Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Map/Reduce

Large Scale Duplicate Detection

Prof. Felix Naumann, **Arvid Heise**

Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

What is Big Data?

“collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications” [http://en.wikipedia.org/wiki/Big_data]

→ terabytes, petabytes, in a few years exabytes

Challenges

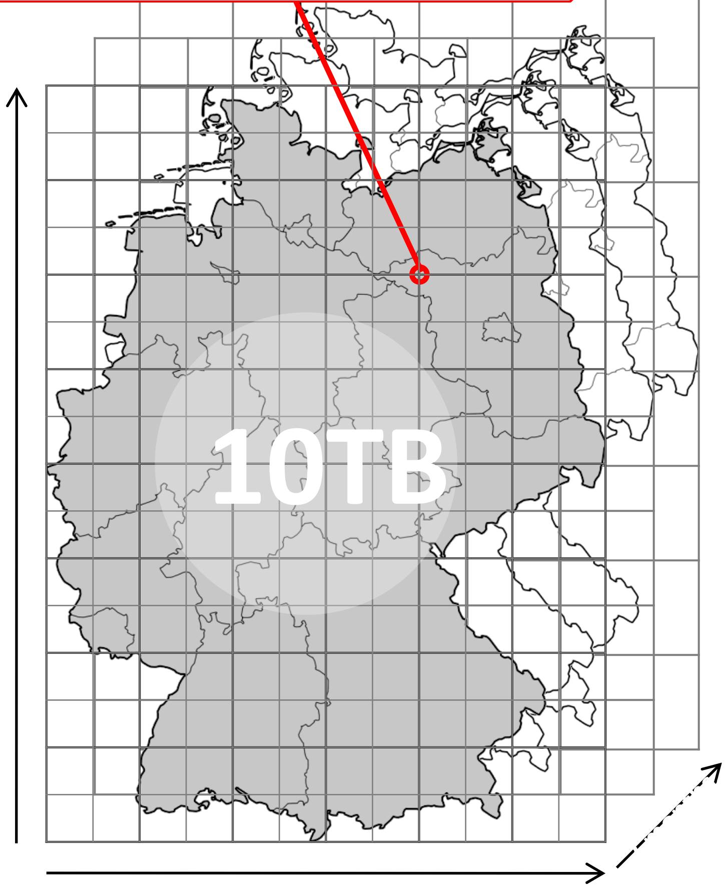
- Capturing, storage, analysis, search, ...

Sources

- Web, social platforms
- Science

Example: Climate Data Analysis

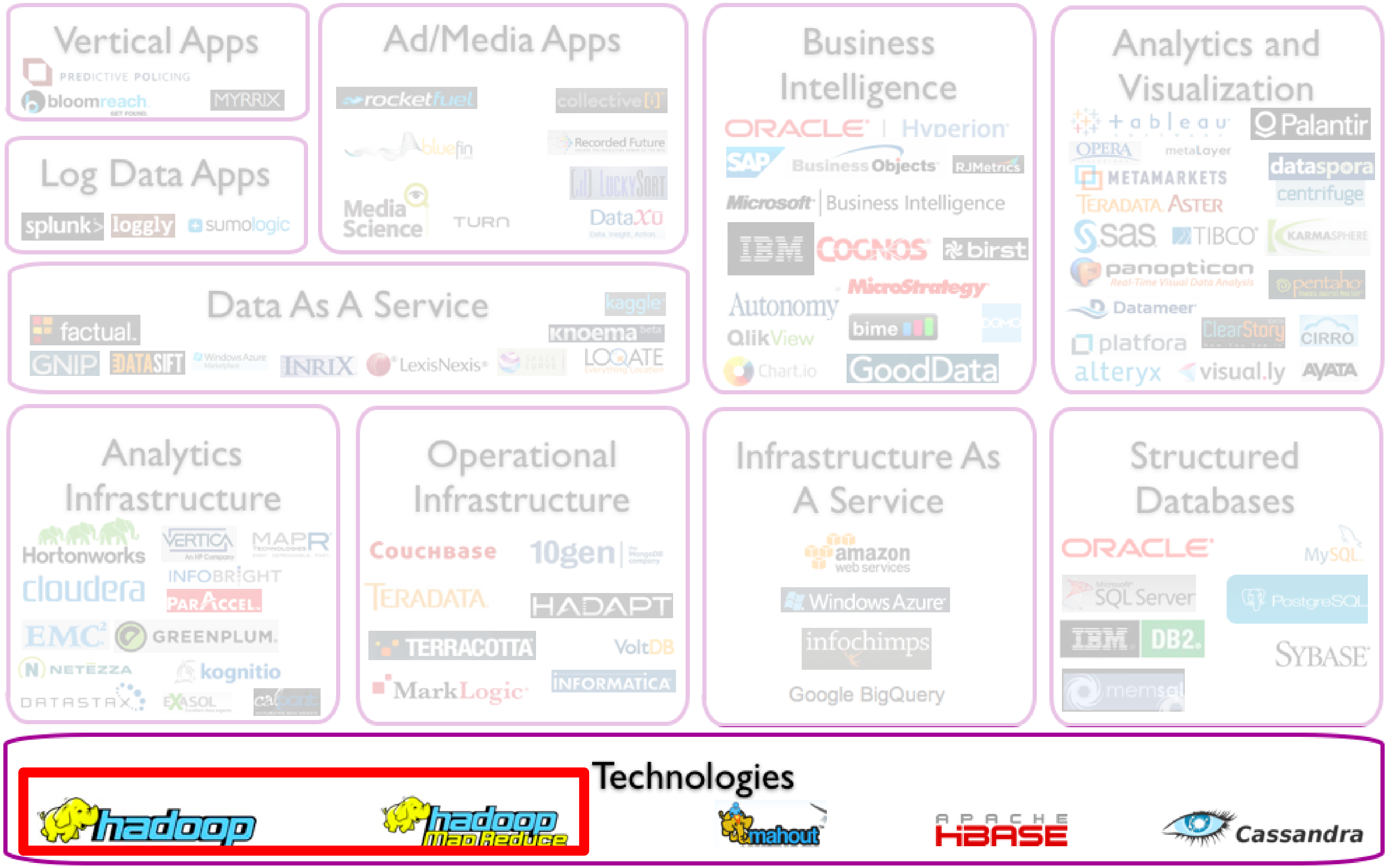
```
PS,1,1,0,Pa, surface pressure
T_2M,11,105,0,K,air_temperature
TMAX_2M,15,105,2,K,2m maximum temperature
TMIN_2M,16,105,2,K,2m minimum temperature
U,33,110,0,ms-1,U-component of wind
V,34,110,0,ms-1,V-component of wind
QV_2M,51,105,0,kgkg-1,2m specific humidity
CLCT,71,1,0,1,total cloud cover
…
(Up to 200 parameters)
```



- Analysis Tasks on Climate Data Sets
  - □ Validate climate models
  - □ Locate „hot-spots“ in climate models
    - ◇ Monsoon
    - ◇ Drought
    - ◇ Flooding
  - □ Compare climate models
    - ◇ Based on different parameter settings

- Necessary Data Processing Operations
  - □ Filter, aggregation (sliding window), join
  - □ Advanced pattern recognition

Big Data Landscape



Vertical Apps

Ad/Media Apps

Business Intelligence

Analytics and Visualization

Log Data Apps

Data As A Service

Analytics Infrastructure

Operational Infrastructure

Infrastructure As A Service

Structured Databases

Technologies

Copyright © 2012 Dave Feinleib dave@vcdave.com blogs.forbes.com/davefeinleib

Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

Programming Model

- Inspired by functional programming concepts map and reduce
- Operates on key/value pairs

Map

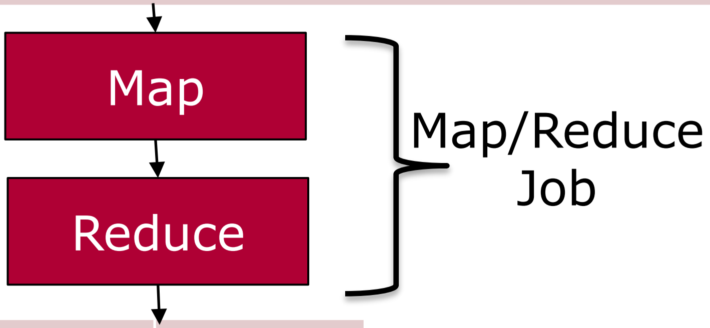
- Process key/value pairs individually
- Generate intermediate key/value pairs
- Example (LISP):
  (mapcar ’1+ ’(1 2 3 4)) ⇒ (2 3 4 5)

Reduce

- Merge intermediate key/value pairs with same key
- Example (LISP):
  (reduce ’+ ’(1 2 3 4)) ⇒ 10

