We Don't Know Enough to make a Big Data Benchmark Suite - An Academia-Industry View

Yanpei Chen, UC Berkeley / Cloudera
ychen2@eecs.berkeley.edu / yanpei@cloudera.com

1. Overview

Benchmarks facilitate performance comparison between equivalent systems. They inform procurement decisions, configuration tuning, features planning, deployment validation, and many other efforts in engineering, marketing, and customer support. Benchmarks are important when the underlying system enjoys sufficient maturity such that the priority moves beyond chaotic feature addition and debugging, and sufficient customers and vendors exist such that performance matters. Big data systems are entering this phase.

Characteristics of big data systems present unique challenges for benchmarking efforts. These include (1) system complexity, which makes it difficult to develop mental models, (2) use case diversity, which complicates efforts to identify representative behavior, (3) data scale, which makes it challenging to reproduce behavior, and (4) rapid system evolution, which requires benchmarks keep pace with changes in the underlying systems.

The position of this paper comes from an unprecedented empirical analysis of seven production workloads of MapReduce, an important class of big data systems. The main lesson we learned is that we do not know much about real life use cases of big data systems at all. Without real life empirical insights, both vendors and customers often have incorrect assumptions about their own workloads. Scientifically speaking, we are not quite ready to declare anything to be worthy of the label "big data benchmark." Nonetheless, we should encourage further measurement, exploration, and development of stopgap tools.

2. Real-life MapReduce Workloads

We collected large-scale, long-term MapReduce workloads traces at customers of Cloudera, a leading vendor of enterprise Apache Hadoop, and at Facebook, a leading Hadoop user. The workloads span diverse industries including social networks, e-commerce, media, telecommunications, and retail. Table 1 summarize the workloads. Key insights are:

- There is a new class of MapReduce workloads for interactive, semi-streaming analysis that differs considerably from the original use case of purely batch computations.
- There is a wide range of behavior within this workload class, such that we must exercise caution in regarding any aspect of workload dynamics as "typical".
- Some prior assumptions about MapReduce such as uniform data access, regular diurnal patterns, and prevalence of large jobs no longer hold.
- Workloads constantly evolve, such that even for the same cluster, design insights need to be periodically refreshed.

Trace	Machines	Length	Date	Jobs	Bytes moved
CC-a	<100	1 month	2011	5759	80 TB
CC-b	300	9 days	2011	22974	600 TB
CC-c	700	1 month	2011	21030	18 PB
CC-d	400-500	2+ months	2011	13283	8 PB
CC-e	100	9 days	2011	10790	590 TB
FB-2009	600	6 months	2009	1129193	9.4 PB
FB-2010	3000	1.5 months	2010	1169184	1.5 EB
Total	>5000	≈ 1 year	-	2372213	1.6 EB

Table 1: Summary of MapReduce traces. CC is short for "Cloudera Customer". FB is short for "Facebook". Bytes moved is input + shuffle + output data sizes for all jobs.

The full workload analysis is under publication review. To spur discussion, we list some quantitative observations:

- Data access patterns: Skew in data accesses range between an 80-1 and 80-8 rule. Temporal locality exists, and 80% of data re-accesses occur on the range of minutes to hours.
- Load variation over time: The cluster load is bursty and unpredictable. Peak-to-median ratio in cluster load range from 9:1 to 260:1.
- Common job types: All workloads contain a range of job types. Over 90% of all jobs are characterized by 10s of KB to GB of data, a range of data patterns between the map and reduce stages, and durations of 10s of seconds to a few minutes. Other job types appear with a wide range of frequencies.
- SQL-like programming frameworks: The cluster load that comes from Hive, Pig, and other such frameworks is up to 80% and at least 20%. Additional tracing at the Hive, Pig, and HBase levels is required.

3. Components of a Benchmark Suite

The diversity of workload behavior means that we should be extremely careful in considering any one aspect of the observed behavior to be "representative" and thus worthy of including in a benchmark. The following lists some challenges associated with building a general benchmark for MapReduce. Overcoming these challenges require additional scientific advances in understanding and quantifying the performance of large scale computer systems.

Pre-populating the data.

The data sizes, skew, and temporal locality all affect performance and therefore should be captured in a good benchmark. Such a benchmark needs to pre-generate data that reflects real life data access patterns.

Generating the processing stream.

A representative processing stream needs to capture the job sizes, shapes, sequences, and submission rate variations over time. It is non-trivial to understand which features of the processing stream we can safely omit for a large range of performance comparison scenarios.

Mix MapReduce and SQL-like frameworks.

Cluster management systems need to multiplex jobs written in the native MapReduce API and from SQL-like frameworks (Hive, Pig, and HBase). A benchmark should include both types of processing in realistic mixes.

