Performance measurement of a Hadoop Cluster

Technical white paper

Created: February 8, 2012

Last Modified: February 23 2012

Contents

Introduction

In recent years with the advent of popular social networks like Facebook (1) and LinkedIn, (2) the world's population has become more connected than ever. According to a McKinsey report, (3) there were 5 billion mobile phones in use in 2010 and 30 billion pieces of content shared on Facebook every month. This massive generation of of content, otherwise known as Big Data, has been growing at a rate of 40% per year, 80% of which is unstructured data. Organizations are constantly facing challenges in dealing with the acquisition, storage, processing and visualization of the stark volume and structure of the data. IDC (4) further boosts this claim forecasting that over 90% of data will be unstructured, containing rich information that is difficult to analyze. In addition, the total amount of data stored will rise to 2.7 Zettabytes (ZB), up 48% from 2011.

The Big Data Puzzle

Information has traditionally been stored in tables consisting of columns, which represent attributes, and rows containing primary value and associated attributes. These relational databases have semantics attached to the data which make analysis very fast and easy. Unstructured data, however, lacks such inherent semantics which makes it very time consuming and difficult to analyze. This type of data includes images, videos, audio files and web enabled workloads among others.

Organizations are facing a very critical challenge today due to the sheer volume of data being generated every year. This data, whether structured or unstructured, can offer invaluable insights and information about the internal details of an organization and can serve as a comprehensive knowledge base for planning and forecasting. Current data analysis methods prove largely inadequate when looking to gain any amount of valuable insight from such large, multi-terabyte, and even petabyte, datasets. Corporate sectors such as media, healthcare, manufacturing, and retail are facing the challenge of analyzing this data. Using advanced Big Data methods for analysis can lead to significant cost savings. The FBI estimates that 10% of transactions within federal healthcare programs, like Medicare and Medicaid, are fraudulent and costs nearly $150 billion a year (5). As healthcare data is spread across five humungous databases, it is almost impossible to detect a fraud in real time. Oak Ridge National Labs has submitted a proposal to unify all these databases to perform fraud detection using the "Jaguar" supercomputer. Similar methods can be applied to other domains to effectively detect patterns and behaviors from a very large data pool.

Apache Hadoop and MapReduce

Processing massive amounts of data requires a parallel compute and storage infrastructure. Traditionally, High Performance Computing (HPC) uses Massively Parallel Processing (MPP) to tackle computationally intensive problems such as simulation, modeling, weather prediction, protein docking, etc. Self-contained compute resources are connected together with high speed interconnects such as Infiniband or 10G Ethernet. As nodes do not share physical resources such as memory and CPU, a *Shared Nothing* MPP infrastructure is highly scalable. Google utilizes this approach to scale up and add additional nodes without slowing down the network.

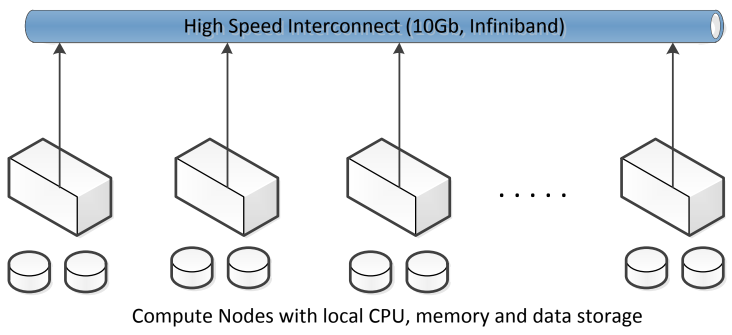
High Speed Interconnect (10Gb, Infiniband)
. . . . .
Compute Nodes with local CPU, memory and data storage

Figure 1. Share Nothing architecture

MapReduce is a programming paradigm and parallel programming framework which distributes data among multiple commodity nodes and processes it in "data-local" fashion (6). The framework operates on data stored in a parallel file system which is easily scalable with additional nodes (7). Apache Hadoop is an open source implementation of the MapReduce framework and the file system needed to store the data. Hadoop utilizes the powerful Hadoop Distributed File System (HDFS); a highly scalable, parallel file system optimized for very large sequential data sets. Compared to existing high throughput computing (HTC) approaches like Condor, which requires a dedicated storage area, HDFS distributes data across all nodes within the network. The MapReduce engine distributes the workload across all nodes such that the data to be processed is local and effectively works to conserve network bandwidth. HDFS has built in protection against rack level failure by means of distributed replication. Each data block in HDFS is replicated on different nodes and, due to its rack-aware nature, rack-level and node-level failures do not disrupt cluster operations.

Hadoop follows a master-slave model and consists of following processes:

- NameNode – Maintains HDFS namespace on master node (1 per cluster).
- SecondaryNameNode – Performs checkpoint functions and maintains change logs. Can also be installed on the master node for a small cluster (1 per cluster).
- DataNode – Handles all data allocation and replication and is installed on each slave node (1 to many per cluster).
- JobTracker – Schedules job execution and keep track of cluster wide job status (1 per cluster)
- TaskTracker – Allocates system resources for execution of tasks. Runs on compute nodes in conjunction with data node (1 to many per cluster).