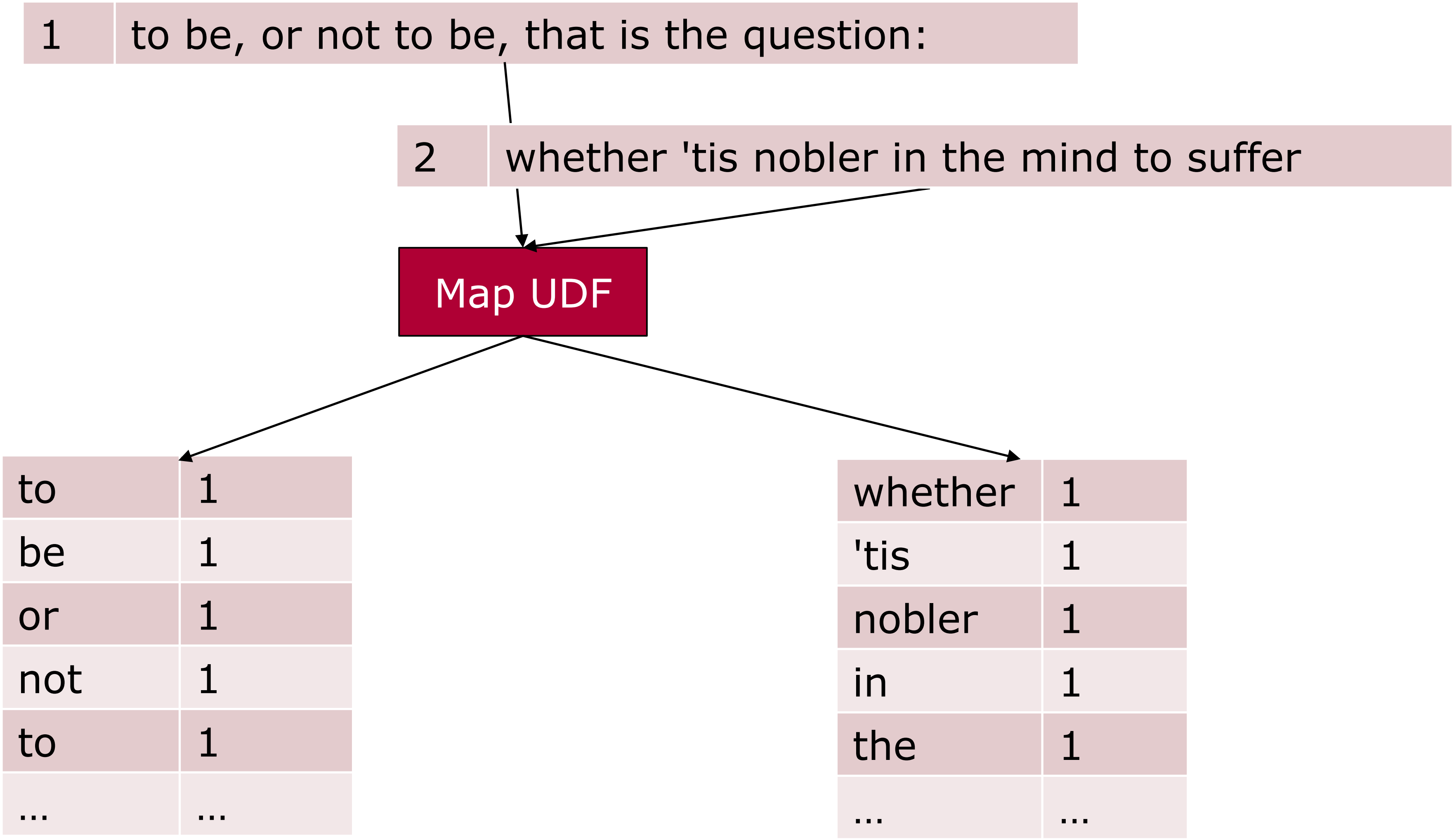
Programmer's Perspective: Word Count

1	to be, or not to be, that is the question:
2	whether 'tis nobler in the mind to suffer
3	the slings and arrows of outrageous fortune,
4	or to take arms against a sea of troubles
…	…



To	4
Be	2
Or	2
Not	1
…	…

Programmer’s Perspective: WC Map



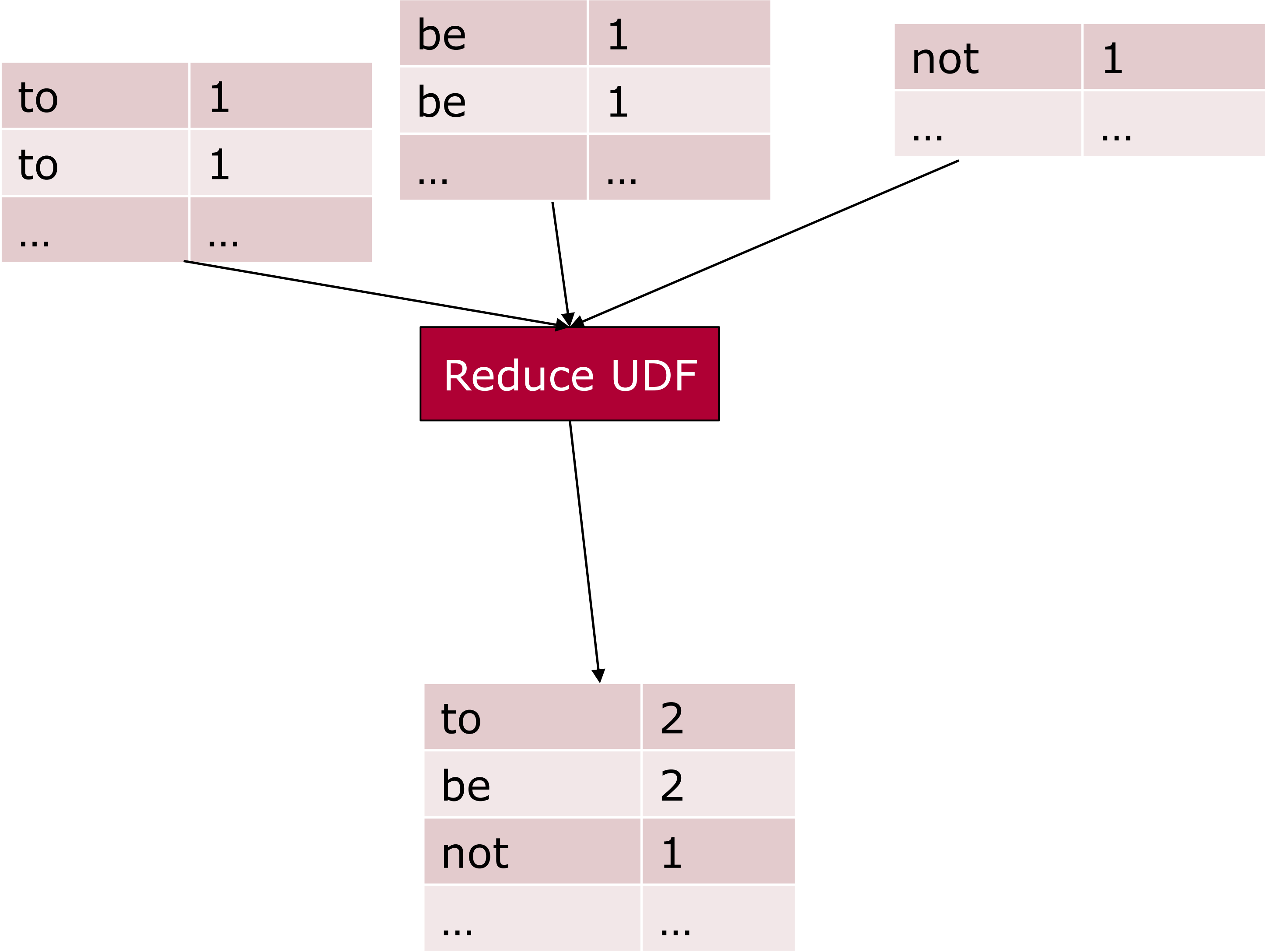
1	to be, or not to be, that is the question:
2	whether 'tis nobler in the mind to suffer

