Developing System Performance Metrics for Cloud Computing Based on Hadoop

Rema Hariharan, Gabriele Jost, Sanjiv Lakhanpal, Dave Raddatz, AMD (Advanced Micro Devices, Inc.)

Abstract:

This short white paper describes our efforts to establish techniques and tools to identify optimization opportunities for Hadoop workloads. Suitable performance metrics and relevant benchmark use cases are a crucial component to achieve these goals. We discuss efforts to define suitable metrics for cloud computing in general, briefly describe hardware and software components that impact Hadoop performance, and look into means to measure and analyze Hadoop performance.

1. Introduction

We live in an age when the amount of data -- let it be measured, collected, or computer-simulated -- is increasing constantly. A recent US Air Force article stated that sensor capabilities have increased so much that there are currently no techniques available to analyze the data. Another trend is the rise of cloud computing. It is therefore natural to investigate infrastructure for large-scale data analysis and storage on networks of commodity processors. Both are provided by Hadoop. The goals of our efforts are to establish techniques and tools to identify optimization opportunities for Hadoop workloads in terms of code optimization, CPU performance, node performance, and cluster performance. Suitable performance metrics and relevant benchmark use cases are a crucial component to achieve these goals.

2. Benchmarking the Cloud

Cloud computing refers to applications and services that run on a distributed network [1]. Resources are pooled by virtualization and are accessed by common internet protocols and networking standards. Cloud models employ different deployment (public, private, hybrid, community) and service (IaaS, PaaS, and SaaS) models. This introduces a plethora of factors that impact the performance of cloud computing. The SPEC Open Systems Group (OSG) Cloud working group has been investigating application workloads suitable for benchmarking. It lists four key metrics relevant in the context of cloud benchmarking [2]:

- Elasticity
  - *Provisioning interval*, or the lag between when a resource is requested and when it is actually available.
  - *Agility*, or the ability of the provider to track the needs of the workload.
  - *Scale-up*, or the improvement in response times with increased amount of resources.
  - *Elastic speed-up*, or the improvement in throughput as an additional resource is added on the fly.
- Throughput
- Response time
- Variability

Variability is a metric that tracks the temporal nature of performance. Resources presented to the customer may not be identical every time, and the access path or system resources, or both, may have a variable background load at different times that affects the overall perceived performance of the cloud system.

Metrics like power, price, and reliability are also relevant; however, they are not easy to measure accurately in every scenario. Benchmarks used to measure the performance of cloud systems must include these metrics when relevant. Additionally, cloud benchmarks must be virtualization-friendly and capable of running across various load levels.

Cloud benchmarks would fall under two broad categories: white-box benchmarks and black-box benchmarks. Benchmarking white boxes is an effort to measure the performance of systems that are marketed by hardware vendors and purchased by cloud vendors. In this case, the engineering details of every component being measured are known with no element of uncertainty.

Black-box benchmarking is of interest to end customers or users of the cloud. This involves benchmarking systems with broad details as specified in a language described by the cloud vendor, with the exact hardware and location of the system under test likely to change with time.

Predominant use cases prevalent in the cloud include social networking, including a web front end and back-end networking, data analytics, data warehousing/mining, memory cloud, and HPC workloads. Of these scenarios, the most predominant uses for cloud providers include the web interface layer, memory cloud, and data analytics. Currently, most data analytics workloads run using a MapReduce framework.

3. Hadoop Performance

According to [3], "The good news is that Big Data is here. The bad news is that we are struggling to store and analyze it." The best known components of the Hadoop ecosystem are a distributed file system HDFS with built-in reliability and MapReduce programming model for efficient and reliable data analysis. Other components support serialization, persistent data storage, cross-language RPC, a distributed data base, distributed warehouse, a data flow language, and more.

The Hadoop ecosystem evolves constantly. From a cloud perspective, we will consider Hadoop running in an IaaS model on private clouds. In our efforts to analyze Hadoop performance for various workloads, we will consider all levels in a system under test (SUT) that impact Hadoop performance: BIOS settings, OS (various flavors of Linux), network drivers, hypervisor, JVM runtime, compiler, libraries, the Hadoop ecosystem and application-specific components. We also plan to analyze the performance impact on the various levels, and the outcome of the analysis should serve as a basis to characterize workloads in terms of performance metrics.

Our major goal is to determine suitable performance metrics to quantify Hadoop performance and make it comparable among different types of SUTs, workloads, deployments, and service models. We also aim to provide feedback to the open source software community such as Apache, openJDK, and gcc. The findings will be interesting to hardware vendors and cloud providers to optimize their SUTs for Hadoop.

4. Techniques and Tools for Hadoop Big Data Workload Analysis

We aim to establish a SUT measurement tool set to measure and quantify SUT performance metrics and make them comparable. Although the focus of the current effort is Hadoop, lessons will be applicable to other software stacks. We are interested in using public domain software for performance analysis on different levels. This approach will allow the Hadoop user community to conduct similar analysis and

benchmarking in their environment. From our perspective, SUT performance analysis encompasses the software stack and the underlying hardware in a CPU within one cluster node and across the cluster.

Code optimization:

This will aim to gather performance profiles based on time and hardware counters, mapped to subroutines, source code, and instructions. Metrics to consider include instructions per clock cycle (IPC) and memory behavior (cache and TLB) per subroutine. Scalability of the software regarding the number of cores in the CPU in one cluster node and, if applicable, across the whole cluster will also be considered.

Some examples for Linux profilers are *gprof* or *perf*. The AMD CodeAnalyst performance analyzer [4] for x86 architectures is built on OProfile [5] for the Linux platform and is available as a free download. It allows time-based, hardware event-based, instruction-based, and other kinds of profiling. The generated statistics show time spent in each subroutine and can be drilled down to the source code or instruction level. This data points to hot spots during program execution and opportunities for code optimization. It can also reveal the impact of using different compilation flags and compilers.

Another possibility is to instrument the application code itself to observe the flow of activity. However, this option may be too time consuming to be fruitful.

CPU performance:

We will profile system-wide hardware performance counters and profile user, system, and idle time during workload execution. These tools are useful for looking at CPU-centric metrics like cache hits/misses, IPC data, etc.

Node and Cluster Performance:

In addition to possibly using hardware performance counters to evaluate node performance by looking at CPU-to-memory and CPU-to-I/O performance, standard Linux performance tools like dstat, vmstat, iostat, and netstat can be used to observe CPU, memory, and disk and network utilization in a node. Network monitoring can also determine how the communication infrastructure impacts cluster performance. For more focused monitoring, the workload can be instrumented to make calls to these OS performance monitoring tools to observe performance during different phases of the workload execution.

Depending on the workload, quality of service metrics can be used to evaluate either node or cluster performance. If the workload driver doesn't already gather this type of information, then it could be instrumented to attempt to get some type of response time or latency information for either a node or cluster environment. We will also monitor power consumption because performance/watt is a relevant metric.

5. Background of the authors

The authors of this paper have experience in performance analysis, code profiling and optimization, and benchmarking and actively participate in SPEC benchmark committees and working groups.

6. References

[1] Barry Sosinsky, *Cloud Computing Bible*, Wiley Publishing, Inc., 2011.
[2] Report on Cloud Computing to the OSG Steering Committee, SPEC OSG Cloud Computing Working Group, to appear 2012.
[3] Tom White, *Hadoop: Definite Guide to Hadoop*, 3rd Edition, O'Reilly, 2012.
[4] http://developer.amd.com/tools/CodeAnalyst/Pages/default.aspx.
[5] http://oprofile.sourceforge.net/about/.