible IDS Management Architecture*, In: Proceedings of 5th International Conference on Information Assurance and Security (IAS'09), IEEE Press, vol. 2, Xi'an, China, pp. 130-134 (August 2009).

[10] Ning, P. and Xu, D.: *Adapting Query Optimization Techniques for Efficient Intrusion Alert Correlation*, Technical Report, North Carolina State University at Raleigh, 2002.

[11] Northcutt, S., Novak, J.: *Network Intrusion Detection: An Analyst's Handbook*, New Riders Publishing, Thousand Oaks, CA, USA (2002).

[12] Tedesco, G. and Aickelin, U.: *Real-Time Alert Correlation with Type Graphs*, In: Proceedings of the 4th international Conference on Information Systems Security (ISS'09), Springer LNCS 5352, Hyderabad, India, pp. 173-187 (2008).

[13] Ning, P., Cui, Y., Reeves, D.: Constructing attack scenarios through correlation of intrusion alerts, In: Proceedings of the *9th ACM Conference on Computer and Communications Security (CCS'02)*, ACM, New York, NY, USA, pp. 245-254 (2002).

[14] Wang, L., Liu, A., and Jajodia, S.: Using attack graphs for correlation, hypothesizing, and predicting intrusion alerts In: *Journal of Computer Communications*, Elsevier, Volume 29, Issue 15, pp. 2917-2933 (September 2006).

[15] Ou, X., Govindavajhala, S., and Appel, A.: MulVAL: A Logic-based Network Security Analyzer, In: *Proceedings of 14th USENIX Security Symposium*, USENIX Association, Baltimore, MD, pp. 8 (August 2005).

[16] Floyd, R.: *Algorithm 97 (SHORTEST PATH)*, In: Communications of the ACM, vol. 5, Issue 6, pp. 345 (1962).

[17] Warshall, S.: *A Theorem on Boolean Matrices*, In: Journal of the ACM, vol. 9, Issue 1, pp. 11-12 (1962).

[18] Noel, S., Jajodia, S.: *Attack Graphs for Sensor Placement, Alert Prioritization, and Attack Response*, In: *Cyberspace Research Workshop*, pp. 3-20 (2007).

Multicore-Based High Performance IPv6 Cryptographically Generated Addresses (CGA)

Hosnieh Rafiee, Ahmad AlSa'deh, and Christoph Meinel
Hasso-Plattner-Institut, University of Potsdam
P.O. Box 900460, 14440 Potsdam, Germany
{Hosnieh.Rafiee, Ahmad.Alsadeh, Christoph.Meinel}@hpi.uni-potsdam.de

Abstract

Neighbor Discovery Protocol (NDP) is one of the main protocols in the IPv6 suite. It is used extensively in several critical functions such as discovering other existing nodes on the same link, determining others' link layer addresses, etc. but it is known to be vulnerable to critical attacks. Thus, SEcure Neighbor Discovery (SEND) is offered to counter the NDP security threats. SEND is computing intensive. SEND is computing intensive. The computing intensive aspect of SEND comes in fulfilling the Hahs2 condition for the address generation of the Cryptographically Generated Addresses (GGA). Unfortunately the CGA computation speed cannot be significantly improved by using a multicore processor because the algorithm used for the CGA generation supports a sequential process. In this paper we will describe an approach that breaks down the CGA algorithms in order to spread their processes over all of the available cores in the computing device. This proposal suggests the automatic detection of the number of cores available on a machine and then creates an equivalent number of working tasks to compute the Hash2 condition. When one task meets the CGA Hash2 condition, the other cores processes stop. To test our approach, several experiments were carried out in the future Soc servers. The results of our experiments show that the CGA creation time can be spedup by increasing the number of cores in the computing device.

1 Introduction

Cryptographically Generated Addresses (CGA) [1] is designed to provide for the authentication of IPv6 addresses and to prevent malicious nodes from taking ownership of others' addresses. Unfortunately CGA, as an essential option in SEcure Neighbor Discovery (SEND) [2], is computationally intense, especially when; meeting high security level requirements. The security parameter (Sec) designates the security level of the CGA address. For Sec values greater than zero, there is no guarantee that the sequential brute-force search will cease after a certain time period. For example, it may take several hours or even days to find the CGA parameters when security level of "2"is being used. This long delay is unacceptable for several applications.

Today multicore processors greatly increase the computational capacity of computers and their use has become the norm. Nearly every computer has a processor with at least two cores. Some desktops have up to 8 CPUs. These CPU numbers are likely to increase in the future. Multicore processors thus provide a potential opportunity for remarkable speed improvements in CGA computation. However a dual-core processor will not automatically double the CGA computational performance because the CGA generation algorithm is a sequential process and will not fully utilized the available CPU cores.

In this paper we will introduce our approach to multicore processing in order to achieve better performance for CGA computations [3]. In our algorithm we do a brute-force search to find a valid modifier in parallel rather than sequentially. The number of cores in the computing device determines the number of parallel threads that will be used to compute the CGA address parameters. The multicore CGA version drastically reduces the CGA generation time. The experiment results are validated and evaluated by comparing the performance of the proposed approach with the conventional approach in the future Soc servers.

This paper is structured in the following manner. Section 2 presents the CGA algorithm and our proposal to use multicore threading in the generation of a CGA. . Section 3 outlines the details of our parallel approach and presents our experimental results using the future Soc servers. Section 4 concludes the work and describes future steps.

2 Project Description

CGA was first proposed as a mechanism for authenticating location updates in Mobile IPv6 [4]. Later, CGAs were standardized in the context of the SEcure Neighbor Discovery (SEND) [2] in order to protect Neighbor Discovery (ND) for IPv6 [5] and the IPv6 Stateless Address Autoconfiguration [6] against known attacks [7].

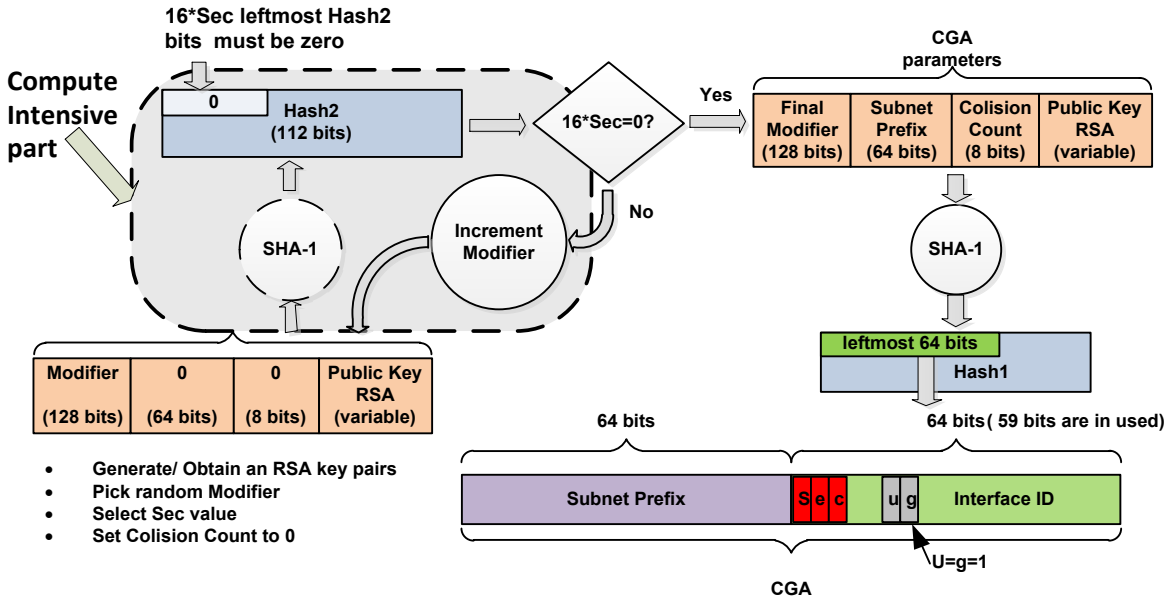


Figure 1: CGA Generation Algorithm

CGA was also proposed to prevent Denial-of-Service (DoS) attacks and to authenticate the Binding Update messages in Mobile IPv6 [8, 9]. The main disadvantage of using CGA for these purposes is the computational cost. This high computational cost is due to the fact that the CGA algorithm is a sequential process and thus, as it stands now, does not lend itself to relief in processing time by running in a multicore environment. CGA computations may take a long time especially when a high security value is used.

The fact is that satisfying the Hash2 condition is actually the most computational expensive part of CGA generation algorithm

2.1 Cryptographically Generated Addresses (CGA) Algorithm

In CGA, the interface identifier portion of the IPv6 address is created from a cryptographic hash of the address owner's public key and other auxiliary parameters. Since the 64-bit are not enough to provide sufficient security against brute-force attacks in the foreseeable future, the standard CGA uses the Hash Extension in order to increase the security strength above 64 bits. The address owner computes two independent hash values (Hash1 and Hash2) by using the public key and other parameters. The Hash Extension (Hash2, or a portion of it) value sets an input parameter for Hash1. The combination of the two hash values increases the computational complexity for generating a new address and thus the cost of using brute-force attacks. The CGA generation algorithm should fulfill two conditions [1]:

1. The leftmost 64-bit of Hash1 is set to equal the interface identifier. The Sec, "u" and "g" bits are ignored for this comparison.

2. The 16×Sec leftmost bits of Hash2 are equal to zero.

The security parameter (Sec) indicates the security level of the generated address being used against the brute-force attacks. Increasing the Sec value by "1" adds 16 bits to the length of hash that the attacker must break. The Sec is an unsigned 3-bit integer having a value between "0" and "7". The CGA parameter data structure contains the following parameters:

1. Modifier (128-bit): initialized to a random value.
2. Subnet Prefix (64-bit): set to the routing prefix value advertised by the router at the local subnet.
3. Collision Count (8-bits): is a counter used for Duplicate Address Detection (DAD) to ensure the uniqueness of the generated address.

4. Public Key (variable length): set to the DER-encoded public key of the address owner.

5. Extension Field: variable length field for future needs.

A schematic of the CGA generation algorithm is shown in Figure 1. CGA generation begins by determining the address owner's public key and selecting the proper Sec value. The process continues with the Hash2 computation loop until the Final Modifier is found. The Hash2 value is a hash value comprises of a combination of the Modifier and the Public Key which is concatenated with a zero-value of Subnet Prefix and Collision Count. The address generator tries different values of the Modifier until the 16×Sec-leftmost-bits of Hash2 reaches zero. Once a match is found, the loop for the Hash2 computation terminates. At this point the Final Modifier value is saved and used as an input for the Hash1 computation. The Hash1 value is a hash combination of the entire CGA parameter data structure.

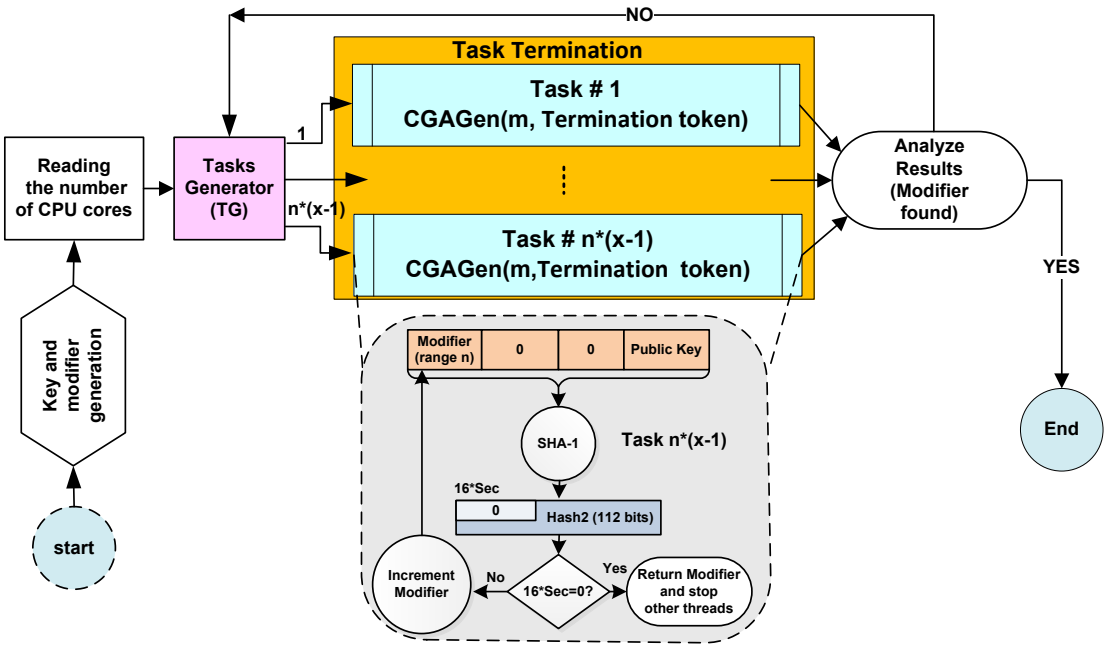


Figure 2: Flowchart of the Proposed Parallel CGA Generation Process

After the interface identifier (IID) is derived from Hash1 the hash value is truncated to the appropriate length (64-bit). The Sec value is encoded into the three leftmost bits of the interface identifier. The 7th and 8th bits from the left of IID are reserved for a special purpose. Finally the Duplicated Address Detection (DAD) process is done to ensure that there is no address collision within the same subnet.

3 Results and Achievements

We implemented a multicore-based CGA and conducted several practical experiments by running the system in a Future SOC infrastructure. We could then divide the CGA sequential algorithm into smaller tasks according to the number of cores available in the Future Soc VMs. We then assigned each task in order to run them in parallel. Even though the CGA algorithm speedup is influenced by tightly-coupled dependencies between the parts of the algorithm, we could achieve a speedup in the parallel mode. In the following subsections we explain our CGA parallel algorithm and our results.

3.1 Implementation

CGA generation is a sequential algorithm. Therefore the main challenge of CGA parallelization is to break down the CGA algorithm into parallel parts and then to assemble those individual results from each process into the final result. To parallelize a CGA algorithm two things should be determined: which part of the CGA algorithm can be parallelized and into how many parallel processes can it be broken. The Hash2 condition needs to be broken into a number of tasks because it is the most intensive part of the CGA computation. To utilize all cores on a device, the number of tasks is determined based on the number of CPU cores. In this way, the CGA generation algorithm can scale its performance based on the available CPU cores [3].

Figure2 shows how the CGA generation can be parallelized to use an n-core computing device [3]. In this approach each task is run in one core to prevent the CPU from switching between the parallel tasks during the CGA generation. The Tasks Generator (TG) passes particular continuous sets of modifier for each task, i.e. each task dedicated to do brute-force searches within a certain range. If the task1 process modifier range is from 0 to n, then task2 would process in a range of n+1 to 2*n, and so on. TG also passes a termination token to the CGA Generation (CGAGen) function in order to control and terminate all tasks when one of the tasks succeeds in finding a valid modifier which fulfills the Hash2 conditions. If none of the tasks succeeds in finding the valid modifier within the assigned range, then new tasks with new ranges of modifiers are generated. We use some customized classes in the Task Parallel Library (TPL) [9] to simplify the parallelization management. TPL simplifies this process and provides a number of classes and methods which allows the running of a number of jobs in a parallel fashion. TPL has a Task Factory Class [10] which provides support for creating and scheduling Task Objects without sacrificing their power and flexibility.

3.2 Experiments

We ran our CGA multicore implementation on two VMs with a Windows 7 64-bit operating system. In our experiments the first VM machine was a quad-core CPU and the second VM machine was a single-core CPU. Both machines had 2 GB of RAM. The

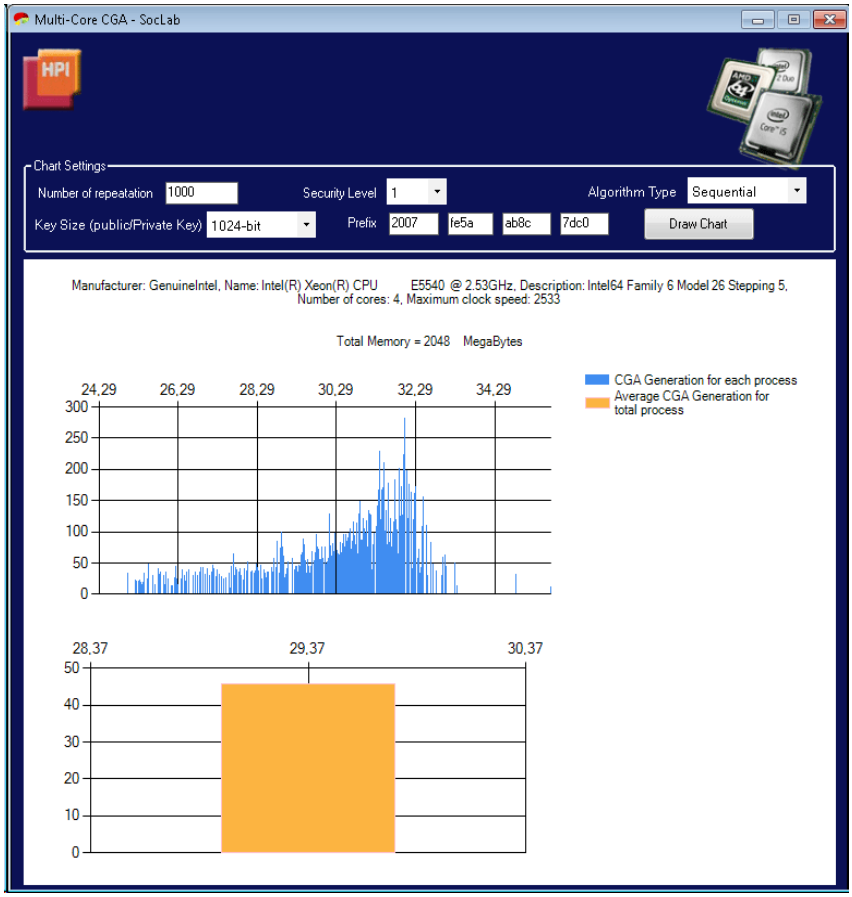


Figure 3: Multicore CGA Analyzer User Interface. The top chart shows the result of each CGA generation time and the bottom chart shows the average time needed to generate the CGA

Table1: CGA Average Generation Time Comparison of Sequential and Parallel

Number CPU cores	CGA average generation time (Milliseconds) 1024-bit RSA key, Sec=1						Percentage of Speedup
	Parallel Mode (CPU usage 85%)			Sequential Mode (CPU usage 29%)			
	Min.	Max.	Avg.	Min.	Max.	Avg.	
1	-	-	-	28	66	47	-
4	1	45	13,36	8	281	45,7	70.7%

CPUs in both VMs were Intel(R) Xeon(R) E5540 with 2.53GHz speed. Figure 3 shows our CGA multi-core analyzer user interface. The user can select the RSA key size, algorithm type and determine the number of samples. All the measurements are done for a RSA key size equal to 1024-bit. The CGA address was generated 1000 times to ensure sufficient samples. The average (avg.), the minimum (min.) and the maximum (max.) values of CGA generation times for both sequential and parallel mode and a Sec value "1" are recorded in Table 1. The CGA Multicore Analyzer has the ability to draw charts and store the output results, such as average, the standard deviation, the variance, etc., in a file.

As shown in Table 1, the parallel approach spedup the CGA computation by 70.1%, when using 4 cores It is clear from the results that the process of computing the CGA benefited from the use of a multicore processor. These results show that even though the sequential process did improve using a multicore processor, the most dramatic improvements were achieved by using the parallel processing approach that we introduced here. In our parallel approach, described above, more mips are used because of the ability to process some aspects of the address calculation functions simultaneously, i.e. in each core, thus lowering the overall process time. Figure 4 shows the CPU performance in multi-core for both parallel and sequential algorithms while running CGA algorithm.

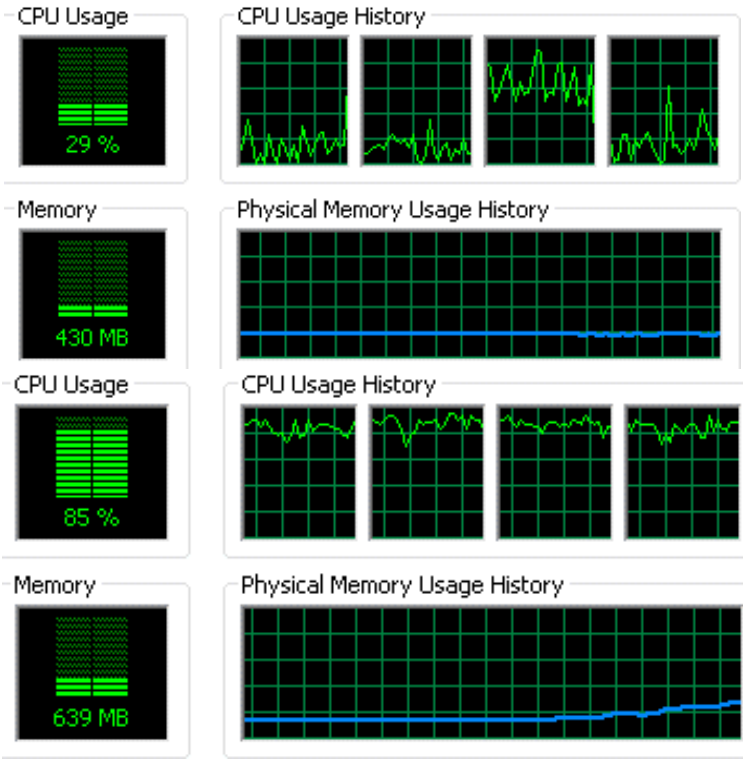


Figure 3: Top figure shows CPU performance in sequential mode and bottom figure shows in parallel mode

4 Future Works

We implemented the multicore CGA using the most CPU power in a computing device. We would like to extend our work to a Linux operating system. We would like to work towards this goal using the following steps:

- Complete our CGA multicore algorithm for the Linux operating platform
- Test the performance of the foregoing algorithm in the future Soc servers.
- Research on better ways to utilize the CPU's memory in order to manipulate the CGA algorithm more efficiently.

References

[1] T. Aura: *Cryptographically Generated Address*, RFC3972, WEBSITE: *http://tools.ietf.org/html/rfc3972* , In: Internet Engi-neering Task Force (March 2005)

[2] J. Arkko, J. Kempf, B. Zill and P. Nikander: *SEcure Neighbor Discovery (SEND)*, RFC 3971 (Proposed Standard), WEBSITE: *http://tools.ietf.org/html/rfc3971*, In : Internet Engineering Task Force (March 2005)

[3] H. Rafiee, A. Alsa'deh, and Ch. Meinel *: Multicore-Based Auto-Scaling SEcure Neighbor Discovery for Windows Operating System*s, In: in Proceedings of the 26th International Conference on Information Networking (ICOIN 2012), IEEE Press, Bali, Indonesia (February 2012)

[4] G. O'Shea and M. Roe : *Child-proof authentication for MIPv6 (CAM),* In: ACM Computer Communications Review, vol. 31, no 2. (2001)

[5] T. Narten, E. Nordmark, W. Simpson, and H. Soliman: *Neighbor Discovery for IP version 6 (IPv6)*, RFC 4861, WEBSITE: *http://tools.ietf.org/html/rfc4861*, In: Internet Engineering Task Force (September 2007)

[6] S. Thomson, T. Narten and T. Jinmei *: IPv6 Stateless Address Autoconfiguration* , RFC 4862, WEBSITE: *http://tools.ietf.org/html/rfc4862,* In: Internet Engineering Task Force (September 2007)

[7] P. Nikander, J. Kempf and E. Nordmark : *IPv6 Neighbor Discovery (ND) Trust Models and Threats*, RFC 3756 (Informational), WEBSITE: *http://tools.ietf.org/html/rfc3756,* In: Internet Engineering Task Force (May 2004)

[8] J. Arkko, C. Vogt and W. Haddad : *Enhanced route optimization for mobile IPv6*, RFC 4866, WEBSITE: *http://tools.ietf.org/html/rfc4866,* In: Internet Engineering Task Force (May 2007)

[9] Microsoft TechNet, Task Parallelism (Task Parallel Library), WEBSITE: *http://msdn.microsoft.com/en-us/library/dd537609.aspx* (2011)

[10] Microsoft TechNet, Task Factory, WEBSITE: *http://msdn.microsoft.com/en-us/library/system.threading.tasks.task.factory.aspx*

Blog- Intelligence Extension with SAP HANA

Patrick Hennig
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
patrick.hennig@student.hpi.uni-potsdam.de

Philipp Berger
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
philipp.berger@student.hpi.uni-potsdam.de

Patrick Schilf
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
patrick.schilf@student.hpi.uni-potsdam.de

Christoph Meinel
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam
office-meinel@hpi.uni-potsdam.de

Abstract

Blog-Intelligence is a blog analysis framework, integrated into a web portal, with the objective to leverage content- and context-related structures and dynamics residing in the blogosphere and to make these findings available in an appropriate format to anyone interested. The portal has by now reached a mature functionality, however, requires ongoing optimization efforts in any of its three layers: data extraction, data analysis and data provision.

1 Introduction

With a wide circulation of 180 million weblogs worldwide, weblogs with good reason are one of the killer applications of the worldwide web. For users it is still too complicated to analyze the heavily linked blogosphere as a whole. Therefore, mining, analyzing, modeling and presenting this immense data collection is of central interest. This could enable the user to detect technical trends, political atmospheric pictures or news articles about a specific topic.
The basis of the Blog-Intelligence project is the big amount of data provided by all weblogs in the world. These data is gathered in the past and in the future by an intelligent crawler.
Blog-Intelligence already provides some basic analysis functionality for the crawled data. Through the improvement of the crawler and the consequent growing amount of data, the analysis gets into big performance issues. Since these performance problems, the analyses are only calculated in a weekly manner to reduce the run time of the analyses algorithms. Therefore the up-to-dateness of the results is not given any more and the web portal is only able to show already deprecated results.

2 Fields of application

With SAP HANA totally new opportunities are coming up. The fast execution of the analysis algorithms provides completely new and better interaction with the system for the end-user. Beside the advantage of exploring the blogosphere in real time, it is possible to provide analyses for the end-user calculated separately for each user with his interests. Furthermore, former time-consuming text and graph analysis algorithm can now get integrated into our framework because SAP HANA offers fast variants of these algorithms. This opens new perspectives onto the data and the blogosphere for the user.
For example, it is now possible to figure out, how and what is discussed about products or companies inside the blogosphere. Traditional providers limit these analyses to the biggest blogs worldwide. With Blog-Intelligence and SAP HANA it gets possible to calculate analyses over all weblogs worldwide.
In addition to the personalized illustration of the blogosphere, companies can figure out how their own weblogs perform and influence the blogosphere. Even the monitoring of competitors' social media influence is imaginable.

3 Used Future SOC Resources

Currently, we are running a small test machine that is embedded into the Future SOC network. This machine runs a SAP HANA instance that is used as test database for our current crawler developement. The development state of the crawler becomes a stable version and the current version is able to crawl fast and more enriched data. Therefore, we observed the SAP

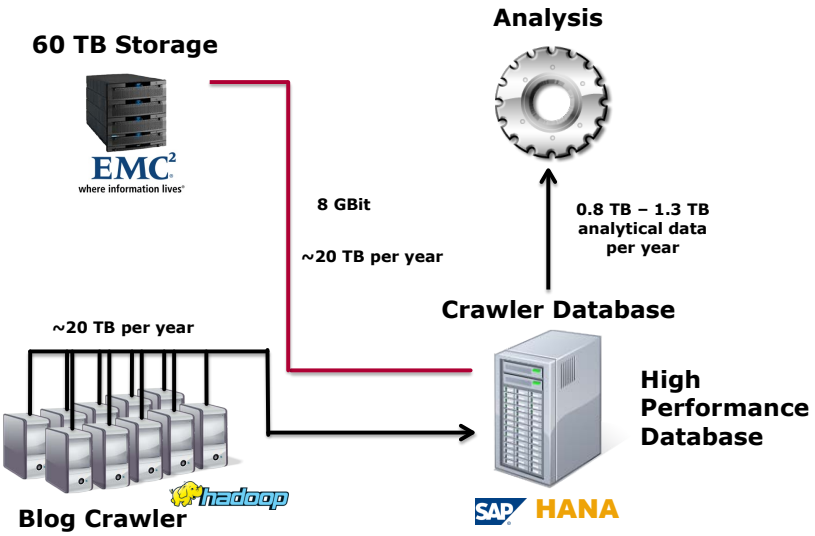


Figure 1. Technical Concept of Blog-Intelligence

HANA instance on our test machine to run out of resources. Hence, we revise our current setup and come up with a new Blog-Intelligence deployment described in the following section.

4 Blog-Intelligence deployment

Our overall technical setup is shown in figure 1. In order to increase the amount of data and to keep the data up-to-date, the weblogs are visited and revisited with the help of a crawler. The crawler is based on the MapReduce framework from Apache. The first structural analyses are done at crawling time like the language detection. The overall detection whether the web site is a blog, a news portal or a regular HTML page is also done by the crawler.
This crawler is executed in parallel and distributed on a Hadoop Cluster and saves the data directly into a SAP HANA instance. The complete extracted data, especially large objects like the original HTML content, is stored as well. Nevertheless, the large data objects get outsourced to a data storage provided by EMC with an overall capacity of 60TB. Although large objects get outsourced, an huge amount of data stays in-memory for the analytics component.

5 Computational Effort Estimation

In the worldwide web approximately 10 million highly active blogs exists with more than one post each week. An important part of these blogs is the news portal blogs, like ”theguardian” with several new posts each day. Of the widespread 180 million weblogs these 10 million weblogs are the most important weblogs. Therefore, we want to store these weblogs inside a SAP HANA database to provide up-to-date real time analyses. As a result, we expect 0.8TB to 1.3 TB of compressed data for our analyses.

6 Next Steps

As mentioned before, the crawler implementation is nearly stable. Hence, we will start a permanent run using the Future SOC resources and create an adequate set of crawled weblog sites. Given this dataset in a running SAP HANA instance, we get able to the test in-memory data analysis algorithms of HANA. We expect to measure a significant performance improvement for the analytics. Thereby, we get able to evaluate real-time and user-centric analysis approaches for the blogosphere. Based on our findings about the blogosphere, we will adapt, develop and re-engineer our existing exploration interfaces for this new level of analytics.

References

[1] P. Berger, P. Hennig, J. Bross, and C. Meinel. Mapping the blogosphere–towards a universal and scalable blog-crawler. In *Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Confernece on Social Computing (SocialCom)*, pages 672–677. IEEE, 2011.

[2] J. Bross, P. Hennig, P. Berger, and C. Meinel. Rss-crawler enhancement for blogosphere-mapping. *IJACSA Editorial*, page 51, 2010.

[3] J. Bross, M. Quasthoff, P. Berger, P. Hennig, and C. Meinel. Mapping the blogosphere with rss-feeds. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, AINA ’10, pages 453–460, Washington, DC, USA, 2010. IEEE Computer Society.

Accurate Mutlicore Processor Power Models for Power-Aware Resource Management

Christoph Meinel, Ibrahim Takouna, and Wesam Dawoud
Hasso-Plattner-Institut (HPI)
University of Potsdam
Potsdam, Germany
{christoph.meinel, ibrahim.takouna, wesam.dawoud}@hpi.uni-potsdam.de

Abstract

In this project, we develop three statistical CPU-Power models based on number of active cores and average running frequency using a multiple liner regression. Our models were built upon a virtualized server. The models are validated statistically and experimentally. Statistically, our models cover 97% of system variations. Furthermore, we test our models with different workloads and different benchmarks. The results show that our models achieve better performance compared to the recently proposed model for power management in virtualized environments. Our models provide highly accurate predictions for un-sampled combinations of frequency and cores; 95% of the predicted values have less than 7% error.

1 Introduction and Project Idea

There are several proposed approaches for power management. Mostly, these approaches consider the CPU frequency and CPU utilization to build power models. For instance, Urgaonkar et al. [1] and Gandhi et al. [2] have adopted non-liner quadric models of power consumption for power management in virtualized environments. The relationship between power consumption of multicore processor and frequency cannot be covered with one fitting curve, as we will see in Section II. For instance, the curve of power consumption of 1 core slightly increases with cores frequency compared to 4 cores and 8 cores. Moreover, Fan et al. [4] have included CPU utilization in their proposed power model. However, using utilization to build a power model for a multicore processor could be not accurate, because the power consumed by a multicore processor with one active core with 100% utilization is more than the power consumed by two active cores each of them 50% utilized for the same workload. We found this result by conducting an experiment using a virtual machine (VM) with a mutlithreaded application. This VM ran with 1 virtual CPU and had 100% utilization and only ran on one physical core. In this scenario, the power consumed by the physical CPU was 26 watts. On the other hand, when the same VM ran with 2 virtual CPU and the VM had the same total CPU utilization 100%. In this scenario, the physical CPU just consumed 17 watts, and each core was 50% utilized. Importantly, both of the configuration gave the same performance. Indeed, the latter could be better due to exploiting the multithreading. Thus, we conclude that only using CPU frequency and CPU utilization only as inputs for power modeling could be inefficient in particular for power estimation of mutlicore processors.

The purpose of this work is to build CPU-Power consumption models that accurately estimate the power consumption of virtualized servers with multicore processor. These models could be employed into power-aware resource management to achieve better power savings. Our work is distinct from others as follows. This project phase presents CPU-Power consumption models taking into account number of the actual active cores N and average running clock frequency F at each sample. It analyzes and evaluates the performance of our proposed models statistically and experimentally. The statistical analysis using the regression R^2 indicates that our models could cover more than 97% of system variations. Experimentally, our proposed models achieve better performance compared to the model adopted by [1][2]. We evaluate models using three different applications with different characteristics (i.e., CPU-intensive, Memory-intensive, and IO-intensive). The results show that 95% of the predicted values have less than 7% error. Furthermore, the maximum prediction error is less than 4% error for Memory-intensive and IO-intensive applications. As future work, we will use these models to build a dynamic optimizer that optimizes number of cores and their frequency settings and dynamically configures a VM to cope with workload and meet power consumption constrains.

2 Used Lab Resou setup

The evaluation experimer PRIMERGY RX300 S5 measurement capability. Xeon(R) CPU E5540 w range is 2.53GHz to 1.5 logical cores. The server ical memory. The experi ized server using Xen-4.1

To build our models, we mark EP Embarrassing E Parallel Benchmarks (NI tails about the characteri found in [6].

Finally, we used xenpm ning frequency and numl the CPU-Power measure to measure the power c our experiments, the pe ered to get accurate pow rizes the system overview Power models developm put of xenpm tool; it illu frequency, performance states (C0-C3). Furthermore, the output demonstrates the percentage of time for each core and for each state.

3 Findings

Several works have used linear models to represent the power consumption of a system or just a processor. These models are based on CPU utilization or other concerned resources such as memory. As current processors have multicores which could operate at different frequency levels at runtime using DVFS, in this section we discuss the relationship between the CPU-Power consumption and CPU-frequency from one side, and the CPU-Power consumption and number of active cores from the other side.

3.1 CPU-Power and frequency relationship

CPU-Power consumption is composed of dynamic and static power. The dynamic power is the important factor for reducing power consumption using DVFS technique. The dynamic consumed power by a CPU with a capacitance c, frequency f, and supplied voltage v is computed by equation $P = c.f.v_{dd}^2$. However, Kim et al. [7] have considered power proportional to the cubic of frequency because the frequency is usually in proportion to the supplied voltage.

$$P_{(F)} = P_{min} + \theta(F - F_{min})^2 \qquad (1)$$

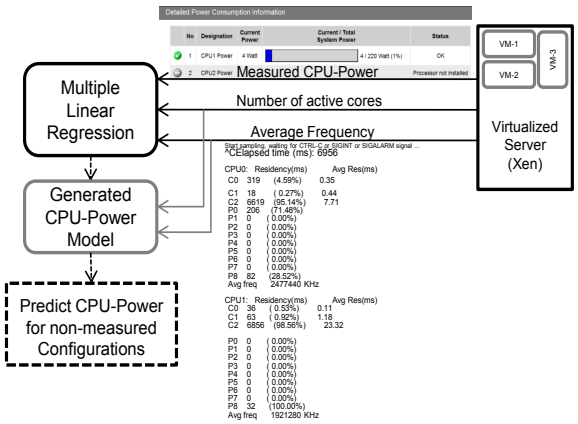
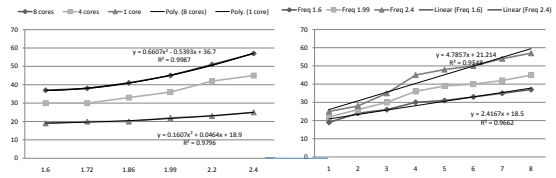


Figure 1: Overview of the system and CPU-Power models development.



