Apache Hadoop Performance-Tuning Methodologies and Best Practices

Shrinivas B. Joshi
Advanced Micro Devices, Inc.
7171 Southwest Pkwy
Austin, TX 78735 USA
shrinivas.joshi@amd.com

ABSTRACT

Apache Hadoop is a Java based distributed computing framework built for applications implemented using MapReduce programming model. In recent years, Hadoop technology has experienced an unprecedented growth in its adoption. From single-node clusters to clusters with well over thousands of nodes, Hadoop technology is being used to perform myriad of functions – search optimizations, data mining, click stream analytics, machine learning to name a few. Although setting up Hadoop clusters and building applications for Hadoop is a well understood area, tuning Hadoop clusters for optimal performance is still a black art. In this demo paper, we will attempt to provide the audience with a holistic approach of Hadoop performance tuning methodologies and best practices. We discuss hardware as well as software tuning techniques including BIOS, OS, JVM and Hadoop configuration parameters tuning.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of systems

General Terms

Performance

Keywords

Performance tuning, Apache Hadoop, Map-Reduce

1. INTRODUCTION

Users who have deployed and tried tuning their Hadoop [1] clusters for the first time will certainly attest to the fact that optimizing Hadoop clusters is a daunting task. Apart from the nature and implementation of Hadoop jobs, hardware, network infrastructure, OS, JVM, and Hadoop configuration properties all have a significant impact on performance and scalability. Using *TeraSort* as a reference workload, this demo paper attempts to educate the audience on challenges involved in performance-tuning of Hadoop setup, tuning best practices, empirical data on effect of various tunings on performance, and some future directions.

2. CHALLENGES

Hadoop is a large, complex framework involving a number of entities interacting with each other across multiple hardware systems. Performance of Hadoop jobs is sensitive to every component of the cluster stack - Hadoop configuration, JVM, OS, network infrastructure, underlying hardware, and possibly BIOS settings. Hadoop supports a large number of configuration properties and a good chunk of these can potentially impact performance. As with any large software system, diagnosing performance issues is a complicated task.

3. EXPERIMENT SETUP

For the purpose of this study we used two different Hadoop cluster configurations.

3.1 Cluster A

The first cluster, Cluster A, has following configuration:

- 7 data nodes, 1 name node: 2 chips/6 cores per chip, AMD Opteron™ [1]2435 @2.6GHz
- 16GB DDR2 800 RAM per node
- 6 x 1TB Samsung SpinpointF3 7200 rpm disks
- Ubuntu 11.04 Server x64, Oracle JDK6 update 25 x64
- TeraSort dataset size – 64 GB

3.2 Cluster B

The second cluster, Cluster B, has following configuration:

- 5 data nodes: 2 chips/4 cores per chip, AMD Opteron ™ 2356 @2.3GHz
- 1 name node: 4 chips/4 cores per chip: AMD Opteron ™ 8356 @2.6GHz
- 64GB DDR2 667 and DDR2 800 RAM
- 6 x 1TB Samsung SpinpointF3 7200 rpm disks
- Ubuntu 11.04 Server x64, Oracle JDK6 update 25 x64
- TeraSort dataset size – 1 TB

While presenting empirical data we highlight performance improvements on the cluster that demonstrated bigger gains amongst the two clusters.

4. PERFORMANCE TUNING

4.1 Hadoop Configuration Tuning

This section contains guidelines on Hadoop configuration parameters tuning procedure.

[1] © 2012 Advanced Micro Devices, Inc. AMD, the AMD Arrow logo, AMD Opteron and combinations thereof are trademarks of Advanced Micro Devices, Inc. HyperTransport is a trademark of HyperTransport Technology Consortium.

Using maximum possible map and reduce slots, identify the optimal number of disks that maximizes I/O bandwidth. On Cluster A, we noticed more than 50% performance improvement while using 5 hard disks as compared to using only 1 hard disk. Experiment with different HDFS block sizes. You may have to re-evaluate optimal block size after other tunings mentioned in subsequent sections. Identify Java heap usage and garbage collections characteristics (GC) of Hadoop framework processes and tune their JVM settings accordingly. On the Hadoop configuration parameter tuning side, start with biggest payoff properties such map output compression and JVM reuse policy. In our experience, enabling map output compression using LZO codec has shown better performance. On Cluster B, we saw close to 30% performance improvement by enabling LZO based map compression as compared to no compression. If you have short running map tasks, enable JVM reuse policy. If memory is not a bottleneck, try to eliminate map-side spills by tuning io.sort.mb Hadoop property. At the least, try to reduce the number of spills. In case there are no spills, tune io.sort.spill.percent Hadoop property. On Cluster B, we noticed close to 20% improvement in performance by avoiding map-side spills and tuning io.sort.factor property. Try to avoid or eliminate intermediate disk I/O operations on reduce side by tuning Java heap sizes. If the reduce functionality is not heap-heavy, try to tune reduce buffer size. Tuning reduce-side configuration properties offered 6% improvement in performance on Cluster B. Tune framework-related resources such as task tracker threads, data node, and name node handler count.