Scaled-down workloads.

It is economically challenging to reproduce behavior at production scale. It is not clear what is the best way to "normalize" workload size (data size, number of jobs, or processing per data) against cluster size (number of nodes, CPU capacity, or available memory).

Empirical models.

The observed workload behavior do not fit well-known statistical distributions. Benchmarks should assume an empirical model of workloads, i.e., the workload traces *are* the model. This departs from some existing approaches [3], where the targeted workloads allow simple models to be used for generating the data and the processing stream.

A truly workload perspective.

Microbenchmarks that execute a small number of jobs one at a time are useful for diagnosing a system subject to very specific processing needs (terasort, Gridmix 1 and 2, HiBench, Hive Benchmark, Pigmix, the benchmarks from [2]). A general MapReduce benchmark should treat a workload as a steady processing stream involving complex and time-varying superposition of many concurrent jobs.

Workload suites.

We should accept that no single set of behaviors is representative. Hence, the benchmark should consist of a suite of multiple workloads. Systems could trade optimized performance for one workload type against more average performance for another, and customers should select solutions targeting their particular workload.

A stopgap tool used at Cloudera.

Cloudera uses a large set of tools for quality assurance, performance testing, and deployment certifcation. One tool is the Statistical Workload Injector for MapReduce (`https://github.com/SWIMProjectUCB/SWIM/wiki`). This is a joint academia-industry effort, with considerable assitance from collaborators at Facebook, and the original developers currently spread between UC Berkeley, Cloudera, VMware, and Splunk. SWIM is New BSD Licensed, and partially address the above challenges. It can pre-populate HDFS using uniform synthetic data, scaled to the number of nodes in the cluster, and replay the workload using synthetic jobs. The methodology is further discussed in [1]. The public SWIM repository includes scaled-down versions of the `FB-2009` and `FB-2010` workloads. An internal version also includes the Cloudera workloads. We are contacting the end customers to seek permission to make public their workloads.

4. Recommendations

It is important to "do the right thing." What is scientifically correct ultimately helps the entire community - customers, vendors, researchers. We advocate a cautious approach to developing a big data benchmark. Something so ambitiously phrased will guide or mis-guide development for a large range of systems and over a long period of time. Our observations of several real life workloads compel us to acknowledge that the science is not quite there yet to declare anything as "*the* big data benchmark." We recommend the following to close the existing knowledge gap.

Share empirical knowledge.

A big data benchmark should be built on empirically substantiated insights. The bar for inclusion in the benchmark suite should be higher than qualitative and unverifiable claims of "our customers do thus." The complexity, diversity, scale, and rapid evolution of big data systems imply that both customers and vendors often have incorrect or outdated assumptions about workload behavior. Therefore, quantitative, empirical data is essential to identifying the set of common use cases. A neutral, cross-organization, joint academia-industry consortium could facilitate data sharing.

Adopt a systems view of the workload.

We should capture workload behavior at the highest level conceptual boundaries of the underlying system, e.g., MapReduce jobs, Hive queries. This allows comparison between systems of the same type, e.g., between MapReduce systems from different vendors. We call this the *system view* of the workload. Tracing capabilities often exists at this level.

Alternate workload views are possible but problematic. The *physical view* captures hardware behavior (CPU, memory, disk, network). It allows hardware tuning of a particular system deployment, e.g., a cluster, but precludes comparison between deployments. The *functional view* captures the system-independent user intent, e.g., the desired outcome of the computation or data management process. It allows comparison between different system types, e.g., between a RDBMS and a MapReduce system. We currently lack knowledge to describe workload behavior at this level.

Advance performance science.

Further advances in performance science will overcome benchmarking challenges such as the lack of rigorous method to scale down a workload or the current inability to capture the function view of workload behavior. Breakthroughs in these areas likely require joint academia-industry efforts - industry supply real life data, use cases, experiences, requirements, and academia combine insights across companies and industries to distill concepts and formulate frameworks. Collaboration should be encouraged.

Learn from existing benchmarking communities.

The desire for rigorous performance comparisons predates the rise of big data. Some established systems have widely accepted benchmarks and qualify for the loosely defined label of "big data." These systems include relational databases, network file systems, scientific computing, and others. Lessons learned from these communities should inform the scientific, engineering, organizational, and community aspects of developing big data benchmark suites.

5. References

[1] Y. Chen et al. The Case for Evaluating MapReduce Performance Using Workload Suites. In *MASCOTS 2011*.

[2] A. Pavlo et al. A comparison of approaches to large-scale data analysis. In *SIGMOD 2009*.

[3] Transactional Processing Performance Council. TPC-* Benchmarks. `http://www.tpc.org/`.