Map UDF

to	1
be	1
or	1
not	1
to	1
…	…

whether	1
'tis	1
nobler	1
in	1
the	1
…	…

Programmer’s Perspective: WC Reduce



to	1
to	1
…	…

be	1
be	1
…	…

not	1
…	…

Reduce UDF

to	2
be	2
not	1
…	…

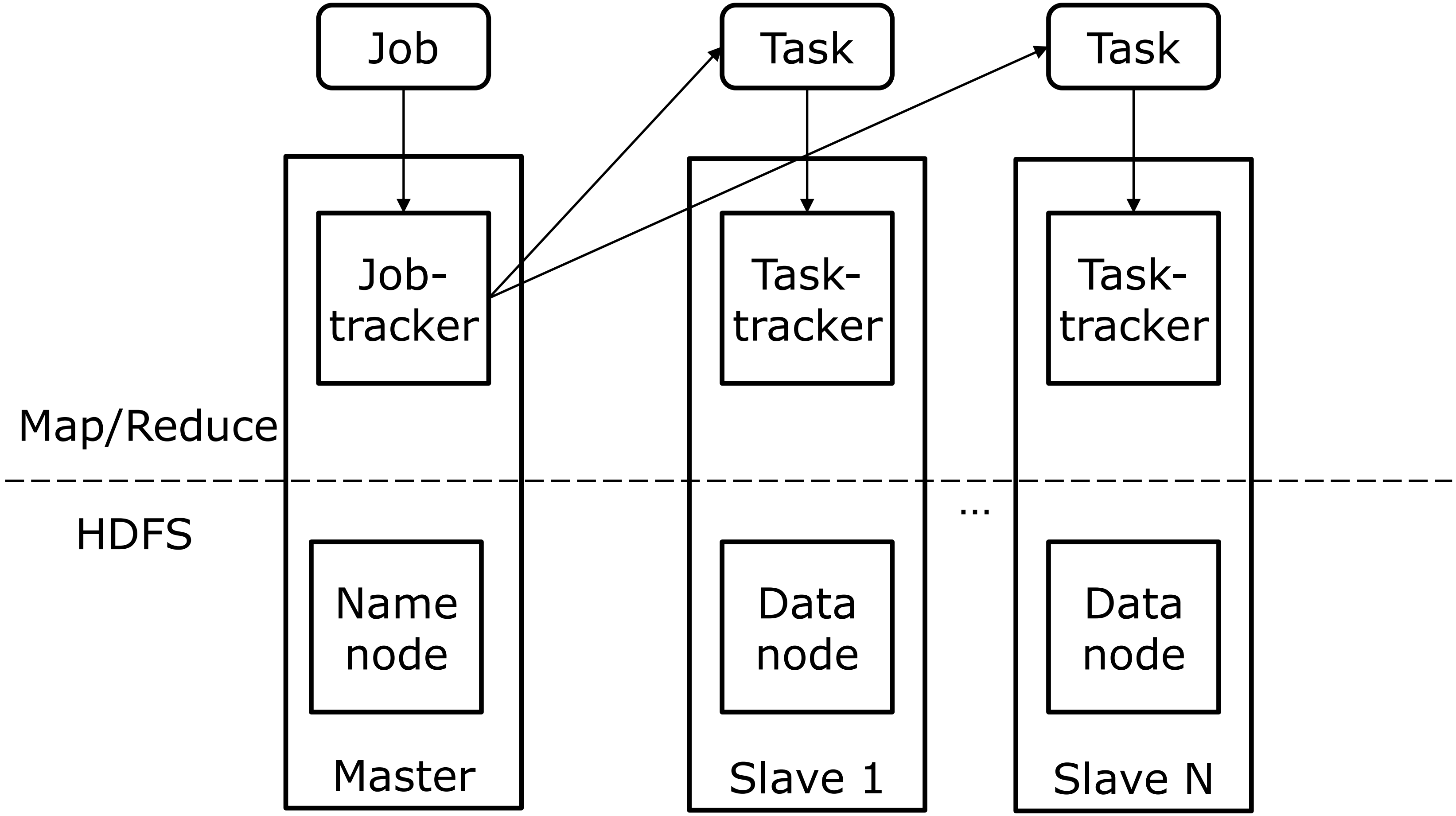
Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

Behind the Scenes

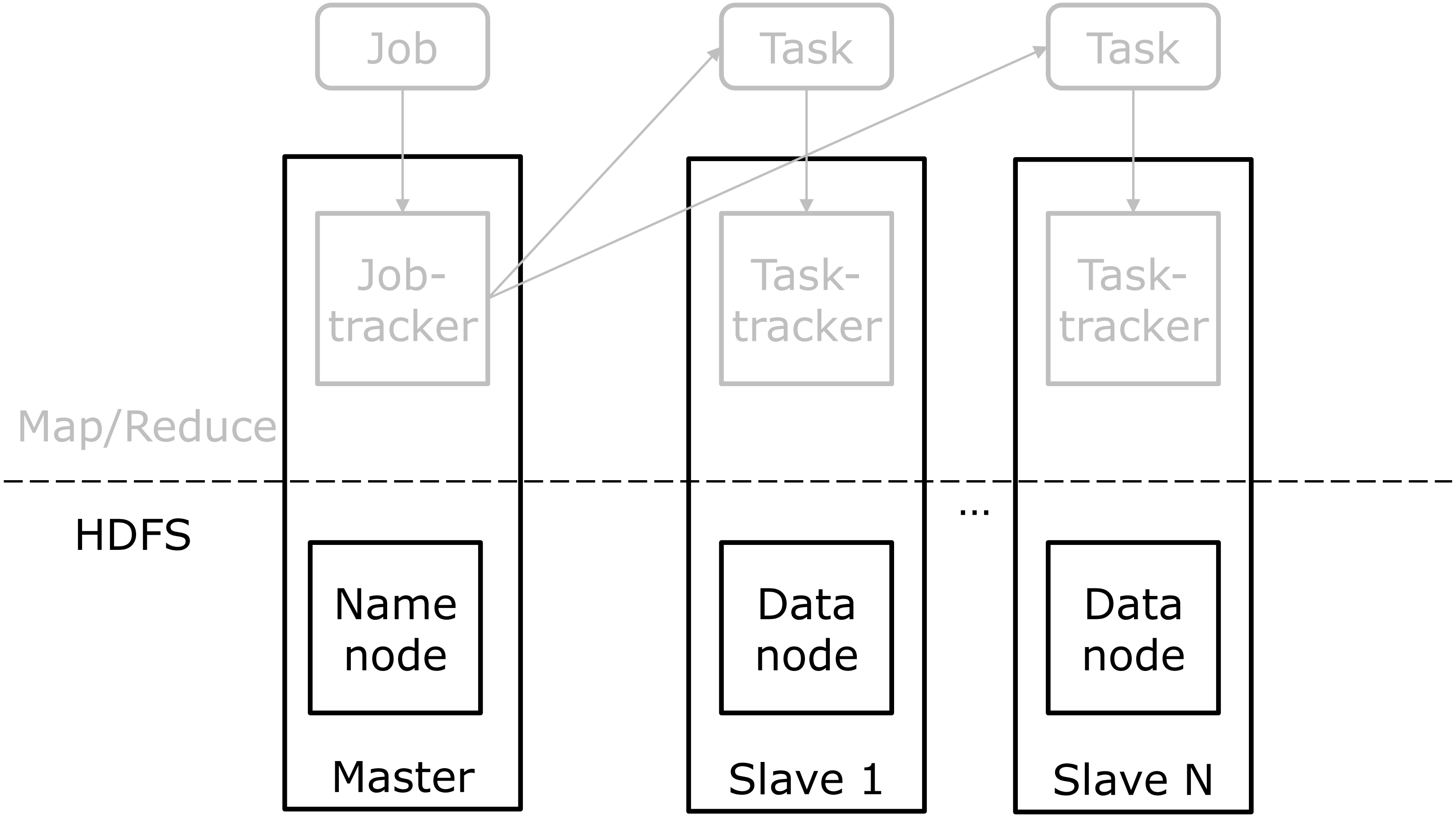
- Map/Reduce framework takes care of
  - Data partitioning
  - Data distribution
  - Data replication
  - Parallel execution of tasks
  - Fault tolerance
  - Status reporting