(a) (b)

Figure 2: CPU-Power consumption relationship with frequency and number of active cores.

To obtain the relationship between CPU-Power consumption and frequency, we ran EP-NPB CPU-intensive benchmark on a virtual machine at different CPU frequencies. We measured the power consumption only for the CPU. Thus, we obtained the curves in Fig. 2-(a). Then, by applying the multiple linear regression, we found that the best relationship could be fit in polynomial linear with regression $R_2 = 0.99$. we could generalize it as a quadric model in equation 1 which resembles the proposed model by [1][2] to estimate the power consumption of a server. Although Fig. 2-(a) shows perfect fitting for each curve of a number of cores, the total system variations cannot be covered with considering only frequency.

Table 1: The determined values of CPU-Power models coefficients and statistics.

Model	θ_2	θ_1	α	C	Std.Err.	R^2
0.Eq.2	6.3	-7.2	0	24.6	7.9	0.29
1.Eq.3	6.3	-7.2	3.3	9.7	1.4	0.97
2.Eq.4	3.8	2.6	3.3	0	1.4	0.99

Hence, we need different values of θ and P_{min} at each number of active cores. For example, to estimate CPU-Power at frequency 1.72 GHz when 8 active cores using equation 1 the best values for θ and P_{min} are 37 watts and 37 $Watt/GHz^2$ respectively. The estimated power is 37.6 watts which is approxi-

mately equal to the measured value 38 watts. Nevertheless, The values of θ and P_{min} should be adapted again to predicted the power when just 4 cores are active. Accordingly, we study the relationship between the power consumption and number of active cores in next section.

3.2 CPU-Power and number of active cores relationship

To estimate the power consumption of multicore processors, we found that it is important to study the relationship between CPU-Power and number of active cores. To this end, we obtained the curves in Fig. 2-(b). The curves have a linear trendline. The relationship is well approximated by a linear model with regression $R^2 0.95$, which means that the power consumption and number of active cores have a strong linear association and can be represented by Equation $P_{(N)} = P_{min} + \alpha.N$. N is number of active cores, and P_{min} is the power consumed by one core running at frequency F. α_F is the slope of the power-to-active cores curve at frequency F. Importantly, each curve has two different slopes. The first one is when the number of active cores is less than 4 cores and the other one is when the number of active cores is more than 4 cores. Moreover, the first slope is greater than the second one. The main reason of this case was that we had a processor with 4 physical cores. Each physical core has two logic cores, and the power consumed by a logical core is less than the power consumed by a physical core.

3.3 CPU-Power estimation models

From previous sections, we found a strong relationship between CPU-Power consumption and both frequency and number of active cores. In this section, we refer to the model adopted by [1][2] as Model-0. Model-0 which is represented by equation 2 does not include number of active cores. Our first model is denoted by Model-1. Model-1 presented in equation 3 is a multiple linear regression with the intercept constant C. Equation 4 represents Model-2. Model-2's intercept constant C is zero. Finally, we removed the first degree term of frequency of Model-2. However, we will study the predication accuracy of these models showing the worst and the best cases for each model.

$$P_{(F,N)} = \theta_2.F^2 + \theta_1.F + C \quad (2)$$

$$P_{(F,N)} = \theta_2.F^2 + \theta_1.F + \alpha.N + C \quad (3)$$

$$P_{(F,N)} = \theta_2.F^2 + \theta_1.F + \alpha.N \quad (4)$$

3.4 Statistical analysis

This section discusses some statistical analysis of our CPU-Power estimation models focusing on Model-1 to show its efficiency to predict the power consumption of a processor.

Table 1 summarizes the determined values of CPU-Power models coefficients and statistics. However, Model-2 is a regression with zero constant. Furthermore, the regression statistics in Table 1 show that the regression R^2 of our models is higher than R^2 of the Model-0. For instance, Model-2 has regression R^2 0.99 which means that these two models could explain 99% of the power variations. The power variations were determined by variations in the independent variables (i.e., frequency and active cores). In contrast, Model-0, which only considers frequency, has regression R^2 0.259. Model-0 explained only 25% of power variations using frequency.

3.5 Performance evaluation

As we discussed models performance statistically, in this section we show and compare the performance of the models experimentally. To achieve this, we conducted three experiments using three different benchmarks of NPB benchmark namely EP, CG, and BT. These benchmarks represent CPU-Intensive, Memory-Intensive, and IO-Intensive applications respectively.

3.5.1 CPU-intensive applications

To evaluate our models against CPU-Intensive applications, we used EP benchmark to generate a workload which was changed with time. As shown in Fig. 3, we started with a low workload which increased with time until it reached its maximum approximately at time 205 sec. Then, it started to decrease after time 250 sec. During the experiment, we measured the CPU-Power consumption every 5 seconds. Then, we computed the estimated power using the four different models. Obviously, the curve shows that Model-0 has a big difference between its estimation and the measured power when low-workload and few of cores are actives (i.e., 1-3 active cores). However, it shows a good performance in high-workload when all the cores are active. This case is similar to estimation power consumption of a processor as a unit regardless of active cores number.

As our models include the number of active cores and the average frequency, they accurately estimated the power in both areas of workload (i.e., low-workload and high-workload). Furthermore, although Model-2 statistically (i.e., regression R^2) is better than Model-1, the experiment demonstrated that Model-1 with constant C achieved better performance than the other models. Finally, slight percentage of error could be observed in our models due to the fact that we considered each logical core as physical core, but as we mentioned before in section II-B the power consumed by a logical core is less than the power consumed by a physical core.

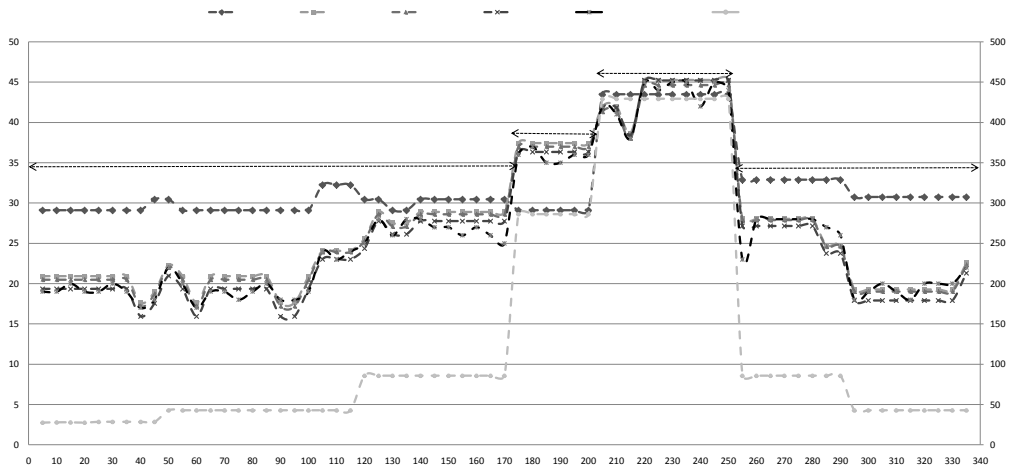


Figure 3: Trace of measured consumed CPU-Power and predicted CPU-Power for the four models.

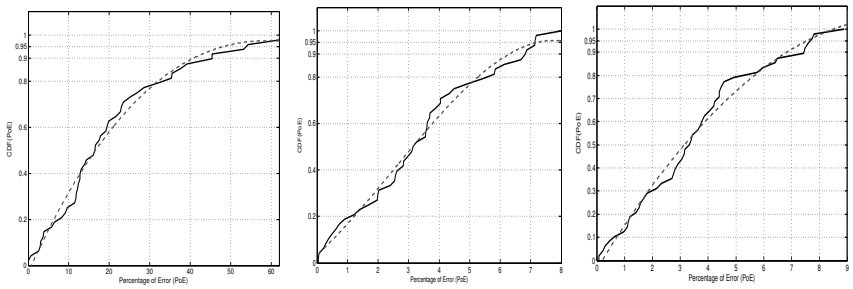


Figure 4: Prediction accuracy of the CPU-Power models.

Now, we discuss the prediction accuracy by computing the percentage of error using the following formula.

$$PoE = |(Estimiated - Measured)/Measured| * 100\%$$

Furthermore, we obtained the Empirical Cumulative Distribution Function of Percentage of Error CDF(PoE). The plot of CDF(PoE) for each model is depicted in fig.4. The x-axis represents the Percentage of Error (PoE), and y-axis shows the percentage of data points (i.e., predicted power values) that achieve error less than each value of x. For instance, 90% of the predicted values using Model-0 have less than 40%, and this error could increase to 50%. On the other hand, our models show that 90% of values were predicted with less than 9% error. Although R^2 value of Model-1 is less than R^2 value of Model-2, Model-1 showed the best results where 95% of the predicted values had error less than 7% error. Generally, the prediction accuracy of Model-1 empirically outperforms the prediction accuracy of Model-2.

Significantly, our proposed models achieve high prediction accuracy due to considering both number of active cores and the average running frequency. Additionally, the accurate readings of CPU-Power that was realized using CPU-Power measurement capability of our server assisted us to build these accurate models. To test our models' ability to predict those un-sampled combinations of frequency and number of cores, we conducted some experiments with different un-sampled combinations frequency 2.53GHz and number of cores. Table 2 presents the results of these experiments. The results proved that our models are capable to predict with high accuracy even un-sampled combinations of frequency and number of cores.

3.5.2 Memory-intensive applications

In this section, we evaluated performance of the models for applications that are considered memory-intensive using CG benchmark. Fig. 5 shows the estimated power versus the measured power. The diagonal line represents the perfect predication line which illustrates the deviation of the estimated values from the measured values. In other words, the predicted value is equal or close to the measured value if it is one of

Table 2: The predicted CPU-Power for un-sampled combination of frequency 2.53GHz and number of cores.

Cores	Measured	Model-0	Model-1	Model-2
3	42	47.09	42.27	41.23
4	45	47.09	45.57	44.53
8	58	47.09	58.77	57.73

the perfect predication line points or close to this line. From Fig. 5, the data points that represent our models are either on or close to perfect predication line. Furthermore, we computed the maximum prediction error of each Model. We found that Model-1 and Model-2 had less maximum prediction error compared to the other two models. However, with less than 6% maximum prediction error for Model-1 and Model-2, these two models are still accurate. The maximum prediction error of Model-0 was 14.4%.

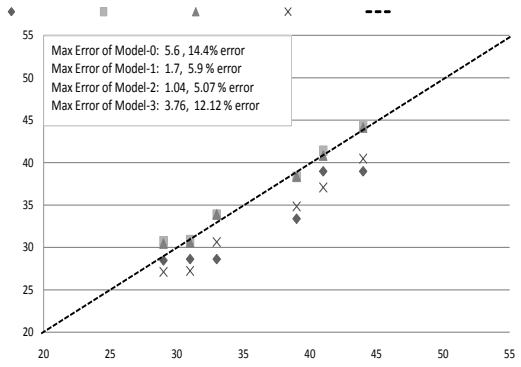


Figure 5: CPU-Power models fit for CG-Memory-Intensive: measured vs. estimated.

3.5.3 IO-intensive applications

As we presented the performance of our models for CPU-intensive and Memory-intensive applications in the previous sections, this section presents performance of the models for IO-intensive applications. We repeated the experiments procedure of the previous section using BT benchmark. Fig. 6 also shows the estimated power versus the measured power. We can see that the predicted values of Model-1 are on perfect predication line or very close to it. In contrast, Model-0 shows large deviation of the perfect line. Moreover, we found that Model-1 and Model-2 had less maximum prediction error compared to the other two models. Model-1 and Model-2 achieved less than 5% maximum prediction error. Finally, the maximum prediction error of Model-0 became worse with 22.07% error.

4 Conclusions and Next Steps

we developed models to estimate the power consumption of multicore processors. Our work is distinguished from previous works in combining number of active cores with P-state of multicore processor. We validated our proposed models using statistical analysis and experimental approach using varied workloads. The results of the experiment showed that our model achieved high accuracy of CPU-Power estimation. As a future work, we will apply our proposed models to

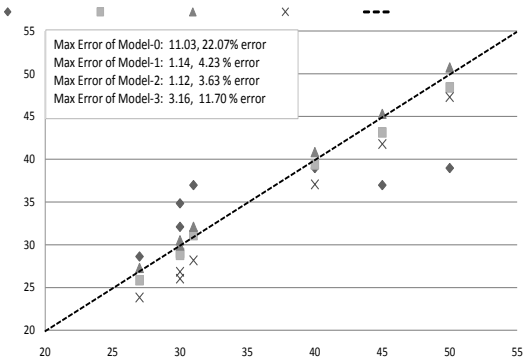


Figure 6: CPU-Power models fit for BT-IO-Intensive: measured vs. estimated.

dynamic power-aware configuration for a virtual machine in terms of number of cores and frequency. This enables new adaptive power management solution for virtualized servers.

References

[1] R. Urgaonkar, U.C. Kozat, K. Igarashi, and M.J. Neely. Dynamic resource allocation and power management in virtualized data centers. *2010 IEEE Network Operations and Management Symposium (NOMS), IEEE*, 2010, pp. 479-486.

[2] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal power allocation in server farms. *In Proceedings of SIGMETRICS*, June 2009.

[3] S. Chatterjee and A. S. Hadi, Regression analysis by example. John Wiley and Sons, 2006.

[4] X. Fan, W.-D. Weber, and L.A. Barroso. Power provisioning for a warehouse-sized computer. I*n Proceedings of the 34th annual international symposium on Computer architecture*, San Diego, California, USA: ACM, 2007, pp. 13-23.

[5] R. V. der Wijngaart. NAS Parallel Benchmarks v. 2.4. NAS Technical Report NAS-02-007, October 2002.

[6] J. Subhlok, S. Venkataramaiah, and A. Singh. Characterizing NAS Benchmark Performance on Shared Heterogeneous Networks. *International Parallel and Distributed Processing Symposium: IPDPS 2002 Workshops*, vol. 2, 2002, pp. 0086.

[7] K. H. Kim, A. Beloglazov, and R. Buyya. Power-aware provisioning of virtual machines for real-time Cloud services. Concurrency and Computation: Practice and Experience. John Wiley and Sons, 2011.

VMs Core-allocation scheduling Policy for Energy and Performance Management

Christoph Meinel, Ibrahim Takouna, and Wesam Dawoud
Hasso-Plattner-Institut (HPI)
University of Potsdam
Potsdam, Germany
{christoph.meinel, ibrahim.takouna, wesam.dawoud}@hpi.uni-potsdam.de

Abstract

In this phase of the project, we investigate the sensitivity of a VM performance running scientific multithreading applications to changes in clock frequency and VM performance dependency on Domain-0 for IO-intensive applications. Then, using sensitivity analysis to schedule VM to suitable core with suitable frequency settings. Currently, our work is built on a static heterogenous system, next we are going to investigate building a dynamic heterogenous system to realize power consumption proportional to a service requirements. However, our test environment showed that we can gain power savings up to 17%.

1 Project Idea

Merging between multi-core processors and virtualization technologies has prompted us to investigate the possibility of achieving power saving for such combination for scientific multithreading applications. In this project, we investigated the advantages of virtualizing heterogeneous multicore systems where they could provide better performance per watt compared to homogeneous processors [1,2,3]. A single processor will contain hundreds of cores that vary in some micro-architecture features such as clock frequency, cache size, power consumption, and others [4], but these cores exploit the same instruction-set architecture. A single chip might have several complex cores and many simple cores. The simple cores are characterized as low-speed clock frequency, cache size, and low power consumption while fast cores are equipped with high-performance features such as high-speed clock frequency, cache size, and high power consumption. Consequently, their potential to achieve different levels of performance that meet applications heterogeneity has prompted researchers in the operating systems domain to implement heterogeneous aware schedulers [5,6,7].
With heterogeneity of applications' characteristics, a Hypervisors' scheduler is efficient if it assigns a virtual CPU (vCPU) to run on the appropriate cores based on the application characteristics in terms of CPU-intensive, Memory-intensive, or IO-intensive. Further, it should have knowledge of the physical processors' architecture and their characteristics such as cores' clock frequency. By this knowledge, VMs with CPU-intensive applications should be assigned to complex fast cores to be executed faster. Generally, scientific applications are CPU-intensive, multithreaded, and fewer CPU stalls due to infrequent memory accesses or I/O operations. On the other hand, I/O-intensive could be assigned to simple slow cores without losing significant performance and achieving the power savings. However, Hypervisors' scheduling-policy is based on the round-robin algorithm to ensure fairness among VMs. Emerging heterogeneous system and virtualization bring more power savings and better resources utilization. This combination needs a new scheduler, which schedules each VM to an appropriate core based on its characteristics.
we used NAS Parallel Benchmarks [8] as CPU-intensive application and netperf benchmark [9] as I/O-intensive application. We denoted performance sensitivity to CPU clock frequency as "performance-frequency sensitivity" and performance dependency on Domain-0 as "performance-Domain-0 dependency". Our scheduling-policy based on these two categories: "performance-frequency sensitivity" and "performance-Domain-0 dependency" to assign a vCPU to the appropriate core. Consequently, the results showed good performance improvements for VMs with CPU-intensive applications and for VMs with IO-intensive applications as well. Finally, our heterogeneous experimental environment achieves promising power savings reach to 17% which theoretically could reach to 45% . The power savings are gained from this architecture, which runs on two cores with high frequency and other two cores with low frequency.

2 Used SOC Lab resources

The evaluation tests were performed on Fujitsu PRIMERGY RX300 S5 server. It has a processor of Intel(R) Xeon(R) CPU E5540 with 4-cores and the frequency range is 2.53GHz to 1.59GHz. the server is equipped with 12GB physical memory. Additionally, We used ServerView Remote Management to monitor power consumption for CPU.

3 Findings

In this section, we analyzed sensitivity of VMs' performance to changes in CPU clock frequency for VMs that run CPU-intensive and IO-intensive applications. Then, we illustrated dependency of VMs' on Domian-0 for VMs with IO-intensive applications.

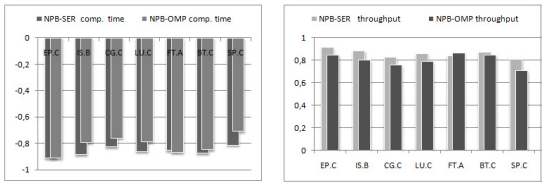
3.1 VMs with NBP Analysis

To analyze VMs performance-frequency sensitivity, we used NBP-SER and NPB-OMP benchmarks as CPU-intensive programs. In this experiment, we pinned vCPUs of Domain-0 to cores (0,1) and vCPUs of VMs were pinned to the another two cores (2,3) to avoid Domain-0's influence on the VMs; in other words, to prevent Domain-0 from being queued with the VMs in the same queue. First, the experiment was run while the cores (2,3) were set to run with high frequency F_F=2.53GHz as fast cores. Then, it was run again after changing frequency settings of the cores (2,3) to low frequency F_S=1.59GHz as slow cores. Finally, we used the price elasticity of demand economics formula to determine program's completion time and throughput sensitivity of clock frequency. We considered T the completion time and Th the throughput as the demand, and F clock frequency as the price. $E_{T,F}$ is the completion time sensitivity of clock frequency, and $E_{Th,F}$ is throughput sensitivity of clock frequency.

$$E_{T,F} = \frac{T_F - T_S}{F_F - F_S} * \frac{F_F + F_S}{T_F + T_S} \quad (1)$$

$$E_{Th,F} = \frac{Th_F - Th_S}{F_F - F_S} * \frac{F_F + F_S}{Th_F + Th_S} \quad (2)$$

Due to the inverse relationship between CPU frequency and completion time, $E_{T,F}$ values are negative, so completion time increases as CPU frequency decreases and vice versa. On the other hand, $E_{Th,F}$ values are positive because of the direct relationship between CPU frequency and throughput. Program speedup depends on program characteristics, so it does not have a liner relationship with CPU frequency. However, CPU-intensive programs might have a semi-liner relation with frequency because of either infrequent memory accesses or I/O operations.



(a) (b)

Figure 1: Performance-frequency sensitivity for NPB-OMP and NPB-SER versions run on a VM with two vCPUs. (a)sensitivity of completion time to frequency, and (b)sensitivity of throughput to frequency.

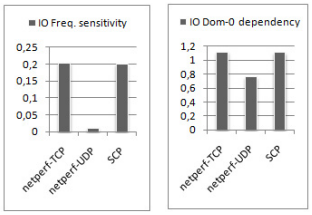
Figure 1 shows NPB-OMP and NPB-SER benchmarks performance-frequency sensitivity (i.e., completion time and throughput). NPB benchmark each program has different memory access behavior and various inter-process communication patterns. These characteristics determine sensitivity of a program to frequency changes. For example, the completion time of EP-OMP and EP-SER programs had the same sensitivity and they gained the highest sensitivity. The similarity between these two programs is that EP-OMP a multithreaded program but has negligible inter-process communication and EP-SER is a single thread program without inter-process communication. Further, EP-OMP is seldom memory access compared with CG-OMP and LU-OMP. Generally, NPB-SER programs sensitivity to frequency changes was higher than NBP-OMP due to the sequential execution of instructions in NPB-SER and inter-process communication patterns or IO operations in some of NBP-OMP programs such as CG and BT respectively. On the other hand, NBP-OMP programs with intensive inter-process communication were less sensitive to frequency such as CG-OMP and LU-OMP. FT, a mixed type program, almost had the same sensitivity in NPB-SER and NPB-OMP. Unlike LU-OMP, BT-OMP includes a number of I/O operations that do not need synchronization among its threads.

3.2 VMs with I/O Analysis

We analyzed sensitivity of VMs performance with I/O-intensive to CPU frequency. Then, as I/O operations depend on Domain-0, we tested VMs performance-Domain-0 dependency.

3.2.1 CPU Frequency Sensitivity

In this experiment, we ran netperf with TCP-STREAM and UDP-STRAEM options to test I/O performance-frequency sensitivity using formula 2. The setting of this experiment was the same setting when we tested VM with NBP sensitivity. As shown in figure 2-(a), TCP test is more sensitive to core frequency than UDP due to the nature of TCP-packet; UDP does neither message fragmentation nor reassembly. Further, the



(a) (b)

Figure 2: Performance-frequency sensitivity and Domain-0 dependency for NPB-OMP and NPB-SER versions run on a VM with two vCPUs. (a) performance-frequency sensitivity, and (b) performance-Domain-0 dependency.

aggregate costs of non-data touching overheads consume majority of the total software processing time. The non-data touching overheads come from as network buffer manipulation, protocol-specific processing, operating system functions, data structure manipulations (other than network buffers), and error checking[16]. To validate our test, we used SCP application TCP-based to transfer a 500 MB file between two VMs and we found the same results obtained using netperf-TCP.

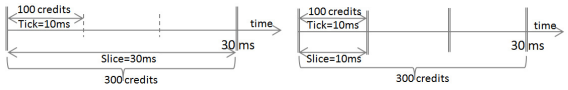
3.2.2 VMs with I/O Domain-0 Dependency

In this experiment, we ran netperf benchmark with TCP-STREAM and UDP-STRAEM options to test I/O performance-Domain-0 dependency. For this end, we reversed the scenario of VM performance-frequency sensitivity, so the cores (2,3) settings were not changed and were set to high frequency F_F=2.53GHz where VMs were pinned in cores (2,3). On the other hand, The cores (0,1) were set to high frequency F_F=2.53 GHz where Domain-0 was pinned. Then, we ran it again while the frequency of cores (0,1) is low F_S=1.59GHz. Finally, we computed the performance-Domain-0 dependency using formula (2). The result of this experiment is shown in figure 2-(b). It illustrates that both netperf-TCP and netperf-UDP depend on Domain-0 for commutation between to VMs, but netperf-TCP depends on Domain-0 more than netperf-UDP.
The conclusion is that applications based on TCP protocol are frequency sensitive and they are Domain-0 dependant as depicted in figure 2-(a) and figure 2-(b) respectively.

4 PERFORMANCE EVALUATIONS

In this section, we evaluated our improved scheduling-policy with the following rules:

- The weight of VM is proportional to the number of vCPUs.
- CPU-intensive vCPU should not being queued with I/O-intensive vCPU. Further CPU-intensive vCPU should be placed in the fast pCPU's queue and I/O-intensive vCPU in the slow pCPU's queue.
- A virtual machine with CPU-intensive application and a single vCPU should be placed in fast pCPU's queue to accelerate the sequential execution.
- The time-slice for the fast pCPU's queue is 30ms and time-slice for slow cores is 10ms as show in figure 3. We chose the value 10ms for the short slice as one tick to avoid high context switching and to keep consistent credit accounting.
- The settings of cores in the experiments are the fast cores (0,1) with F_F=2.53GHz and the slow cores (2,3) with F_S=1.59GHz.



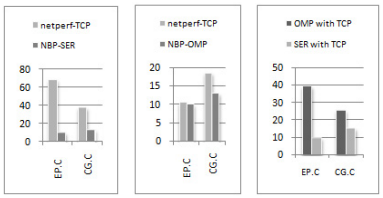
(a) (b)

Figure 3: Scheduling time-slice modifications. (a) time-slice = 30ms for fast cores, and (b) time-slice = 10ms for slow cores. The accounting period of vCPU is 30ms for both fast and slow cores.

4.0.3 I/O and CPU-intensive Isolation

In this experiment, we created three VMs one with two vCPUs and the other two each has one vCPUs. We ran netperf on the two VMs with one vCPU for testing TCP and UDP bandwidth channels between them. The VM with two vCPUs used to run NPB-SER, then NBP-OMP. We ran the three VMs with our new scheduling-policy. First, we used EP and CG programs in NPB-SER with netperf, then EP and CG of NPB-OMP were used. We pinned the VMs with I/O to the slow cores (2,3) and the VM with CPU-intensive was pinned to the fast cores (0,1). Performance improvement for both I/O and CPU-intensive VMs compared to the default scheduler is illustrated in figure 4. Figure 4-(c) shows that the performance gain of CG.C is better than EP.C. Indeed, EP.C has negligible inter-process communication compared to CG.C which has also memory accesses. On the other hand, netperf-TCP throughput when co-hosted with VM that ran NBP-SER is better than when co-hosted with VM that ran NBP-OMP. As seen in figure 3-(b), netperf depends on Domain-0 and NPB-SER is a single thread test that gave Domain-0 chance to be scheduled in fast cores and improve I/O operations for netperf-TCP. The aggregate average gain is depicted in figure 4-(c).

Obviously, isolating CPU-intensive vCPUs from IO-intensive vCPUs was the main reason for performance improvement. Using isolation eliminated the sources of delay that effect on CPU-intensive vCPUs performance.



(a) (b) (c)

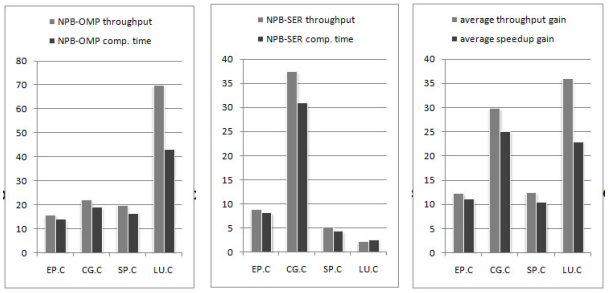
Figure 4: I/O and CPU-intensive Isolation Performance improvements; netperf-TCP run on a VM with one vCPU,and NPB-OMP run on a VM with two vCPUs. (a) Throughput gain for NPB-OMP and netperf-TCP benchmark, (b) throughput gain for NPB-OMP and netperf-TCP benchmark, and (c) the average improvement of the overall system.

4.1 VMs with sensitive Inter-process Comm.

In this experiment, we tested the performance gain for inter-process communication intensive such as CG and LU of NPB-OMP version. The performance of NPB-OMP benchmark in VM is near to the performance in physical server as long as the vCPUs are less than pCPUs, and LU-OMP is the most sensitive program to communication delay [10]. For testing inter-process communication intensive program performance improvement, we created one VM with one vCPU and another VM with four vCPU. Nevertheless, we had five vCPUs in addition to four vCPUs for Domain-0. The performance gain is illustrated in figure 5 where figure 5-(a) shows Throughput gain and completion time speedup for NPB-OMP while figure 5-(b)illustrates Throughput gain and completion time speedup for NPB-SER. Figure 5-(c) shows the average aggregated performance gain for NPB programs with two versions. Nevertheless, LU-OMP gained about 70% performance improvement. This improvement due to changing the time-slice of the slow pCPUs' to 10ms which increases scheduling frequency. Increasing scheduling frequency gave chance for inter-process communication and synchronization. Further, decreasing time-slice decreases holding time when vCPU status "busy blocking" holds pCPU [11]. A lot of "busy blocking" wastes pCPU cycles and degrades the overall system performance.

5 Next Steps

The next steps of our project will be as follows:



(a) (b) (c)

Figure 5: CPU-intensive with inter-process communication intensive performance improvements; NPB-OMP run on a VM with four vCPUs and NPB-SER run on a VM with one vCPU. (a) Throughput gain and completion time speedup for NPB-OMP, (b) throughput gain and completion time for NPB-SER, and (c) the average improvement of the overall system.

- Analyzing energy characteristics of scientific mutlithreading applications executed on VM with multi-Virtual-CPU to provide a dynamic mechanism for saving energy while satisfying performance requirements. As we studied the NPB benchmarks performance sensitivity of changing in CPU frequency, Our method is to determine number of virtual-CPUs for a virtual machine and cores frequency settings in order to minimize energy consumption.
- We would like to use SOC LAB resource that has 64-cores which enable many configuring combinations of Frequency, Voltage, and number of cores where the used machine only has 4 cores. Using 64-cores machine also could be useful to apply our idea of cores clustering based on CPU clock frequency and other features.

References

[1] K. Asanovic, R. Bodik, B. Catanzaro, J. Gebis, and P. Husbands, "The Landscape of Parallel Computing Research: A View From Berkeley," UC Berkeley Technical Report UCB/EECS-2006-183, 2006.

[2] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguade, "Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors," *IEEE Computer Architecture Letters 5(1):4*, 2006.

[3] R. Kumar, K. I. Farkas, and N. Jouppi et al, "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," *In Proc. of MICRO 36*, 2003.

[4] S. Borkar, "Thousand Core Chips-A Technology Perspective," *in Proc. of the DAC*, 2007.

[5] R. Kumar, Dean M. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas, "Single-ISA Heterogeneous Multicore Architectures for Multithreaded Workload Performance," *in Proc. of the 31st Annual International Symposium on Computer Architecture*, 2004.

[6] R. Kumar, D. M. Tullsen, and P. Ranganathan et al, "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," *in Proc. of ISCA*, 2004.

[7] M. Becchi and P. Crowley, "Dynamic Thread Assignment on Heterogeneous Multiprocessor Architectures," *in Proc. of the Conference on Computing Frontiers*, 2006.

[8] R. V. der Wijngaart, "NAS Parallel Benchmarks v. 2.4", NAS Technical Report NAS-02-007, October 2002.

[9] R Jones, "NetPerf:a Network performance benchmark," http://www.netperf.org.

[10] C. Xu, Y. Bai, and C. Luo, "Performance Evaluation of Parallel Programming in Virtual Machine Environment," *In Proc. of Sixth IFIP International Conference on Network and Parallel Computing*, pp. 140-147, 2009.

[11] H. Chen, H. Jin, K. Hu, and J. Huang, "Dynamic Switching-Frequency Scaling: Scheduling pinned Domains in Xen VMM," *in Proc. of 39th International Conference on Parallel Processing*,pp. 287-296, 2010.

[12] D. Shelepov and A. Fedorova, "Scheduling on Heterogeneous Multicore Processors Using Architectural Signatures," *in Proc. of the Workshop on the Interaction between Operating Systems and Computer Architecture*, in conjunction with the 35th International Symposium on Computer Architecture (Beijing, China, June 21-25, 2008). WIOSCA '08.

Paralleli ation of Elementary Flux Mode Enumeration for Large-scale Metabolic Networks

Fahad Khalid
Hasso-Plattner-Institut,
University of Potsdam
14482 Potsdam, Germany
fahad.khalid@hpi.uni-potsdam.de

Abstract

Elementary Flux Modes (EFM) is a concept from Systems Biology, describing a set of minimal pathways in the metabolic network of an organism. Under steady-state conditions, the problem of finding EFMs in a metabolic network is mathematically equivalent to the enumeration of extreme rays of a polyhedral cone. This problem is computationally challenging and currently mostly solved with sequential algorithms, which are inappropriate for the analysis of large-scale metabolic networks.

In this project we aim to implement a parallel solution to the problem that is targeted towards hybrid many-core architectures. So far, we have utilized the hardware available in the FutureSOC Lab to understand the memory consumption behavior of the algorithm on shared-memory architectures. We have identified memory consumption as a priority issue, and are currently working towards designing a better solution.

1 Introduction

This project is aimed at solving a computational problem from the field of Systems Biology. It is a collaborative effort between the Max Planck Institute of Molecular Plant Physiology and the Hasso-Plattner-Insitute at the Universtiy of Potsdam.

The following sections provide a brief introduction to the problem, the computational challenge that it poses and also findings from the experiments conducted so far.

1.1 Metabolism

The metabolism is a collection of chemical reactions responsible for supporting life in a living organism. They include reactions that breakdown organic matter in order to harvest energy; as well as those that utilize energy to construct the required chemicals e.g. proteins and nucleic acids. When considered at the scale of the whole organism, these chemical reactions form networks that consist of a very large number of chemicals interacting with each other.

1.2 The study of Metabolic Networks

Analysis of metabolic networks serves an important tool in Systems Biology research. Following are two examples of how a thorough understanding of the structure and function of metabolic networks can be useful:

Diseases such as cancer cause significant alterations in the intracellular metabolism [1, 2]. This is due to the fact that metabolic activities in tumor cells are different from healthy cells. Therefore, understanding the metabolic pathway flux of tumor cells can help identify drug targets that may help develop improved therapies for cancer patients.

There exist in nature several organisms that on an intracellular level produce chemicals useful for human consumption. However, the rate of production of these chemicals is optimized by evolution for objectives such as growth and reproduction of the organism [3]. In order to optimize the metabolism of these chemicals for human needs, the rate of production has to be altered. In metabolic engineering, a promising method to achieve this is the mathematical prediction of metabolic fluxes.

1.3 Elementary Flux Modes

Elementary Flux Modes (EFMs) describe a set of minimal pathways in the metabolic network of an organism. A system can have multiple steady states, and each of these unique states can be expressed as a nonnegative linear combination of elementary flux modes.

1.4 Enumeration of EFMs

This project deals with the problem of enumerating all EFMs of a given metabolic network, for a given steady state. The following major steps are involved in the enumeration process:

1. The metabolic network under consideration is represented as a node weighted directed hypergraph. The chemical reactions in-

volved in the network are represented by the edges of the hypergraph, while the vertices represent the chemical compounds involved in those reactions. These chemical compounds are referred to as *metabolites*.

2. The incidence matrix of this hypergraph is created. This matrix shows the relationship between the metabolites and reactions involved in the network. This matrix is known as the *Stoichiometric matrix*. Throughout the rest of this document, we shall represent the stoichiometric matrix as S.
3. Under steady state conditions, the following relationship holds:

$$Sv = 0$$

where S is the stoichiometric matrix, and is flux vector.