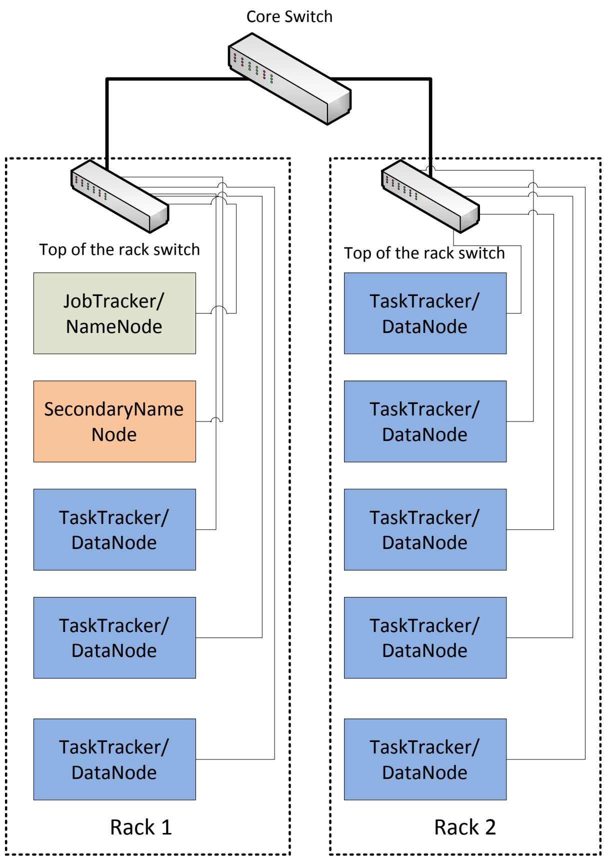

Figure 2. Typical Hadoop cluster

AMAX's PHAT-Data (Peta-Scale Hadoop Analytics Technology) (8) provides a complete turnkey solution for Apache Hadoop. The PHAT-Data product line offers a highly scalable, tunable and easy to deploy platform bundled with Apache Hadoop designed to solve Big Data acquisition, storage and analysis problems, and is thoroughly tested and certified production-ready to be deployed on large scale. The PHAT-Data turnkey solution includes a complete software stack of Apache Hadoop, configured application specific and tuned for optimum efficiency and maximum performance by our dedicated Hadoop engineering team.

Performance tuning for Hadoop

The Hadoop framework consists of a multitude of components layered horizontally and vertically. For optimum performance, each framework parameter needs to be carefully chosen and adjusted based on cluster size, type of workload, amount of compute resources available and number and type of hard disk drives in each DataNode. Each application has specific nuances which need to be tackled in a unique manner as one set of parameters may not be suitable for all workloads. A few points worth considering are:

1. Hadoop operates on large sequential files with very large chunk sizes which usually range from 128MB to 512MB. To handle such a large logical chunk, the file system should be tested for

sequential data write bottlenecks. For example, XFS formats much quicker than ext4 but file deletes can be significantly slower in XFS compared to ext4.

2. HDFS is built around the assumption that failure is a norm rather than exception. Inherent block replication features offer protection against node level and rack level failures which makes RAID unnecessary for data drives.
3. Choose the number of mappers, reducers and task slots carefully as too many task slots can cause memory exceptions and repeated job failures. Adjust sort buffer size for each map task such that sufficient memory is available for the operating system. If the IO buffer is full, the buffer is saved on disk resulting in a spill file which can significantly slow down job execution. Decide on the number of task slots such that per task buffer size reduces the possibility and creation of a spill file.
4. A workload can be categorized as CPU bound, IO bound or both. Choose a hardware subsystem such that the CPU core to memory ratio is optimum.
5. Since Hadoop operates on large chunk sizes, significant network traffic during shuffle phase and replication is probable. Block compression offers an attractive solution to reduce network traffic and disk IO without burdening the CPU.
6. Link aggregation on the network offers an attractive solution for load balancing and automatic failover. Ensure that switching infrastructure supports link aggregation and protocol on switches is compatible. (e.g. IEEE Link aggregation and control Protocol (LACP) vs Cisco Port Aggregation Protocol (PAgP)).
7. Consider using jumbo frames which reduces network traffic and CPU utilization. NICs with TCP offloading can further reduce CPU consumption.
8. Choice of operating system and Java framework version governs thread and energy efficiency. Linux kernel version 2.6.30 onwards and Java 6u14 or later is recommended for a production environment.