Hadoop Architecture



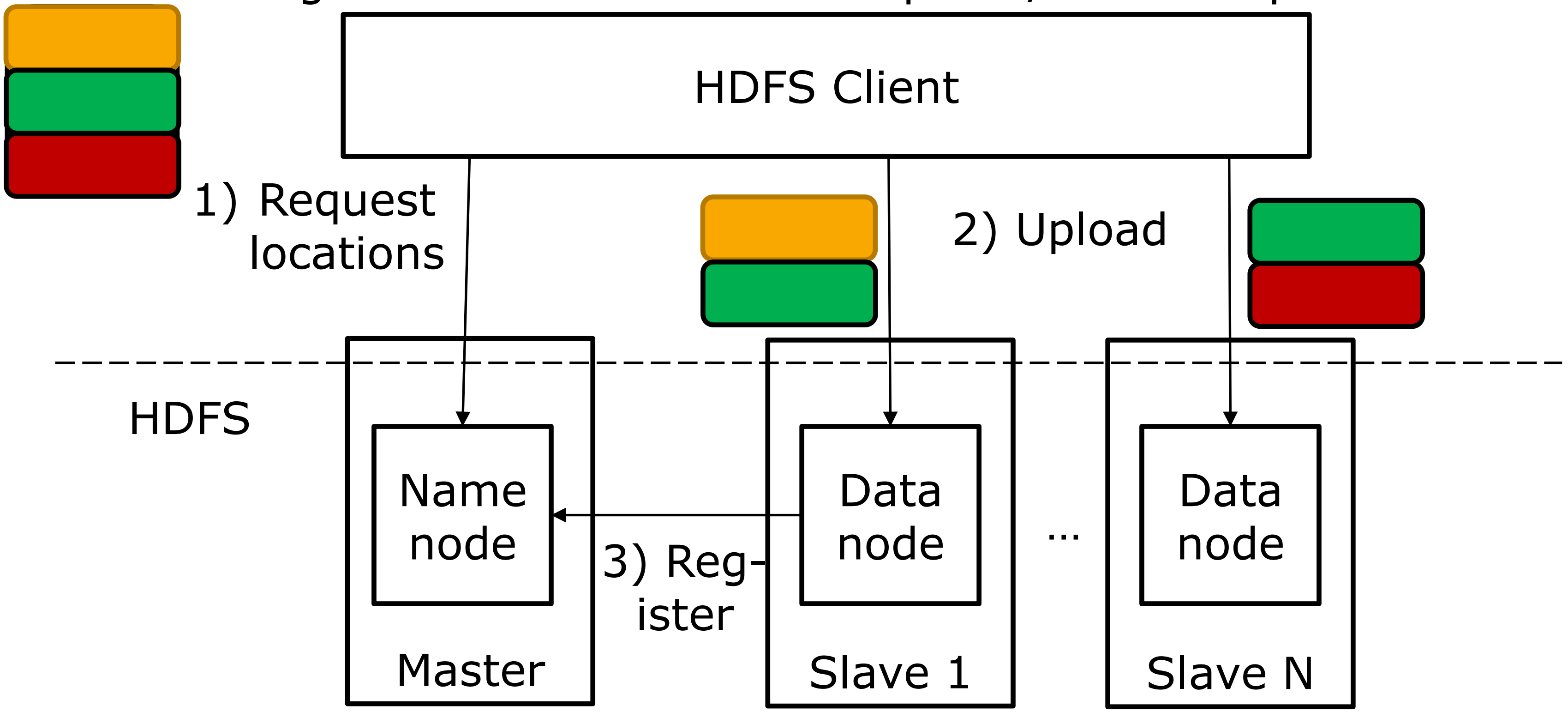
HDFS Upload

- First step: User uploads data to HDFS



HDFS Upload

- Block/split-based format (usually 64 MB)
- Splits are replicated over several nodes (usually 3 times)
- In average: each slave receives #Split*3/#Slaves splits