4.2 JVM Configuration Tuning

In this section we discuss different JVM command-line switches and their potential impact on performance of Hadoop workloads.

On 64-bit Oracle JDK6 update 25 JVM, compressed pointers are enabled by default. If you are using an older version of JDK and compressed pointers are disabled, experiment with enabling them. Compressed pointers reduce memory footprint. We saw more than 3% improvement in performance by enabling compressed pointers on Cluster A. Biased-locking feature in Oracle HotSpot JDK improves performance in situations with un-contended locks. Given the architecture of Hadoop framework, biased locking should generally improve performance. On Cluster A, we noticed more than 5% improvement by enabling biased locking. Oracle JVM optimizations enabled by command-line flags such as AggressiveOpts, UseCompressedStrings, and UseStringCache can have an impact on Hadoop performance. Try experimenting with these flags. We, however, noticed 2% degradation in performance by enabling AggressiveOpts flag on Cluster A. Verify whether the JVM is running out-of-code cache. Increase code cache size if necessary. Experiment with UseNUMA and UseLargePages JVM flags. Perform detailed GC log analysis and tuning of the map and reduce JVM processes. We noticed a 3% improvement in performance by tuning GC flags on Cluster A.

4.3 OS Configuration Tuning

This section presents information about impact of tuning Linux OS properties on Hadoop performance.

Certain Linux distributions support EXT4 as the default file-system type. If you are using another type of file system, experiment with EXT4 file system. We noticed 9% performance improvements by using EXT4 file-system over EXT3 on Cluster A. By default, every file read operation triggers a disk write operation for maintaining last access time of the file. Disable this logging using noatime, nodirtaime FS attributes. Experiment with other FS tuning attributes such as extent, flex_bg, barrier etc. On Cluster B, we noticed 15% improvement in performance by using noatime FS attribute. Linux kernels support 4 different types of I/O schedulers – CFQ, deadline, no-op, and anticipatory. Experiment with different choices of I/O scheduler, especially CFQ and deadline. On Cluster B, CFQ scheduler performed 15% better than deadline scheduler. Linux OS limits such as max open file descriptors and epoll limits can have an effect on performance, experiment with these limits. We, however, saw regression of 1% by increasing open fd limit to 16K from its default value of 1K on Cluster A.

4.4 BIOS Configuration Tuning

In this section we discuss some of the BIOS parameters that could potentially impact Hadoop performance.

Native command queuing (NCQ) feature of modern hard drives helps improve I/O performance by optimizing drive head movement. Experiment with AHCI option in BIOS, which can be used to enable NCQ mode. When all the CPU cores on the hardware are not fully utilized, the processor could be downgrading CPU frequency and other resources such as HyperTransport^{TM} links. Experiment with ACPI and other power-related BIOS options. We noticed 1% performance improvement by disabling power saving mode in the BIOS. This observation was made on Cluster A. On some AMD processor-based systems, NorthBridge frequency and width are dynamically tuned to reduce power consumption. If memory bandwidth is a bottleneck, experiment with options that can be used to modify NorthBridge frequency and width settings. On Cluster A, we noticed 2% improvement in performance by tuning NorthBridge settings. Modern AMD processors support a feature called HT assists (a.k.a. probe filters). This feature reduces traffic on memory interconnects at the expense of some portion of L3 cache. Experiment with HT assists settings; disable it if your job is sensitive to L3 cache size.

5. CONCLUSIONS AND FUTURE WORK

This demo paper discussed some of the best practices in tuning different components of the software and the hardware stack running Apache Hadoop framework. We were able to achieve speed-up factor of 4.2x on Cluster A and 2.1x on Cluster B running TeraSort workload. Configuration tuning of all the components of Hadoop stack is an important exercise and can offer a huge performance payoff. Different Hadoop workloads will have different characteristics, so it is important to experiment with different tuning options. We want to perform a similar study in multi-tenant environments and in the cloud environment. We want to explore JVM and JDK optimizations targeting peculiar characteristics of Hadoop framework and the jobs running on top of it.

6. REFERENCES

[1] Apache Hadoop. http://hadoop.apache.org/