4. The EFMs that we are looking for, reside in the nullspace matrix corresponding to this system of linear equations.
5. An algorithm is used to get the nullspace matrix, and then systematically search for solution vectors that qualify as EFMs. The algorithm used in this project is the Nullspace algorithm [4].

1.5 The Computational Challenge

The core computational challenge presented by the problem of EFM enumeration is that of combinatorial explosion. The number of EFMs grows very rapidly with the size of the network, and therefore puts very high demands on both computation and memory.

A parallelization based approach to the solution enables us to exploit the inherently parallel architecture of modern processors, thereby reducing the time that it takes to enumerate EFMs. However, the attempts so far have only marginally increased the capacity to compute EFMs, and it is still not possible to enumerate EFMs for genome scale networks.

2 Our Approach

The traditional algorithmic model in HPC is aiming at parallelized shared-nothing architectures with deep memory hierarchies, as with the IBM BlueGene architecture and MPI as programming model. This is contrary to one of the most recent trends, where CPU and GPU hardware is mixed for speeding up specific steps in the algorithmic processing. Also, the current trend in multi-core and many-core architectures provides the opportunity to exploit shared-memory architectures with multiple cores.

Considering the above mentioned trends, our approach is to tailor the algorithm and implementation in such a way, that not only can we take advantage of distributed memory cluster like environments, but also hybrid shared-memory parallel architectures within a single box.

2.1 Current Status

The progress so far is in line with the proposed project plan. A state-of-the-art algorithm has been selected for parallelization [5]. While adapting the algorithm to shared-memory architectures, a major memory consumption bottleneck was identified. Work is currently in progress on optimizing memory consumption, as well as further adapting the algorithm for efficient execution on shared-memory architectures.

The challenge now is to optimize memory consumption for shared-memory architectures. This requires efforts both on the algorithmic and implementation levels. The use of FutureSOC machines is vital for the verification and validation of the required improvements. This is due to the fact that the memory bottleneck is only apparent for a large number of processes running in parallel, which requires the use of high-end hardware available in the FutureSOC lab.

3 Results

The results are presented in tabular form in the attached Appendix.

The experiments were conducted using two of the machines available in the FutureSOC Lab. Hyperthreading was enabled on both machines, which means that total number of hardware threads available on each machine is 128.

Table 1 presents the execution times of networks of different sizes, executed using different number of processes. The most important message here is that memory consumption increases almost linearly with the number of processes. The consequences are apparent in row 7. The maximum number of processors that can compute this Yeast network is 16. If we increase the number of processes to 32, the amount of memory consumption increases as well, and the program eventually runs out of memory. The Yeast network in the last row could not be computed at all. It consumes all the memory even if executed using just 2 processes. Therefore, parallel execution of the last network is not possible on this machine due to the inept memory consumption behavior.

Table 2 presents the results of execution on a similar machine, but this time with 2 terabytes of memory (which is considerably larger than was available on the first machine). As can be seen here, the Yeast network that could only be computed with a maximum of 16 processes on the first machine, can be computed with even 128 processes. This results in a significant reduction in execution time. Fort the last network, however, we again have an upper bound on the number of processes. In this case, with even 2

terabytes of memory, not more than 32 processes can be used in parallel.

3.1 Conclusions

As can be inferred from the results presented above, parallel computing is indeed as suitable approach to solve this problem in two ways:

1. Minimizing execution time for currently computable network sizes.
2. Extending the computable network size to more than that computable with serial processing alone.

However, considerable work is required in order to make the computation feasible for genome-scale networks. The major bottleneck at the moment is that of memory consumption, which is what we would like to improve upon in the near future.

References

[1] R. J. DeBerardinis, J. J. Lum, G. Hatzivassiliou, and C. B. Thompson, "The Biology of Cancer: Metabolic Reprogramming Fuels Cell Growth and Proliferation," *Cell Metabolism,* vol. 7, pp. 11-20, 2008.

[2] L. G. Boros, N. J. Serkova, M. S. Cascante, and W.-N. P. Lee, "Use of metabolic pathway flux information in targeted cancer drug design," *Drug Discovery Today: Therapeutic Strategies,* vol. 1, pp. 435-443, 2004.

[3] H. U. Kim, T. Y. Kim, and S. Y. Lee, "Metabolic flux analysis and metabolic engineering of microorganisms," *Molecular BioSystems,* vol. 4, pp. 113-120, 2008.

[4] C. Wagner, "Nullspace Approach to Determine the Elementary Modes of Chemical Reaction Systems," *The Journal of Physical Chemistry B,* vol. 108, pp. 2425-2431, 2004.

[5] D. Jevremović, C. T. Trinh, F. Srienc, C. P. Sosa, and D. Boley, "Parallelization of Nullspace Algorithm for the computation of metabolic pathways," *Parallel Computing,* vol. 37, pp. 261-278, 2011.

4 Appendix

Hewlett Packard DL980 G7 – 2, 8 x Xeon X6550, 128 GB RAM					
#	Network Size (Original)	Network Size (Compressed)	# of Processes	Time taken	# of EFMs
1	E. coli 47 × 59(21)	26 × 38(13)	128	1s	44354
			16	0.6s	
			1	3s	
2	E. coli 41 × 61(19)	26 × 40(12)	128	0.9s	38002
			16	0.62s	
			1	2.5s	
3	E. coli 49 × 64(19)	26 × 41(12)	128	< 2s	92594
4	E. coli 50 × 66(19)	27 × 43(13)	128	3.1s	188729
5	E. coli 50 × 66(28)	29 × 45(19)	128	41s	1224785
			64	53s	
			32	57s	
6	S. cerevisiae 62 × 78(31)	35 × 54(17)	128	42s	1515314
			64	48s	
			32	63s	
7	S. cerevisiae 62 × 80(31)	38 × 57(19)	16	1 hour 45 minutes	13322463
			32 and >	could not be computed	
8	S. cerevisiae 63 × 83(34)	40 × 61(23)	-	-	-

Table 1 Execution results with 128 Gigabytes of RAM

Hewlett Packard DL980 G7 – 1, 8 x Xeon X7560, 2048 GB RAM					
#	Network Size (original)	Network Size (Compressed)	# of Processes	Time taken	# of EFMs
1	S. cerevisiae 62 × 80(31)	38 × 57(19)	128	~ 32 minutes	13322463
2	S. cerevisiae 63 × 83(34)	40 × 61(23)	32	~ 5.2 hours	68874836
			64 and >	could not be computed	

Table 2 Execution results with 2 Terabytes of RAM

Early Anomaly Detection in SAP Business ByDesign Report March 2012

Felix Salfner
SAP Innovation Center
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam

Peter Tröger, Eyk Kny
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam

Abstract

The management of cloud computing infrastructures on operator side is a true challenge. An ever-growing number of servers, the heterogeneity of software, necessary elastic load handling, energy consumption and other non-functional aspects have to be taken into account – continuously and in an adaptive fashion.
In this report we discuss one specific idea for smart cloud operation. Anomaly detection triggers preventive maintenance activities when some part of the system is about to enter an erroneous state. Examples for such activities are administrator alarming and automated load migration. We combine this approach with the idea of semi-automated root cause analysis to reduce time-to-repair and improved availability. Prototypical experiments for this approach were conducted with SAP's enterprise cloud infrastructure called Business ByDesign. All measurements and prototypical implementations were performed in the HPI FutureSOC lab.

1 Introduction

Cloud computing is currently one of the predominant trends in computer science and IT industry. Its potential impact on IT infrastructures and software platforms cannot be underestimated: The cloud paradigm will change the way how IT companies make business as well as how end-users (private and corporate) perceive software and IT. It moves the burden of IT infrastructure management and operation away from users, and shifts it to specialized providers that can guarantee fast, reliable, and secure operation of the cloud infrastructure.
Satisfying the users' expectations turns the management of cloud computing infrastructures into a true challenge. An ever-growing number of servers, the heterogeneity of software, multiple interacting mechanisms to elastically react on changing load, the consideration of energy consumption and other non-functional aspects have to be taken into account. Operating such systems requires intelligent management tools to deliver the guaranteed service-level to the user in a profitable way.

One of the key features of system monitoring at the scale of a cloud computing center is to aggregate low-level monitoring events in order to present only critical events to the operation personnel. To achieve this goal, current monitoring systems rely on temporal as well as spatial correlation mechanisms and infrastructure-centric rule-based aggregation. These techniques usually have a snapshot-like view on the system and ignore problems that evolve in the system over a long period of time. In our project, we focus on such evolving phenomena and explore a new monitoring technique that is based on computing correlations between monitoring signals in order to identify the spreading of problems within the system.
This report summarizes the first project phase and documents our project results. In summary we have:

- created a general monitoring and computation infrastructure for anomaly detection,
- implemented the necessary signal correlation algorithm using SAP's in-memory database HANA,
- identified potential anomaly signal data sources in SAP Business ByDesign, and have
- collected data from other use cases, i.e., from the Python BuildBot infrastructure and the TACC Ranger High-Performance Computing System.

2 Approach

In 2010, Oliner et al. [3] have presented an algorithm to infer shared influence in complex production systems. The algorithm is based on anomaly signals, which are real values between zero and one indicating how much a measurement signal deviates from "normal behavior". By computing the correlation between any pair of anomaly signals, the "spreading" of anomalies across the system can be quantified. More precisely, if there is a high correlation between the anomaly signals of two components, we can speculate that the abnormal states of both components are related.
The time lag at which the correlation reaches its maximum can be used to determine the temporal interde-

pendency between the two components, i.e., to identify, which component is the source (or initiator) and which one is the target of the problem (suffering from the anomaly of the source component). For further analysis, the interdependencies can be visualized in a so-called *structure of influence graph (SIG)*. A simple exemplary graph is depicted in Figure 1. The thickness of each edge represents the strength of influence above some threshold. If the edge is directed, there is a significant time lag between the anomaly signals at both components. More specifically, Component A shares some influence with Component B (no significant time lag between anomaly signals) and a strong influence with Component C (in this case the anomaly of Component A precedes the one of C). This can help to detect anomalies early and can help administrators to troubleshoot the problem more quickly.

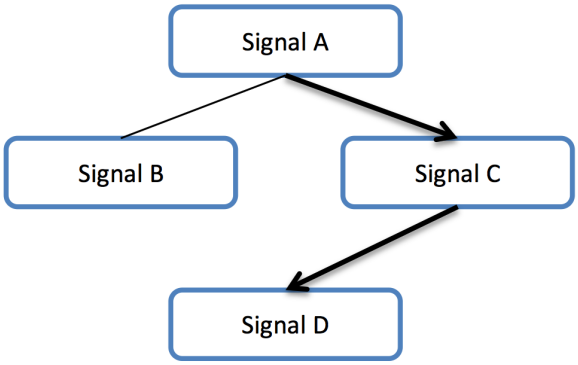


Figure 1. Influence graph example

In a follow-up paper [2], Oliner and Aiken have presented an improved online version of their approach, in which dependent signals are compressed using principal component analysis and special algorithms are used to compute correlations efficiently during runtime. The same authors also introduced a query language and graphical representation for SIGs [1], which will serve as a basis for representing SIGs in the management console.

3 Application to SAP Business ByDesign

SAP *Business ByDesign (ByD)* is a complex enterprise software that provides business cloud services in a scalable and multi-tentant enabled fashion. In our research, we use ByD as an example for a complex cloud operation infrastructure that can benefit from automated anomaly detection and handling. In order to apply the original concepts from Oliner et al. in such a business framework, we identified several mandatory extensions to the original approach:

- True complex cloud operation environments have heterogeneity as default. We must consider the heterogeneous multi-layer structure of modern systems as basic foundation of the targeted solutions. This includes differing virtualization solutions, operating systems, application servers and application monitoring interfaces. Specifically, we need to target multi-layer anomaly signal generation and normalization as foundational concepts.

- The computation of anomaly correlations and their translation to management actions must be realized in adherence to the overall system structure. Reporting, incident generation and computationally intensive analysis tasks must be performed as part of the overall system operation. For this reason, we decided to realize the correlation computation and storage in the In-Memory HANA database environment. A second reason for this choice is that the task of monitoring data analysis resembles the tasks that are typically performed in business data exploration.

- In order to achieve true proactive behavior, we need to apply prediction techniques for the anomaly signal correlation result. This would allow to early identify specific patterns in the anomaly spreading through the system. The In-Memory HANA database technology provides the necessary computational and data access performance to accomplish that.

Figure 2 shows the system design of our current prototype. Different ByD components are expected to feed anomaly signal generators with information about their current health condition. The signals of the multi-layer and multi-component generators are summarized to one signal and sent to the central storage and correlation engine. While Oliner et al. originally proposed a *principal component analysis* (PCA) for this, we focus on the collection and transmission of time-synchronized generator values. An according technical approach is described in the next section.

4 Anomaly signal collection

In order to collect a number of anomaly signals from heterogeneous system components, we designed an interoperable monitoring environment. All anomaly signals are transported to a central data storage facility, which is assumed to be performant enough for handling the incoming request load. This is a reasonable assumption for the ByD environment since the number of servers is significantly smaller than in the scenarios investigated by Oliner et al.
Our framework focusses on easy setup and implementation overhead for the signal generators, since the overall solution shall not add much development overhead to the product teams. We also needed to bridge the gap between non-synchronized monitoring data

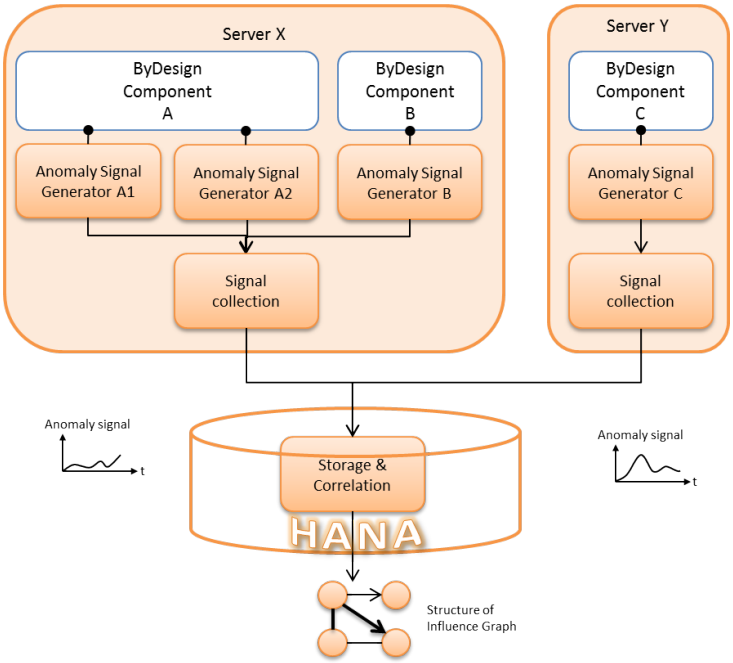


Figure 2. Computing structure of influence graphs using SAP's in-memory database HANA

fetching and synchronous data availability assumed by the correlation framework.

The monitoring code for signal generation must execute under truly diverse conditions. One example are operating system sensors, which are typically developed in native C language. On the other hand, high-level software sensors may be developed in Java or ABAP. Anomaly signal generators also encode very domain-specific knowledge. They will in many cases be programmed by domain experts. The monitoring framework should hence allow maximum flexibility in order to put a minimum burden on anomaly signal generator programmers.

Figure 3 shows our resulting infrastructure for a multi-server environment.

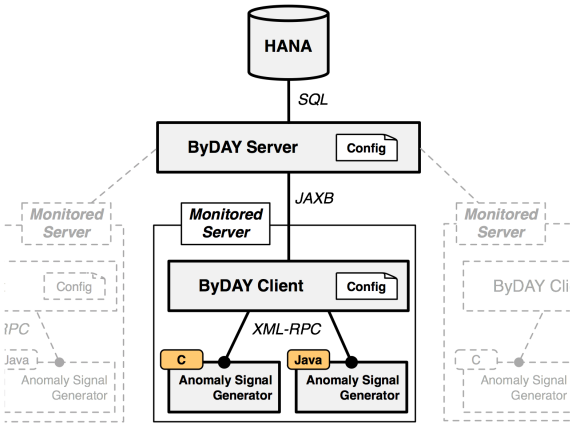


Figure 3. Monitoring infrastructure for anomaly signal collection.

Our typical client-server architecture consists of the *ByDAY server* and one *ByDAY client* per server. The data between both entities is exchanged via JAXB serialization of Java objects. The setup procedure performs time synchronization (or appropriate time shift determination) between the two parties, in order to get a consistent time mapping of sensor values.

The support for heterogeneous generator implementations is ensured by the usage of XML-RPC as protocol between the *ByDAY client* and the domain-specific generator instances. While the wire encoding of XML-RPC truly adds some overhead, it must be noted that this kind of communication only happens locally between the *ByDAY client* and the generator instances. All generators acting as XML-RPC server have to implement two methods:

```
<boolean> setup(<struct> config)
<double> pullSignal(<int> signalID,
                    <dateTime> timestamp)
```

The ByDAY client starts all anomaly signal generators (on the same machine) and calls their `setup()` procedure to configure generator-specific parameters. This enables to configure all anomaly signal generators of one machine to be configured using just one configuration file (the one of the ByDAY client).

Our approach to bridge the gap between asynchronously available (raw) monitoring signals and the need to collect anomaly signals for one system-wide synchronized timestamp is the following: Each anomaly signal generator acquires its raw monitoring data asynchronously. Synchronization is achieved by the ByDAY client asking for the anomaly signal value at some given timestamp in the past. It is generator's

duty to store enough history so that this request can be answered. If there is no value for the exact timestamp requested, it is the generator's duty to provide an approximate value for the requested timestamp, as the according approximation, rounding and logging activities are signal-specific. To request an anomaly signal, the ByD client - in coordination with the ByDAY server - will periodically call the `pullSignal()` method to ask for the anomaly value of the given signal at the given timestamp.

In order to limit the need for monitoring data storage, the ByDAY client sets a config parameter `maxTimeLag` along with the `setup()` call, which determines the maximum time delta into the past for which the ByDAY client can request a value. In order to enable an anomaly signal generator to serve multiple signals, `setup()` also requires a config parameter `signalID`.

After having collected the anomaly signals of all local anomaly signal generators, the ByD server sends back the data as timestamp-based tuple to the ByDAY server, which persists it for later processing in the HANA database.

5 HANA-based correlation engine

Investigating the root cause of a problem that has occurred in a cloud environment frequently requires to analyze monitoring data and to compare the system behavior of the current faulty case to system behavior at other fault-free times. In order to do this, an efficient analysis framework has to enable the operator to quickly move forth and back in the monitoring data, to drill down at specific points in time or to get an overview on higher levels of granularity. These requirements make root cause analysis similar to business data exploration. In-memory database technology is a key enabling technology to perform on-the-fly data analysis — and we believe that it has the potential to also boost efficiency for root cause analysis. For this reason, we implemented the correlation algorithm with on-the-fly data aggregation on the database level.

Our approach builds on the tight integration of the SAP HANA in-memory database and the statistical computing framework R. Figure 4 shows the setup. The implementation of the *ByDAY server* was possible through the usage of the HPI FutureSOC Lab HANA installation.

6 Collecting Test and Real-world Data

In order to test the described prototypical implementation, we started to collect historical execution and failure logs from different clue infrastructures.

6.1 ByDesign execution traces

SAP Business ByDesign is a complex cloud enterprise software platform that is built on top of a cloud infrastructure. The system is operated and maintained by multiple groups and we first focused on getting an overview of the monitoring data that is available. The various data sources reach from infrastructure-level monitors, e.g., from network switches, up to incidents reported by external users of Business ByDesign. We also have talked to experts that integrate monitoring data from various sources and architectural levels to perform impact analysis and event correlation. We have identified data sources that can be used once we have finished and tested our monitoring data collection framework.

6.2 Buildbot execution traces

Buildbot is a continuous integration system designed to automate the build/test cycle. By automatically rebuilding and testing the tree each time something has changed, build problems are pinpointed quickly, before other developers are inconvenienced by the failure. The Python interpreter development is using a Buildbot infrastructure[1] for continuous integration purposes. Based on personal contact with Dr. Martin v. Löwis, we were able to gather a set of log files for the Python Buildbot infrastructure. We are currently developing an anomaly generator implementation that can replay the logs and generate anomaly signals.

6.3 TACC Ranger execution traces

The Texas Advanced Computing Center (TACC) operates their largest HPC supercomputer, Ranger, with different software and hardware facilities at a large scale. Ranger can perform 579.4 trillion operations per second (or teraflops), and is nearly 30,000 faster than today's desktop computers. Based on personal contact with Edward Chuah, we were able to gain failure logs from the Ranger operation over a time period of 24 months. We are currently preparing a log replay anomaly signal generator similar to the Buildbot case in order to test our framework also with this data set.

7 Initial Results

We have performed experiments using artificial data generators to test our monitoring and computation infrastructure and to see how the approach scales. Figure 5 plots the computation time needed to compute the correlation between all signals as a function of the number of signals. The family of curves in the plot show computation time if the correlation is computed for a maximum time lag of 50, 240, and 1500 samples,

[1] http://bit.ly/zrjRec

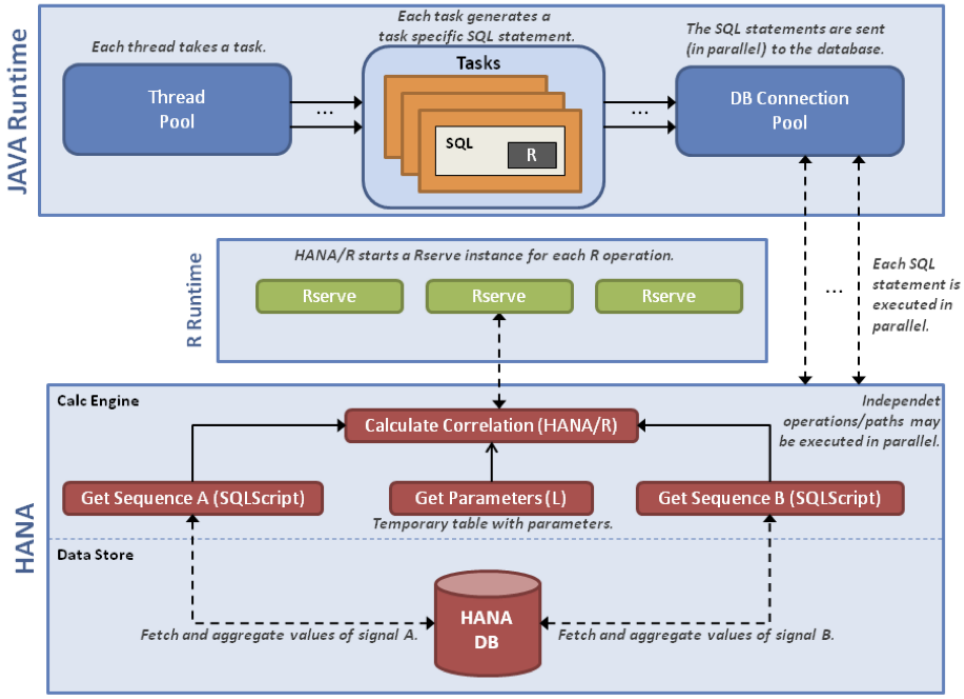


Figure 4. Overview of the HANA-based correlation engine.

i.e., for each pair of signals, the correlation is computed with an offset of $[0 \ldots 50]$ samples.

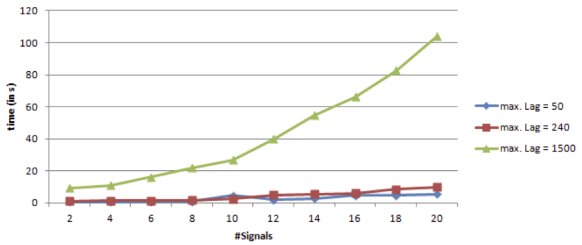


Figure 5. Computation time for a varying number of signals with 50000 sample points each.

8 Conclusions and Next Steps

Operating a cloud computing infrastructure at minimum cost while at the same time offering service at a guaranteed level is a challenge. This project attempts to adapt a new correlation-based monitoring data analysis method to the cloud context. One major new aspect is to include in-memory databases in the picture to facilitate on-the-fly data analysis and exploration. In this first phase of the project we have built the infrastructure and have collected several datasets that we are going to analyze in the next phase. Further topics will include sophisticated methods for structure-of-influence graph analysis and visualization.

References

[1] Adam Oliner and Alex Aiken. A query language for understanding component interactions in production systems. In *24th ACM International Conference on Supercomputing*, pages 201–210, New York, NY, USA, 2010. ACM.

[2] Adam Oliner and Alex Aiken. Online Detection of Multi-Component Interactions in Production Systems. In *Dependable Systems and Networks*, pages 49–60. IEEE, 2011.

[3] Adam J. Oliner, Ashutosh V. Kulkarni, and Alex Aiken. Using Correlated Surprise to Infer Shared Influence. In *Dependable Systems and Networks*, pages 191–200, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

ECRAM (Elastic Cooperative Random-Access Memory) HPI Future SOC Lab Project Report March 2012

Kim-Thomas Rehmann, Kevin Beineke, Michael Schöttner
Institut für Informatik, Heinrich-Heine-Universität Düsseldorf,
Universitätsstraße 1, 40225 Düsseldorf, Germany
E-Mail: Kim-Thomas.Rehmann@uni-duesseldorf.de

Abstract

The ECRAM storage implements in-memory objects for data-centric cloud applications. In order to make objects availe on shared-nothing architectures, ECRAM replicates objects. An adaptive replication mechanism analyses object access patterns to switch dynamically between update and invalidate protocol. We have evaluated the impact of different replication strategies on performance using HPI Future SOC Lab's resources. The experiments demonstrate that adaptive replication scales better than update and invalidate replication for the applications under examination.

1 Project Idea

The ECRAM project addresses current research topics related to storage for cloud applications. ECRAM simulates a shared in-memory storage on shared-nothing clusters. The object-based storage provided by ECRAM is comparable to a key-value store, where the keys are object identifiers and the values contain binary object data. In contrast to most existing key-value stores, ECRAM increases availability by replicating data. It synchronizes objects optimistically in order to allow atomic multi-object operations.
ECRAM does not restrict the format of objects it stores. It neither associates any semantics with object content. Applications can therefore implement row-based or column-based tables, or they can define a flexible data format, such as graph structures.
Cloud applications often have read-optimized object access patterns. Access patterns with prevailing reads are a benign workload for optimistic concurrency control. Therefore, ECRAM implements optimistic multi-version concurrency control using the concept of memory transactions. In contrast to traditional DBMS transactions, memory transactions operate on replicated rather than partitioned data. Replication relieves ECRAM from the need to handle distributed transactions. Transactions can access distributed storage, but they always execute on a single node. The colocation of in-memory storage and application code allows ECRAM to restart transactions transparently for applications.
By caching replicas in the application address space, ECRAM achieves zero-latency object accesses at the best. Transparent access detection using virtual memory hardware simplifies application development. ECRAM's adaptive caching mechanism is able to switch dynamically between update and invalidate semantics and to prefetch objects to reduce access latency. Adaptive caching monitors and analyses access patterns to predict future accesses.
Our experiments on HPI Future SOC Lab resources investigate the performance of ECRAM's adaptive replication on a multicore machine with huge main memory. ECRAM can run on conventional compute clusters over TCP/IP networking as well as on a single machine over the loopback network interface.

2 Future SOC Lab Resources

We are executing our experiments on a Hewlett Packard ProLiant DL980 G7 Server. The server is equipped with 8 Xeon Nehalem X7560 CPUs, each having 8 hyper-threaded cores. The CPU clock rates are 2,27 GHz, the L3 caches 24 MB large, and Turbo Boost is enabled. The DL980 has 128 GB of RAM. Given that our experiments run in main memory of the single machine, they do not use the hard disks, neither the Fibre Channel network card. The DL980 boots Ubuntu Server 10.10 from a local harddisk and mounts the home file-system from a NAS device.

3 Findings

Our previous project report from October 2011 has documented the initial results of running ECRAM on Future SOC Lab. In the Winter 2011/2012 period, we have assessed ECRAM's performance more in detail with a special focus on adaptive replication.

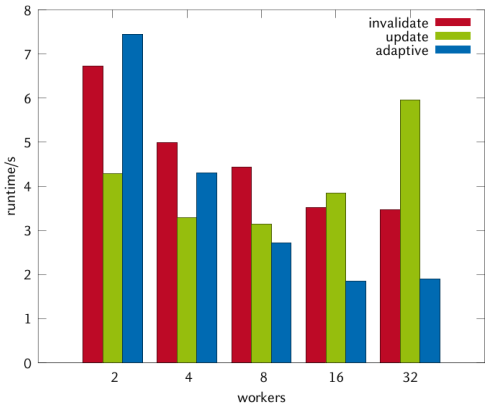


Figure 1. Execution time of raytracer using ECRAM on DL980, image size 1800x2400 pixels

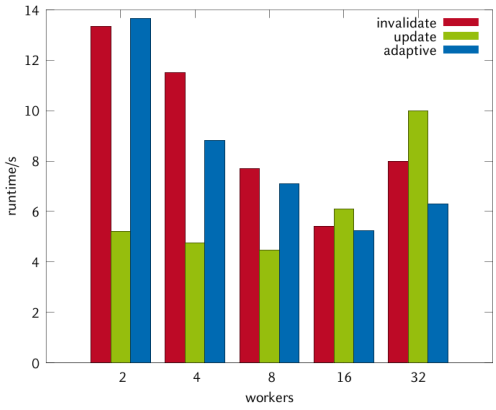


Figure 2. Execution time of KMeans using ECRAM on DL980, 100,000 points in 3D space with 16 cluster centers

Figure 1 shows the runtime of the raytracer application with an image size of 1800x2400 pixels and 228 objects in the scene graph. The raytracer application is implemented using ECRAM's framework for in-memory MapReduce [1]. In the map phase, the worker nodes calculate the pixels in disjoint regions, and in the reduce phase, the output image is produced. The raytracer's map phase is an *embarrassingly parallel* workload, because each pixel in the output image is traced independent of any other pixel. The raytracer's reduce phase simply collates all regions computed by different nodes into the final output image. The invalidate protocol causes the reduce jobs to pull the output from the map jobs, which increases the amount of synchronous network messages and thereby the overall latency. The update protocol distributes all data generated to all other nodes, most of which do not access the data lateron. The bandwidth required to tranfer the huge amount of data limits scalability. The adaptive replication protocol notices that the output of the map jobs is afterwards accessed by the reduce jobs. It sends updates directly from map workers to reduce workers, which keeps the required bandwidth low and at the same time avoids synchronous requests to pull replicas.

Figure 2 shows the runtime of the KMeans application with 100,000 points in the 3D space, which are grouped into 16 clusters. The KMeans algorithm is generally not so well parallelizable, because the iterative execution often updates data. However, the adaptive replication protocol often successfully predicts where data is required in the next round, so it scales better than the invalidate or update protocol.

4 Next Steps

In the context of cloud computing, network and machine failures are common. Therefore, the reliability of distributed storage is an important research topic. Until now, ECRAM provides reliability by means of replication in volatile memory. We are currently working on providing persistency within ECRAM by logging and storing objects to durable storage such as harddisks, flash memory, solid state drives and phase-change memory. Each of these storage technologies has its own characteristics, such that persistent storage should implement different policies for different technologies. If time and resources permit, we intend to evaluate our distributed in-memory file-system, which is based on ECRAM and FUSE, on Future SOC Lab.

ECRAM enhances fault tolerance of parallel applications by executing each thread in a private protection domain. However, using this approach, different threads executing on the same machine cannot benefit from their colocation. OS-level IPC mechanisms could allow them to access shared data directly. Direct sharing can potentially boost the performance of applications on powerful multicore machines like the ones in HPI Future SOC Lab. We plan to implement direct sharing techniques in ECRAM and evaluate them on Future SOC Lab.

References

[1] Kim-Thomas Rehmann and Michael Schöttner. An in-memory framework for extended MapReduce. In *Proceedings of the Seventeenth IEEE International Conference on Parallel and Distributed Systems 2011 (ICPADS 2011)*, Tainan, Taiwan, 12 2011.

KONECT Cloud – Large Scale Network Mining in the Cloud

Report, Winter 2011/2012

Dr. Jérôme Kunegis
WeST – Institute for Web Science and Technologies
University of Koblenz–Landau
kunegis@uni-koblenz.de

Abstract

In the Winter 2011/2012 run at the Future SOC Lab, we used the KONECT framework (Koblenz Network Collection) to compute ten different network statistics on a large collection of downsampled versions of a large network dataset, with the goal of determining whether sampling of a large network can be used to reduce the computational effort needed to compute a network statistic. Preliminary results show that this is indeed the case.

1. Introduction

Networks are everywhere. Whenever we look at the interactions between things, a network is formed implicitly. In the areas of data mining, machine learning, information retrieval and others, networks are modeled as *graphs*. Many, if not most problem types can be applied to graphs: clustering, classification, prediction, pattern recognition, and others. Networks arise in almost all areas of research, commerce and daily life in the form of social networks, road networks, communication networks, trust networks, hyperlink networks, chemical interaction networks, neural networks, collaboration networks and lexical networks. The content of text documents is routinely modeled as document-word networks, taste as person-item networks and trust as person-person networks. Databases, the data storage method *par excellence*, are essentially complex network storage engines.

In fact, a majority of research projects in the areas of Web Mining, Web Science and related areas uses datasets that can be understood as networks. Examples are social networks, hyperlink networks, bipartite rating graphs, and many more. Unfortunately, results from different papers cannot be compared easily if they use different datasets. Therefore, the goal of the KONECT project is to provide a consistent access to network datasets. To achieve this goal, KONECT contains a large number of network datasets (about 150) of all different types. This also has the advantage that a typical study may use several networks for their research, since all networks offered by KONECT are provided in the same data format, which is chosen to be easily usable from a large number of programming language and environments. The KONECT project also provides the code for the computation of the ten network statistics which we investigated in this project.

In the run of the Winter 2011/2012 Future SOC Lab Call, KONECT was used to compute statistics on a large collection of downsampled versions of a single large social network, in order to test the feasibility of sampling as a strategy to reduce the computing overhead of analysing very large networks. Our preliminary results indicate that sampling can indeed be used effectively to significantly reduce the computational overhead when calculating many diverse network statistics.

2. KONECT

KONECT, the Koblenz Network Collection, is a framework for the analysis of large network datasets, as well as a collection of such datasets. KONECT currently contains 147 network datasets. In addition to these datasets, KONECT also consists of code to generate statistics and plots about them, which can be viewed on the KONECT website[1]. Of the 147 datasets, the largest is the Twitter social network with 42 million nodes and 1.5 billion edges [2], and the smallest dataset is the metabolic network of *C. elegans*, with 453 nodes and 4,596 edges [1]. Figure 1 shows a scatter plot of all network by the number of nodes and the network's density, i.e. the average degree of nodes.

3 Architecture

KONECT is both a collection of datasets and code to generate statistics and plots, as well as a website showing all these. Starting with datasets publically available on the World Wide Web, the different components of KONECT generate the standard network format files, compute all network statistics, generate

[1] konect.uni-koblenz.de

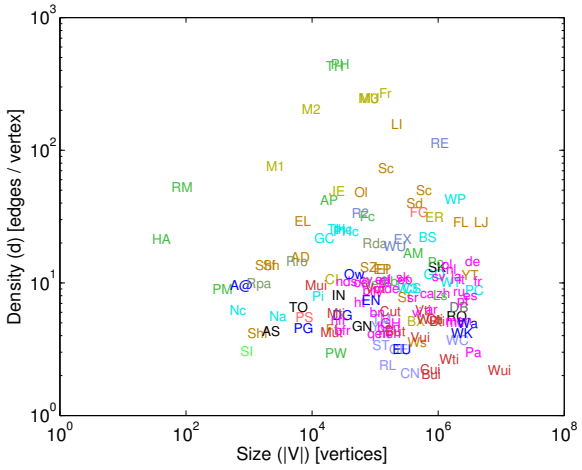


Figure 1. An overview of the size of all datasets: all networks of KONECT arranged by the size (the number of nodes) and the density (the average number of neighbors of all nodes). Each network is represented by a code whose color corresponds to the network category (e.g., social network or hyperlink network). The complete list of networks and their codes is available online at konect.uni-koblenz.de/networks.

graphical analyses, evaluate link-prediction methods and finally generate the KONECT website on which all results are browsable. All code is available under the GNU GPL on konect.uni-koblenz.de. The architecture of the KONECT system consists of the following components:

- Extraction code that reads out publically available network datasets from the Web and generates standard-format files. The extraction code is specific to each network extracted, and generates dataset files in a simple, unified file format. This file format is designed to be usable from within as many programming languages as possible, and therefore uses only text which is easy to parse, but still is flexible enough to allow for the variety of structural properties in all network datasets.