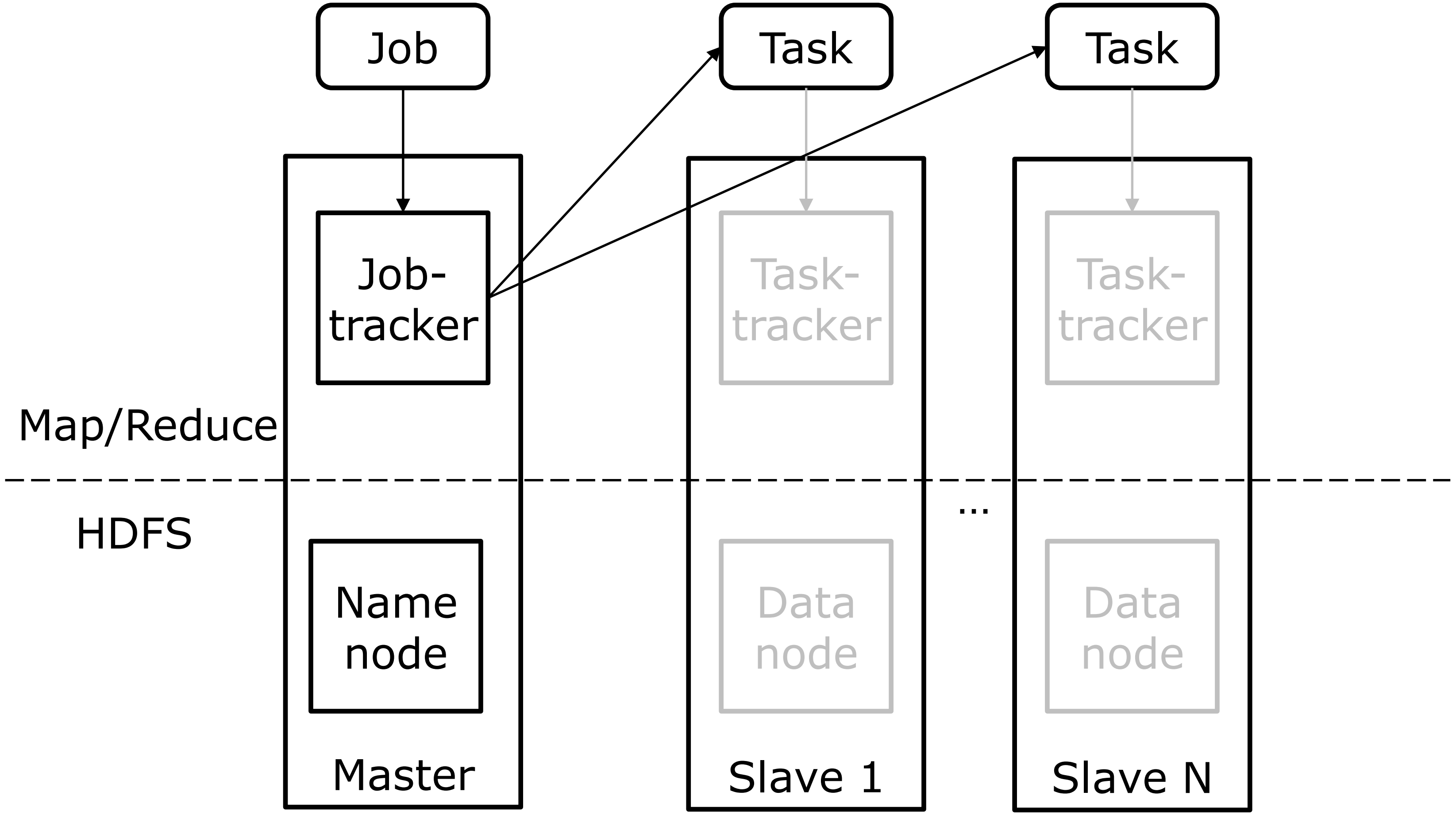


Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

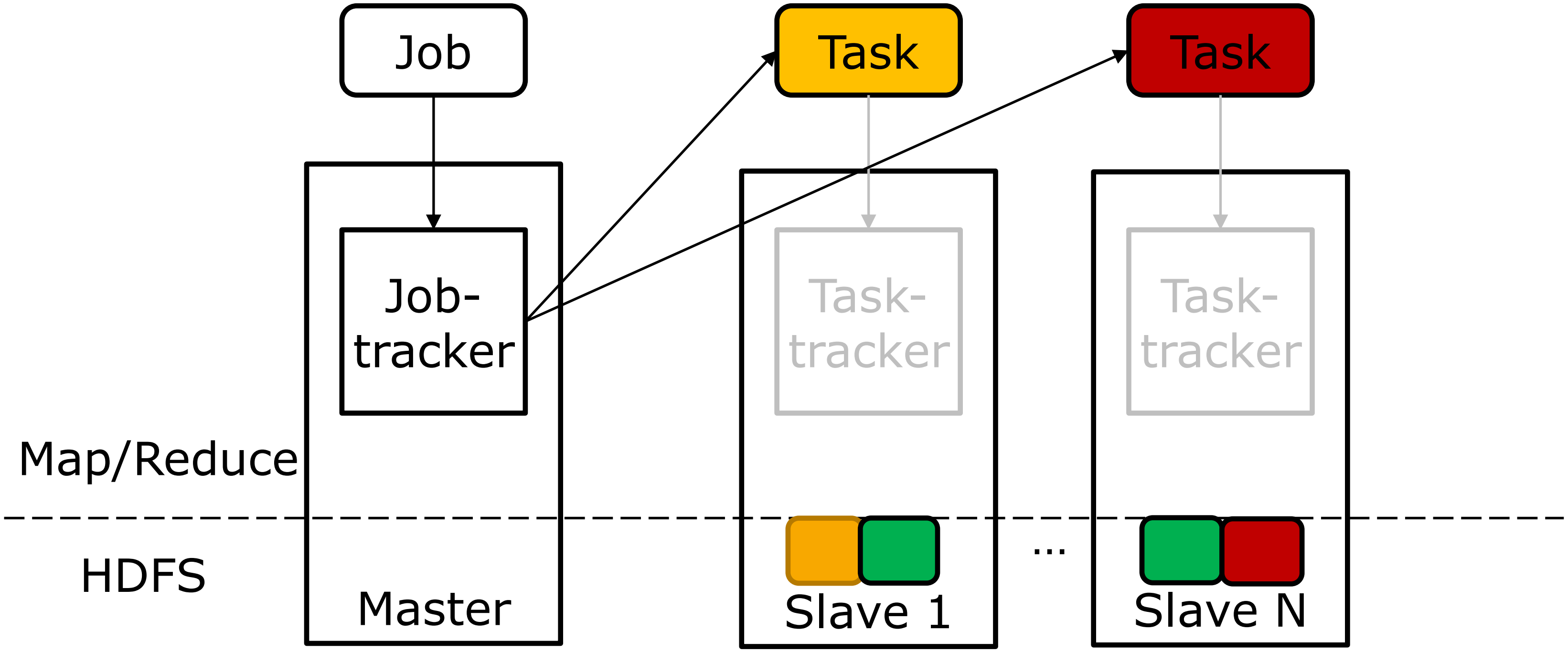
Job Submission

- Second step: User submits job



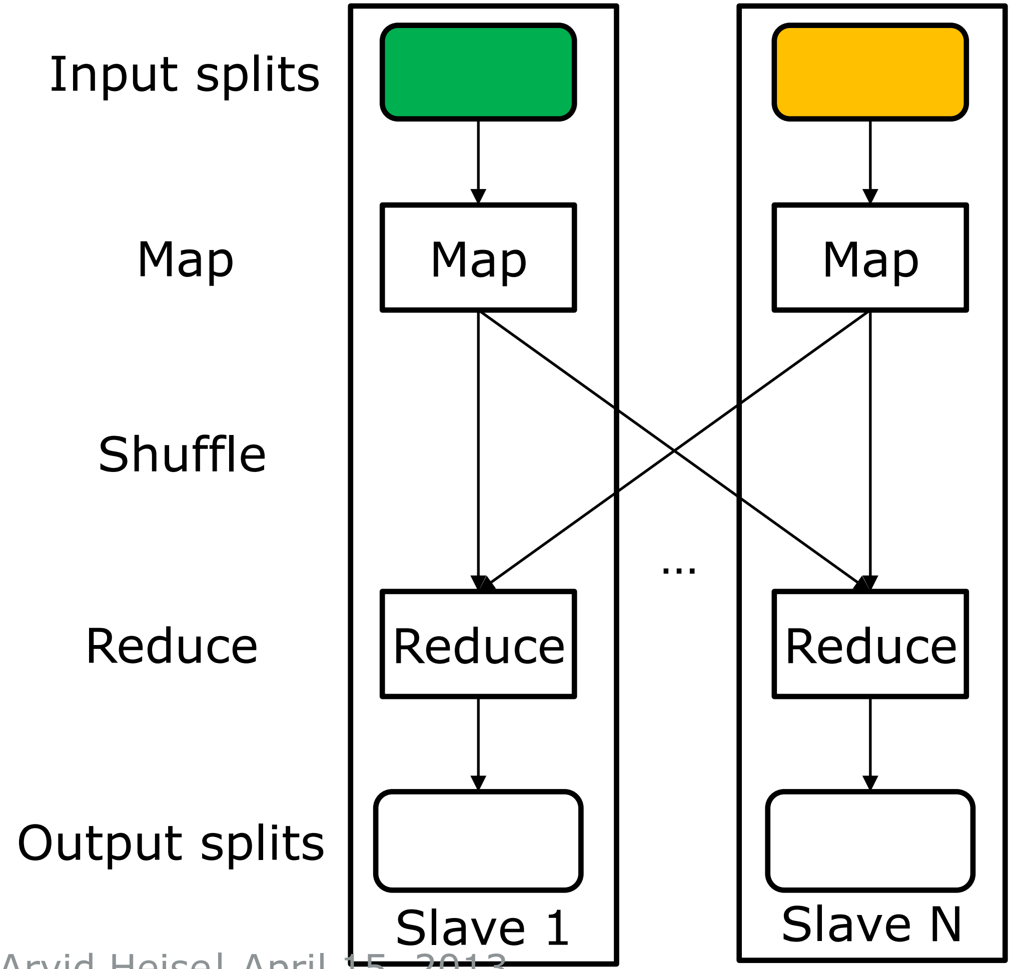
Job Submission

- Job tracker allocates resources for submitted job
- Uses name node to determine which nodes processes what
- Distributes tasks to nodes



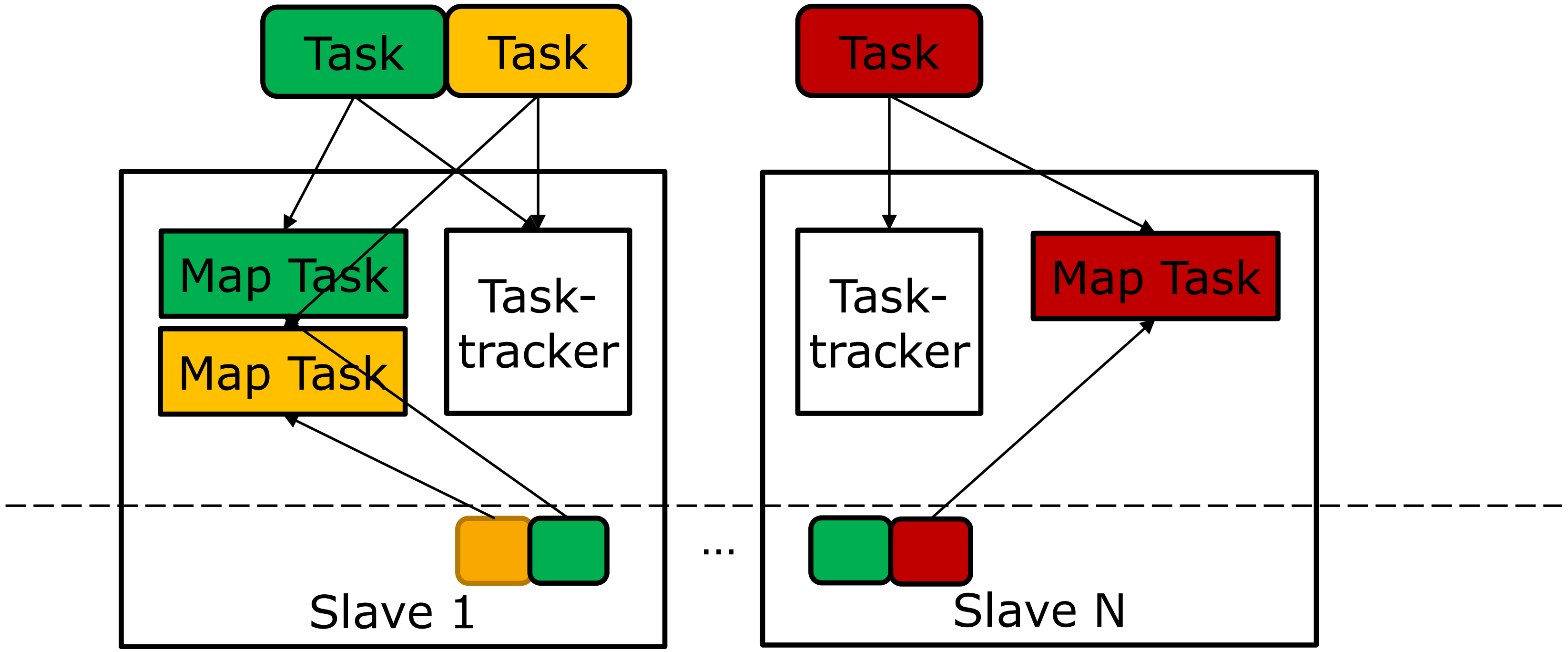
Job Execution

- Third step: job execution



Map tasks

- Third step: job execution, map task
- Nodes process tasks indepently
- Task tracker receives tasks and spawn one map process per task



