Benchmarking Abstractions

Len Wyatt
Microsoft
LenWy@Microsoft.com

A key use for benchmarks is to help technology users ("customers") make technology choices. I assert that there are two classes of Big Data customer organizations to consider:

- Technology specialist organizations ("implementers"), which generally have a specific application area and the expertise to develop custom applications and hardware environments. This is the group that gave rise to the Big Data movement. It is not an accident that Big Data grew out of scientific circles and large web companies, where there was high motivation to solve specific problems.
- Technology user organizations ("consumers"), which more commonly need to make choices among off-the-shelf technologies (whether commercial or open source) to deploy. These organizations need expeditious solutions for their Big Data needs; in many cases a "good enough" solution that can be expeditiously deployed is preferable over an optimal solution that takes longer.

For purposes of this discussion, I will stipulate that there are many potential benchmarks of specific system components that may be useful to implementers. The computer industry has a long history of benchmarks of system components that are very useful to specialists. Classic examples would be the Dhrystone or Whetstone benchmarks of arithmetic performance in a CPU, but others exist for a wide array of components, including most of the SPEC benchmarks. I have complete confidence in the group of people gathered at this Workshop on Big Data Benchmarking to come up with a variety of interesting and useful benchmarks for this group.

The concern I wish to address is: How can we use benchmarks to help the consumers? Another class of benchmarks in the computer industry is benchmarks designed to measure the performance of complete systems ("end-to-end") under a workload that is intended to be representative of a real-world need. The TPC benchmarks all fall in this category, the best known being TPC-C for transaction processing systems and TPC-H for decision support systems. These benchmarks are intended to help consumers evaluate technologies.

An issue with benchmarks such as the TPC benchmarks is that vendors will go to extraordinary lengths to create optimal configurations for the benchmark requirements. Often these configurations do not reflect the realities of customers. In fact, there is a widely held myth that benchmarks are meaningless for this very reason.

This issue of vendor optimization is compounded by the choices of tools that consumers will deploy. For example the Hadoop ecosystem is enriched by languages such as HiveQL and Pig Latin. These and other tools that are layered on the Map/Reduce framework are much more tractable ways for analysts, data scientists, and even developers to work with large data sets. They are clearly less efficient than custom-coded M/R jobs, but users prefer the higher-level tools because they are far more productive with them. People time is optimized in preference to machine time. But what does that mean for benchmark development? Should a benchmark be developed purely for the task to be done, or for the tool in which the task is done?

Consider a trivial benchmark that might be defined to create a simple aggregation over a huge data set in HDFS. For that problem, a vendor implementing the benchmark would most likely define a custom M/R job since that is the highest-performance solution. However, a customer might prefer to use Hive for this, a level of abstraction that is more efficient for the user. To that customer, the vendor benchmark based on an M/R job might not seem to be relevant[1]. Alternatively, one might specify the benchmark using HiveQL. In the fast-changing world of Big Data, one has to wonder whether HiveQL has a sufficiently precise and stable definition that it is productive to define benchmarks on it. In addition, another customer might prefer to use Pig for the exact same problem. If the benchmark is defined using HiveQL, then a Pig Latin solution to the same problem would be a different benchmark!

This example is meant to illustrate a point: In defining a benchmark, it is important to use a level of abstraction that has flexibility to accommodate a variety of tools, durability to survive the fast rate of change in Big Data, and utility to represent workloads that are meaningful to consumers.

A related reason why customers adopt technologies like Hadoop is the reduced time to prototype and time to solution. Again, people time is optimized in preference to machine time. But again, benchmark implementers typically take the exact opposite approach: large amounts of development time are spent creating an optimal solution. Might a benchmark somehow capture the time to solution for the implementation? Given the wide disparity of programmer skills, this seems like a difficult problem, yet it is one of the key criteria being evaluated by customers.

Note 1: It fails the first of Jim Gray's criteria for a good benchmark, which are relevance, portability, scalability and simplicity. These are good concepts for us all to keep in mind at this workshop.

Len Wyatt is a Principal Program Manager at Microsoft, working at the intersection of the Big Data ecosystem with the traditional Data Warehousing ecosystem. His background includes Data Warehousing, Business Intelligence, On-Line Analytical Processing, ETL and a dash of system performance. Len has a nasty habit of getting involved with unreasonable projects, such as creating the biggest OLAP cube ever, or setting the "ETL World Record." Len is the instigator and chair of the TPC committee developing a Data Integration benchmark.