- Analysis code to generate network statistics, including network statistics over time and various matrix decompositions. This code is written in Matlab and in the C++ programming language, and is optimized for scalability to very large networks.

- Code to generate plots from the computed numbers. This code is written in Matlab. There are currently more than 23,000 plots in KONECT. These include plots generated for each individual network (e.g., the degree distribution) and plots comparing all networks in this paper).

- Code that implements various link prediction algorithms. In different types of networks, the generalized link prediction problem is known by different names, for instance friend recommendation in social networks, rating prediction in bipartite rating graphs or trust prediction in trust networks. This code implements a large selection of basic and advanced link prediction algorithms, and an evaluation of these. This allows link prediction algorithms to be compared across a large number of networks, something which is done only in a small number of studies.

- The website konect.uni-koblenz.de is finally generated from the generated plots and numerical statistics.

4 Graph Sampling

In the Winter 2011/2012 run, we used a single node in the Future SOC Lab with 1 TiB of RAM to investigate graph sampling in the following way. Sampling is the operation on a graph that consists in finding a subset of a graph such that the resulting graph can be used instead of the original graph to greatly reduce the runtime when computing network statistics such as the network diameter or spectral norm.
Taking a very large graph dataset containing 200 million edges, we computed samplings of it containing from 5% to 100% of all edges, in steps of 5%. For each sample size, we generated twenty different samplings randomly, and computed all KONECT network statistics for them. As a result, we can observe by what amount the individual statistics are changed by the sampling, and thus whether sampling will affect the statistics, and if it does, in what way.
Preliminary results are given in Figure 2. As preliminary observations, we can cite the following:

- The effective diameter is not affected by a sampling of at least 60% of all edges.

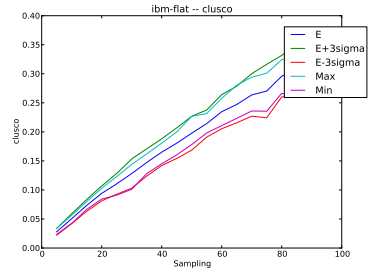
- The spectral norm is affected in a linear fashion with respect to the sampling coefficient.

- The size of the largest connected component, the controllability, the (normalized) degree distribution entropy and the power-law exponent are affected by sampling, but in a very predictable way.

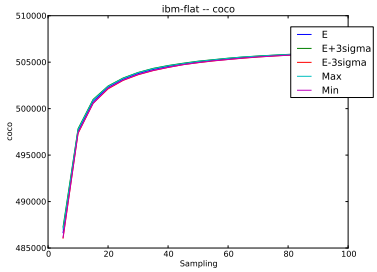
We plan to publish these and other results in order to give network researchers tools to analyse very large networks with computing ressource smaller than those available at the Future SOC Lab.

5. Conclusion

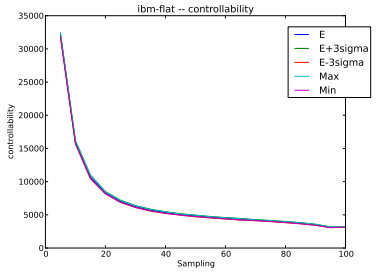
We thus conjecture that graph sampling can be used to make the computation of certain network statistics



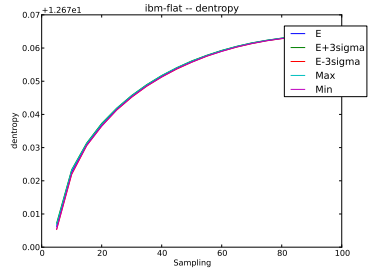
(a) Clustering coefficient



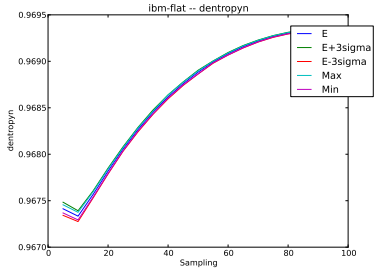
(b) Largest connected component



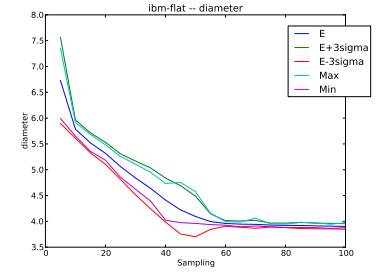
(c) Controllability



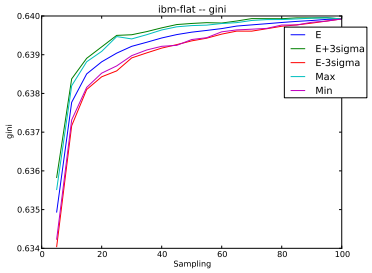
(d) Degree distribution entropy



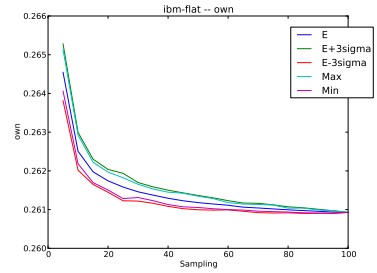
(e) Normalized degree distribution entropy



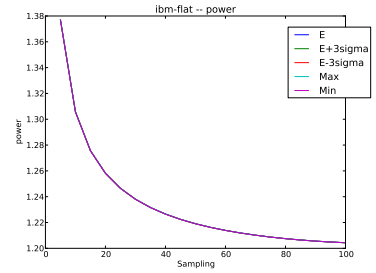
(f) Effective diameter (90 percentile)



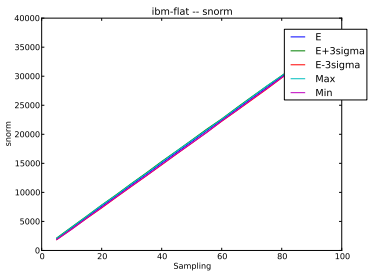
(g) Gini coefficient of degree distribution



(h) Balanced inequality ratio



(i) Estimated power-law exponent



(j) Spectral norm

Figure 2. The results of graph sampling. Each plot represents one network statistic, and shows the sampling coefficient (i.e., the amount of retained edges) on the X axis, and the measured network statistic on the Y axis.

more efficient, especially those for which no closed-form expression in the sampled graph exists, such as the effective network diameter.
In the next semester at the Future SOC Lab, we plan to used the KONECT framework to integrate more very large networks into the KONECT system, such as those from Wikipedia, the Semantic Web, Twitter and other sources, where the number of edges in the networks exceeds one billion. Other planned tasks, all using the KONECT framework, include the evaluation of link prediction algorithms and other machine learning tasks on these very large graphs.

6 Acknowledgments

First of all, we thank the Future SOC Lab for providing the intruments to run KONECT. Furthermore, we thank everyone who has released network datasets to the public, in particular Jure Leskovec, Alan Mislove, Chris Bizer, Munmun De Choudhury, Christian Thurau, Andreas Hotho, Alan Said, Robert Wetzker, Nicolas Neubauer, Pan Hui, Eiko Yoneki and the Wikimedia Foundation. Thomas Gottron, Daniel Dünker and Martina Sekulla have assembled several network datasets in KONECT. The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n^{o} 257859, ROBUST, as well as from the German Research Foundation (DFG) under the MULTIPLA project (grant 38457858).

References

[1] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72(2):027104, 2005.

[2] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Proc. Int. World Wide Web Conf.*, pages 591–600, 2010.

Integrated Management Support with Forward Business Recommendations

Prof. Dr. Rainer Thome
Dipl.-Kff. Patricia Kraft
Chair in Business administration
and business computing
Joseph-Stangl Platz 2
97070 Würzburg
thome@wiinf.uni-wuerzburg.de
pkraft@wiinf.uni-wuerzburg.de

Dr. Andreas Hufgard
Dipl.-Kff. Stefanie Krüger
IBIS Labs
Mergentheimer Str. 76a
97082 Würzburg
Hufgard@ibis-thome.de
skrueger@wiinf.uni-wuerzburg.de

Abstract

Data provisioning plays an increasingly pivotal role in business today. Decision-makers need relevant information at their fingertips in time to make well-informed, sustainable decisions. The Forward Business Recommendations approach attempts to provide just this. The team at IBIS Labs would use a project extension to analyze various scenarios to find an optimum balance between user benefit and practicability.

1 Rule based Business Matrix Processing and Forward Business Recommendations

The primary object at project launch was to create a rule base that could be used to extract relevant information from the ERP system. This original idea formed the foundation of Rule based Business Matrix Processing, which defined processes whose sequence and results were monitored with the aid of these rules.

The next phase consisted in applying Forward Business Recommendations to forecast potential problems and future business opportunities. One example is the order-to-cash process. It analyzes opportunities and their impact on future stock and liquidity levels, and on project deadlines, to deliver vital information early on, so decision-makers have ample time to respond. For instance, if FBR forecast a liquidity squeeze, management could trigger redeployment of capital or arrange for loan negotiations with the bank before liquidity levels plummeted. The organization would gain flexibility, which would enable decision-makers to avert critical situations. Processes are similar in every business, but companies' dissimilarities must also be modeled in a rule base. For this reason, the project team explored customization options and integrated them into the prototype, where appropriate.

The FBR prototype contains a model of the opportunity-to-production process. It was conceived using tools from the SAP Business ByDesign solution and implemented externally, because in the beginning stages, direct integration into ByDesign was not possible. Team members therefore utilized the vast capabilities offered by SAP ByDesign Excel Add-in's integrated data export to deposit corresponding rule logic outside of ByDesign.

The upshot: the prototype furnishes a concrete recommendation for each opportunity already contained in ByDesign and each new one entered. It suggests how decision-makers should best handle this specific opportunity and reveals effects on the company's future development. For example, if decision-makers receive an opportunity and

the organization has all components for it in stock, the prototype will recommend offering a discount as an incentive and so, increase the probability of implementing this opportunity. When sales reps meet with a client, they cannot know current and future stock levels for all products. Often sales reps are not even authorized to view this information. In this way, reps receive the vital information in the form of a recommendation, without having to search through data from other user departments.

2 Status quo

The prototype was recently upgraded to be compatible with the new SAP ByDesign release FP 3.0. This enabled the project team to test new options available in the current release and their relevance to the prototype.

Furthermore, team members examined the possibilities in the ByDesign Studio development environment, Software Developer Kit (SDK). They also gained valuable instruction and project experience by developing enhancements in SAP Business ByDesign.

3 Current developments

The project team is currently working to design an integrated solution for the FBR prototype. Creating a solution that can work within ByDesign would afford a great number of advantages but would also have a few disadvantages.

3.1 Advantages

One advantage of development using SAP ByDesign Studio is that the solution is integrated into the original product. The user would not notice a difference between the core ByDesign system and the app, because everything would have the same look and feel.

Besides being more user-friendly, integration with ByDesign would be a huge advantage. It would shorten communication channels and vastly increase possibilities for data exchange between user departments. Whereas an external solution requires that data be made available to it, an internal solution could independently trigger a targeted revaluation whenever a change was made to the data pool (as an extreme example), and so, deliver up-to-the-minute data. This degree of currency is a decisive factor in day-to-day business.

3.2 Disadvantages

While the advantages mentioned are great, it is important to remember that the conditions for internal ByDesign solutions are very strict. In some cases, users have access to a larger body of data when the solution is external, rather than integrated. The Public Solution Model (PSM) includes data sources accessible to the public and SAP has released only these data sources for use in apps.

In addition, the ERP system's rigid division into work center views makes it more difficult to integrate FBR. The team will address this problem and implement a workable solution in the next project phase (see Section 4).

3.3 Evolving the new app

In summary, the project team has completed the design of the prototype, revision of the first set of rules and definition of possibilities in ByDesign Studio. The team can now move to the next phase, in which it evolves the prototype into an app.

4 Next steps and required resources

The following project phase is divided into two stages. The first stage consists in evaluating each of the scenarios and weighing costs and benefits. This includes analyzing usage in the scenarios for each user group and comparing practicability of the scenarios.

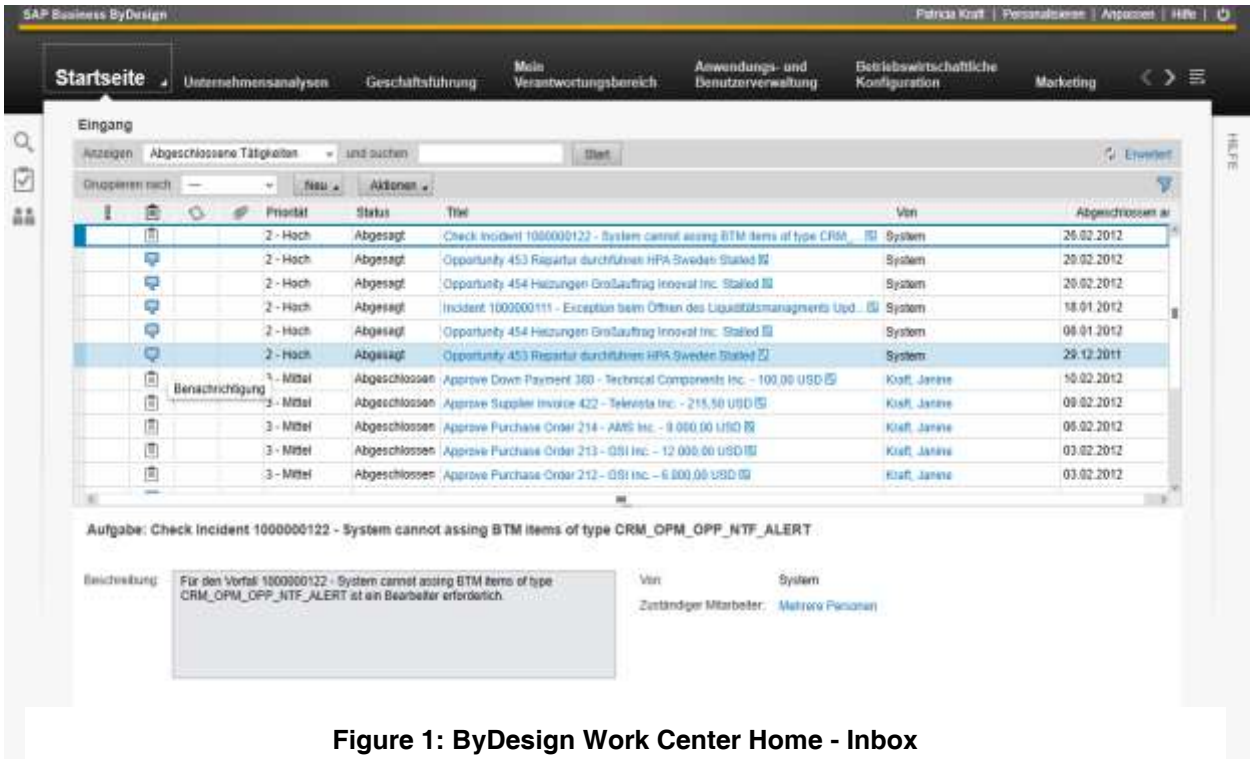


Figure 1: ByDesign Work Center Home - Inbox

4.1 Scenario 1: Integrated alarm function

The first scenario offers the greatest potential for integration. The developers will try to generate alarms, warnings and notifications in the background. A notification will appear on the screen only when a significant incident occurs. However, this solution cannot be employed using ByDesign Studio in its current release, because ByDesign Studio has no back-end access, and background operations require deliberate user interaction. This solution would be the most convenient for the user, since s/he would receive the recommendation and a concrete reference without having to actively search for the data.

For example, if during a specific period a very high-volume opportunity generated a critical liquidity level as a result of related purchase orders, the manager could be alerted to this as soon as the opportunity was created, so that s/he could take appropriate action. This would greatly increase the company's flexibility and ensure certainty with respect to company resources.

4.2 Scenario 2: Work center-oriented

This scenario addresses the work centers. A monitoring function would be deployed in a specific location in the system and used equally by all employees. A new work center view would appear within an existing one. For example, the New Business work center view might appear with reference to opportunities. Alternatively, the Purchase Orders and Purchase Requisitions work center view might pop up when purchase orders needed to be executed ASAP in order to implement a last-minute opportunity that promises high probability.

4.3 Scenario 3: User-oriented

The third scenario addresses individual users. If a change were made in their application area that could influence their actions, they would be notified of the change. This could be done by inserting an activity overview (Figure 1) on the start screen. In this scenario, as in the first, back-end processing presents a problem. If these obstacles were not present, it would be possible for the system to send a message to the user when, for example, s/he needs to deal with a certain opportunity that cannot be implemented.

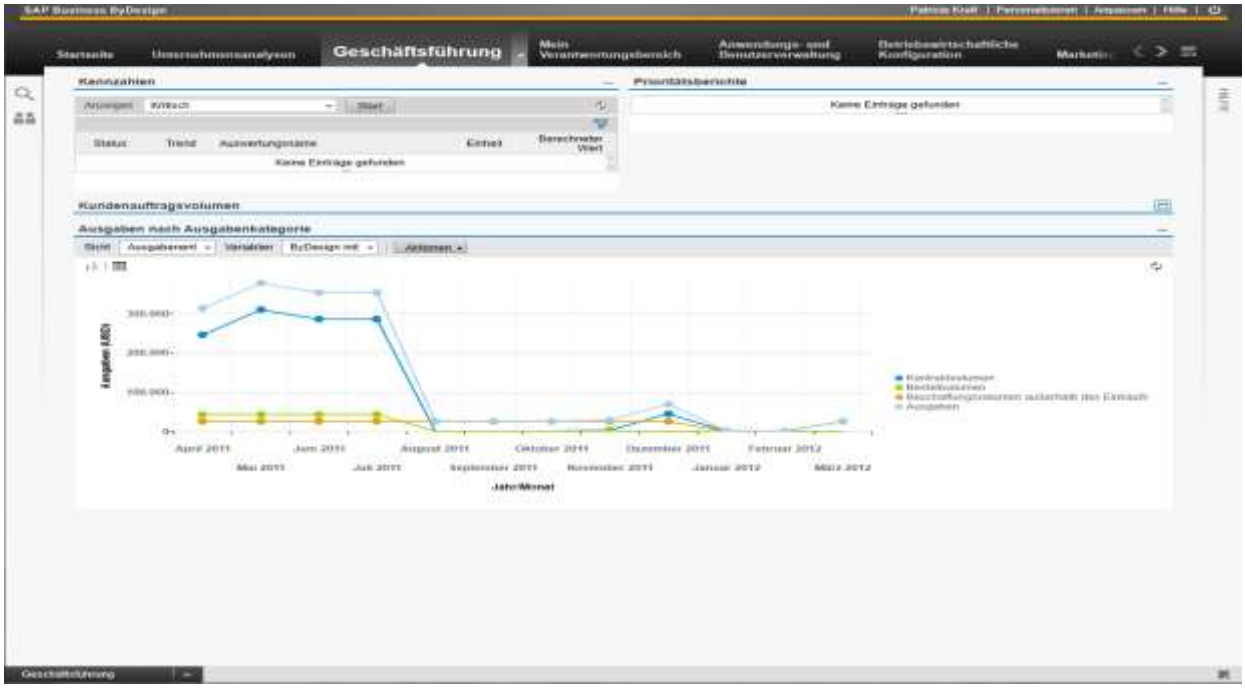


Figure 2: ByDesign Corporate Performance work center

4.4 Scenario 4: Central work center-oriented

The fourth and final scenario is the easiest to implement. It consists in creating a separate, centralized work center in which all data is grouped as in the Corporate Performance work center (Figure 2). However, this view would be developed for management, who would then pass on the necessary recommendations and instructions to the respective employees. One drawback is that executives would be the only ones furnished with an overview of the entire organization. Business processes would not be supported directly at transaction level.

4.5 Required resources

Finally, the question of necessary resources remains. To deploy the prototype in ByDesign, the project team requires a ByDesign system with a connection to SDK. It would be best to use the latest version of ByDesign Studio (FP 3.5) because of marked programming improvements. Up to now the team has had access to a ByDesign Studio based on release FP 3.0. But because of the Public Solution Model, far fewer development options are possible.

Service-Based 3D Rendering and Interactive 3D Visualization

Benjamin Hagedorn
Hasso Plattner Institute for
Software Systems Engineering
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
benjamin.hagedorn@hpi.uni-potsdam.de

Jürgen Döllner
Hasso Plattner Institute for
Software Systems Engineering
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
doellner@hpi.uni-potsdam.de

Abstract

This short report describes our work in the context of the HPI Future SOC Lab. This work generally aims on exploiting high performance computing (HPC) capabilities for service-based 3D rendering and service-based, interactive 3D visualization. A major focus is on the application of HPC technologies for the creation, management, analysis, and visualization of and interaction with virtual 3D environments, especially with complex 3D city and landscape models.

1 Latest Work and Results

During the last project phase, the Computer Graphics Systems group at the HPI did continue its research and development on fundamental concepts and techniques in the area of service-based 3D rendering and service-based, interactive 3D visualization. The techniques developed so far rely on and take advantage of multi-core/multi-threading processing capabilities, the availability of large memory, and GPGPU systems as provided by the HPI Future SOC Lab.

1.1 Massive Texture Processing

We successfully continued our research on the processing of very large 3D city model data (including massive 3D geometries and 2D texture data) and its service-based and image-based visualization and distribution [1]. Especially, we experimented on the optimization of workflows and tools for the preprocessing of large sets of raster data by parallel computation and usage of massive main memory. Based on earlier work on generally reducing processing times, we have been able to enhance our algorithms for the preprocessing and optimization of massive texture data, which forms a major input for high-quality 3D rendering techniques. Through this, we could dramatically increase the visual quality of the rendering output leaving the texture data footprint nearly the same.

1.2 Web View Services

Also, the Future SOC system forms part of the implementation of an experimental web view service currently developed within an ongoing international initiative at the Open Geospatial Consortium OGC. As one milestone in that context, the so called OGC 3D Portrayal Interoperability Experiment has been finished in September 2012 and results have been published as an OGC Public Engineering Report [2].

2 Next Steps

To continue our research in the area of service-based 3D rendering and service-based, interactive 3D visualization, we would like to continue our scientific work using the Future SOC Lab.

In particular, we will continue our research on server-based 3D rendering techniques as well as on design and construction of service-based, cloud-enabled systems for interactive real-time 3D visualization. For example, we are planning to continue our work to research and development on service-based technologies for assisted interaction and camera control in massive virtual 3D city models. Based on previous work in exploiting the Future SOC Lab capabilities for processing massive 3D point cloud data (such as generated through laser scanning), we are planning to design and implement a fast technique and system for automatically classifying large 3D point clouds. Also, we also planning to extend our work on processing spatially related data to the preprocessing, and on-demand exploration and analysis of 3D trajectory information (such as flight information data).

References

[1] J. Döllner, B. Hagedorn, J. Klimke: Server-Based Rendering of Large 3D Scenes for Mobile Devices Using G-Buffer Cube Maps. 17th Int. Conference on 3D Web Technology, 2012.

[2] B. Hagedorn, Ed.: "Web View Service Discussion Paper, Version 0.3.0, OGC 09-166r2." Open Geospatial Consortium Inc., 2010.

Smart Wind Farm Control

Patrick Böwe
University of Oldenburg
Department of Computing Science
Uhlhornsweg 84
D-26129 Oldenburg
patrick.boewe@uni-oldenburg.de

Ronja Queck
University of Oldenburg
Department of Computing Science
Uhlhornsweg 84
D-26129 Oldenburg
ronja.queck@uni-oldenburg.de

Michael Schumann
University of Oldenburg
Department of Computing Science
Uhlhornsweg 84
D-26129 Oldenburg
michael.schumann@uni-oldenburg.de

Deyan Stoyanov
University of Oldenburg
Department of Computing Science
Uhlhornsweg 84
D-26129 Oldenburg
deyan.stoyanov@uni-oldenburg.de

Benjamin Wagner vom Berg
University of Oldenburg
Department of Computing Science
Uhlhornsweg 84
D-26129 Oldenburg
benjamin.wagnervomberg@uni-oldenburg.de

Abstract

The occurrences of fossil sources of energy are restricted and the energy extraction gets more and more expensive. In order to meet the energy demand in the long term, renewable energy technologies are promoted. Most notably, wind energy plays a central role in this context. The number of installed and operating wind turbines has risen rapidly over the past years. There are two types of wind farms – onshore and offshore. One of the biggest cost-factors of both types is maintenance. While onshore wind farms are relatively easy to maintain, offshore wind farms cause high maintenance costs. There are a variety of reasons for this: restricted means of transportation, dependency on meteorological conditions and a more complex supply chain.
The project Smart Wind Farm deals with the identification of average lifetimes of the individual components of wind turbines in order to optimize the maintenance work for the offshore sector. By using different analyses, the current condition and approximate remaining lifetime of the wind turbine components shall be identified. This knowledge can be used to replace components with a short remaining lifetime proactively in a routine maintenance. This way, unscheduled repairs can be minimized.
SAP HANA will be used in a different context to realize these goals. Normally, in-memory technology (and Business intelligence in general) is used in the business domain. In this project, SAP HANA is applied by engineers.

1 Wind Farms

Against the backdrop of global challenges like climate change, growing energy demand, constantly rising prices for primary fossil fuels as well as the Fukushima nuclear disaster, renewable energies are providing an increasingly important contribution to the energy sector [1]. In the Federal Republic of Germany, the last nuclear power plants will be decommissioned by the end of 2022. Until then, the renewable energies will become the supporting pillar of the future energy supply, making up at least 35 percent of the energy mix. In the year 2050, 50 percent of the energy mix will be created by renewable energies [2]. Energy scenarios have shown that wind energy will play a central role in generating electricity in 2050. This requires a massive expansion of wind energy plants, and offshore wind parks in particular. Wind energy offers the most economic and effective potential for expanding renewable energies in the short and medium term [3]. Hence, the number

of wind turbines has risen rapidly over the past years (see Figure 1).

Generally, the capacity of a wind turbine (WT) is defined by its rotor diameter. The rotor diameter determines the proportion of the wind flow which is available to the WT for conversion into electric energy. The energy of the wind flow rises to the third power to the wind speed, which increases with the height above ground level. By building higher towers, the turbines can use increased wind speed and thus realize a higher return.

In order to evaluate and compare wind turbines, the annual energy supply is related to the rated output. This number is called full load hours and depends on the local conditions [5].

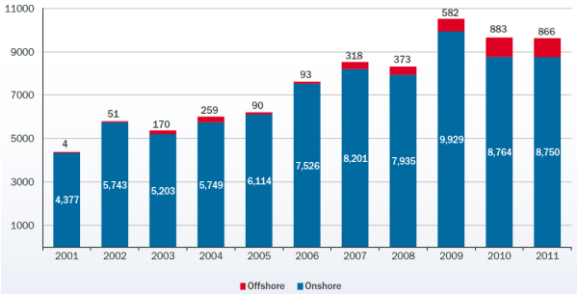


Figure 1: Annual onshore and offshore installations in MW [4]

A distinction is made between offshore and onshore wind farms. Onshore wind farms are located on the main land and can be subdivided into landscape categories. A normal onshore wind turbine produces 2 to 3 MW. The German average for a new WT is between 1552 and 1667 full load hours. All German WTs which were built before 2002 are qualified for repowering. Repowering means the replacement of older wind turbines with more modern multi-megawatt machines.

Offshore wind farms are located at sea. Modern offshore wind turbines produce around 5 MW. Because of increased average wind speed, the revenues are a lot higher than on the main land. Offshore wind farms can generate between 3000 to 4500 full load hours [5].

2 Maintenance of Wind Turbines

Wind turbines have a planned lifespan of 20 years. During this period, many main components have to be maintained or replaced.

The maintenance of offshore WTs is a lot more problematic than onshore WTs, because they can only be reached by ship and helicopter. Therefore, offshore maintenance causes costs which are six times higher than those on the mainland [1].

For ships, the wave height determines significantly the access of an offshore wind turbine. Usually, weather conditions with a wave height above 1.5 m are called “weather days”, because the WTs cannot be reached hazard-free. The annual number of “weather days” for different German offshore wind farms is shown in Figure 2 [5].

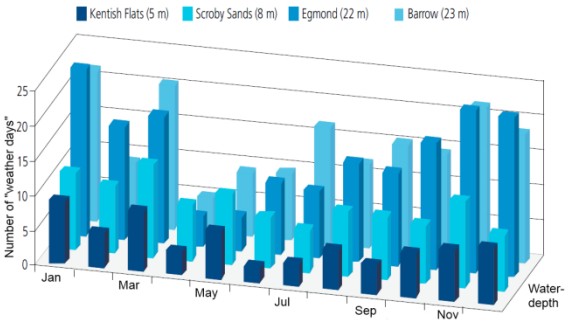


Figure 2: Accessibility of offshore wind farms [5]

In addition to the above-mentioned difficulties (restricted means of transportation and dependency on meteorological conditions), offshore maintenance also has a more complex supply chain. It is very important to ensure a reliable and cost-efficient parts supply. Since the topic of offshore wind farming is still very new, it has not been possible to standardize any maintenance concepts yet. The long-term reliability of wind turbines is still unknown. Hence, no spare part storage concept from other industries may be adopted [1].

3 Objective Target

The main objective target is to develop a wind farm management maintenance system based on new technologies and scientific approaches. The principal objective targets of this system can be separated into the following topics:

Proactive management system

The proactive management system intends to evaluate all relevant physical values, which are provided by the offshore wind park. Real-time monitoring and reporting should be based on a dataset containing 400 records per second per turbine. Particularly, the use of any averages for faster calculations as well as the reduction of storage space like in other systems has to be avoided. Furthermore, the system should provide an automated error detection and error classification unit.

Realistic forecasts for the cost/income ratio

Based on the large amount of physical values and supplementary data, like historical data or maintenance efforts, a more realistic cost/income ratio forecast should be developed.

Exact forecasts for maintenance periods

Focusing on the turbine maintenance, the target is to forecast lifetime estimation and breakdowns. Forecast reports and pre-alerting for all turbine components should be created automatically by the system. Based on different researches in this area, it is possible to develop algorithms which make the system able to generate these reports and alerting using weather data, resource data, operational data and maintenance history data.

On demand statistic functions for physical research

Resting on new database technologies, more complex analyses can be executed on a larger dataset. Finally, the period of computation is shorter. Thus, faster responses are possible to improve the workflow in research, like developing algorithms or analyzing complex diagrams.

4 Reasons to Use SAP HANA for this Project

The main task of the proactive maintenance of offshore turbines is to calculate the average remaining life expectancy. As mentioned in chapter 3, the current analyses are performed on aggregated data, although more granular data could lead to more precise calculations. An advantage is that the 400 sensors of the wind turbine are already delivering data on a per second basis. In order to analyze this data set, which is increased by a factor of 600 compared to aggregated data on a 10 minute basis, SAP HANA has to be used.

By using SAP HANA, the creation of OLAP cubes can be omitted. Hence, more data can be analyzed and new analyses can be performed directly on the data. SAP HANA offers the possibility for the engineers to create new analyses, test them directly and subsequently continue their work with new findings.

SAP HANA and Business Intelligence Tools will be used to achieve the objective targets of the Smart Wind Farm Control project. The use of these information technologies for optimization and support of science/engineer researches makes this project unique.

5 New Insights

One of the first insights gained from working with SAP HANA is the easy handling as well as the clean and organized layout. It offers many interfaces to integrate its in-memory technology into almost any IT landscape. The project group has developed a virtual wind farm in order to simulate sensor data. This data source can be connected to SAP HANA via JDBC. Consequently, it is possible to simulate workflow conditions, like real-time data transfer processes. Besides storing this operational data, it is also used as a source for analytical processes. This way, one of the most important features of the product SAP HANA can be used – one database containing transactional and analytical processes.

However, there are some features which are not supported by SAP HANA at the moment. A crucial aspect of the maintenance of wind farms is alerting. This feature is only possible when using workarounds with 3rd party software. Hopefully, SAP HANA will integrate this feature in future releases.

SAP HANA should not be used for simple reporting or data storage but instead for complex workflows, calculations and evaluations. Otherwise, the in-memory technology just leads to resource waste because there are no actual added values. SAP HANA offers benefits when the amount of data to be processed is very large and it is used for both transactional and analytical processes. Furthermore, in-memory technology enables real-time reporting [6].

6 Further Steps and Outlook

Currently, the project group has only limited access to real data and has to use a virtual wind farm to simulate the data sets instead. The project group is conducting negotiations with the companies ForWind and ENBW in order to receive larger data sets.

Once the larger data sets are inserted into SAP HANA, different analyses can be tested. As mentioned in chapter 5, HANA offers no alerting function. Therefore, the project group is implementing a workaround in a Java environment.

The proactive maintenance can only be fully tested on real data sets. As soon as those are given, these analyses will have to be refined. If the data sets contain information on the repair and replacement of the individual components, the results can be checked.

The project group could then calculate the improvement rate of their analyses compared to currently applied procedures.

References

[1] J. Westerholt: *Entwicklung eines Ersatzteilbevorratungskonzeptes für die Instandhaltung von Offshore Windenergieanlagen*. Diplomarbeit. Bremen, 2012.

[2] D. Böhme, W. Dürrschmidt, M. van Mark, F. Musiol, T. Nieder, T. Rüther, M. Walker, U. Zimmer, M. Memmler, S. Rother, S. Schneider, K. Merkel: *Erneuerbare Energien in Zahlen - Nationale und internationale Entwicklung*. Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. 1. Auflage, Berlin, 2012.

[3] Bundesministerium für Wirtschaft und Technologie: *Energiekonzept für eine umweltschonende, zuverlässige und bezahlbare Energieversorgung*. Niestetal, 2010.

[4] J. Wilkes, J. Moccia, M. Dragan: Wind in power – 2011 European statistics. European wind energy association. 2012

[5] S. Pfaffel, V. Berkhout, S. Faulstich, P. Kühn, K. Linke, P. Lyding, R. Rothkegel: *Windenergie Report Deutschland 2011*. Fraunhofer Institut für Windenergie und Energiesystemtechnik. Kassel, 2012.

[6] SAP HANA http://www.sap.com/solutions/technology/in-memory-computing-platform/hana/overview/index.epx

Measurement of Execution Times and Resource Requirements for single user requests

Alexandru Danciu
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3,
85748 Garching, Germany
danciu@in.tum.de

Helmut Krcmar
Technische Universität München
Chair for Information Systems
Boltzmannstr. 3,
85748 Garching, Germany
krcmar@in.tum.de

Abstract

In an ERP environment with limited resources and imposed service levels, user requests have to be scheduled according to their priority and expected response time. For predicting the response time of an incoming user request, the impact of the request on the resource consumption has to be known. Measuring the impact of single user requests on the hardware resource consumption represents a difficult challenge, especially in a production ERP environment. The reason is that a large amount of requests are processed at any time. Therefore only the impact of the whole set can be observed. The sequential execution and measurement of single user requests in an experimental ERP environment would be very time consuming. This research proposes an approach for measuring the execution times and resource requirements of single user requests in an ERP environment.