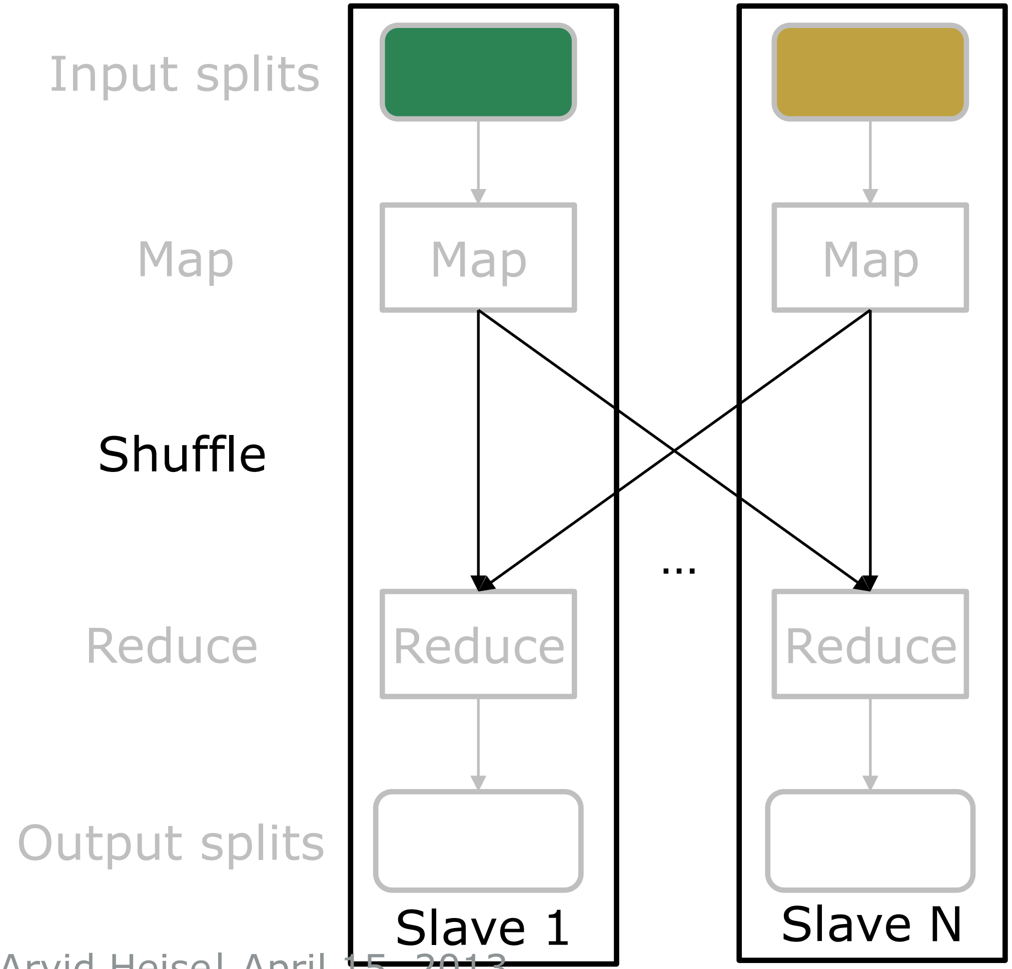
Map Execution

- Task tracker receives input as *map waves*
- Each wave consists of at most #processors splits
- Spawns a new JVM(!) for each split
- Each wave has at least ~6s overhead

- For each split, the map task reads the key value pairs
- Invokes the map UDF for each map task
- Collects emitted results and spills them immediately to a local file

- Optionally reuses JVM to reduce time per wave

Job Execution, Shuffle



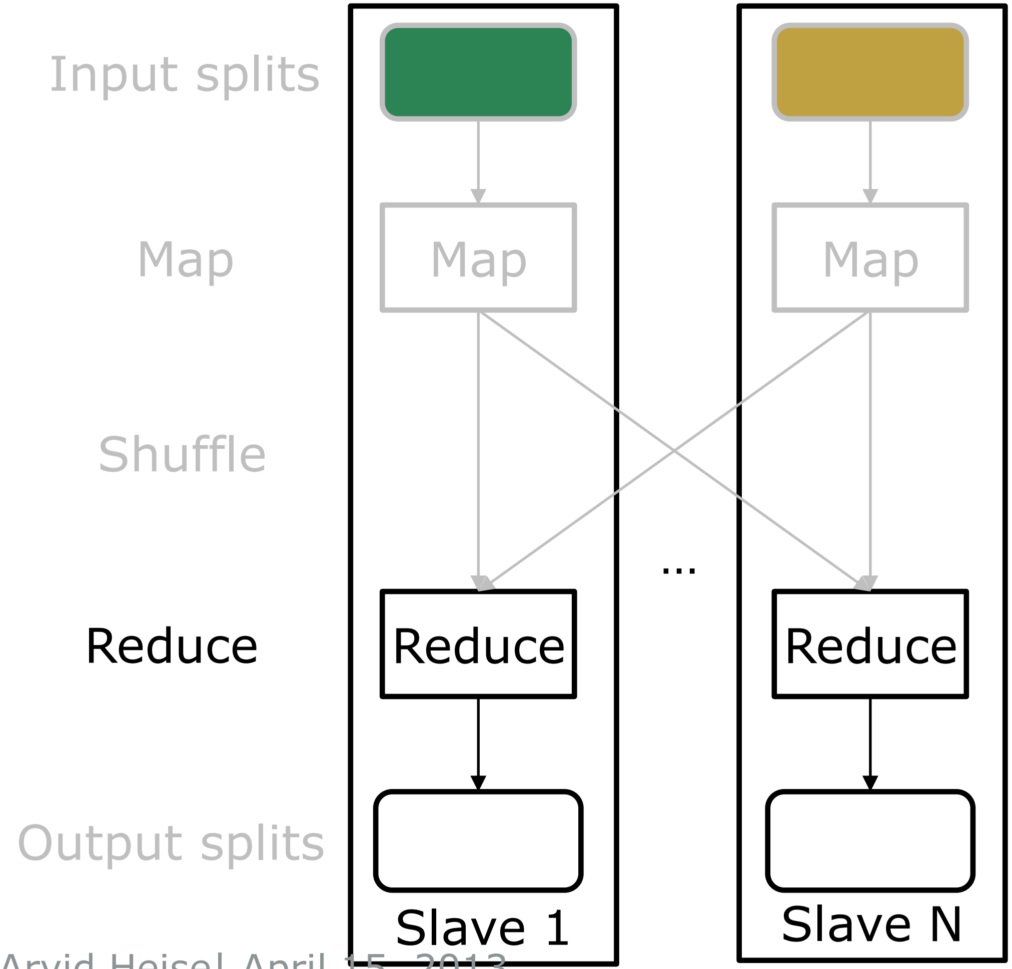
Shuffle

- Partitioner distributes data to the different nodes
  - Uses unique mapping from key to node
  - Often: key.hashCode() % numReducer

- Key/Value-pairs are serialized and sent over network
- Spilled to local disk of the reducer
- Sorted by key with two-phase merge sort

- Usually most costly phase

Job Execution, Shuffle



Reducer Execution

- Basic idea
  - Scans over sorted list
  - Invokes reducer UDF for subset of data with same keys
- In reality, a bit more complicated
  - Provides reducer UDF with iterator
  - Iterator returns all values with same key
  - UDF is invoked as long as there is one element left
  - Only one scan with little memory overhead
- Stores result on local disk
- Replicates splits (two times)

Combiner

- Local reducer
- Invoked in map phase for smaller groups of keys
  - Not the complete list of values in general
  - Preaggregates result to reduce network cost!

- Can even be invoked recursively on preaggregated results

Word Count Recap, Data Upload

- During upload, split input
- (In general, more than one line)

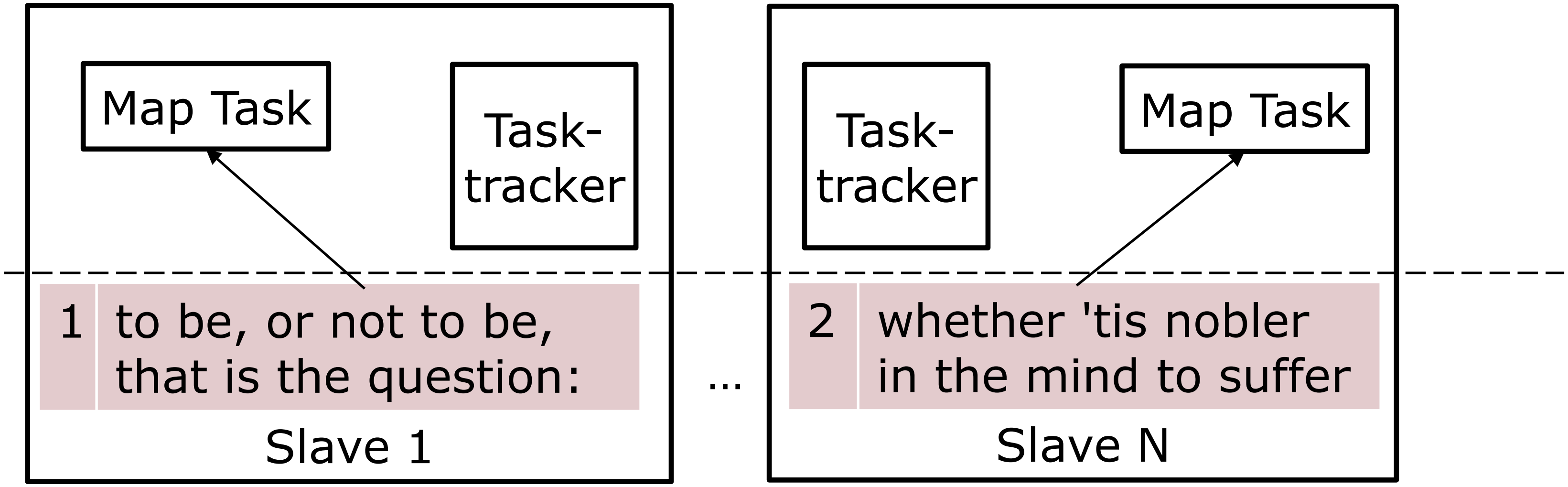
1	to be, or not to be, that is the question:
2	whether 'tis nobler in the mind to suffer
3	the slings and arrows of outrageous fortune,
4	or to take arms against a sea of troubles
…	…

1	to be, or not to be, that is the question:

2	whether 'tis nobler in the mind to suffer

Word Count Recap, Map Phase

- For each input split invoke map task
- Map task receives each line in the split
- Tokenizes line, emits (word, 1) for each word
- Locally combines results!
    - □ Decreases I/O from #word to #distinct words per split (64MB)



Word Count Recap, Shuffle+Reduce

- Assigns each word to reducer
- Sends all preaggregated results to reducer
    - For example, (to, 3512)
- Reducer sorts results and UDF sums preaggregated results up
- Each reducer outputs a partial word histogram

