Processing Speed in Not-So-Big-Data Engineering Systems

When building data engineering systems speed often is one of the key goals, especially for interactive systems. This poster presents various aspects of **processing speed considerations** for these kinds of systems and introduces **Flare**, an Apache Spark engine replacement made to improve Spark's processing time to those of classical relational database systems while keeping most of it's flexibility.

Bottlenecks

A bottleneck occurs when a single system component is limiting the overall system performance. There are different bottleneck categories:

- **Compute-bound**: all CPU capacity exhausted
- **I/O-bound**: system is mostly waiting for data transfer from/to main memory, secondary storage or the network
- **Memory-bound**: a too small amount of main memory is limiting processing speed

Over the past decades, CPU-speed has been increasing faster then bandwidth, such that many classical approaches were memory-bound, leading to methodologies that bring computation closer to the data, like map-reduce and the development of algorithms for distributed systems.

Abstraction

Modern data engineering systems like Apache Spark don't only provide classic ACID guarantees like typical RDBS do, they also scale out to support parallel shared-nothing architectures and cope with node failures. Spark also supports domain specific languages and user defined functions that are especially useful for non-relational workloads as seen in Machine Learning applications.

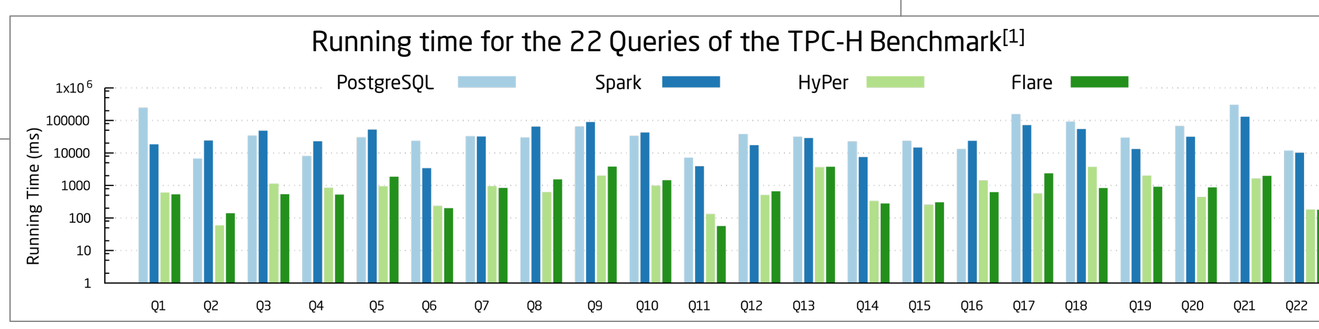
All this variety adds to a lot of abstraction and indirection, which creates a big overhead when processing big amounts of data. Spark is optimized to scale out, but *"[m]any times, big data is not that big, and often computation is the bottleneck"*[1] when running on a single machine while still utilizing Spark's querying flexibility.

TPC Benchmarks

The Transaction Processing Performance Council (TPC) is a non-profit consortium founded in 1988 consisting of various businesses in the IT sector. The TPC develops benchmark for transaction processing and database systems in the area of trading (C), banking (E), web (App) and decision-support (H) with the goal to improve comparability between solutions.[2]

[1] Essertel, Grégory & Tahboub, Ruby & Decker, James & Brown, Kevin & Olukotun, Kunle & Rompf, Tiark. (2017). Flare: Native Compilation for Heterogeneous Workloads in Apache Spark. (https://arxiv.org/pdf/1703.08219.pdf)

[2] TPC Homepage (http://www.tpc.org)



Flare

Flare is an Apache Spark backend meant to bring processing speeds for certain workloads up to par with best-of-breed query engines. The white paper presents 3 levels and concludes with a comparison based on TPC-H and heterogonous machine learning workloads.

A key assumption with Flare is that many workloads are in fact not *Google-scale* and thus can be reasonably run on a single sufficiently scaled up machine. [1]

Level 1 replaces Spark's Java Execution Environment with native C-code generation.

Level 2 drops support Sparks support for cluster execution, but is able to compile whole queries at once, reducing a big amount of overhead.

Level 3 uses *Lightweight Modular Staging* to generate Code from User Defined Functions.

(think ML kernel computations)

Felix Rindt
5. Semester BA-ITSE
based on Tiark Rompf's talk on "A Programming Language and Compiler View on Data Management and Machine Learning Systems", 14.01.2020

Ringvorlesung Practical Data Engineering
Winter Term 2019/2020
Hasso Plattner Institute, Potsdam, Germany