1 Introduction

Enterprise resource planning (ERP) systems support the execution of business processes within large companies. Because their performance, availability and scalability are very critical for companies, these systems are often implemented as distributed systems [5]. A prerequisite for exploiting the advantages of distributed computing is the allocation of resources based on the system load [1]. Scheduling algorithms require information on the execution time of each incoming request [4]. Information on the resource requirements of requests is desired for load balancing purposes [2].

The goal of this research is the development of an approach for measuring the execution times and resource requirements of single user requests.

2 HPI Future SOC Lab resources

During this research the hardware and software resources of the HPI Future SOC Lab were not used at all. Instead, several interviews were conducted with performance management experts from SAP AG. The interviews first aimed at the elicitation of functional and non-functional requirements for measuring the response times and resource requirements of requests in productive ERP environments. Based on the identified requirements, existing means for measuring these performance metrics in the context of SAP ERP systems were evaluated.

3 Conclusions

This research identified the tools for measuring response times and resource requirements supported by SAP ERP systems which are most suitable for productive environments.

Using distributed statistical records, performance measurements such as executions time and resource utilization can be collected across multiple heterogeneous systems for each request. This type of monitoring is a standard functionality even in productive system and does not influence the measured values.

The main challenge identified during this research is the operationalization of the term request and the mapping of requests to user activities. To overcome this challenge, we will focus on user interactions via external RFC and web service calls. This way, a function or web service call can be mapped directly to a user activity.

An important limitation of the proposed approach is the lack of a measurement for the I/O caused by single requests. Using distributed statistical records it is possible to measure the database calls, but not the file and network I/O.

4 Further work

Further research is based on the assumption that resource requirements and execution times of activities depend on the context, e.g. the input parameters and environmental factors like tasks executed in parallel. We will focus on the development of an approach for estimating the response times and resource demands for incoming user requests. Techniques for identifying correlations between the input parameters and the performance of single requests will be identified in literature. The proposed approach will be instantiated in a SAP landscape. Finally, the approach will be evaluated. The evaluation will be performed using multiple methods based on [3]. First an analytical evaluation - an architecture analysis - will be performed to evaluate the ability of integrating the approach in system landscapes. Second, a controlled experiment will be used to evaluate the prototypical implementation of the approach. The feedback provided by the evaluation will be used to improve the approach.

References

[1] Casavant, T.L.; Kuhl, J.G. (1988): A taxonomy of scheduling in general-purpose distributed computing systems. In: IEEE Transactions on Software Engineering, Vol. 14 (1988) No. 2, pp. 141-154.

[2] Devarakonda, M.V.; Iyer, R.K. (1989): Predictability of process resource usage: a measurement-based study on UNIX. In: IEEE Transactions on Software Engineering, Vol. 15 (1989) No. 12, pp. 1579-1586.

[3] Hevner, A.R.; March, S.T.; Park, J.; Ram, S. (2004): Design science in information systems research. In: MIS Quarterly, Vol. 28 (2004) No. 1, pp. 75-105.

[4] Iverson, M.A.; Ozguner, F.; Potter, L. (1999): Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. In: IEEE Transactions on Computers, Vol. 48 (1999) No. 12, pp. 1374-1379.

[5] Yung-Terng, W.; Morris, R.J.T. (1985): Load Sharing in Distributed Systems. In: IEEE Transactions on Computers, Vol. C-34 (1985) No. 3, pp. 204-217.

Future SOC Lab Spring Term Project Activities Report:
Benchmarking for Efficient Cloud Operations

Multitenancy Project Team
SAP Innovation Center Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany

Abstract

This project's main focus in the now-ending lab term was on the efficiency of cloud services provided in a multitenant environment. Several approaches to multitenancy have been implemented and evaluated using a multitenant-variant of the CBTR (Composite Benchmark for Transactions and Reporting) benchmark. Tenant workloads were varied in intensity and composition (analytical and transactional shares) and the effects on throughput and response time were measured. The results allowed interesting insights into the behavior of the approach implementations, but have been classified as company confidential and therefore must not be presented in this report. A proposal for continuation of this project in the coming lab term concludes this short report.

1 Report

The idea behind the cloud is to offer computing as a service with seemingly endless capacity that can be added or removed on demand [1]. The cloud customer keeps the data inside the cloud infrastructure and has access to the performance of a data center to execute complex operations on it. Through the network, data can be accessed in an easy way with various devices.

Cloud computing turns the IT investments of companies that move into the cloud into operational expenditures that they pay for the consumed services to a cloud provider. Consequently, the risk of correctly dimensioning the infrastructure as well as the need to keep capital expenditures as well as administrative costs at viable levels is transferred to the cloud provider. Multitenancy, i.e. consolidating several customers onto the same infrastructure, is one method to achieve higher utilization and therefore a more efficient cloud operation. [2]

In the now-ending lab term this project investigated several ways of accomodating several customers on one server, such as shared machine, shared database process, and shared table [2]. These approaches were compared with respect to their effect on throughput and on response time using two server machines of the Future SOC Lab. One served as database server (32 cores) and the second one as client machine (64 cores) from which the client requests were submitted and where the performance were measured.

The experiments employed CBTRmt, our multitenancy extension of CBTR, which is a mixed-workload benchmark based on SAP's Order-to-Cash process [3] comprising four analytical-type queries and nine transactional-type queries. Our extension allowed us to simulate a larger number of client organizations (for most experiments we selected 20 tenants) with various data sizes and request behavior in a highly concurrent manner.

In a first batch of experiments, we used analytical load, then added transactional load (20% and 80% share) and finally simulated time zone-diverse tenant localizations, i.e. not all tenants were active at all times, but followed an overlapping diurnal load pattern with one group entering night time while the other group woke up.

The results that we gathered allowed interesting insights into the approach implementations and their effect on system performance, but have been classified as company confidential and therefore must not be detailed in this public report.

2 Project Proposal for Next Lab Term

In the coming lab term we would like to continue our investigations for a further approach to realize multitenancy based on the virtualization platform Xen.

3 Acknowledgements

The project members wish to thank Bernhard Rabe of Future SOC Lab/HPI for his invaluable administrative support to setup and run the numerous experiments that were conducted as part of the project.

4 References

[1] Peter Mell, Timothy Grance: *The NIST Definition of the Cloud Computing*. NIST Special Publication 800-145. 2011.

[2] Dean Jacobs, Stefan Aulbach: *Ruminations on Multi-Tenant Databases*. BTW 2007: 514-521

[3] Anja Bog, Hasso Plattner, Alexander Zeier: *A mixed transaction processing and operational reporting benchmark.* Springer Science + Business Media, LLC 2010.

Instant Intrusion Detection using Live Attack Graphs and Event Correlation

Amir Azodi, David Jaeger, Feng Cheng, Christoph Meinel

Hasso Plattner Institute
University of Potsdam
14482, Potsdam, Germany

{*amir.azodi, david.jaeger, feng.cheng, meinel*}
@hpi.uni-potsdam.de

***Abstract*—Today, with the growing complexity of computer networks and companies dependence on such networks, securing them is more important than ever. Intrusion Detection systems have become a necessary tool to protect networks and privileged data from falling into the wrong hands. However IDS systems are limited by the amount of environmental information they process; including network and host information. Additionally the speed at which they receive such information can have a tremendous effect in the time needed to detect attacks. We have pinpointed three areas of Event Normalization, Alert Gathering and Event Correlation, where improvements can lead to better and faster results produced by the IDS.**

I. Introduction

The current state of IDS systems does have some weaknesses. One of the biggest problems is the inability to detect some of the more complex attacks; due to lack of access to the information which would have led to a successful detection. An underlying reason for this problem is the limitations of the viewpoint of any single sensor in the network. An example of such a scenario is the access to log files. Log files can contain large amounts of information regarding any attempts to breach security of a system. This problem becomes more evident when application logs are considered. The possibility to correlate between events gathered from different logs and systems, provides a unique level of access to information needed to detect the more advanced and well hidden attacks [3].

The rest of this report is organized as follows.

- Review of HPI Security Analytics Lab: In this section an overview of the legacy Security Analytics Lab (SAL) is given as background information.
- Updates - Design and Architecture: This section outlines the changes that were made to the legacy SAL system in order to incorporate new features and possibilities for detecting attacks.
- Leveraging Live Environment Information for Instant Attack Detection: This section describes the gathering of information in order to generate a sound and complete network graph to be used in conjunction with a comprehensive vulnerability database for the generation of a live attack graph.
- Extracting Attack Activities from Logging Information: Discusses the methods behind extracting all possible log information and to normalize them into one single format.
- Results and Achievements: In this section particular attention is given to the deliverables and completed objectives of the new phase, so far.

II. Review of HPI Security Analytics Lab

The SAL represents a system to encounter the challenges and provide a high level of security in a network. Different Log Gatherers and IDS Sensors provide a variety of data sources for the complex analysis on the In-Memory based platform. A multi-core-supporting architecture is the foundation for high performance as well as real-time and forensic analysis. Using efficient algorithms and various visualization techniques supports security operators with the challenging task of defending the network by identifying and preventing attacks [2].

The present deployment of SAL on the FutureSOC Lab has the following features listed below. This is a feature freeze instance and excludes the latest changes implemented into SAL. The latest changes will be added as part of the next deployment phase throughout the next six months.

- Detection of complex attack scenarios
- In-Memory based platform with up to 2 TB of main memory
- Multi-Core support with thousands of cores
- Correlation of events from a variety of data sources
- Utilization of environment information represented by attacks graphs
- Ranking of complex alert dependency graphs
- Visualization of attack scenarios and complex alert relations

III. Updates: Design and Architecture

The new design follows the 3 level architecture or chain of command system. First there is a component named

Enforcer/Gatherer, which is in charge of gathering and sending information relating to the host it is installed on. This information includes log files, system data such as installed software, available hardware and system status. At the second level a component named *Agent* receives the information from the Enforcer and normalizes it into CEE[5] events or control messages before sending them to the Server. Finally the *Server* will make the information persistent and continues on to generating an attack graph and correlating events received in order to find attacks. Figure 1 illustrates this workflow.

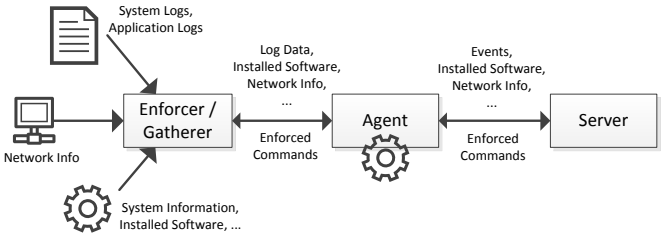


Figure 1. Message workflow in the SAL

IV. Leveraging Live Environment Information for Instant Attack Detection

The design proposed in this report allows for complete access to the host information of every host on which the Gatherer is deployed. This information includes, but is not limited to, log files (e.g. syslog), memory information, processor information, network information (e.g. hostname, IP addresses and MAC addresses) and application information (e.g. their CPE [6], which allows SAL to compile a list of current vulnerabilities present on any host).

Having imported all the information into SAL, a network graph can be created. Using a vulnerability database we can then construct a comprehensive attack graph to be used by SAL. In order to always have the most up-to-date attack graph, the SAL system continuously updates its underlying data structures, i.e. the network graph and the vulnerability database. Therefore at any moment an up-to-date attack graph can be compiled from the live information present in the underlying data structures.

V. Supporting Intrusion Detection with Event Log Information

A common way for reporting intrusions are security alerts, which are generated by so called intrusion detection systems. Generally, these alerts are generated as a result of observed activities in an environment that are potentially harmful. However, an alert is not a guarantee that an actual intrusion has taken place. In addition, alerts are only limited to detected malicious activity, but they do not give information about the attackers activities before and after his intrusion attempt. An approach to solve the shortcomings of an intrusion detection with alerts only is the consideration of additional details about the attacker activities. One type of such information are event logs. They are produced by applications, monitoring systems, operating systems, etc. and are give an insight into all observed events in an environment. The next subsections show how these event logs can be gathered from different locations in an environment, how they are normalized into an automatically processable format [4].

A. Information Gathering

Event logs usually originate from a multitude of sources, such as IDSs, firewalls or applications, where each source produces a different log representation and output format. Examples for log representations are Syslog or the IDMEF specification and typical output formats are binary, CSV and XML. In order to read events from the log sources, the previously mentioned gatherers are employed. They can be attached to the log files on their host and are able to send the file content to their centrally connected agent.

B. Normalization

The received event logs are interpreted event by event, which makes it necessary to split the sequential event stream into single events. The SAL realizes this splitting with the help of regular expressions that are on the matching of event separators. Once the events have been extracted from the event stream, they are still in a format being specific to the generating event source. Therefore, each event is further transformed into a common and easily processable representation. In the concrete SAL implementation, the *Common Event Expression* (CEE)[5] has been chosen as the common representation of events, to which all custom event are normalized to. The CEE makes use of event containers that contains the properties of the event as key-value-pairs.

In order to normalize events to the CEE format, information from the raw events have to be extracted and put into the corresponding fields. This extraction of event properties from the raw events is performed with the support of regular expression, or more specifically with the group mechanisms in regular expressions. The idea is to first define the format of a single raw event in the expression and then surround structured information with regular expression groups, so that they can be matched and finally used as the properties of the event.

VI. Results and Achievements

The mechanisms that have been described in the last two sections allow the SAL to react on live changes in the network, verify intrusions and uncover intrusions that could not be detected with the legacy SAL. In addition, it is possible to correlate the discovered intrusion attempts to the nodes in an attack graph. The described changes are mainly realized by interpreting dynamic and static environment

information as well as using event logs instead of only alert logs.

VII. Conclusion

As we work to secure networks from ever more sophisticated attacks, it is clear that the industry is in need of more intelligent protection mechanisms that actively protect our networks. As part of this project we aim to design intelligent solutions that consider a wide array of information relating a network and are therefore better able to detect intrusions. The initial results of our method show considerable promise. In the next phases of the project, we will focus finishing a POC implementation of our design principles and functional requirements. We also look to publish the results of our POC implementation and the effectiveness of the system as a whole in subsequent reports aimed at the Future SOC Lab.

References

[1] Sebastian Roschke, Feng Cheng, Robert Schuppenies, and Christoph Meinel: Towards Unifying Vulnerability Information for Attack Graph Construction , in Proceedings of 12th Information Security Conference (ISC'09), Springer LNCS, Pisa, Italy, pp. 218-233 (September 2009).

[2] Christoph Meinel, Andreas Polze, Alexander Zeier,Gerhard Oswald, Dieter Herzog, Volker Smid, Doc DErrico and Zahid Hussain (Eds.): Proceedings of the Fall 2010 Future SOC Lab Day , Technical Report of Hasso-Plattner-Institute, Heft 42 (2011).

[3] Sebastian Roschke: Towards High Quality Security Event Correlation Using In-Memory and Multi-Core Processing , PhD Thesis, Hasso Plattner Institute at University Potsdam (May 2012).

[4] David Jaeger: Monitoring in Scenario-based Security Experiments , Master Thesis, Hasso Plattner Institute at University Potsdam (August 2012).

[5] The CEE Board: CEE Overview. Available from: http://cee.mitre.org/docs/overview.html.

[6] Common Platform Enumeration:. Available from: http://cpe.mitre.org/.

Exploiting Heterogeneous Architectures for Algorithms with Low Arithmetic Intensity

Fahad Khalid
Hasso-Plattner-Institute, University of Potsdam
14482 Potsdam, Germany
fahad.khalid@hpi.uni-potsdam.de

Andreas Polze
Hasso-Plattner-Institute, University of Potsdam
14482 Potsdam, Germany
andreas.polze@hpi.uni-potsdam.de

Abstract

The Scientific Computing community has witnessed a significant increase in applications that utilize GPUs as accelerators; complementing CPU based processing. However, the feasibility of porting an algorithm to the GPU is dependent on a complex combination of factors that include arithmetic intensity, data access patterns and data reuse. E.g. algorithms with very low arithmetic intensity are less likely to gain performance if implemented on a GPU.

In this report, a problem from Systems Biology is taken as a case study for utilizing heterogeneous computing to improve the performance of an algorithm with low arithmetic intensity. It is shown that the Map-Reduce structural pattern for parallel applications can be used to split the algorithm into two phases; where the Map phase can be optimized for a GPU implementation, while the Reduce phase can be efficiently implemented on the CPU. The result is a significant performance gain over a serial CPU-only implementation. The results presented were obtained by running the algorithms on the FutureSOC hardware.

In addition, the idea of Constraint-based Adaptive Memory Management is introduced; targeted towards combinatorial algorithms.

1 Motivation

In recent years, research community in the High Performance Computing (HPC) and Scientific Computing sectors has witnessed an increasing application of Heterogeneous Computing [1]. The term Heterogeneous Computing (as used in this document) refers to the design of applications that can harness the power of both the CPU and the GPU for problems amenable to massive parallelism.

The rapid adaption of the scientific community to Heterogeneous Computing is often attributed to the GPUs' capability to perform massive amounts of arithmetic operations, at very high speeds. Majority of the silicon chip area in a GPU is dedicated to a number of arithmetic processing units. This makes it possible for the GPU to execute a very large number of arithmetic instructions in parallel. However, since most of the chip area is committed to arithmetic processing, only small amounts of low latency memory resides on-chip. Therefore, a memory hierarchy (similar to the traditional CPU based systems) is employed, but with very small sizes for low latency memory as compared to the CPU.

A significant number of algorithms have been successfully ported to GPUs, attaining up to 100x speedup over serial execution on a CPU [2]. However, due to the aforementioned architectural limitations (w.r.t. low latency memory size), this success holds only for a certain set of algorithms that can take full advantage of the GPU architecture. This set of algorithms typically shows one important execution characteristic; i.e. high arithmetic/numerical intensity (compute-to-memory access ratio). The most common example for such an algorithm is matrix-matrix multiplication [3].

Nevertheless, even for algorithms with high arithmetic intensity, porting and optimizing the algorithm is a tedious task that requires investing a significant amount of time and effort. Moreover, several important algorithms have low arithmetic intensity e.g. Sparse Matrix-Vector Multiplication [4]. With the advent of Big Data revolution, the significance of such algorithms is further amplified; since this involves algorithms associated with processing of very large datasets. For such algorithms, CPU-only architectures (equipped with the much larger low latency memory) would appear more suitable.

In the context of Heterogeneous Computing, the above considerations raise the following important question:

Is it possible to effectively utilize Heterogeneous Computing for algorithms with low arithmetic intensity? Or must such algorithms be executed on CPU-only systems?

In the sections to follow, the above posed question is approached by looking at a low arithmetic intensity algorithm from the domain of Systems Biology. The

algorithm has already been successfully parallelized on both shared-memory [5] and distributed-memory [6] CPU based architectures. Here, a heterogeneous architecture based parallelization is presented.

2 Heterogeneous Computing for Enumeration of Elementary Flux Modes

The set of all Elementary Flux Modes (EFM) represents the complete set of minimal pathways in the metabolic network of an organism [7]. Under steady-state conditions, the problem of enumerating EFMs is mathematically equivalent to the enumeration of extreme rays of polyhedral cone [8]. The Nullspace algorithm [9] is known to be the most efficient algorithm for EFM enumeration. It consists of the following steps:

1. Solve a system of linear equations to get the Kernel matrix
2. Process the Kernel matrix to remove redundancies and permute it so that it is feasible for further processing
3. For each row in the Kernel matrix (EM Matrix):
    a. Generate Candidate EM vectors
    b. Verify elementarity using Rank tests
    c. Update the EM matrix

The most compute intensive part of the algorithm is Candidate Generation. Following is the pseudocode for the serial candidate generation algorithm [6]:

- Input: Matrix A, Matrix B – where both are bit matrices
- Output: Candidate column pairs of the form:
    - $\{(a,b) | a \in A \text{ and } b \in B\}$
- *For each column in MatA*
    - *For each column in Mat*B
        - $candidate = col(A) \vee col(B)$
        - $nonZeros = popCount(candidate)$
        - $If\ (nonZeros \leq$ maxNonZerosAllowed)
            - *Keep the index pair corresponding to* (a,b)

Note: Index values are retrieved from another pair of arrays available as input

The core of the algorithm is based on an element-wise binary operation (bitwise OR) between the two input matrices. The $popCount()$ function is implemented as a lookup operation that computes the *Hamming weight* i.e. the number of set/on bits in the vector. Let the size of Matrix A be m, and size of Matrix B be n, then the total number of binary operations is $m \times n$. In the worst case, this leads to a result matrix that grows quadratically as a function of input size.

As can be inferred from the above description, the serial Nullspace algorithm has very low arithmetic intensity. In the sections to follow, the experience of parallelizing the Nullspace algorithm for execution on NVIDIA GPUs is presented.

2.1 Parallel Candidate Generation Model for GPU

Since the generation of each candidate vector is independent of the others, the parallelization strategy used is one where each thread computes a single candidate vector. This is a data parallel model that results in a massively parallel GPU application.

2.2 A Na ve Kernel

Following is the pseudocode for the kernel based on memory partitioning with the given index algebra:

- $indexMatC = (blockIndex \times blockDim + threadIndex) + (gridId \times gridDim \times blockDim)$
- $uniqueId = indexMatC + (partitionId \times partitionDim)$
- $indexMatA = (uniqueId\ /\ period) \times columnLength$
- $indexMatB = (uniqueId\ \%\ period) \times columnLength$
- $candidate = MatA[indexMatA] \vee MatB[indexMatB]$
- $nonZeros = popCount(candidate)$
- $result[indexMatC] = (nonZeros \leq maxNonZeros)$

Given the massively parallel nature of the GPU, it would appear that the above kernel would perform magnitudes faster than the serial CPU-only algorithm. The results, however, show that the GPU performs even worse than the serial CPU-only code. Figure 1 shows that the GPU performance degrades with the increase in the input size. Overall, the serial CPU-only implementation outperforms the naïve GPU implementation.

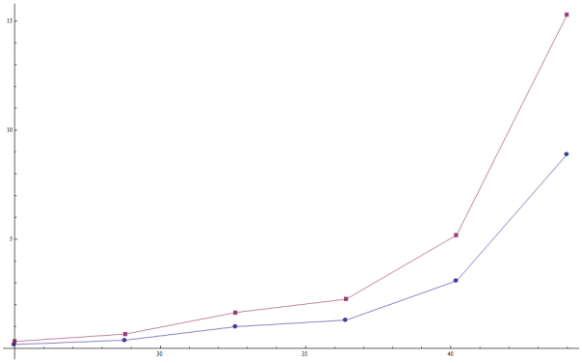


Figure 1: Performance comparison between the serial CPU-only implementation (blue) and the na ve GPU code (red). X-axis represents iterations of the Nullspace algorithm, which correspond to the growing input si e. Y-axis represents the time taken in seconds.

An analysis of the GPU kernel shows that every column pair results in a write to the result matrix; which translates to a total of $m \times n$ global memory write operations. Given that each result value is a 64-bit integer, this results in a very large result array. Therefore, most of the time is spent in transferring the result data structure between the device and the host.

2.3 Employing the Map-Reduce Structural Pattern for Parallel Applications

In the aforementioned pseudocode for the GPU kernel, the final step is a comparison that results in a Boolean value i.e. '0' or '1'. However, as per the serial version of the algorithm, this is just an intermediate step that eventually leads to fetching and storing the index pair corresponding to the input elements. The kernel has been designed in a way so that only the Boolean value is computed on the GPU, and the index pair computation is left as a post-processing step to be performed on the CPU. This division of work between the GPU and the CPU corresponds to the *Map-Reduce structural pattern* [11] with the following two phases:

- *Map* → The GPU kernel. Generates candidates and stores Boolean values as the result. Each result value represents the decision whether the pair of input vectors should be stored.
- *Reduce* → A post-processing step performed on the CPU. Parses the result data structure populated by the GPU. For all values where the decision is positive, the corresponding pair of indices is fetched and stored in a dynamic data structure. This phase is implemented as a shared-memory parallel procedure.

One of the benefits of implementing a heterogeneous Map-Reduce pattern is that the GPU kernel is relieved from executing further memory intensive index fetch operations. Also, the massively parallel nature of a GPU application results in concurrency issues (mutual exclusion) if a dynamic global data structure is used to store the resulting index values. Therefore, relegating such operations to the CPU, results in a relatively efficient kernel.

2.4 Introducing the Compression Factor

A significant advantage of employing the Map-Reduce pattern is the possibility to exploit the Boolean nature of the result computed per thread, in order to reduce the number of global memory write operations by a constant factor. Since each result value is a Boolean, instead of storing the result as an integer, it is stored as a single bit. Therefore, with $com\ ression Factor = 64$, 64 results can be stored in one element of the result array (assuming 64-bit unsigned integers are being used in the result data structure). This makes it possible to compute $com\ ression Factor$ number of candidates per thread.

As a result, size of the result data structure for the Map phase is reduced by a factor of $com\ ression Factor$. Following are two major advantages of the reduction in result size:

1. Previously, the time spent in data transfer between the device and the host overshadowed the time spent on computation. With the $com\ ression Factor$ scheme, the balance is shifted in the favor of computation time i.e. time spent in device-host-device data transfers is negligible in comparison to the time spent in kernel execution. This results in better utilization of the GPU resources.
2. Much more efficient user-managed caching [12] schemes can now be employed.

The performance results after implementing the Map-Reduce pattern with $compressionFactor$ are shown in Figure 2. The heterogeneous algorithm outperforms the serial CPU version by achieving a relative speedup of ~3.5x. Please note that this speedup does not include all the typical performance optimizations [13] applied to CUDA[1] code. Efforts to further improve the performance by applying such optimizations are underway. These optimizations are well known within the CUDA programmers' community, and therefore, will not be discussed in detail here.

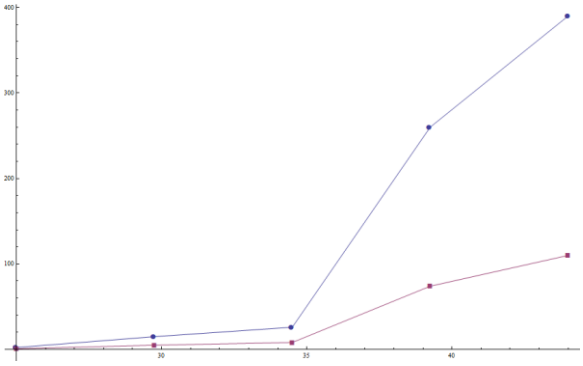


Figure 2: Performance comparison between serial CPU-only implementation (blue) and the heterogeneous Map-Reduce based implementation with compressionFactor (red). X-axis represents iterations of the Nullspace algorithm, which correspond to the growing input si e. Y-axis represents the time taken in seconds.

2.5 Brief Overview of Planned Optimi a-tions

Following is a brief discussion of further optimizations applicable to the heterogeneous implementation:

- *Coalesced Global Memory Access and Result Caching:* Due to the use of $com\ ression Factor$, access to the global memory is no longer coalesced from threads in the same block. An improved index algebra, coupled with a cooperative caching mechanism is being implemented to ensure coalesced accessed to the global memory. This is expected to result in a significant performance improvement.
- *Tiled Input and Caching:* Tiling can be used to implement efficient caching for the input data. However, it is not clear at the moment if this will result in a significant performance gain.
- *Asynchronous Map-Reduce:* The current implementation executes the Map and Reduce phases

[1] NVIDIA CUDA, http://www.nvidia.com/cuda

in sequence. A multi-threaded asynchronous implementation may lead to better overall performance.

In addition to the above mentioned possible optimizations, several typical CUDA optimizations will be implemented.

3 Constraint-based Adaptive Memory Management

The Candidate Generation algorithm (as presented in the previous Section), is combinatorial in nature. Such combinatorial algorithms often deal with the exploration of very large solution spaces. This exploration process can lead to the intermittent generation of a very large number of combinations, only some of which might eventually be required by the algorithm.

The intermittent *combinatorial explosion* results in excessive memory consumption, which is only temporary. Nevertheless, if the consumption exceeds the amount of available physical memory, the program can crash.

A possible approach to control the memory consumption behavior of such an algorithm is to apply an upper-bound on the maximum amount of memory that can be consumed by a process at a given time. In the case of EFM enumeration, this means that each thread running in parallel must not process more than a certain number of candidate vectors per iteration of the Nullspace algorithm.

We intend to design a constraint definition system, coupled with a constraint enforcement runtime that makes it possible to define resource constraints for individual processes/threads. This requires altera-tion of the program flow, so that more iterations are executed with less data per iteration. Formally, it can be defined as the problem of *Optimal Dynamic Partitioning of Program Flow*.

At this point, the proposal is in the idea phase. We intend to develop it further into a prototype implementation during the next phase of our FutureSOC Lab project.

4 Conclusions

In order for software designers to take full advantage of heterogeneous computing, generic patterns in low arithmetic intensity algorithms need to be identified that can make it possible to gain significant performance gains over multi-threaded shared-memory CPU-only implementations. A positive result in this direction is presented by utilizing the Map-Reduce structural pattern to improve algorithm efficiency for an algorithm from Systems Biology.

A set of further optimizations will be implemented for this application, which will provide further insight into the full potential for utilizing Heterogeneous Computing for this particular problem. Moreover, research will be carried out in order to see if the results can be generalized for a broader class of algorithms.

Moreover, we intend to develop a constraint-based adaptive memory management system. This is expected to result in an effective and efficient memory consumption behavior for combinatorial algorithms.

References

[1] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A Survey of General-Purpose Computation on Graphics Hardware", Computer Graphics Forum, vol. 26, pp. 80-113, 2007.

[2] V. W. Lee, C. Kim, J. Chhugani, M. Deish-er, D. Kim, A. D. Nguyen, N. Satish, M. Smelyan-skiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU", presented at the Proceedings of the 37th annual international symposium on Computer architecture, Saint-Malo, France, 2010.

[3] A. Buluç, J. R. Gilbert, and C. Budak, "Solving path problems on the GPU", Parallel Com-puting, vol. 36, pp. 241-253, 2010.

[4] J. D. Davis and E. S. Chung, "SpMV: A Memory-Bound Application on the GPU Stuck Be-tween a Rock and a Hard Place", Microsoft Research Silicon Valley, Technical Report14 September 2012 2012.

[5] M. Terzer and J. Stelling, "Accelerating the Computation of Elementary Modes Using Pattern Trees", in Algorithms in Bioinformatics. vol. 4175, P. Bücher and B. Moret, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 333-343.

[6] D. Jevremović, C. T. Trinh, F. Srienc, C. P. Sosa, and D. Boley, "Parallelization of Nullspace Algorithm for the computation of metabolic pathways", Parallel Computing, vol. 37, pp. 261-278, 2011.

[7] S. Schuster and C. Hilgetag, "On elementary flux modes in biochemical reaction systems at steady state", J. Biol. Syst, vol. 2, pp. 165–182, 1994.

[8] J. Gagneur and S. Klamt, "Computation of elementary modes: a unifying framework and the new binary approach", BMC Bioinformatics, vol. 5, p. 175, 2004.

[9] C. Wagner, "Nullspace Approach to Deter-mine the Elementary Modes of Chemical Reaction Systems", The Journal of Physical Chemistry B, vol. 108, pp. 2425-2431, 2004.

[10] S. Klamt and J. Stelling, "Combinatorial Complexity of Pathway Analysis in Metabolic Networks", Molecular Biology Reports, vol. 29, pp. 233-236, 2002.

[11] K. Keutzer, B. L. Massingill, T. G. Mattson, and B. A. Sanders, "A design pattern language for engineering (parallel) software: merging the PLPP and OPL projects", presented at the Proceedings of the 2010 Workshop on Parallel Programming Patterns, Carefree, Arizona, 2010.

[12] M. Silberstein, A. Schuster, D. Geiger, A. Patney, and J. D. Owens, "Efficient computation of sum-products on GPUs through software-managed cache," presented at the Proceedings of the 22nd annual international conference on Supercomputing, Island of Kos, Greece, 2008.

[13] NVIDIA, "CUDA C BEST PRACTICES GUIDE," Design GuideJanuary 2012 2012.

[14] J. Mora, "Do Theoretical FLOPS Matter for Real Application Performance?", presented at the HPC Advisory Council Spain Workshop, 2012.

Using In-Memory Computing for Proactive Cloud Operations
Future SOC Lab Report October 2012

Felix Salfner, Marcus Krug
SAP Innovation Center, Potsdam

Peter Tröger, Eyk Kny
Hasso Plattner Institute, Potsdam

Abstract

The management of cloud computing infrastructures on operator side is a true challenge. An ever-growing number of servers, the heterogeneity of software, necessary elastic load handling, energy consumption and other non-functional aspects have to be taken into account – continuously and in an adaptive fashion.
Proactive cloud operations tries to trigger preventive maintenance activities when some part of the system is about to enter an erroneous state. Examples for such activities are administrator alarming and automated load migration. We combine this approach with the idea of semi-automated root cause analysis to reduce time-to-repair and improved availability. In-memory computing provides a well-suited technology for this. We have implemented a prototype to analyze feasibility of the approach and analyzed data sets from HPC clusters.

1 Introduction

Cloud computing is currently one of the predominant trends in computer science and IT industry. Its potential impact on IT infrastructures and software platforms cannot be underestimated: The cloud paradigm will change the way how IT companies make business as well as how end-users (private and corporate) perceive software and IT. It moves the burden of IT infrastructure management and operation away from users, and shifts it to specialized providers that can guarantee fast, reliable, and secure operation of the cloud infrastructure.
Satisfying the users' expectations turns the management of cloud computing infrastructures into a true challenge. An ever-growing number of servers, the heterogeneity of software, multiple interacting mechanisms to elastically react on changing load, the consideration of energy consumption and other non-functional aspects have to be taken into account. Operating such systems requires intelligent management tools to deliver the guaranteed service-level to the user in a profitable way.
One of the key features of system monitoring at the scale of a cloud computing center is to aggregate low-level monitoring events in order to present only critical events to the operation personnel. To achieve this goal, current monitoring systems rely on temporal as well as spatial correlation mechanisms and infrastructure-centric rule-based aggregation. These techniques usually have a snapshot-like view on the system and ignore problems that evolve in the system over a longer period of time. In our project, we focus on such evolving phenomena and explore a new monitoring technique that is based on computing correlations between monitoring signals in order to identify the spreading of problems within the system.
This report summarizes the second project phase and documents our project results. In summary we have:

- optimized the monitoring and computation infrastructure for anomaly detection, which was developed in the first phase
- optimized the computation of signal correlation using SAP's in-memory database HANA,
- built a web-based front-end for interaction with structure-of-influence graphs
- developed a new graphical representation for correlation of hierarchical monitoring signals
- investigated ways how to use structure-of-influence graphs for the prediction of upcoming failures

2 Approach

Oliner et al. [4] have presented an algorithm to infer shared influence in complex production systems. The algorithm is based on anomaly signals, which are real values between zero and one indicating how much a measurement signal deviates from "normal behavior". Anomaly signals are obtained from monitoring signals by applying a transformation function that encodes local domain knowledge. By computing the correlation between any pair of anomaly signals, the "spreading" of anomalies across the system can be quantified. More precisely, if there is a high correlation between

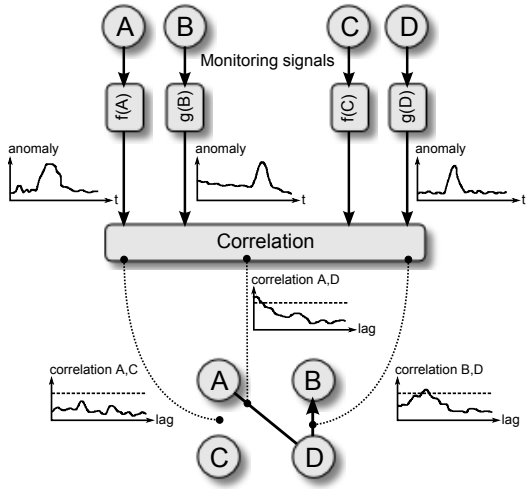


Figure 1. Creation of structure-of-influence graphs.