- Client is responsible for putting output splits together

Behind the Scenes

- Map/Reduce framework takes care of
  - □ Data partitioning
  - □ Data distribution
  - □ Data replication
  - □ Parallel execution of tasks
  - □ Fault tolerance
  - □ Status reporting

Fault Tolerance

On Map/Reduce level

- Each task tracker sends progress report
- If a node does not respond within 10 minutes (configurable)
  - It is declared dead
  - The assigned tasks are redistributed over the remaining nodes
  - Because of replication, 2 nodes can be down at any time

On HDFS level

- Each data node sends periodic heartbeat to name node
- In case of down time
  - Receives no new I/O
  - Lost replications are restored at other nodes

Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

Record Reader

- For WC, we used LineRecordReader
  - Splits text files at line ends (‘\n’)
  - Generates key/value pair of (line number, line)

- Hadoop users can supply own readers
  - Could already tokenize the lines
  - Emits (word, 1)
  - No mapper needed

- Necessary for custom/complex file formats
- Useful when having different file formats but same mapper

Dealing with Multiple Inputs

- Map and reduce take only one input
- Operations with two inputs are tricky to implement

- Input splits of map can originate in several different files
  - Logical concatenation of files
- Standard trick: tagged union
  - In record reader/mapper output (key, (inputId, value))
  - Mapper and reducer UDFs can distinguish inputs

Join

- Reduce-side join
  - Tagged union (joinKey, (inputId, record))
  - All records with same join key are handled by same reducer
  - Cache all values in local memory
  - Perform inner/outer join
    - Emit all pairs of values with different inputIds
  - May generate OOM for larger partitions

- Map-side join
  - Presort and prepartition input
  - All relevant records should reside in same split
  - Load and cache split
  - Perform inner/outer join

Secondary Grouping/Sort

- Exploit that partitioner and grouping are two different UDFs

- Map emits ((key1, key2), value)
- Partitioner partitions data only on first key1
- All KV-pairs ((keyX, ?), ?) are on the same physical machine
- However, reducer is invoked on partitions ((keyX, keyY), ?)

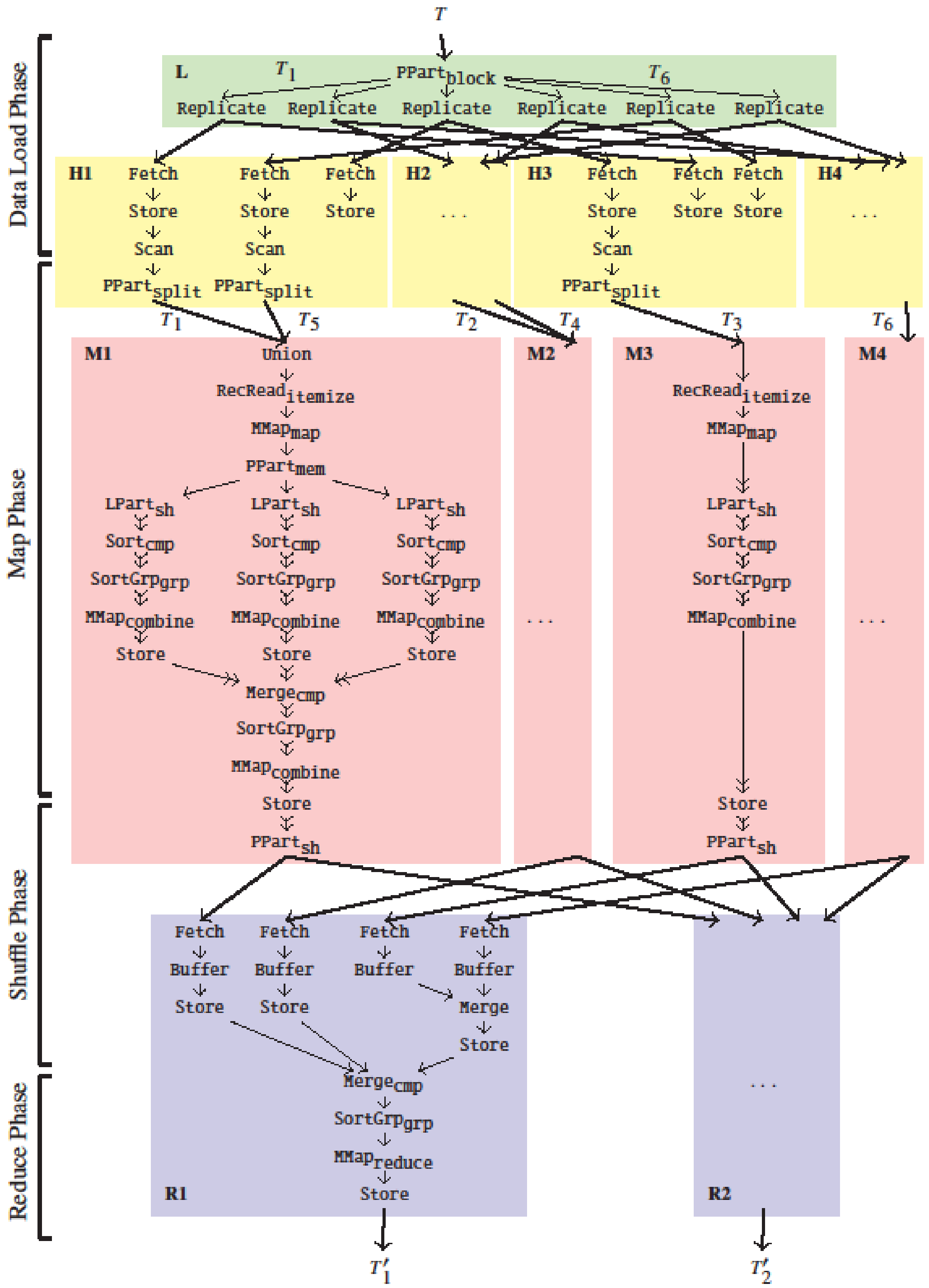
- Useful to further subdivide partitions
  - Join data could also be tagged ((joinKey, inputId), record)
  - Only need to cache one input and iterate over other partition
- Hadoop Reducer always sorts data
  - Data is grouped by first key and sorted by second key

Side-effect Files

- Sometimes even these tricks are not enough
- Example: triangle enumeration/three way join
- SELECT x, y, z WHERE x.p2=y.p1 AND y.p2=z.p1 AND z.p2=x.p1

- Cohen's approach with two map/reduce jobs
- Generate triad (SELECT x, y, z WHERE x.p2=y.p1 AND y.p2=z.p1)
- Probe missing edge with a reducer on input data
- Huge intermediate results on skewed data sets!

- Way faster: one map/reduce job
- Generate triad and immediately test if missing edge is in data
- Needs to load data set into main memory in reducer
- Might run into OOM



Complete pipeline in

Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, Jörg Schad. PVLDB 3(1): 518-529 (2010)

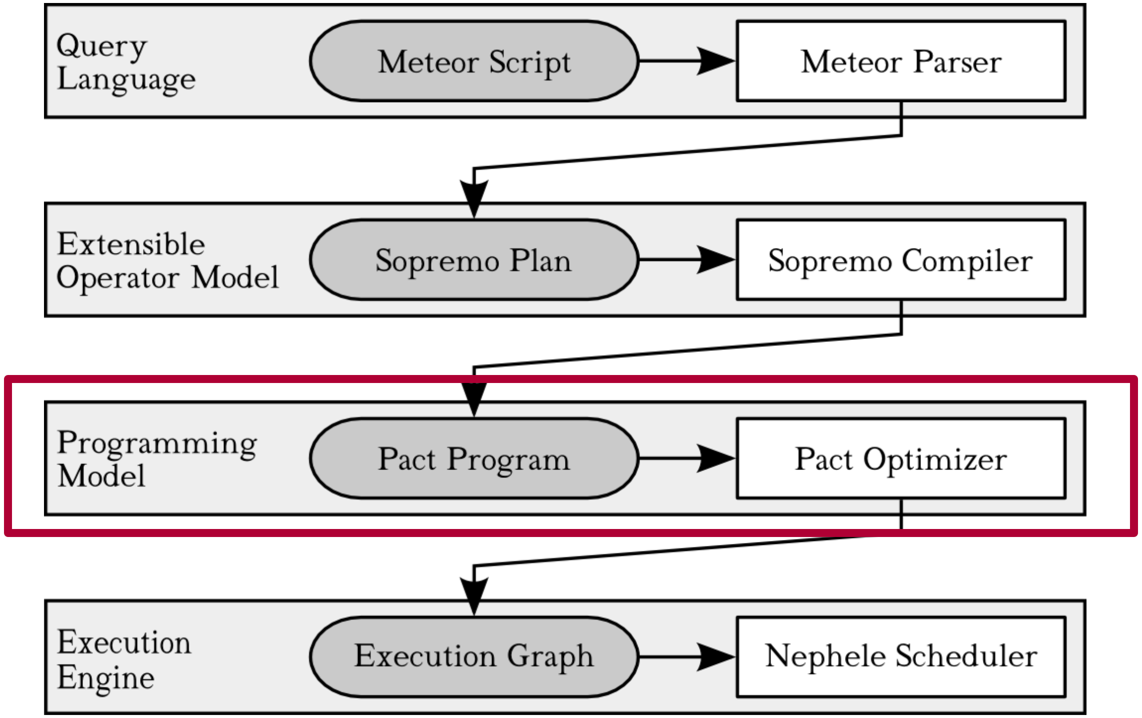
More than 10 UDFs!