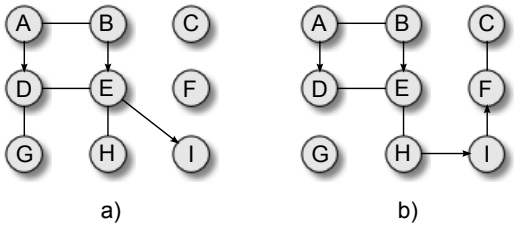


Figure 2. Prediction by computing similarity between two structure of influence graphs: Currently observed (a) which resembles the beginning of a previous failure occurrence (b).

the anomaly signals of two components, we can speculate that the abnormal states of both components are related.

The time lag at which the correlation reaches its maximum can be used to determine the temporal interdependency between the two components, i.e., to identify, which component is the source (or initiator) and which one is the target of the problem (suffering from the anomaly of the source component). For further analysis, the interdependencies can be visualized in a so-called *structure of influence graph (SIG)*. The nodes of the SIG are monitoring signals and an edge is added to the graph if there is a significant correlation between the two signals. The edge is directed, if the time lag of maximum correlation is larger than some threshold. Figure 1 depicts the procedure of generating a SIG.

The primary use case for SIGs is root cause analysis. Given a SIG, which represents how anomalies have spread throughout the system, the initiating first anomaly can be detected and the component, to which the anomaly signal belongs, appears to be a good candidate for further inspection. Please note that such analysis is based on correlation, which can but not necessarily does represent causation.

3 Predicting Failures

SIGs have been used successfully for root cause analysis in complex systems, such as high-performance computing centers. As an extension, we propose to use extended SIGs for an early detection of anomaly spreading that might eventually turn into a failure. We accomplish this by checking whether a currently observed SIG might develop into a SIG that is known to have led to a system failure in the past.

As can be seen in Figure 2, this is not an easy task since two graphs will probably never be equal, i.e., there can be missing or additional edges, the spreading of anomalies might exhibit short "detours" or "shortcuts", and even if the edges are the same their weight tuples will most likely differ. For these reasons, the comparison of SIGs has to be based on some metric of similarity. Examples include:

- Gromov–Hausdorff distance, which builds on a mapping of graphs to metric spaces and measuring how far the two spaces are from being isometric.
- Sequence-based approaches such as proposed in [5] that build on the assumption that two graphs are similar if they share many sequences of vertices and edges.
- Scoring-based approaches that score the existence (and similarity of the weight tuple) of an edge in both graphs.
- Graph isomorphism algorithms or their approximations (see, e.g., [1]).

Computing such similarity metrics is computationally challenging. Hence, computational complexity will play a significant role in the evaluation of metrics. The bi-variate structure of edge weights might put additional complexity to the evaluation. Additional aspects to be considered include the efficient storage of SIGs or their signatures in a database of failure SIGs as well as efficient ways to search for failure SIGs matching the currently observed SIG. We also investigate ways to simplify the graph by, e.g., collapsing cliques of vertices, clustering, etc.

In addition to the search for the best similarity metric, there are two general approaches to deal with the aspect of spreading. In the first approach, we compare the currently observed SIG to stored SIGs that were previously saved at system failure time. The second approach compares the current SIG *development*, i.e., a sequence of several successive SIGs, to stored SIG developments from previous failure cases.

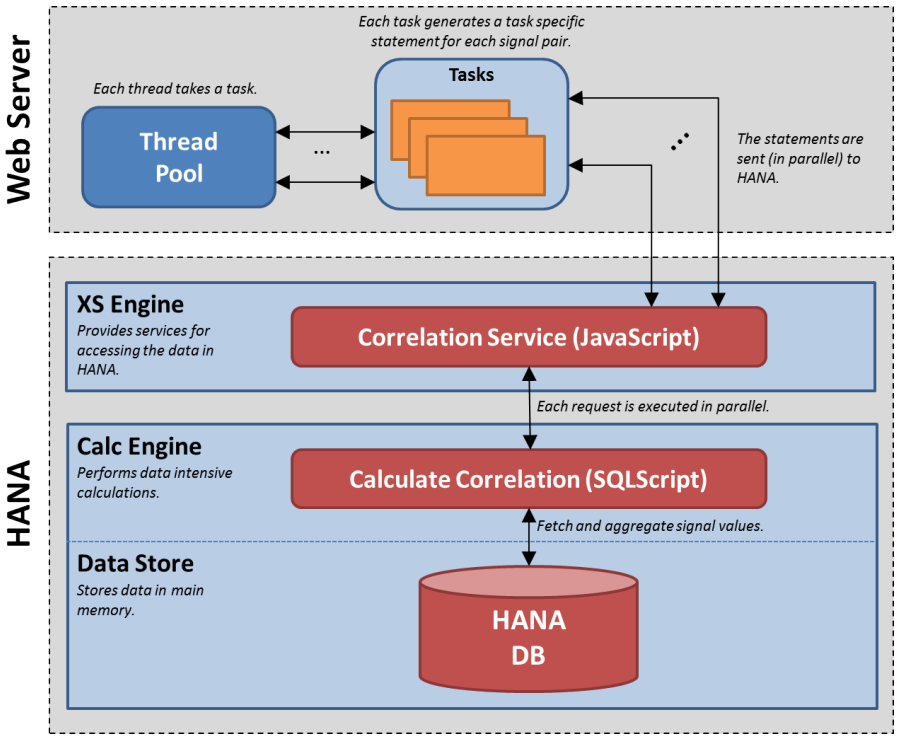


Figure 3. Overview of the HANA-based correlation engine.

4 A Use-case for In-memory Computing

Investigating the root cause of a problem that has occurred in a cloud environment frequently requires to analyze monitoring data and to compare the system behavior of the current faulty case to system behavior at other fault-free times. In order to do this, an efficient analysis framework has to enable the operator to quickly move forth and back in the monitoring data, to drill down at specific points in time or to get an overview on higher levels of granularity. These requirements make root cause analysis similar to business data exploration. In-memory database technology is a key enabling technology to perform on-the-fly data analysis — and we believe that it has the potential to also boost efficiency for root cause analysis. For this reason, we implemented the correlation algorithm with on-the-fly data aggregation on the database level. Our approach builds on SAP's HANA in-memory database. This means that all computationally complex and data-intensive operations have been implemented in SAP's computational database language *SQLScript*. Figure 3 shows the setup. The implementation was supported by hardware of the FutureSOC lab.

5 Dealing With Complex SIGs

SIGs can very quickly become too large to be understandable. Even medium-sized systems can have several thousand monitoring variables. The number of monitoring variables becomes prohibitive rather quickly as the number of correlations that need to be computed grow quadratically. However, in real-world systems measurement signals (and hence also anomaly signals) are organized in a hierarchical fashion. For example, application queue length measurements belong to some application, which runs on a specific server, which resides in some rack in the data center. One example for such a signal hierarchy is shown in Figure 4.

The present hierarchy of signals can be leveraged for four general ways to reduce complexity:

1. Selection of anomaly signals
2. Aggregation of anomaly signals
3. Algorithmic simplification
4. Hierarchical visualization

In the first approach, complexity is reduced simply by selecting a subset of anomaly signals, for which a SIG is computed and for which a graph is rendered. For example, a SIG can be computed only for anomaly signals that belong to one specific server. Another example is the selection of temperature monitoring signals across the entire system. In order to facilitate the user of a SIG-based tool to perform such a selection the tool needs to provide facilities to navigate and slice the signal hierarchy.

In the second approach, data of anomaly signal is condensed via aggregation. Aggregation can be performed either on the time axis, which simplifies computation of each single correlation but does not reduce

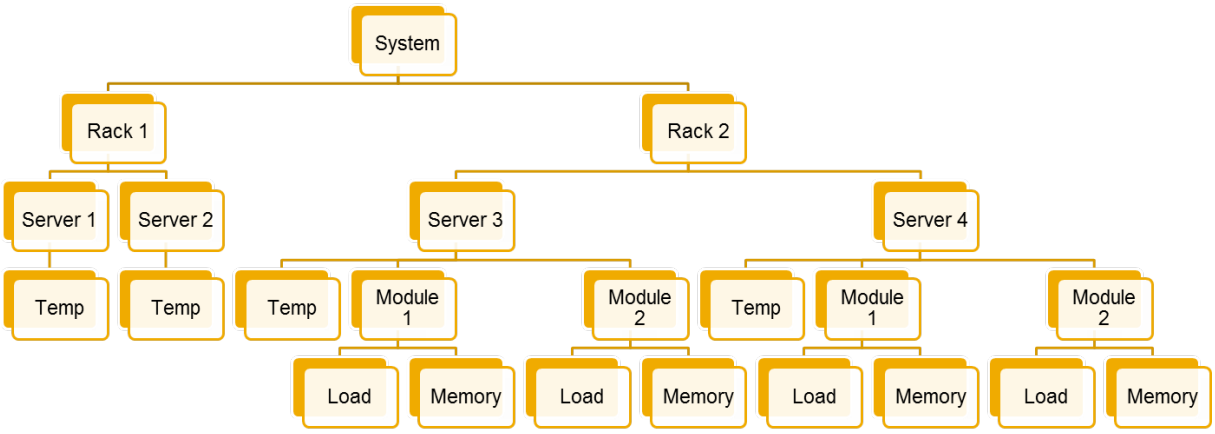


Figure 4. Example hierarchy of monitoring signals

the number of correlations and it does also not simplify the resulting SIG. A second way of aggregating anomaly signals again makes use of the hierarchy: For example, a SIG can be computed on rack level. That means that for example all temperature anomaly signals that belong to one rack are combined into one temperature anomaly signal. The same approach can also be applied to anomaly signals that do not derive from the same domain: for example, all anomaly signals of a rack, i.e., memory, load, and temperature signals, are aggregated on rack level. This is possible because anomalies are normalized to the range $[0, 1]$.

Algorithmic simplifications include various machine learning techniques such as clustering, principal component analysis (as has been proposed by [3]), or stochastic techniques. Another promising candidate are frequency-based techniques that operate in the frequency domain rather than the temporal domain, which would also speed up computation of correlations tremendously. More advanced topics include network exploration, path extraction, etc.

Although not reducing the computational complexity, visualizations of the SIG play an important role in enabling the user to get maximum information out of a SIG. An example is shown in Figure 5. In Graph (a) some clusters and shared influence can be identified. However, it is almost impossible to defer any information from Graph (b). This was the motivation for investigating new ways to visualize SIG data. Following some discussions with the research group by Prof. Döllner, we had the idea to adapt a hierarchical edge bundles visualization [2], as it is also used in software visualization. An example for such a graph is shown in Figure 6. In such a graph, the hierarchy of anomaly signals is indicated by rings, where the innermost elements are single anomaly signals (leafs in the tree of Figure 4), and the outer rings combine elements to larger units. For reasons of confidentiality, the labels in the graph have been blurred. The correlation itself is indicated by vector bundles in the middle of the circle. In the figure, we used a color gradient from green to red to indicate the direction of the arc. Future versions will feature grading to transparency and line thickness to indicate strength of a correlation.

6 Conclusions and Next Steps

Operating a cloud computing infrastructure at minimum cost while at the same time offering service at a guaranteed level is a challenge. This project attempts to adapt a new correlation-based monitoring data analysis method to the cloud context. One major aspect is to include in-memory databases to facilitate on-the-fly data analysis and exploration. A second major focus of the work is on reducing computational complexity using advanced pre-processing techniques and to make structure-of-influence graphs easier to consume by exploring new graphical representations. Future work will focus on the exploration of simplification techniques as well as the usage of structure-of-influence graphs for predicting upcoming problems.

References

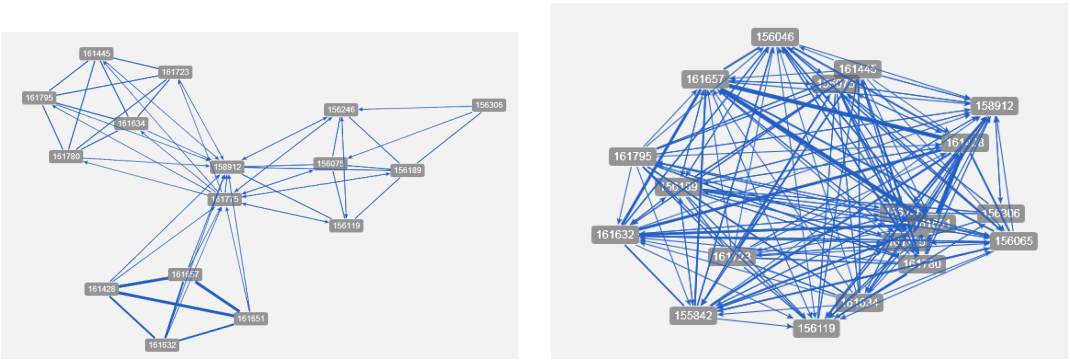
[1] Matthias Dehmer, Frank Emmert-Streib, and Jürgen Kilian. A similarity measure for graphs with low computational complexity. *Applied Mathematics and Computation*, 182:447–459, November 2006.

[2] Danny Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *Transactions on Visualization and Computer Graphics*, 12:741–748, 2006.

[3] Adam Oliner and Alex Aiken. Online Detection of Multi-Component Interactions in Production Systems. In *Dependable Systems and Networks*, pages 49–60. IEEE, 2011.

[4] Adam J. Oliner, Ashutosh V. Kulkarni, and Alex Aiken. Using Correlated Surprise to Infer Shared Influence. In *Dependable Systems and Networks*, pages 191–200. IEEE Computer Society, 2010.

[5] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. Web Graph Similarity for Anomaly Detection. Technical Report 2008-1, January 2008.



(a) (b)

Figure 5. Two examples for SIG visualizations.

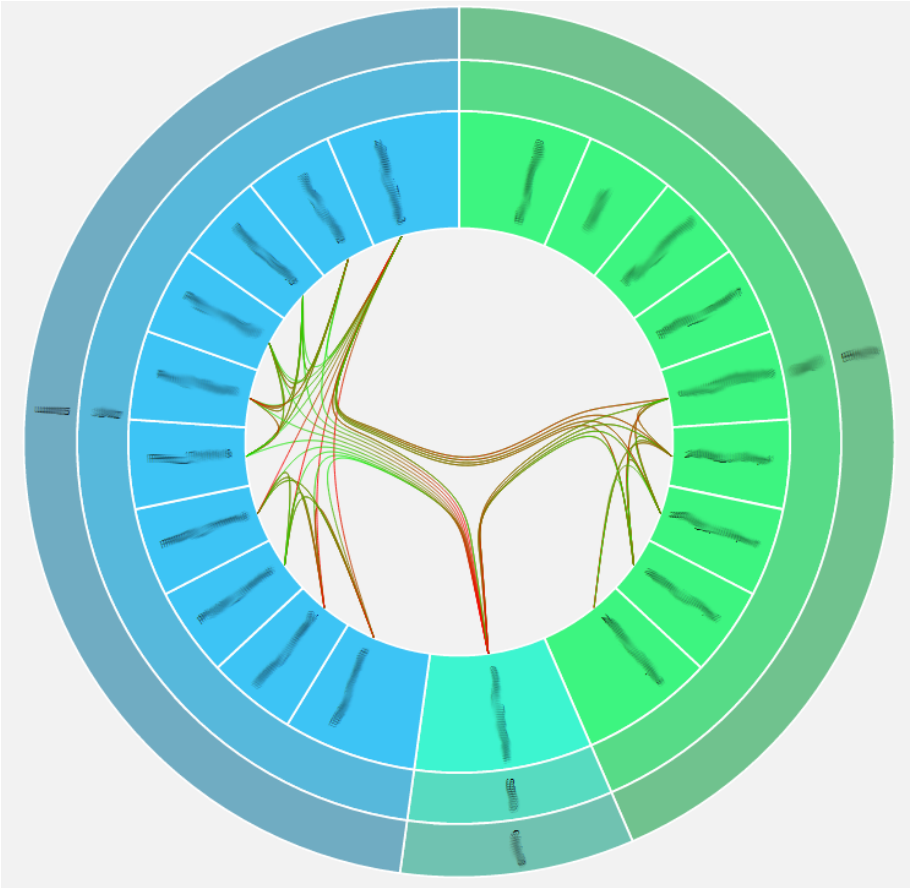


Figure 6. Sunburst visualization of SIGs

Evaluation of Multicore Query Execution Techniques for Linked Open Data

Ahmed Imad Aziz, Heiko Betz and Kai-Uwe Sattler

Database and Information Systems Group
Ilmenau University of Technology
{*first.last*}@tu-ilmenau.de

Abstract

LODcache – a project of the Database & Information Systems Group at Ilmenau University of Technology – aims at developing an in-memory database system for managing Linked Data in RDF format and processing SPARQL queries. One of the main goals of this project is to investigate techniques for exploiting modern hardware architectures such as many-core CPUs, cache hierarchies, large main memory, and co-processing.
In this context, the specific focus of the project performed at the HPI Future SOC Lab was to evaluate and improve newly developed indexing and parallelization techniques. In this paper, we give a brief report on the main results of this work.

1. LODcache Architecture

The idea of the LODcache system is to build a Memcached-like system for Linked Open Data which fetches data from the sources, materializes them in an in-memory cache database and evaluates queries on this database only. Furthermore, query processing in LODcache leverages features of modern hardware architectures: it follows the in-memory processing paradigm using hash indexes and stores data in a compressed and columnar-oriented way. In addition, it uses the Intel Thread Building Blocks library to exploit parallel processing techniques on multicore architectures.
The architecture of LODcache comprises two main components: a storage layer and a query engine. At the storage layer triple data are organized in chunks of fixed size where they are stored column-wise. Each triple is stored and indexed three times: using a hash-based subject index, predicate index, and object index. Furthermore, all triples on the chunks assigned with an index are stored in the order of the index. Finally, the triple components encoded using a dictionary compression scheme.

The query engine provides the standard set of query operators for implementing a SPARQL algebra including different join implementation to evaluate basic graph patterns (BGP).
As an in-memory database system, LODcache faces in principle the problems of the memory wall [1] such as latency, limited memory and bus bandwidth. These problems become even worse when there is more than one CPU sharing the same memory, and the solution applicable to the single processor system is less effective when considered under multiprocessor or multicore assumptions. Therefore, parallelization strategies for efficient query processing on big datasets need to deal with various issues such memory architecture effects, cache obliviousness but also with data skew and load balancing issues.

2. Parallel LODcache

In database query processing different parallelization strategies exist on the query level: inter- and intra-query parallelism. Inter-query parallelism allows multiple queries to run in parallel, while intra-query parallelism considers the parallel execution of a single query. In our work, we have chosen two different intra-query strategies. The first approach is based on *parallel query branches* by duplicating the query execution tree multiple times and executes all branches in parallel. The second strategy is an intra-operator strategy for processing joins called *pipeline join*. Here, the available input data is split into partitions and assigned to join works running in parallel.

2.1. Parallel Query Branches

The idea of parallel query branches is shown in Figure 1. The right-hand side shows a sequence of operators which forms an operator tree. The left-hand side shows the same query but with three parallel query branches which are coordinated by the `parallel_do` operator as the root node. However, the parallel version is not an exact copy of

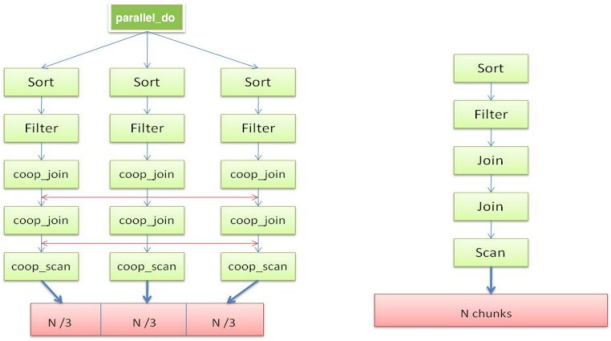


Figure 1. Parallel query branches vs. serial query execution plan

the original execution plan: the join and scan operators are replaced by corresponding `coop_join` and `coop_scan` implementations.
Parallel query branches can be easily derived from serial query execution plans just by replacing the join and scan operators, replicating the whole plan to multiple branches, partition the input, and adding a new root operator `parallel_do`. Finally, each branch is assigned to a single physical thread.

The `coop_scan` Operator. The `coop_scan` operator is quite similar to the normal scan operator. Every `coop_scan` operator receives one part of the available input chunks for initial load balancing. Due to possible data skew, the chunks do not all contain the same number of triples and in the end, the sum of all triples over all assigned chunks can be different between different branches. To address this issue, a *work stealing* strategy is used.
After processing all available chunks, the `coop_scan` operator asks his right-hand neighbor from another branch to share work. If the neighbor has more than one chunk not processed yet, the asker "steals" some of these chunks. Otherwise, it keeps the last chunk and nothing is transfered. In this case, the next branch is asked until none of the neighbors has any chunk not processed yet. For synchronization purposes each chunk contains only a single atomic variable for indicating the processing state.
Using the same physical thread for processing data reduces the operation system overhead. Only the requested data between the local memory and processing unit must be transfered.

The `coop_join` Operator. The default `index_join` processes its outer input relation by looking up the join value in an index for the inner relation. Here, a problem is that it is unknown how many chunks have to be processed from the inner relation, i.e. the workload per chunk is not known which makes a direct mapping of input workload to worker threads unfair. To address this, a strategy like work stealing would be useful, but a direct application would result in significant overhead due to the finer granularity in the join operator.
A solution is the so-called `coop_join` operator, which implements *work sharing* between all `coop_join` operators of the same level in different branches. When one operator instance has no more work to do, it asks its siblings for portions of the remaining work; full chunks and parts of a single chunk can be transferred. Here, no explicit synchronization is required, because only the sibling can decide which chunk can be accessed and transferred. On the other hand, the asker has to register in the givers list and the giver has to check periodically if some asker is available. Periodical checking the giver list is negligible compared to the waiting time between sending a request and receiving a response. The overhead can be further reduced, when the asker performs busy waiting instead of thread sleeping.

2.2. Pipelined Join

The `pipelined_join` approach uses the TBB[1] parallelization approach, which is called task- based load balancing. This method splits up the available work into small partitions. Every partition is mapped to a task and finally the task can be mapped to a physical thread. Faster threads have to process more tasks than others. The more tasks and the smaller the partitions, the higher is the load balancing between threads. It is a portable approach for almost every application.
Unfortunately, increasing the number of tasks and decreasing the partition size is not for free, but results in an overhead for each created task. The problem size must be big enough; otherwise the overhead is too huge and the performance will decrease. The number and size of task limits the load balancing between threads if the problem size is too small to be divided into many smaller tasks. There is also no possibility to share work between different tasks.
The goal of the `pipelined_join` operator is to find a way to make use of the task-based load balancing strategy, without the need for any work sharing or stealing. The work task should be complex enough to be parallelized, it needs a dividable nature and finally it should be applicable to our query execution tree. A good place for applying this strategy is the index join operator, which implements an index-based nested loops strategy. The `pipelined_join` operator just creates tasks for processing partitions of input data. However, the challenge is to collect the results of all tasks in a single result chunk. If all tasks have

[1] Intel Threading Building Blocks library is a highly optimized library for implementing parallelized software. It can be found at `http://threadingbuildingblocks.org`.

to access a single chunk list to store their results, we would need to pay the cost of concurrency for each result tuple. If we would allow each task to return its own chunk of result tuples we would end up with a large number of result lists. The implementation of the pipeline join relies on the fact that although we create many tasks we have no more than C tasks active at any moment on the C threads. Thus, we need only C containers to be alternatively mapped to each of the active tasks.

2.3. TBB Scheduler

Assigning tasks to physical threads is done by the TBB Scheduler. For small queries and unbalanced operator trees it is not efficient to use all available processing units of the system. Therefore, a combination of intra- and inter-query parallelization is necessary to really utilize the available cores. We address this issue by providing our own implementation of a TBB scheduler.
The scheduler takes the first query, uses the parallel branches of the operator tree and assigns every branch to a physical thread. Before the last branch is started the scheduler is called again in order to schedule the subsequent query. This query is again split into branches but not started before any free physical thread is available. The process stops when the scheduled query has at least one pending task waiting to start executing. This helps to improve CPU utilization, because we have always tasks ready to run when any thread becomes free when there are enough queries waiting in the queue.

2.4. Boost Mode

Using the default TBB task-based load balancing approach has two disadvantages. The first is the logical creation of tasks by using the TBB scheduler. The second is the assignment of created tasks to threads, which is also handled by the TBB scheduler and needs support from the operating system. The latter one is more expensive than the first one and should be avoided. It happens, when the TBB scheduler does not find any free task to execute. The physical thread is going to sleep. A solution is to deny sleeping by using active waiting. Every thread, which has no work, is waiting active for some given time. If a new query arrives during the waiting phase, it splits up in tasks, which are assigned to waiting threads.

3. Evaluation

In this section, we give a brief overview on the main results of our experimental evaluation. The hardware platform was a *Hewlett Packard DL980 G7-1* server and was provided by the Future SOC Lab of the Hasso-Plattner-Institute (HPI). The server contains 8 Xeon

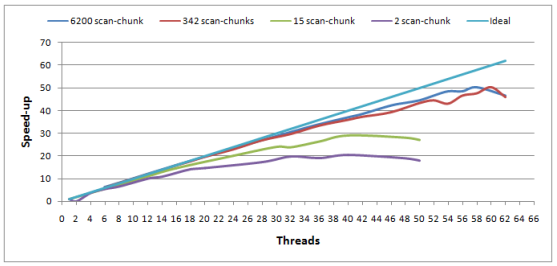


Figure 3. Speedup with respect to different number of chunks and threads.

(Nehalem EX) X7560 CPUs, each with 8 physical and logical cores, and 2048 MB RAM. The total sum of physical cores is 64, with 64 logical cores. Other hardware specifications are 2.26 GHz clock frequency, 32 KB data and 32 KB instruction set per core (L1 cache). The L2 cache provides 256 KB per core and the L3 cache has 24 MB accessible by all cores.
Apart from several micro benchmarks for measuring memory latencies, NUMA effects etc. we have performed several tests with different queries on different data sizes. In the following we present the results for a single query execution plan where we have varied the number of chunks (input size) and the number of parallel working threads to determine the possible speedup. Furthermore, only the join operation is considered, because executing SPARQL on triple stores results in very join-intensive query execution plans and more than 99% of runtime is spent by the join operator. Thus, the result can be generalized for almost all queries which uses joins. More results including the results of the micro benchmarks can be found in [2].

Parallel Query Branches. The parallel query branch experiment uses the `coop_scan` and `coop_join` operators. The speedup graph in Figure 3 shows the speedup variation with respect to different numbers of chunks at scan level. More chunks at scan level results in better speedup, because the distribution among branches is easier. The 2-chunks series gained a speedup of 21 by using 40 threads in parallel. Here, the speedup is achieved mainly by to the `coop_join` operator. The 15-chunks series touches the 30 by using 40 threads. Both 342 and 6200-chunk series starts with quite linear behavior and the deviation from the ideal speedup is not noticeable with less than 32 threads. After 20 respective 28 threads the slope is decreasing. The maximum achieved speedup is 51 with 57 respective 61 threads.
The load unbalancing in terms of runtime between different branches is shown in Figure 4. In the worst case, the unbalance is always less than 7%.
Figure 5 shows the ratio between useful work and overhead on the 342-chunk example. The overhead in-

```
$1 := coop_scan(p-index, "<http://www.w3.org/2000/01/rdf-schema#label>");
$2 := coop_join($1, s-index, _0, =[_1, "<http://...>"]);
$3 := coop_join($2, s-index, _0, =[_1, "<http://...>"]);
$4 := filter($3, >[_8, 845]);
$5 := coop_join($4, s-index, _0, =[_1, "<http://...>"]);
$6 := filter($5, =[_11, "<http://...>"]);
$7 := coop_join($6, s-index, _0, =[_1, "<http://...>"]);
$8 := filter($7, =[_14, "<http://...>"]));
$9 := parallel_do($8, NUM_CORES);
printer($9)
```

Figure 2. Example query execution plan.

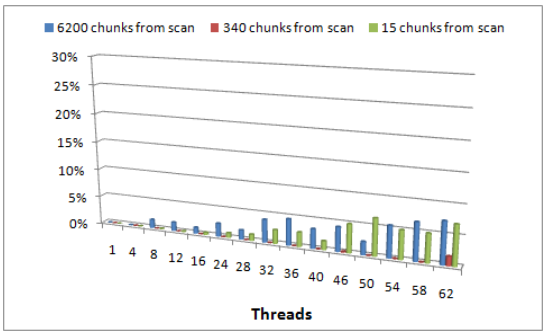


Figure 4. Load unbalancing in terms of runtime.

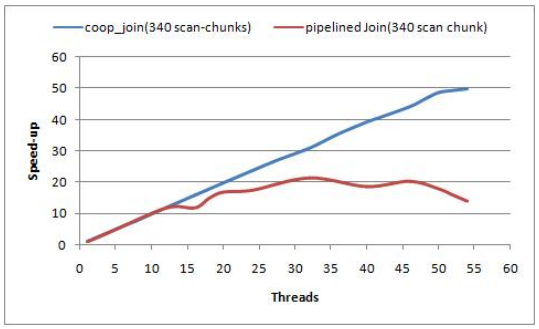


Figure 6. Comparison between both approaches.

cludes the times for accessing the memory and cache synchronization. The average overhead is about 5% for the `coop_join` operator, but it reaches 10% for higher number of threads.

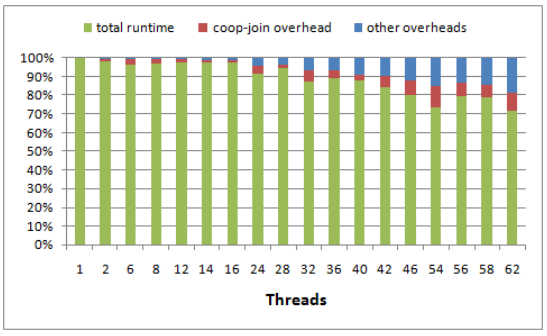


Figure 5. Overhead and useful work distribution.

In summary, our `coop_join` operator achieves a good speedup but has some overhead. Work sharing between different branches with the `coop_join` operator should be used if the number of chunks at the scan level is small or if no waiting queries exist. In case of waiting queries, the `coop_scan` parallelization is more useful by scheduling additional queries. In general work stealing has a smaller overhead than work sharing.

Pipelined Join The pipelined join approach uses the `pipelined_join` operator for join processing. For scanning data, a single thread was used to provide the input data for all `pipelined_join` tasks. A combination of `pipelined_join` and `coop_scan` results in smaller speedup than shown here.
The speedup of the `pipelined_join` operator is shown in Figure 6. In all cases, the speedup does not reach more than 15. Compared to the parallel query branches approach (maximum speedup of 51) the pipelined join approach has shown less useful.

4. Conclusions & Outlook

Today, multicore systems are widely available, but developing software for utilizing these architectures is still a challenging task. Several hardware details have to be considered, otherwise the overhead will reduce the possible speedup of parallelization.
In LODcache we have chosen a parallelization approach based on the ideas of work stealing and work sharing for balancing the work between different threads in case of data skew and keeping the utilization of the CPU cores high. These strategies are implemented in different dedicated query operators such as coop_scan, coop_join and pipelined join. Further-

more, all these operators work in a cache-aware way by processing larger chunks of triples represented in a columnar memory layout.
Using the provided hardware at the HPI Future SOC Lab we were able to evaluate these strategies with big data sets on a CPU with 64 cores. Our evaluation results show a linear speedup up to 28 cores. The maximum achieved speedup is up to 51 by using 60 threads. We found also only a small overhead for synchronization between the threads for work stealing (up to 10% for coop_join) and 7% unbalancing between branches in terms of runtime.
In future work, we plan to further investigate main memory layouts taking the characteristics of NUMA into account. Furthermore, dynamic scheduling of queries, i.e. assigning cores to incoming queries, to improve the query throughput is another issue we will address.

References

[1] Peter A. Boncz, Martin L. Kersten, Stefan Manegold: Breaking the Memory Wall in MonetDB. Commun. ACM 51(12): 77-85, 2008.

[2] Ahmed Imad Aziz: Parallelization Techniques for In-Memory Processing of Linked Data. Master's Thesis, Ilmenau University of Technology, Department of Computer Science and Automation, September 2012.

Next Generation Sequencing: From Computational Challenges to Biological Insight

Cornelius Fischer
cfischer@molgen.mpg.de

Annabell Witzke
witzke@molgen.mpg.de

Sascha Sauer
Nutrigenomics and Gene Regulation
Otto-Warburg Laboratory
Max Plank Institut for Molecular Genetics, Ihnestr. 63-73, 14195 Berlin, Germany
sauer@molgen.mpg.de

Abstract

Next generation sequencing (NGS) is changing the way researchers approach analysis of biological information. However, one of the main bottlenecks in NGS applications is the computational analysis of experimental data. Using the Future SOC Lab resources we established and used a computational pipeline for the analysis of sequencing data that we recently generated. We found that the provided resources worked very robust and fast. This enabled us to move rapidly from raw NGS data to initial biological insights.

1. Project idea

Transcription factors are crucial proteins that are involved in controlling gene expression. Thus, activities of transcription factors determine how cells function and respond to environmental stimuli. Many transcription factors bind directly to DNA close to the genes they regulate. The determination of the genomic location of transcription factor binding sites provides important insights into the mechanism of regulation that will potentially be used to develop drugs for prevention and treatment of metabolic disorders. We therefore analysed the binding sites of a transcription factor of interest (TFI) that is involved in metabolic and inammatory signal integration. Transcription factor binding is often DNA sequence specific. However, it becomes clear that features beyond DNA sequences define where transcription factors bind. It has previously been shown that transcription factor binding correlates with regions of open chromatin that represent active regulatory elements [3]. Therefore, we determined the openness of chromatin that accompanies the binding of the TFI. Collectively, our studies aimed to provide an initial framework for understanding and investigating TFI-dependent gene regulations in human cells by the use of integrative analysis strategies.

2. Used Future SOC Lab Resources

We used a Hewlett Packard DL980 G7 server that was equipped with eight 8-core Intel Xeon X6550 processors and 128 GB of RAM running Ubuntu Server 12.04 LTS. This powerful system was perfectly suited for our approach. The only disadvantage regarding flexibility was the restriction to weekly user time slots.

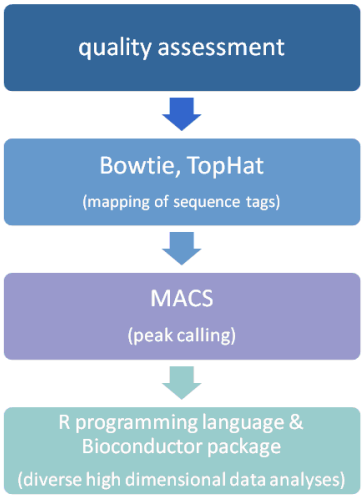


Figure 1. Dynamic pipeline for computational analysis of high-throughput NGS data.

3. Methods and tools

Genomic binding sites of the TFI were determined by sequencing the DNA that was bound by the TFI,

a technique referred to as chromatin immunoprecipitation coupled with NGS (ChIP-seq). Open chromatin was investigated using formaldehyde-assisted isolation of regulatory elements coupled with NGS (FAIRE-seq). The computational pipeline for NGS data analysis is given in Figure 1. The ChIP-seq and FAIRE-seq derived unltered 36bp raw reads (about 23 GB) were uniquely aligned to the human reference genome February 2009 assembly (GRCh37/hg19) using the fast and memory-efficient short read aligner Bowtie [4]. Systematic bias was corrected prior to downstream analysis using the tool BEADS [2]. Post-mapping analysis workflows involved determination of genomic regions significantly enriched in aligned tags by the use of the peak calling algorithms MACS [7] and F-Seq [1]. The defined peak intervals were further characterized relative to genome features using Bedtools [5] and R [6].

4. Findings

The Future SOC Lab resources helped us to rapidly analyse data generated by NGS. Using the multi-core architecture we were enabled to systematically test several settings for sequencing tag mapping, filtering and peak calling. Initially, we used the established pipeline to generate genome-wide density maps of TFI binding and chromatin accessibility (Figure 2).

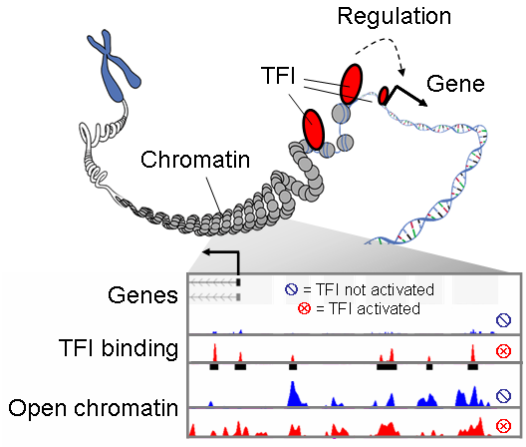


Figure 2. Density profiles generated from NGS data help to investigate transcription factor binding and chromatin states on a genome-wide scale.

4.1. Genome-wide identication of TFI binding sites

We applied peak calling using calibrated settings and dened a high-condence set of genome-wide TFI binding sites with 5734 peak intervals if TFI was activated and 1130 peak intervals if TFI was not activated. These binding regions were integrated with gene expression data to predict a putative gene regulatory network of the cellular response to TFI activation (Figure 3).

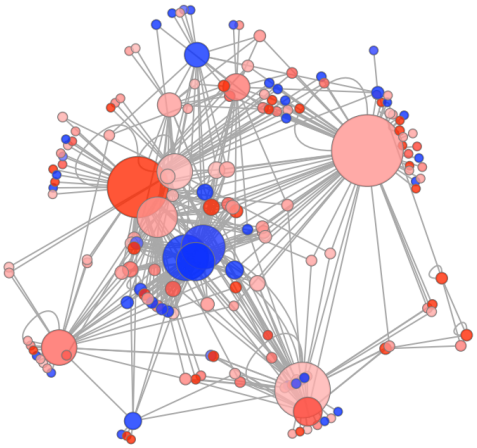


Figure 3. Network prediction that connects the identified TFI target genes with the global transcriptional response. Node colors indicate expression levels of genes (red: up-regulated, blue: down-regulated). The node size is proportional to the relative connectivity to other factors.

4.2. Quantitative relationship between TFI binding and open chromatin

To quantify the relation of TFI binding and chromatin status, genomic regions with high chromatin accessibility over background were delineated with liberal stringency [3] and 170k sites were identied encompassing 60Mb of the human genome (2.2%). Non-promoter sites of open chromatin (91%) showed dynamic activation-dependent changes while promoter-associated sites (9%) were relatively constant. 72% of the TFI bound regions were restricted to sites of open chromatin generating a high-confidence set of putatively functional binding regions.

5. Next steps

Further participation in the HPI Future SOC Lab would enable us to break new ground in data analysis and to extract biologically meaningful information from the generated NGS data. Using sequencing data that we recently generated we were not exploiting the whole potential of the provided resources. Thus, the provided capacities will enable us to integrate publicly available genome-wide data from other research

groups with our own data in future work. Therefore, the already established computational pipeline shall be used to help to identify and understand the mechanism of metabolic pathologies and inflammatory processes.

References

[1] A. P. Boyle, J. Guinney, G. E. Crawford, and T. S. Furey. F-seq: a feature density estimator for high-throughput sequence tags. *Bioinformatics (Oxford, England)*, 24(21):2537–2538, Nov. 2008. PMID: 18784119.

[2] M.-S. Cheung, T. A. Down, I. Latorre, and J. Ahringer. Systematic bias in high-throughput sequencing data and its correction by BEADS. *Nucleic Acids Research*, 39(15):e103, Aug. 2011. PMID: 21646344 PMCID: PMC3159482.

[3] K. J. Gaulton, T. Nammo, L. Pasquali, J. M. Simon, P. G. Giresi, M. P. Fogarty, T. M. Panhuis, P. Mieczkowski, A. Secchi, D. Bosco, T. Berney, E. Montanya, K. L. Mohlke, J. D. Lieb, and J. Ferrer. A map of open chromatin in human pancreatic islets. *Nature genetics*, 42(3):255–259, Mar. 2010. PMID: 20118932.

[4] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 10(3):R25, 2009. PMID: 19261174.

[5] A. R. Quinlan and I. M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics (Oxford, England)*, 26(6):841–842, Mar. 2010. PMID: 20110278.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[7] Y. Zhang, T. Liu, C. A. Meyer, J. Eeckhoute, D. S. Johnson, B. E. Bernstein, C. Nusbaum, R. M. Myers, M. Brown, W. Li, and X. S. Liu. Model-based analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137, 2008. PMID: 18798982 PMCID: PMC2592715.

ECRAM (Elastic Cooperative Random-Access Memory) HPI Future SOC Lab Project Report September 2012

Kim-Thomas Rehmann Kevin Beineke
Michael Schöttner
Institut für Informatik, Heinrich-Heine-Universität Düsseldorf,
Universitätsstraße 1, 40225 Düsseldorf, Germany
E-Mail: Kim-Thomas.Rehmann@uni-duesseldorf.de

Abstract

The ECRAM storage system is an experimental platform for data-centric cloud applications. In contrast to a traditional one-copy serializable database, ECRAM allows highly concurrent accesses to storage objects. It replicates objects in order to make them availe on shared-nothing architectures. An adaptive replication mechanism analyses object access patterns to switch dynamically between update and invalidate protocol. We have evaluated the impact of different replication strategies on performance using HPI Future SOC Lab's resources. The experiments demonstrate that adaptive replication scales better than update and invalidate replication for the applications under examination.

1 Project Idea

The ECRAM project addresses current research topics related to storage for cloud applications [6]. ECRAM simulates a shared in-memory storage [3] on shared-nothing clusters. The object-based storage provided by ECRAM is comparable to a key-value store, where the keys are object identifiers and the values contain binary object data. In contrast to most existing key-value stores, ECRAM increases availability by replicating data. It synchronizes objects optimistically in order to allow atomic multi-object operations.
ECRAM does not restrict the format of objects it stores. It neither associates any semantics with object content. Applications can therefore implement row-based or column-based tables, or they can define a flexible data format, such as graph structures.
Cloud applications often have read-optimized object access patterns. Access patterns with prevailing reads are a benign workload for optimistic concurrency control. Therefore, ECRAM implements optimistic multi-version concurrency control using the concept of memory transactions. In contrast to traditional DBMS transactions, memory transactions operate on replicated rather than partitioned data. Replication relieves ECRAM from the need to handle distributed transactions. Transactions can access distributed storage, but they always execute on a single node. The colocation of in-memory storage and application code allows ECRAM to restart transactions transparently for applications.
By caching replicas in the application address space, ECRAM achieves zero-latency object accesses at the best. Transparent access detection using virtual memory hardware simplifies application development. ECRAM's adaptive caching mechanism is able to switch dynamically between update and invalidate semantics and to prefetch objects to reduce access latency. Adaptive caching monitors and analyses access patterns to predict future accesses.
ECRAM can run on conventional compute clusters over TCP/IP networking as well as on a single machine over the loopback network interface. Our experiments on HPI Future SOC Lab resources investigate the performance of ECRAM's adaptive replication on a multicore machine with huge main memory.
This report is structured as follows. Section 2 sketches design and implementation of the ECRAM distributed in-memory storage and of EMR, the in-memory MapReduce implementation based on ECRAM. Section 3 describes the Future SOC Lab resources used for experiments with EMR, and Section 4 presents the results of the measurements. Section 5 concludes this report.

2 Using ECRAM for in-memory MapReduce

The ECRAM in-memory storage implements update-anywhere objects on shared-nothing clusters. In order to make objects available on all participating nodes ECRAM replicates objects. Therefore, consistency handling is an important part of its functionality. ECRAM's builtin consistency handling enables lock-free implementation of applications and of programming frameworks such as EMR.

2.1 The ECRAM distributed in-memory storage

ECRAM's consistency handling bases on memory transactions in the sense of distributed transactional memory [2, 7]. A memory transaction is a sequence of read or write accesses by one computing node. If the transaction's read and write set does not intersect with transactions committed in the meantime, the transaction is said to be valid and commits its changes atomically to the shared storage. If ECRAM detects an invalid transaction, it transparently rolls back the speculative transaction's changes and restarts it. For efficient comparison of read and write sets, ECRAM has a conflict unit size of 4 KB. In order to ensure the global serializability of transactions, ECRAM's current implementation uses a central validator node. Object content is transferred directly between computing nodes.

Accessing objects outside transactions is possible with ECRAM, however read accesses may return out-of-date content, and changes are not stored permanently. In the course of optimizing ECRAM for MapReduce workload, we have developed a relaxed transactional consistency targeted towards the needs of MapReduce applications, which we describe in the next subsection.

ECRAM implements three replication modes. The invalidation mode replicates objects only on demand. To avoid frequent accesses to outdated object versions, ECRAM multicasts object invalidations using transaction commit notifications. Invalidation mode consumes least bandwidth, but requires synchronous communication when accessing recently modified objects. The update mode sends the content of each modified objects along with the commit notifications. Update replication has a high bandwidth requirements, but ensures that replicas are almost always available without network communication. The third replication mode, adaptive replication, monitors object accesses to detect object access patterns. In case of regular access patterns, the adaptive mode replicates objects only to those nodes that will probably access the version of the object soon. Besides replicating objects for performance, ECRAM creates backup replicas to improve fault tolerance in case of failures.

2.2 The EMR Framework for In-memory MapReduce

In order to demonstrate the utility of distributed in-memory storage for MapReduce, we have developed the EMR extended MapReduce framework [6]. MapReduce applications running on EMR share information by means of the ECRAM[1] distributed in-memory storage. The EMR framework implements an extended MapReduce model comprising one-pass, iterative and on-line execution of the MapReduce sequence. Figure 1 illustrates the execution model of in-memory extended MapReduce.

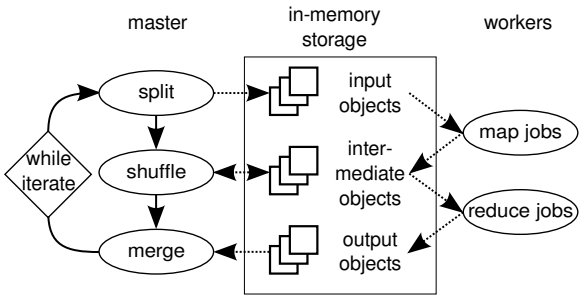


Figure 1. Execution model of in-memory MapReduce

The EMR framework stores application data as replicated objects in the ECRAM in-memory storage. During MapReduce execution, the EMR framework automatically prepares input, intermediate and output objects according to the application's specification. Furthermore, worker nodes can create objects on their own using the function `ecram_alloc`, which returns zero-initialized objects of the requested size. If an object is not needed any longer, it must be destroyed using `ecram_free`. Objects in ECRAM may contain references to other objects, which allows applications to implement distributed data structures.

In general, transactional consistency is well-suited for irregular access patterns with low conflict probability. However, for applications that conform strictly to iterative MapReduce with alternating, stateless map and reduce phases, transactional consistency is slightly too strong because the sequence of map and reduce phases eliminates concurrent accesses. If the programmer can preclude access conflicts, validation of transactions is dispensable. We have implemented a special weak consistency which encapsulates accesses in transactions but assumes the absence of conflicts and omits the validation phase. The programmer selects non-validating transactions when calling `ecram_bot`.

Like other MapReduce frameworks, EMR comprises a scheduler for map and reduce jobs [5]. EMR stores work-queues, jobs and configuration data in ECRAM to take advantage of automatic replication and consistency. To avoid busy waiting for conditions, EMR uses the `ecram_wait` function that blocks until a storage object fulfills a given condition. Each work-queue has a field specifying the current number of jobs in the queue, which serves as a condition variable for the `ecram_wait` synchronization primitive.

EMR keeps a queue of map and reduce jobs for each worker with the intention to minimize contention on the queues. High contention would lead to high transaction conflict rates and therefore degrade perfor-

[1] ECRAM is an acronym for *elastic cooperative random-access memory*.

mance. For a given number of worker nodes n, the first n jobs are assigned round-robin, and successive jobs are distributed randomly to the nodes. A more advanced load leveling policy could be implemented in a future version of EMR.

In some MapReduce applications, job execution time has significant outliers [1]. EMR implements a simple work-stealing approach to counteract workers idling while others still have jobs in their queue. If a worker finds that he is about to block on his empty queue, he scans the work-queues of his peers for jobs to steal from them. Given that work-queues are stored as shared objects, there is no danger of deadlocks or lost jobs.

3 Future SOC Lab Resources

We have executed our experiments on a Hewlett Packard ProLiant DL980 G7 Server. The server is equipped with 8 Xeon Nehalem X7560 CPUs, each having 8 hyper-threaded cores. The CPU clock rates are 2,27 GHz, the L3 caches 24 MB large, and Turbo Boost is enabled. The DL980 has 2 TB of RAM. Given that our experiments run in main memory of the single machine, they do not use the hard disks, neither the Fibre Channel network card. The DL980 boots Ubuntu Server 10.10 from a local harddisk and mounts the home file-system from a NAS device.

4 Findings

Our previous project reports from October 2011 and Mai 2012 have documented the initial results of running ECRAM on Future SOC Lab. We had confirmed that ECRAM runs on HPI Future SOC Lab and scales comparably, and in some cases significantly better than on a distributed compute cluster. In the Spring 2012 period, we have assessed ECRAM's performance more in detail with a special focus on adaptive replication. Considering that ECRAM is a fully distributed system, running several ECRAM programs on one multicore machine constitutes a best case test environment because of the low latency and the high bandwith.

Figure 2 shows the runtime of the raytracer application with an image size of 1800x2400 pixels and 228 objects in the scene graph. The raytracer application is implemented using ECRAM's framework for in-memory MapReduce [6]. In the map phase, the worker nodes calculate the pixels in disjoint regions, and in the reduce phase, the output image is produced. The raytracer's map phase is an *embarrassingly parallel* workload, because each pixel in the output image is traced independent of any other pixel. The raytracer's reduce phase simply collates all regions computed by different nodes into the final output image. The invalidate protocol causes the reduce jobs to pull the output

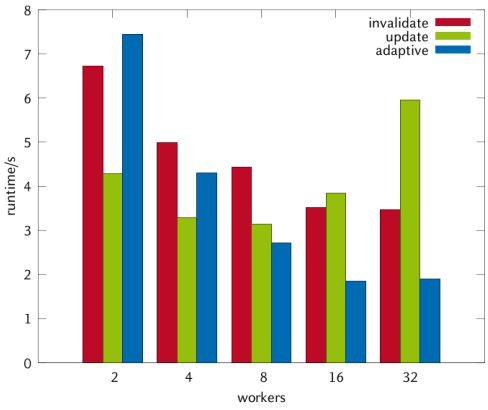


Figure 2. Execution time of raytracer using ECRAM on DL980, image size 1800x2400 pixels

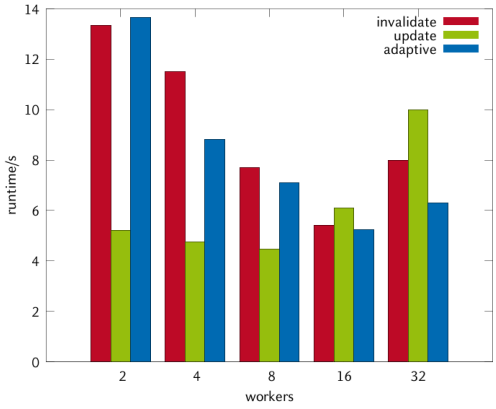


Figure 3. Execution time of KMeans using ECRAM on DL980, 100,000 points in 3D space with 16 cluster centers

from the map jobs, which increases the amount of synchronous network messages and thereby the overall latency. The update protocol distributes all data generated to all other nodes, most of which do not access the data lateron. The bandwidth required to tranfer the huge amount of data limits scalability. The adaptive replication protocol notices that the output of the map jobs is afterwards accessed by the reduce jobs. It sends updates directly from map workers to reduce workers, which keeps the required bandwidth low and at the same time avoids synchronous requests to pull replicas.

Figure 3 shows the runtime of the KMeans application with 100,000 points in the 3D space, which are grouped into 16 clusters. The KMeans algorithm is generally not so well parallelizable, because the iterative execution often updates data. However, the adaptive replication protocol often successfully predicts where data is required in the next round, so it

scales better than the invalidate or update protocol.

5 Next Steps

In the context of cloud computing, network and machine failures are common. Therefore, the reliability of distributed storage is an important research topic. Until now, ECRAM provides reliability by means of replication in volatile memory. We are currently working on providing persistency within ECRAM by logging and storing objects to durable storage such as harddisks, flash memory, solid state drives and phase-change memory. Each of these storage technologies has its own characteristics, such that persistent storage should implement different policies for different technologies. If time and resources permit, we intend to evaluate our distributed in-memory file-system [4], which is based on ECRAM and FUSE, on Future SOC Lab.

ECRAM enhances fault tolerance of parallel applications by executing each thread in a private protection domain. However, using this approach, different threads executing on the same machine cannot benefit from their colocation. OS-level IPC mechanisms could allow them to access shared data directly. Direct sharing can potentially boost the performance of applications on powerful multicore machines like the ones in HPI Future SOC Lab. We plan to implement direct sharing techniques in ECRAM and evaluate them on Future SOC Lab. Given that ECRAM was designed for data sharing in shared-nothing compute clusters, we are also interested in evaluating ECRAM's performance on a cluster of multicore machines.

References

[1] Ganesh Ananthanarayanan, Srikanth Kandula, Albert Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris. Reining in the outliers in Map-Reduce clusters using Mantri. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–16, Berkeley, CA, USA, 2010. USENIX Association.

[2] Maurice Herlihy and Victor Luchangco. Distributed computing and the multicore revolution. *SIGACT News*, 39(1):62–72, 2008.

[3] Hasso Plattner and Alexander Zeier. *In-Memory Data Management*. Springer, Berlin and Heidelberg, Germany, 2011.

[4] Kim-Thomas Rehmann and Michael Schöttner. Adaptive meta-data management and flexible consistency in a distributed in-memory file-system. In *Proceedings of the Twelfth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2011)*, Gwangju, Korea, 10 2011.

[5] Kim-Thomas Rehmann and Michael Schöttner. Applications and evaluation of in-memory mapreduce. In *Proceedings of the Third International IEEE Conference on Cloud Computing Technology and Science 2011 (CloudCom 2011)*, Athens, Greece, 12 2011.

[6] Kim-Thomas Rehmann and Michael Schöttner. An in-memory framework for extended MapReduce. In *Proceedings of the Seventeenth IEEE International Conference on Parallel and Distributed Systems 2011 (ICPADS 2011)*, Tainan, Taiwan, 12 2011.

[7] Paolo Romano, Luis Rodrigues, Nuno Carvalho, and Joäo Cachopo. Cloud-TM: harnessing the cloud with distributed transactional memories. *SIGOPS Oper. Syst. Rev.*, 44:1–6, April 2010.

Analysis of CPU/GPU data transfer bottlenecks in multi-GPU systems for hard real-time data streams

Uri Verner
Technion IIT
Technion City, 32000
Haifa, Israel
uriv@cs.technion.ac.il

Assaf Schuster
Technion IIT
Technion City, 32000
Haifa, Israel
assaf@cs.technion.ac.il

Avi Mendelson
Technion IIT
Technion City, 32000
Haifa, Israel
mendlson@cs.technion.ac.il

Abstract

This project is a part of our effort to extend the framework to multi-GPU systems. In this scope, we focused on the analysis of data transfers between a host and multiple GPU devices, for streaming workloads. We identified important bottlenecks in data transfers, and showed that data transfers to different GPUs are interdependent. By changing the configuration of GPU cards in the Tesla machine, we increased the overall bandwidth between the main memory and the GPUs, and stabilized it.

1 Introduction

Modern real-time data stream processing systems typically use high-bandwidth connections both internally and to external sources of data, and have high computational demands. The increasing amounts of collected data and the demand for its processing have far passed the compute capabilities of traditional CPU-only systems. Compute accelerators, such as FPGAs, DSPs and GPUs, are a common way to increase a systems' computing capabilities for specific tasks. However, processing real-time data on a heterogeneous system poses great challenges in data transfer, scheduling, load balancing, software design, synchronization, and more.

Previously, we have developed a high-throughput software framework for processing of multiple real-time streams on a heterogeneous system with a multi-core CPU and a single GPU [1]. The framework receives a configuration of the workload – the number of data streams and their compute and latency requirements – and statically distributes the streams between the CPU and the GPU in a way that guarantees that their execution will not result in a deadline miss. Then, each stream is processed on the device it was assigned.

This project is a part of our effort to extend the framework to multi-GPU systems. In this scope, we focused on the analysis of data transfers between a host and multiple GPU devices, for streaming workloads. We identified important bottlenecks in data transfers, and showed that data transfers to different GPUs are interdependent. The last observation is contrary to the basic assumption that each data transfer to a GPU takes a unique path over a separate, per-GPU, PCI-Express bus.

2 CPU/GPU data transfers

We begin by describing the underlying hardware architecture of a large multi-GPU system.

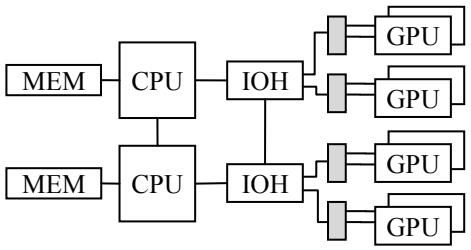


Figure 1: System architecture of a multi-GPU platform

2.1 Architectural description

Figure 1 depicts the high-level architecture of a large GPU-based system. There are five major types of components in the CPU/GPU data transfer pipeline: main memory, multicore CPUs, I/O hubs (IOH), PCI Express switches (gray boxes in Fig. 1), and GPUs.

An on-chip memory controller connects the multicore CPUs to the memory bus. The CPUs are linked with fast interconnects between themselves and to the I/O hubs. These interconnects communicate directly with the memory controller. Thus, I/O hubs also have access to main memory.

I/O hubs connect to GPUs over a PCI-Express tree-shaped bus hierarchy. Each I/O hub supports two links. Each link is multiplexed using a PCI Express switch to two or more links, and eventually these links are attached to GPUs.

2.2 Streaming data transfers

In GPU applications, data is most commonly copied from main memory to GPU memory, and back. In this work, in the context of extending our hard real-time stream processing framework to multi-GPU systems, we concentrate on transferring data streams to multiple GPUs and back continuously and simultaneously. Therefore, we redirect input streaming data from main memory to the GPUs over the following path: main memory → CPU → IOH → PCIe switch → GPU memory. Output is redirected over the same path in the opposite direction.

2.3 Transfer bandwidth and predictability

The workload for our framework is hard real-time data streams. Hence, it is essential that the latency of data processing will be predictable. Data transfers are a major part of processing, so the framework needs to be able to compute a tight bound on their duration.

Transfer bandwidth to and from the GPUs is also an issue of great importance to the system, since its throughput is limited by the effective bandwidth for data transfers in and out of the GPUs.

In this project, we examine how the latency and throughput of a data transfer between the main memory and a GPU are influenced by concurrent data transfers to different GPUs.

3 Experiments

We ran benchmarks that measure the latency and throughput of data transfers between the host and the GPUs. In each experiment, a data chunk of 128MB is transferred to or from one or more GPUs. The data transfers are concurrent and are always done in the same direction (host-to-device or device-to-host). The results were calculated as an average over 100 transfers.

We used the Tesla machine in the Future SOC lab for the experiments. The architecture and initial setup of the machine is illustrated in figure 2. The system has two Xeon (Nehalem) E5620 processors and supports 8 GPUs, but only has four GPUs of type NVIDIA Tesla C2050. The main board in use was TYAN FT72B7015, which has eight x16 PCI Express 2.0 slots. The CPUs and the IOHs are connected with point-to-point QPI links.

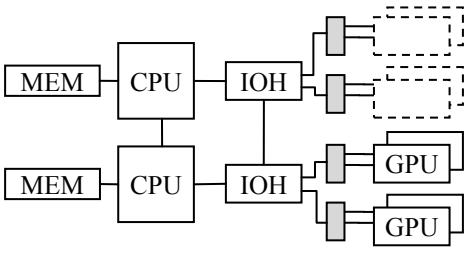


Figure 2: Architecture of the Tesla machine

3.1 Bandwidth maximi ation by card rearrangement

The bandwidth test results for the system were inconsistent; repeating executions of the same experiment gave different results. At its highest value, the bandwidth to the four GPUs was limited by twice the bandwidth of a PCIe link, due to the use of multiplexing PCIe switches. Hence, we recommended that the configuration of GPUs be changed in a way that no two GPUs share a PCIe switch, as illustrated in figure 3.

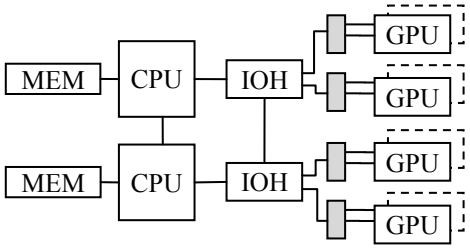


Figure 3: Architecture of the Tesla machine with altered configuration of GPUs

As a result, the benchmark results stabilized and returned similar values in repeating executions. The total bandwidth improved as well.

3.2 Latency of data transfers

Predictable and persistent latency of data transfer operations is critical to our framework, since hard real-time stream processing requires a priori scheduling. For each data transfer, we measured the time from issue to completion. We ran experiments that transfer data from the host to all possible groups of one or more GPUs.

The results showed that the latencies of data transfers to different GPUS are inter-dependant. Moreover, in some cases, the transfers are serialized, and the order of serialization is arbitrary.

4 Conclusions and discussion

High bandwidth and predictable latency of data transfers between the host and the GPUs are essential to the hard real-time data stream framework. System architecture and its configuration have great influence on these parameters.

We increased the total bandwidth between the main memory and the GPUs, and made it stable, by rearranging the GPU cards so that no two GPU cards share a PCI Express switch.

The latency of a data transfer to a GPU depends on concurrent transfers to other GPUs. Hence, the stream processing framework cannot schedule data transfers independently. As a result, we recommend that the framework perform data transfers to all the

GPUs in a single stage. The latency of this stage can be predicted by dividing the total amount of transferred data and divide it by the aggregate bandwidth to all GPUs. Even though it is not known in which order the transfers are finished, the overall transfer time is stable and can easily be predicted.

Further research is required to determine the reasons for the inconsistent latency results and the bottlenecks for throughput.

References

[1] U. Verner, A. Schuster and M. Silberstein: Processing data streams with hard real-time constraints on heterogeneous systems. *Proceedings of the 2011 ACM International Conference on Supercomputing*, June 2011.

KONECT Cloud – Large Scale Network Mining in the Cloud

Report, Spring 2012

Dr. Jérôme Kunegis
WeST – Institute for Web Science and Technologies
University of Koblenz–Landau
kunegis@uni-koblenz.de

Abstract

In the Spring 2012 run at the Future SOC Lab, we used the KONECT framework (Koblenz Network Collection) to compute network statistics, distributions, plots, matrix decompositions and link prediction evaluations on the network of KONECT. The results are used on the project website konect.uni-koblenz.de, and internally at the University of Koblenz–Landau for network analysis research.

1 Introduction

KONECT (the Koblenz Network Collection [2], konect.uni-koblenz.de) is a project to collect network datasets, provide a framework for the computation network statistics, and display them online. One of the project's goals is to include a high number of diverse network datasets of many different types. Currently, KONECT holds 186 network datasets from 15 categories; an overview is given in Figure 1 and Table 1. Networks in KONECT are also diverse in their structure – they can be directed, undirected, bipartite, unweighted, have multiple edges, signed, have ratings, and timestamps. At the same time, KONECT strives to implement and compute many different numerical network statistics, analysis plots, distributions, matrix decompositions and link prediction algorithms. This combination leads to a high number of different network-statistic combinations that must be computed. In addition to providing a website in which the network statistics are navigable and browsable, the computed statistics are used for network analysis research. For instance, papers studying measures of diversity and power law have used the data compute by the KONECT project [3, 4].

2 Performed Analyses

The analyses computed in KONECT include the following; a detailed definition for each of these is given in the KONECT Handbook [1].

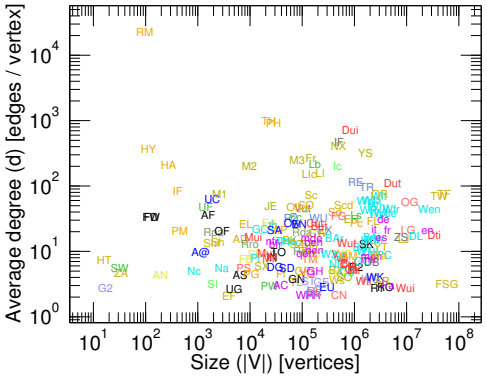


Figure 1. All networks in KONECT arranged by the size (the number of nodes) and the average number of neighbors of all nodes. Each network is represented by a two- or three-character code. The color of each code corresponds to the network category as given in Table 1.

The numerical analyses computed include the connected components, the diameter, the radius, the estimated power-law exponent, the Gini coefficient, the degree distribution entropy, the algebraic connectivity, the algebraic conflict, the frustration, the bipartivity, the number of triangles and the clustering coefficient.

The computed plots include the edge weight and degree distributions, the Lorenz curve, spectral plots and hop plots. A sample of computed plots is shown in Figure 2, showing a screenshot of the Plots section of the KONECT website.

Computed matrix distributions include the eigenvalue decomposition, the singular value decomposition and DEDICOM-type decompositions (Decomposition into Directed Components) of various characteristic graph matrices including the adjacency matrix, the normalized adjacency matrix, the Laplacian matrix, asymmetric matrices, complex matrix, and other, experimental decompositions.

Computed link prediction algorithms include preferential attachment, counting connecting paths, graph kernels and solving logistic regression models.

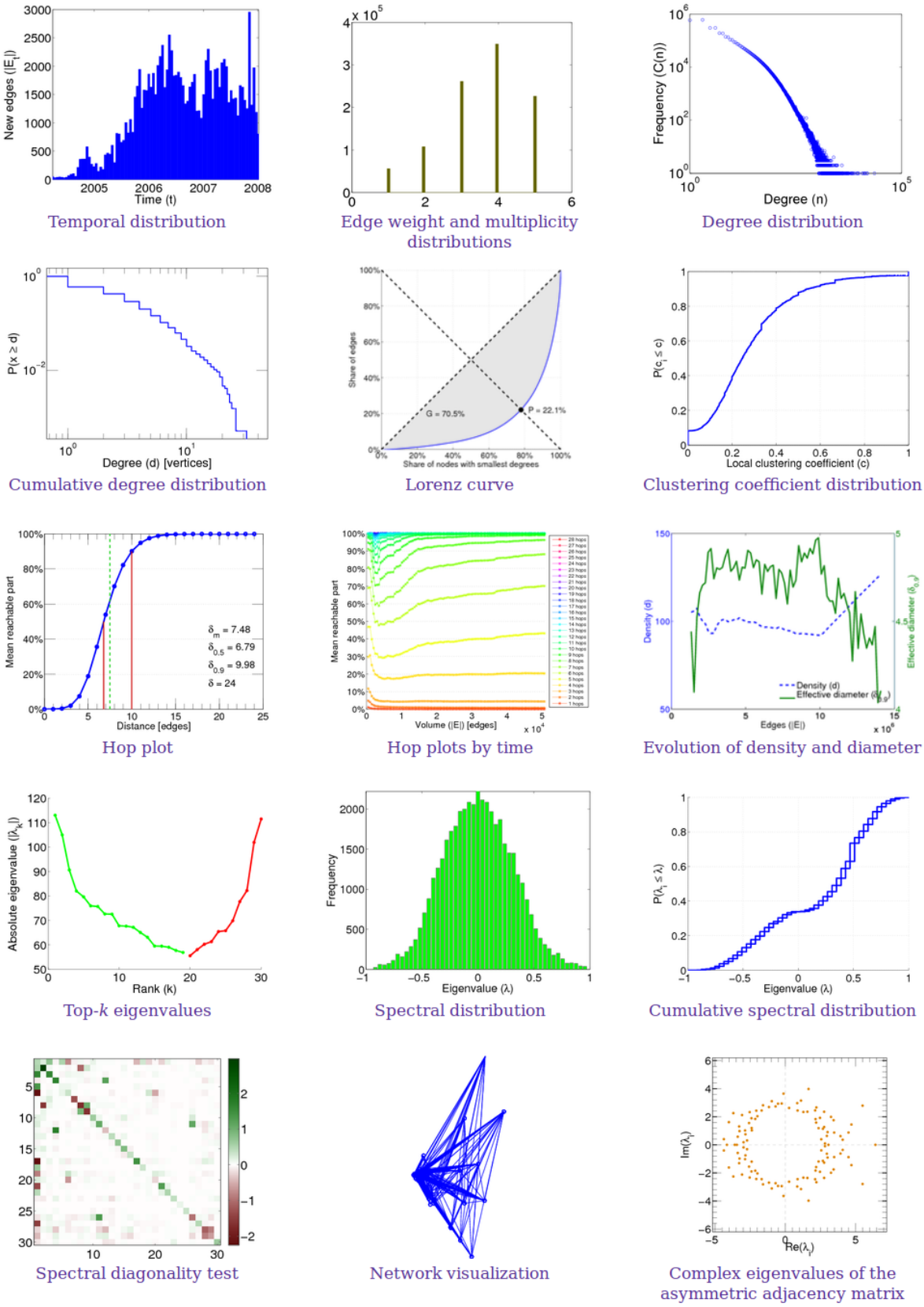


Figure 2. Network analysis plots computed in KONECT and shown on the KONECT website.

Table 1. The network categories in KONECT. Each category is assigned a color, which is used in plots, for instance in Figure 1. The given dataset counts are current as of April 2013.

	Category	Vertices	Edges	Count
	Affiliation	Actors, groups	Memberships	8
	Authorship	Authors, works	Authorships	18
	Co-occurrence	Items	Co-occurrences	2
	Communication	Persons	Messages	8
	Contact	Persons	Interactions	4
	Features	Items, features	Properties	5
	Folksonomy	Users, tags, items	Tag assignments	17
	Interaction	Persons, items	Interactions	14
	Lexical	Words	Lexical relationships	5
	Physical	Various	Physical connections	13
	Ratings	Users, items	Ratings	11
	Reference	Documents	References	28
	Semantic	Entities	Relationships	1
	Social	Persons	Ties	29
	Text	Documents, words	Occurrences	5

3 Implementation

The KONECT code executed on Future SOC Lab servers is written in the Matlab programming language. The code is available in form of a Matlab toolbox.[1]

We used a single server having 2 terabytes of memory (RAM) and 128 cores. Each computation used between one and seven CPUs in parallel; with the matrix decompositions using the most parallelism. To control the execution, a Makefile-based framework was used, using the `-j` option to enable parallelization. In accordance with our previous experiments with KONECT, the bottleneck for the computation of network statistics is memory, not CPU. Thus, we were able to fully use the 2 terabytes available.

4 Conclusions and Acknowledgements

The computation of network analyses is not finished. First, the available runtime was not enough to execute all analyses for all networks. Furthermore, more network analyses and networks are added to KONECT constantly, and we are constantly computing new data.

We thank the Future SOC Lab for supporting the KONECT project through making available hardware.

KONECT was written by Jérôme Kunegis, Daniel Dünker and Holger Heinz. We also thank everyone who has made available network datasets; see http://konect.uni-koblenz.de/about for a full list.

[1] konect.uni-koblenz.de/toolbox

References

[1] J. Kunegis. Handbook of network analysis [KONECT – the Koblenz Network Collection], 2013.

[2] J. Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Web Observatory Workshop*, 2013.

[3] J. Kunegis and J. Preusse. Fairness on the web: Alternatives to the power law. In *Proc. Web Science Conf.*, pages 175–184, 2012.

[4] J. Kunegis, S. Sizov, F. Schwagereit, and D. Fay. Diversity dynamics in online networks. In *Proc. Conf. on Hypertext and Social Media*, pages 255–264, 2012.