Agenda

- Big Data
- Word Count Example
- Hadoop Distributed File System
- Hadoop Map/Reduce
- Advanced Map/Reduce
- Stratosphere

Overview over Stratosphere

- Research project by HU, TU, and HPI
- Overcome shortcomings of Map/Reduce
- Allow optimization of queries similar to DBMS



Extensions of Map/Reduce

- Additional second-order functions
- Complex workflows instead of Map/Reduce pipelines
- More flexible data model
- Extensible operator model
- Optimization of workflows
- Sophisticated check pointing
- Dynamic machine booking

Intuition for **Pa**rallelization **Cont**racts

Map and reduce are second-order functions

- Call first-order functions (user code)
- Provide first-order functions with subsets of the input data

Define dependencies between the records that must be obeyed when splitting them into subsets

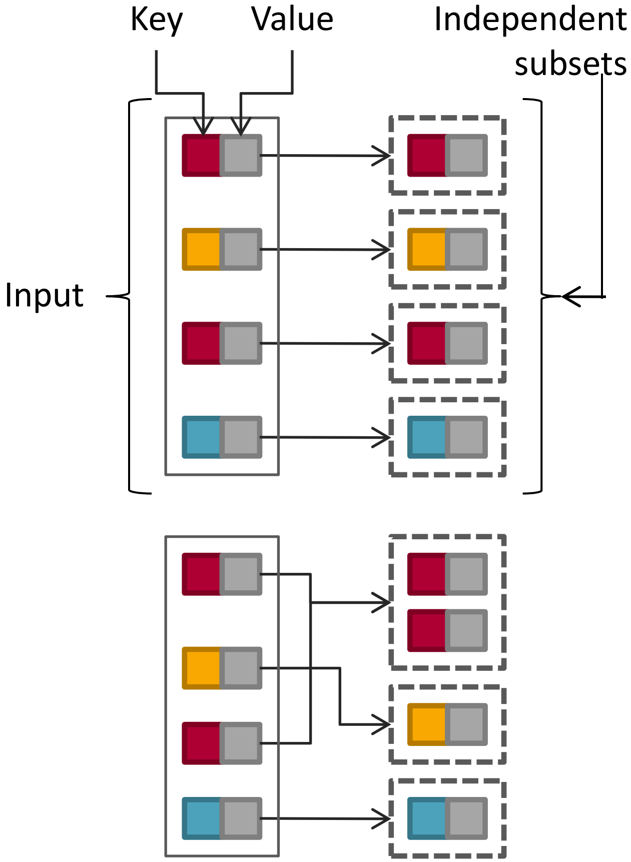
- Contract: required partition properties

Map

- All records are independently processable

Reduce

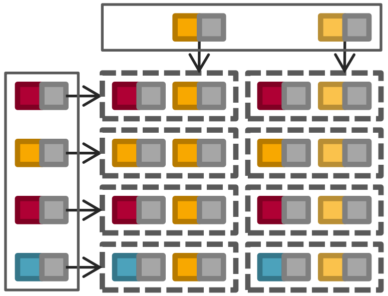
- Records with identical key must be processed together



Contracts beyond Map and Reduce

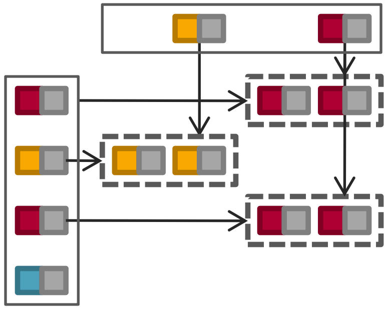
Cross

- Two inputs
- Each combination of records from the two inputs is built and is independently processable



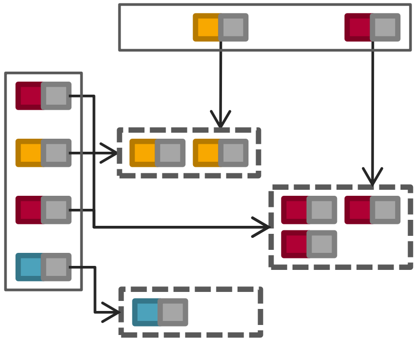
Match

- Two inputs, each combination of records with equal key from the two inputs is built
- Each pair is independently processable



CoGroup

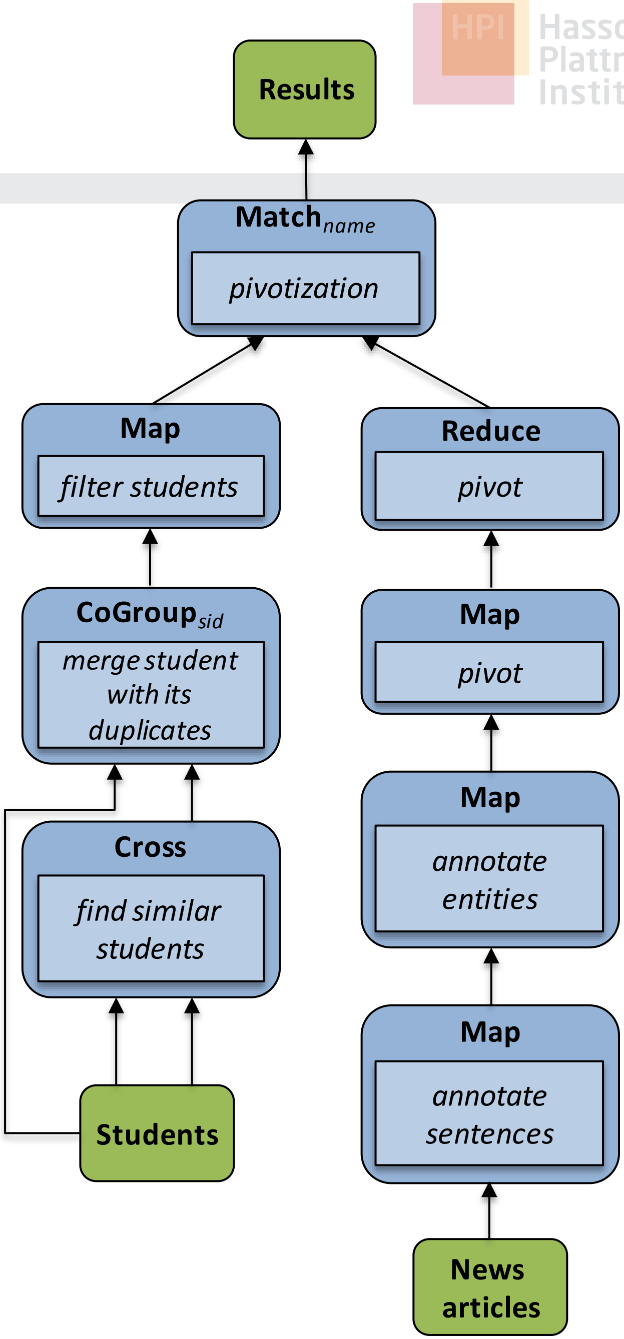
- Multiple inputs
- Pairs with identical key are grouped for each input
- Groups of all inputs with identical key are processed together



Complex Workflows

- Directed acyclic graphs
- More natural programming

- Holistic view on query
  - Map/Reduce queries scattered over several jobs
- Higher abstraction
  - Allows optimization
  - Less data is shipped



Motivation for Record Model

- Key/Value-pairs are not very flexible
- In Map/Reduce
  - Map performs calculation and sets key
  - Reducer uses key and performs aggregation
- Strong implicit interdependence between Map and Reduce

- In Stratosphere, we want to reorder Pacts
  - Need to reduce interdependence

- Record data model
  - Array of values
  - Keys are explicitly set by contract (Reduce, Match, CoGroup)

Record Model

- All fields are serialized into a byte stream
- User code is responsible for
  - Managing the indices
  - Knowing the correct type of the field

- Huge performance gain through lazy deserialization
  - Deserialize only accessed fields
  - Serialize only modified fields

Composite Keys

- Composite keys in Map/Reduce
  - New tuple data structure
  - Map copies values into the fields
  - Emits (keys, value)

- Stratosphere allows to specify composite keys
  - Reduce, Match, CoGroup can be configured to take several indices/types in the record as key

More Documentation

- Project website https://stratosphere.eu/
- **MapReduce and PACT - Comparing Data Parallel Programming Models**
  Alexander Alexandrov, Stephan Ewen, Max Heimel, Fabian Hueske, Odej Kao, Volker Markl, Erik Nijkamp, Daniel Warneke
  In Proceedings of Datenbanksysteme für Business, Technologie und Web (BTW) 2011, pp. 25-44