Adaptive Realtime KPI Analysis of ERP transaction data using In-Memory technology

Prof. Dr. Rainer Thome
Dipl.-Kff. Patricia Kraft
Chair of Business Administration
and Business Computing
Joseph-Stangl Platz 2
97070 Würzburg
thome@wiinf.uni-wuerzburg.de
pkraft@wiinf.uni-wuerzburg.de

Dr. Andreas Hufgard
Dipl.-Kfm. Fabian Krüger
Dipl.-Kfm. Ralf Knauer
IBIS Labs
Mergentheimer Str. 76a
97082 Würzburg
hufgard@ibis-thome.de
fkrueger@ibis-thome.de
knauer@ibis-thome.de

Abstract

The adaptable real-time analysis provides an answer to the greater complexity and size of advanced ERP systems (big data), and increased market demands for speed and quality of decision-related data. In-memory storage enables direct access to document or change data, and allows greater flexibility when selecting by date, organization and many other characteristics. During this project we have been able to gain some basic knowledge about SAP HANA and establish a data connection between HANA and SAP ERP using the SAP Landscape Transformation Server.

1 Setting up the environment

In the beginning of the project we were given access to the Windows Terminal Server (WTS) in order to connect to a SAP HANA instance hosted in Walldorf. This should bridge the time gap until the system in Potsdam at HPI is available. Therefore the Walldorf system was kind of a playground for working through tutorials, creating the first tables and views.

1.1 Difficulties

The HANA instance accessed via WTS was running an old release at the beginning (as of November 2011). This led to a lack of functionality, for instance, it was not possible to upload data to the tables from a local *.csv* file. After the first upgrade we still found some bugs or glitches. Especially importing date-related data was very complicated due to local settings as well as SAP HANA creating columns of the type "timestamp" instead of "date" when using the *.csv* import. The issues have been reported to our contact person at HPI and have been fixed in the following revisions.

1.2 Lessons learned

In SAP HANA you can create Attribute Views, Ana-

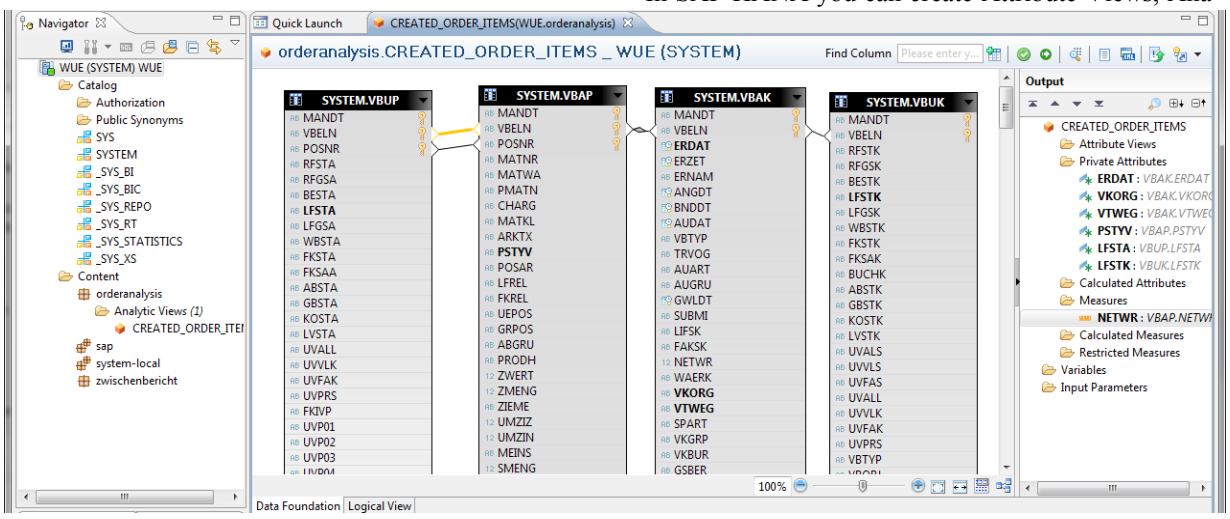


Figure 1: Creating an analytical view in SAP HANA Studio

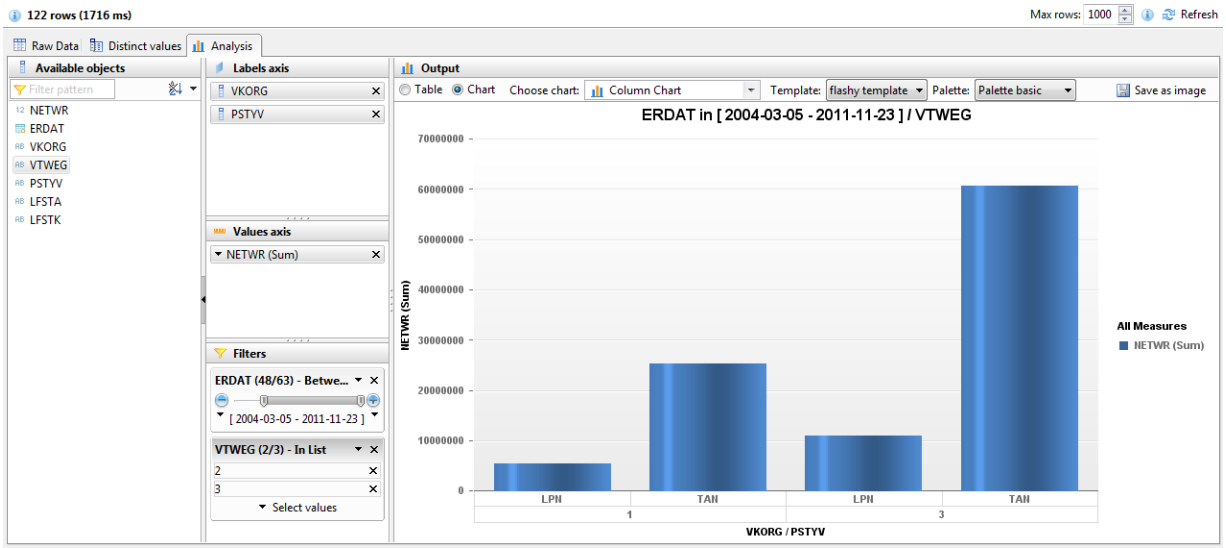


Figure 2: Built-in analysis view in SAP HANA Studio

lytical Views (Figure 1), Graphical Calculation Views, SQL Script Calculation Views and (stored) Procedures. Each of these Objects has some capabilities and restrictions and their complexity increases from the simple graphical Attribute Views to the complex scripted procedures.

For example, an Attribute View has no measures and therefore can be used for master data analysis. Analytical Views are usually used for creating star schemes when combining a transactional table with the corresponding master data. When it comes to time-travel and other special types of Analysis, only scripted views or procedures provide the necessary capabilities. Therefore we also had to acquire some knowledge about SQL Script, the language used to create procedures.

2 Project related progress

The project aim is to create a fully functional prototype of an interactive realtime analysis environment. There are three separate activities needed to achieve this goal: You need a user interface or presentation layer, data to run the analysis on and last but not least the calculations which turn the data into KPIs to be displayed.

2.1 Presentation layer

SAP HANA Studio has an integrated analysis client for data preview (Figure 2). Although it provides a wide range of different chart types and the ability to "drag-and-drop" columns into the label axis, the value axis or filter, it will not be suitable as output medium. First, the preview is limited to a maximum of 5,000 rows, and second, people in the target group for these reports won't be running SAP HANA Studio, which is made for developers. For this project we intend to use "Analysis for Microsoft Excel", which is part of the Business Objects Suite. The connection from Excel to SAP HANA is working flawless and is an easy way to present the data. However, the filtering of Data is not as interactive as in the HANA Studio and organizing a lot of Excel files and sheets can be very confusing. Drill-Down-Possibilities are limited, too. Therefore we plan to use some of the other SAP BusinessObjects tools for presentation. We will need to evaluate those and find one which fits our needs. There is a tool called SAP BusinessObjects Information Steward which looks very promising.

2.2 Connection to SAP ERP

In order to do realtime analysis you need realtime data. That's why we connected SAP HANA to an SAP ERP System by using SAP System Landscape Transformation (Figure 3). The SLT Server creates a database Trigger in the source system (we used a preconfigured IDES system at SAP) and replicates all changes (in the selected tables) to the SAP HANA database. Another fact that requires a continuous stream of data is that we want to do some time-travel analysis. If you have only one point in time where the table is created and initially filled with data, a time travel analysis makes no sense at all. Updating the data in intervals of weeks or month wouldn't do the job either, that's why we rely on realtime replication of the transactional ERP data.

Currently we have the connection between the SAP HANA database at SAP and an IDES system at SAP which is the source system and SLT server at the same time.

Since the HANA System at SAP will be replaced by the one in Potsdam at HPI, we will need to configure a new connection for replication. As we can't create a VPN tunnel from Potsdam to the internal SAP network, we will also need a new SAP ERP system as data source as well as an SLT server.

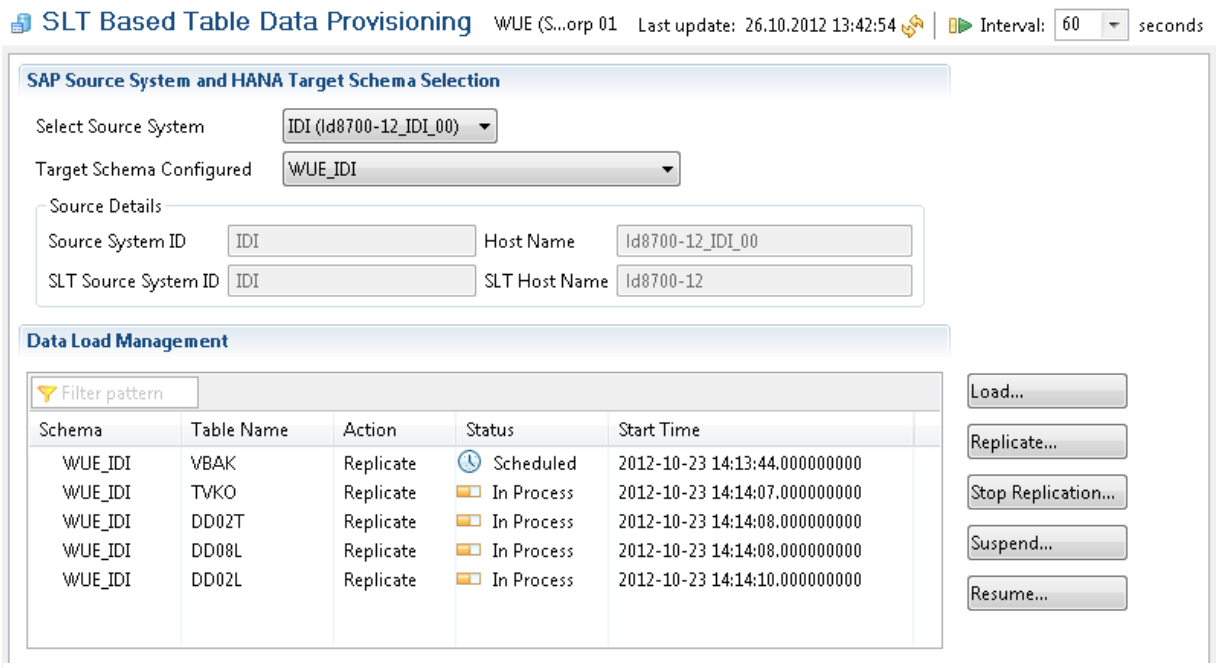


Figure 3: Connection from SAP HANA to SLT and Source System

There is a SAP ERP available at the University of Wuerzburg which contains up-to-date transactional data from case studies in training courses. We would like to use this system, but we would need to get the DMIS addon/license installed in order to allow SLT connections. Additionally an SLT server would have to be set up at HPI in Potsdam.

In case the process of getting the required licenses and addons for the SAP ERP in Wuerzburg is too complicated we would need a SAP ERP and SAP SLT system hosted in Potsdam at HPI.

2.3 KPI Calculations

Since building star schemas is not a real innovation in SAP HANA, we focus our research on the historic tables and time travel. This means we build a KPI which is normally dependent from a reference date.

For example you can calculate the open sales order items by joining the tables VBAK (Sales document: header data), VBAP (Sales document: items data) and VBUP (Sales document: item status). You count the rows where the document category is 'C' (Order), the overall processing status is 'A', the overall delivery status is 'A' and there is no lock or cancellation on header and item level.

You can run this query on every database to get the open orders at this point of time (=now). But on SAP HANA you can classify the tables as 'historic table' and add a time-travel clause to the select statement. By appending 'AS OF UTCTIMESTAMP […]' you can calculate the open orders at any point of time. To make things easier we created a procedure called 'open_orders_as_of' which has a timestamp as input parameter. The result table of this function call is shown in Figure 4.

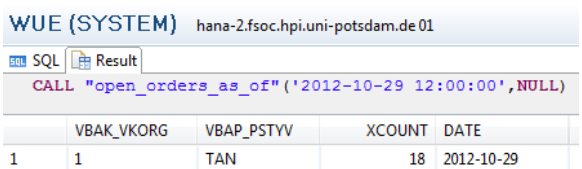


Figure 4: Result table of function call

Once we activated this function we simply call it multiple times for the last X days, union the X result tables into a single one and then we already have the KPI 'Open sales order items' in a timeline for the last days. From now on it would also be easy to add the sum of the net price or add other columns than sales organization and item category.

This historic data can then be used for automated monitoring and triggering alerts as well as context for interpreting the numbers.

3 Further research and needed resources

Until now we have already proven that SAP HANA has all capabilities to enable the interactive reporting we looked for. We created our first KPI and were able to retrieve the values even for dates in the past.

Nevertheless, there is still a lot of research to do:

- The analysis period for the KPI is only static right now. In the scenario of an interactive analysis the user should be able to set the analysis timeframe by himself.
- Until now we only have built one KPI. We need to build other KPIs to prove our con-

cept. There are similar ones like blocked, cancelled and completed orders and some which need to be calculated differently like ‘changed orders’.

- We are planning to make use of the native R Integration to calculate trends, recognize patterns and trigger alerts based on customer specific data from the past. This way the threshold for the alerts can be set automatically without the need of a manual definition.
- As mentioned before we need to find a way to present the data and controls for the interactive analysis. Excel is a first step but we will need to evaluate some of the BusinessObjects tools.

3.1 Needed resources

This project requires the following resources:

- Access to the HANA database at HPI through SAP HANA Studio
- SAP HANA link to an ERP database via SAP landscape trans-formation (SLT) replication server (trigger-based replication)
- SAP Business Objects or Microsoft Excel for visualizing reports

The following requirements have been met and can be used for the project:

- SAP ERP incl. database, located in Wuerzburg, 100/100 Mbit connection

The Impact of Software as a Service

Till Winkler
Humboldt-Universität zu Berlin
10178 Berlin, Germany
till.winkler@wiwi.hu-berlin.de

Abstract

Dieses Dokument beschreibt den Stand und Zwischenergebnisse der empirischen Studie „The Impact of Software as a Service", welche von der Humbodt-Universität zu Berlin (HU) mit Unterstützung des HPI Future SOC Lab und der SAP AG durchgeführt wurde. Insgesamt konnten über diese Kooperation 32 vollständige Datensätze zur Nutzung von verschiedenen SaaS-Lösungen in Anwenderunternehmen gewonnen werden. Präsentiert werden hier einige ausgewählte deskriptive Merkmale dieses Datensatzes. Für weiterführende Auswertungen ist geplant, diese Daten mit denen einer weiteren Erhebung zusammen zu führen und wissenschaftlich auszuwerten.

1. Hintergrund

Die wachsende Verbreitung von Software as a Service (SaaS) ist im Begriff die Zusammenarbeit von Fach- und IT-Bereichen in Unternehmen zu verändern. Dadurch dass weite Teile des Anwendungs- und Infrastrukturmanagements für SaaS von externer Anbieterseite übernommen werden, ändert sich ebenfalls die interne Aufteilung von Rollen und Verantwortlichkeiten (die Governance) zwischen Nutzern und IT Mitarbeitern in Unternehmen.

An der Humbodt-Universität wurden in diesem Kontext verschiedene qualitativ und quantitativ ausgerichtete Arbeiten durchgeführt, welche nun durch die aktuelle Studie „The Impact of Software as a Service" ergänzt werden sollen. Ziel dieser Studie ist insbesondere zu erforschen, inwieweit sich verschiedene Aufteilungen der IT-Verantwortlichkeiten auf die Qualität der internen Zusammenarbeit – und damit auf die Qualität der SaaS-Nutzung insgesamt – auswirken.

Das HPI Future SOC Lab und seine Kooperationspartner (insbesondere die SAP AG) haben sich im Rahmen des vergangenen Future SOC Day bereit erklärt, im Rahmen einer akademischen Partnerschaft dieses Projekt und die notwendige Datenerhebung zu unterstützen.

2. Vorgehen und ausgewählte deskriptive Ergebnisse

Vorgehen und ausgewählte deskriptive Ergebnisse Im Mai-Juni 2012 wurde ein bestehendes Instrument als Online-Fragebogen umgesetzt und im Hinblick auf die Datenerhebung unter SAP-Kunden inhaltlich und Layout-technisch umgestaltet sowie die Umfrage vorbereitet. Qualitätssicherung und Pretests erfolgten durch verschiedene Mitarbeiter der SAP.

Ab dem 2. Juli wurde die Umfrage über verschiedene Online-Kanäle beworben. Hierzu gehörten insbesondere der SAP Blog[1], die LinkedIn Gruppe SAP Business By Design[2], die SAP ByDesign-Gruppe in Xing[3], die Facebook-Seite SAP Cloud Solutions[4] sowie ein Post im Tweet SAP Cloud Solutions über Twitter[5]. Aufgrund des vergleichsweise schwachen Rücklaufs, wurde die Umfrage ab Mitte Juli zusätzlich auch in diversen Usergruppen der SAP AG weltweit gepostet sowie die offizielle Fragebogenlaufzeit auf Ende August verlängert. Abbildung 1 zeigt den Verlauf der (nutzbaren und nicht nutzbaren) Antworten über die Zeit.

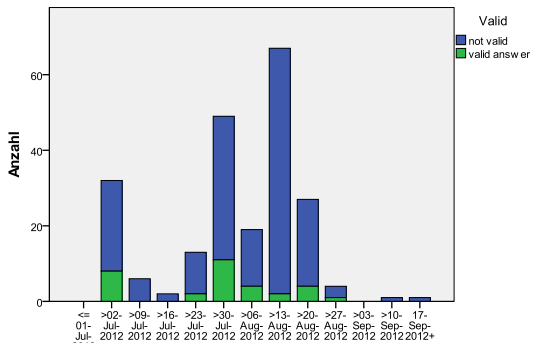


Abbildung 1. Teilnehmer über die Fragebogenlaufzeit.

[1] http://blogs.sap.com/cloud/2012/07/02/scientific-study-among-users-of-sap-cloud-products-please-participate-by-the-end-of-august/
[2] http://www.linkedin.com/groups?gid=138840&trk=hb_side_g
[3] https://www.xing.com/net/prib961b4x/sapbyd/
[4] http://www.facebook.com/SAPCloudSolutions
[5] https://twitter.com/#!/SAPCloud

Der Fragebogen gliederte sich in fünf große Abschnitte:

I Company and Participant Profile

II IT Organisation Characteristics

III SaaS Application Details

IV Business and IT Role in Managing SaaS (focus topic)

V Outcomes of using SaaS

Insgesamt haben 245 Personen den Fragebogen aufgerufen. 84 Personen haben den ersten Abschnitt ausgefüllt, 32 Teilnehmer haben den Fragebogen beendet. Die genaue Teilnahme über die gesamte Länge des Fragebogens (bzw. der „Dropout“) ist in Abbildung 2 dargestellt. Insgesamt bewegt sich der Verlust von Teilnehmern im normalen Bereich. Die Dauer zum Ausfüllen des Fragebogens betrug laut dem Umfrage-Tool durchschnittlich rund 20 Minuten.

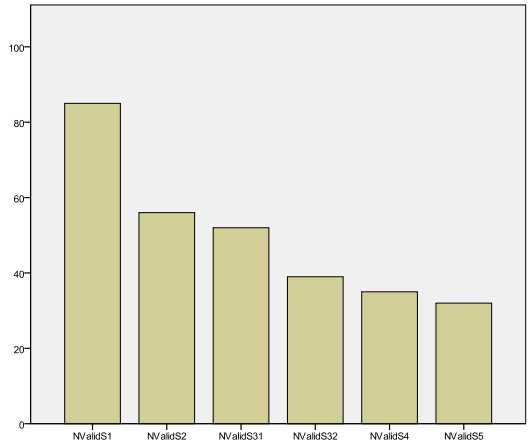


Abbildung 2. Gültige Teilnahmen pro Fragebogenabschnitt.

Die überwiegende Mehrheit der Teilnehmer wurde offenbar über die SAP Usergruppen akquiriert, hierauf deuten neben Abbildung 1 ebenfalls die http-Dereferrer. Hierbei handelt es sich nahezu ausschließlich um internationale Teilnehmer. 80 Prozent haben den Fragebogen auf Englisch ausgefüllt (Standardsprache). Das Herkunftsland wurde jedoch bewusst nicht als explizites Merkmal mit abgefragt. Abbildung 3 zeigt eine Aufschlüsselung der Teilnehmer nach Branchen.

Other: Software; IT services; IT Software Development; IT; Consumer; electrical and electronics; Music Industry; Petrochemicals; Consumer Packaged Goods; Local Government; Energy Management; Medical Devices; Conglomerate: Chemicals, autoparts, food, etc; HR and Payroll service

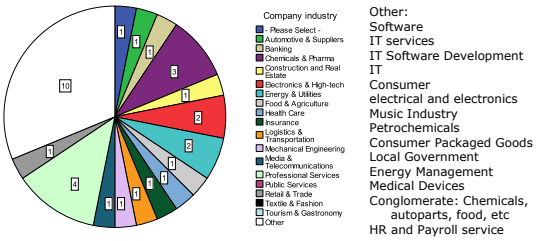


Abbildung 3. Teilnehmer-Branchen.

Die Teilnehmer wurden gebeten, zunächst Auskunft über die Gesamtheit aller SaaS-Anwendungen zu geben, die in Ihrem Unternehmen im Einsatz sind, bevor gezielte Fragen zu einer SaaS-Anwendung ihrer Wahl gestellt wurden. Abbildung 4 zeigt die verwendeten SaaS-Anwendungen nach Anbieter-Plattformen, welche in der Studie aufgeführt wurden. Bemerkenswert erscheint, dass nur 13 der 32 Befragten SAP On-Demand Lösungen einsetzen. Dies spricht für die breite Streuung der Studie, welche offenbar auch von zahlreichen SaaS-Interessierten im Umfeld der SAP wahrgenommen wurde, welche (noch) nicht notwendigerweise SAP Cloud-Kunden sind. Da bei dieser Frage Mehrfachnennungen möglich waren, lässt sich übrigens sagen, dass etwa die Hälfte der befragten Unternehmen (46%) mehr als eine einzige SaaS-Lösung im Einsatz hat.

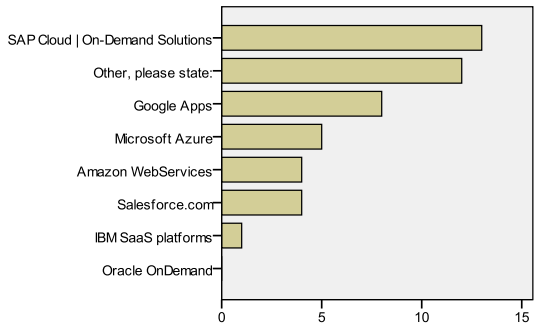


Abbildung 4. Verwendete SaaS-Plattformen (Mehrfachnennungen möglich).

Die im Einsatz befindlichen SaaS-Lösungen wurden ebenfalls funktional untergliedert, siehe Abbildung 5. Aufgrund der Stichprobe kann diese Rangreihung (wie alle anderen Merkmale auch) selbstverständlich nur als rein deskriptiv und nicht als repräsentativ im Sinne einer Marktstudie angesehen werden.

Ziel der Studie ist, verschiedene Aufteilung von Rollen und Verantwortlichkeiten im Bezug auf das Management der SaaS-Anwendung mit bestimmten Erfolgsdimensionen in Zusammenhang zu setzen, um somit Aussagen treffen zu können für welche SaaS-Anwendungen und welchen Umständen welcher Governance-Modus zielführend ist. Vorstudien

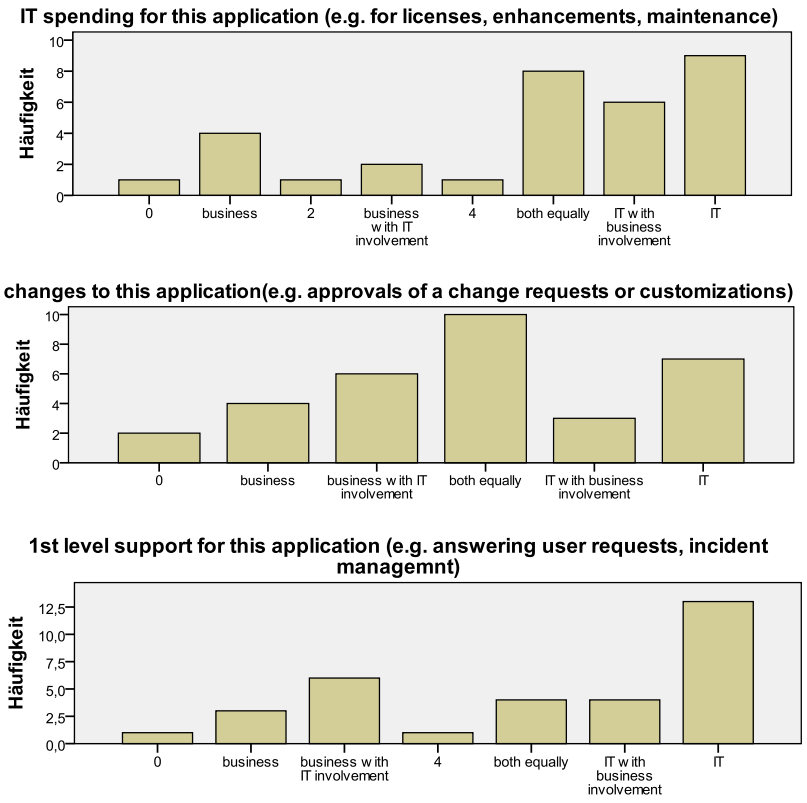


Abbildung 6. Ausgewählte Governance-Items (unabhängige Variablen).

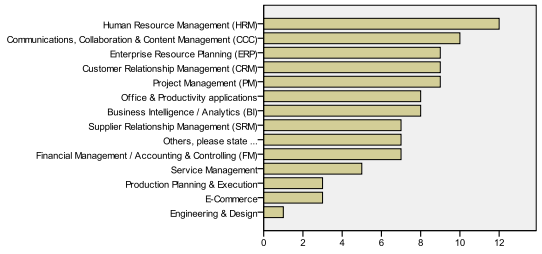


Abbildung 5. Verwendete Anwendungstypen (Mehrfachnennungen möglich).

der Humboldt-Universität legen nahe, dass durchaus große Unterschiede (d.h. statistisch gesehen eine Varianz) in der Governance-Frage existiert, von wem SaaS-Anwendungen gesteuert und gemanagt werden [1] [2]. Diese Vermutung lässt sich durch die vorliegenden Ergebnisse bereits bestätigen.

Abbildung 6 zeigt die Verteilungen einiger ausgewählter Governance-Items aus dem Fragebogen. Demzufolge treffen bei den Befragten offenbar in ca. 2/3 der Fälle IT-Bereiche die Hauptentscheidungen für das SaaS-Anwendungsmanagement (wie z.B. Budgetentscheidungen) und den internen Nutzersupport. In ca. 1/3 der Fälle liegt die Hoheit dagegen eindeutig im Fachbereich. Während die Budgetverantwortung interessanterweise klar geregelt zu sein scheint, kommt es bei Änderungsentscheidungen (Changes) offenbar wesentlich häufiger zu gemischten Formen, welche einen höheren Abstimmungs- und Koordinationsbedarf zwischen Fach- und IT-Bereichen nahelegen (Abbildung 6b).

Im Bezug auf die Erfolgsvariablen wurden die Teilnehmer auf der letzten Seite des Fragebogens gebeten, die jeweilige SaaS-Anwendung mit anderen herkömmlichen Anwendungen in ihrem Unternehmen zu vergleichen.

In Abbildung 7 ist die Verteilung einiger ausgewählter Erfolgsvariablen dargestellt. Interessanterweise lässt sich auch hier eine gewisse Varianz bei den Fragen erkennen (was für eine spätere inferenzstatistische Auswertung von Vorteil sein kann). Beispielsweise scheint es sowohl Beispiele in der Stichprobe zu geben, wo die Arbeitsaufteilung von IT und Fachbereichen zu vergleichsweise viel, als aber auch zu vergleichsweise wenig Gesamtaufwand im SaaS-Anwendungsmanagement führt. Ebenfalls ist ein positiver Ausschlag zu erkennen, was die wahrgenommene Effizienz und Zufriedenheit mit der SaaS-Anwendung angeht. Wohlgemerkt kann dieser allerdings auch auf einen positiven Response-Bias der Teilnehmer zurückzuführen sein (d.h. zufriedene Anwender nehmen u.U. eher an einer freiwilligen Befragung teil).

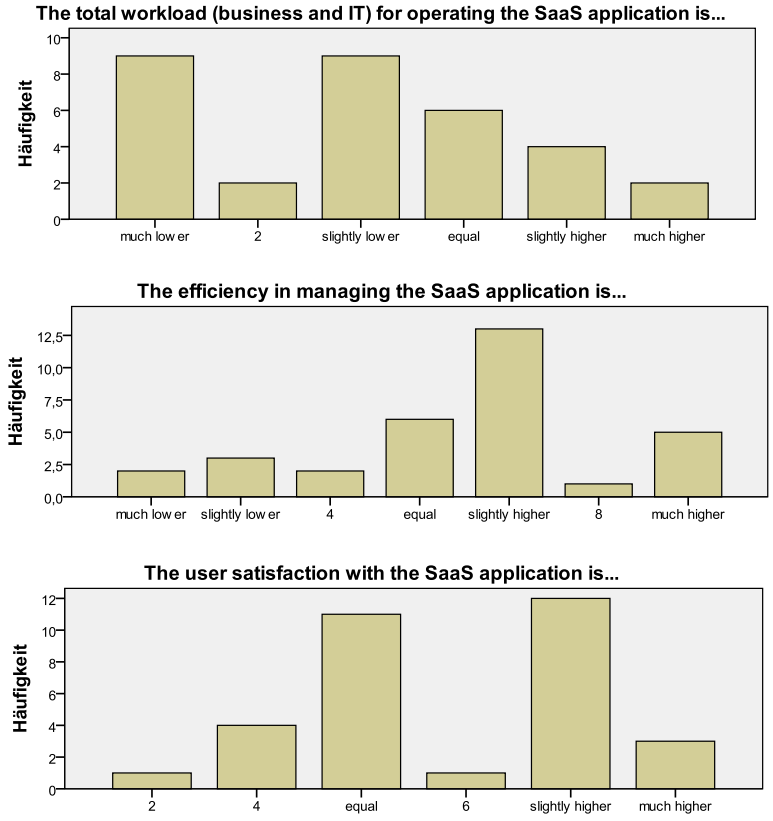


Abbildung 7. Ausgewählte Erfolgsvariablen (abhängig).

3 Nächste Schritte

Da die Stichprobengröße für weitergehende inferenzstatistische Analysen als zu klein erscheint, ist geplant die erhobenen Daten im nächsten Schritt mit denen einer weiteren Befragung zusammen zu führen. Dies verlangt zunächst einer umfangreicheren Datenbereinigung und -analyse sowie im Nachgang einer fundierten Reflexion der Ergebnisse anhand der wissenschaftlichen Literatur. Erwartet wird, dass somit u.a. belastbarere Ergebnisse bezüglich der hypothetisierten Zusammenhänge zwischen den dargestellten Governance- und Erfolgsvariablen erzielt werden können.

4 Danksagung

Wir danken dem HPI Soc Lab und dem Kooperationspartner SAP, insbesondere Herrn Henning Schmitz, für den Zugang zu den verschiedenen Kommunikationskanälen und die engagierte Unterstützung bei der Durchführung dieser Studie.

Literatur

[1] T. J. Winkler, C. Goebel, A. Benlian, F. Bidault, and O. Guenther. The impact of software as a service on is authority - a contingency perspective. *International Conference on Information Systems (ICIS 2011) Proceedings*, 2011.

[2] T. J. Winkler and O. Guenther. Explaining the governance of software as a service applications - a process view. *Multikonferenz der Wirtschaftsinformatik (MKWI 2012) Proceedings*, 2012.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
84	978-3-86956-274-2	**Anbieter von Cloud Speicherdiensten im Überblick**	Christoph Meinel, Maxim Schnjakin, Tobias Metzke, Markus Freitag
83	978-3-86956-273-5	**Proceedings of the 7th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Christoph Meinel, Hasso Plattner, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch (Hrsg.)
82	978-3-86956-266-7	**Extending a Java Virtual Machine to Dynamic Object-oriented Languages**	Tobias Pape, Arian Treffer, Robert Hirschfeld
81	978-3-86956-265-0	**Babelsberg: Specifying and Solving Constraints on Object Behavior**	Tim Felgentreff, Alan Borning, Robert Hirschfeld
80	978-3-86956-264-3	**openHPI: The MOOC Offer at Hasso Plattner Institute**	Christoph Meinel, Christian Willems
79	978-3-86956-259-9	**openHPI: Das MOOC-Angebot des Hasso-Plattner-Instituts**	Christoph Meinel, Christian Willems
78	978-3-86956-258-2	**Repairing Event Logs Using Stochastic Process Models**	Andreas Rogge-Solti, Ronny S. Mans, Wil M. P. van der Aalst, Mathias Weske
77	978-3-86956-257-5	**Business Process Architectures with Multiplicities: Transformation and Correctness**	Rami-Habib Eid-Sabbagh, Marcin Hewelt, Mathias Weske
76	978-3-86956-256-8	**Proceedings of the 6th Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
75	978-3-86956-246-9	**Modeling and Verifying Dynamic Evolving Service-Oriented Architectures**	Holger Giese, Basil Becker
74	978-3-86956-245-2	**Modeling and Enacting Complex Data Dependencies in Business Processes**	Andreas Meyer, Luise Pufahl, Dirk Fahland, Mathias Weske
73	978-3-86956-241-4	**Enriching Raw Events to Enable Process Intelligence**	Nico Herzberg, Mathias Weske
72	978-3-86956-232-2	**Explorative Authoring of ActiveWeb Content in a Mobile Environment**	Conrad Calmez, Hubert Hesse, Benjamin Siegmund, Sebastian Stamm, Astrid Thomschke, Robert Hirschfeld, Dan Ingalls, Jens Lincke
71	978-3-86956-231-5	**Vereinfachung der Entwicklung von Geschäftsanwendungen durch Konsolidierung von Programmier-konzepten und -technologien**	Lenoi Berov, Johannes Henning, Toni Mattis, Patrick Rein, Robin Schreiber, Eric Seckler, Bastian Steinert, Robert Hirschfeld
70	978-3-86956-230-8	**HPI Future SOC Lab - Proceedings 2011**	Christoph Meinel, Andreas Polze, Gerhard Oswald, Rolf Strotmann, Ulrich Seibold, Doc D'Errico
69	978-3-86956-229-2	**Akzeptanz und Nutzerfreundlichkeit der AusweisApp: Eine qualitative Untersuchung**	Susanne Asheuer, Joy Belgassem, Wiete Eichorn, Rio Leipold, Lucas Licht, Christoph Meinel, Anne Schanz, Maxim Schnjakin

ISBN 978-3-86956-276-6
ISSN 1613-5652