Test Cluster Configuration

The test cluster consisted of 1 master node running NameNode, SecondaryNameNode and JobTracker daemons, and 7 data nodes configured as follows: AMAX PHAT-Data 8-node cluster

- CPU: Intel Xeon X5650 2.67GHz 12MB cache HT Enabled
- Memory
  - Master: 48GB DDR3 1333MHz, 4GB DIMMs
  - Data Nodes: 24GB DDR3 1333MHz, 4GB DIMMs
- Storage Controller: LSI MegaRAID SAS 9265-8i RAID Controller
- Hard disk drives: 6x Seagate Constellation ES 1TB SAS Drives in JBOD configuration
- File system: ext4 aligned at 4KB block size
- Linux
  - Distribution: CentOS 6.0 Final amd64
  - Kernel: 2.6.32-71.el6.x86_64
    - nofile=16384

            - nproc=4096
- Java: Sun JRE 64 bit 1.6.0_29-b11
- Network Interfaces
    - 1Gb: Intel 82575EB Gigabit Ethernet Controller
    - 10Gb: Emulex OneConnect (be3) 11102-NX CNA
- Network switch: Extreme Networks Summit X670V 10Gb switch

Following is the list of configuration parameters used in benchmark:

- General parameters
    - Map tasks: 84
    - Reduce tasks: 42
    - Maximum map tasks for one compute node: 12
    - Maximum reduce tasks for one compute node: 6
- HDFS
    - Block size: 128 MB
    - Namenode handler count: 10
    - Replication factor: 3
    - Permission check: disabled
- MapReduce
    - Java child options: -Xmx1024m
    - Mapred ulimit: 2097152
    - Slowstart completed maps: 0.5
    - Compression enabled: Block
    - Io sort buffer size: 256MB
    - Io sort factor: 32

Benchmark procedure

The test process aims at finding bottlenecks posed by the network interface and performance comparison of 10Gb and 1Gb Ethernet interfaces. For this test we have used 2 industry standard benchmarks: TestDFSIO and TeraSort.

TestDFSIO is used to measure performance of HDFS and stresses both thenetwork and IO subsystems. The command reads and writes files in HDFS which is useful in measuring system-wide performance and exposing network bottlenecks on the NameNode and DataNodes. A majority of MapReduce workloads are IO bound more than compute and hence TestDFSIO can provide an accurate initial picture of such scenarios.

The benchmark can be run for writing, using the –write switch, and using –read for the read test. The command line accepts a number of files and sizes of each file in HDFS. As an example, the command for a read test may look like:

```
$ hadoop jar Hadoop-*test*.jar TestDFSIO –read –nrFiles 100 –fileSize 10000
```

TestDFSIO generates 1 map task per file and splits are defined such that each map gets only one file name. After every run, the command generates a log file indicating performance in terms of 4 metrics: Throughput in MBytes/s, Average IO rate in MBytes/s, IO rate standard deviation and execution time. The most notable metrics are throughput and average IO, both of which are based on file size read or written by the individual map task and the elapsed time in performing the task. The throughput for N map tasks are defined as:

$$Throughput\ (N) = \frac{\sum_{i=0}^{N} size\ of\ file_{\ i}}{\sum_{i=0}^{N} execution\ time_{\ i}}$$

And average IO rate is calculated as:

$$Average\ IO\ Rate\ (N) = \frac{\sum_{i=0}^{N} rate_i}{N} = \frac{\sum_{i=0}^{N} \frac{size\ of\ file_{\ i}}{time_{\ i}}}{N}$$

If the cluster has 50 map slots and TestDFSIO creates 1000 files, the throughput can be calculated as:

$$Concurrent\ throughput = Reported\ throughput * Number\ of\ Map\ Slots$$

The IO rate can be calculated in similar fashion. While measuring cluster performance using TestDFSIO may be considered sufficient, the HDFS replication factor (value of dfs.replication in hdfs-site.xml) also plays important role. A lower replication factor leads to higher throughput performance due to reduced background traffic.

TeraSort is accepted as an industry standard benchmark to compare performance of Hadoop clusters. The benchmark tries to sort 1TB of data as quickly as possible using the entire cluster. This benchmark is divided in three phases which are

- TeraGen – Generates a desired size file as an input and usually ranges between 500GB to 3TB.
- TeraSort – Sorts the input file across a Hadoop cluster.
- TeraValidate – Verifies the sorted data for accuracy.

TeraGen data, once generated, can be used in all runs with the same file size. TeraSort stresses the cluster in terms of 3 parameters: IO response, network bandwidth and compute, as well as both layers of the Hadoop framework (MapReduce and HDFS). Once sorted, a single reduce task checks the output of the sort phase for validity. Each test phase constitutes one run of TeraSort. The format for a TeraSort command is:

```
$ hadoop jar hadoop-*examples*.jar <input directory> <output directory>
```

Our test procedure ran 5 iterations of TestDFSIO for Ten 10GB files followed by five iterations of TeraSort using a 1TB file. All results were averaged to minimize errors and test parameters were kept constant across all iterations.

Results

To paint a good picture of performance, each benchmark tool was run 5 times on each 1Gb network interface and results were averaged to reduce error margin. The same process was carried on 10Gb interfaces to get data for comparison.

TestDFSIO Read

The read test benefits from high network bandwidth offered by a 10Gb interface. Even with the background replication factor set to 3, 10GbE per map task performance is significantly better than 1GbE in terms of execution time and overall bandwidth utilization.

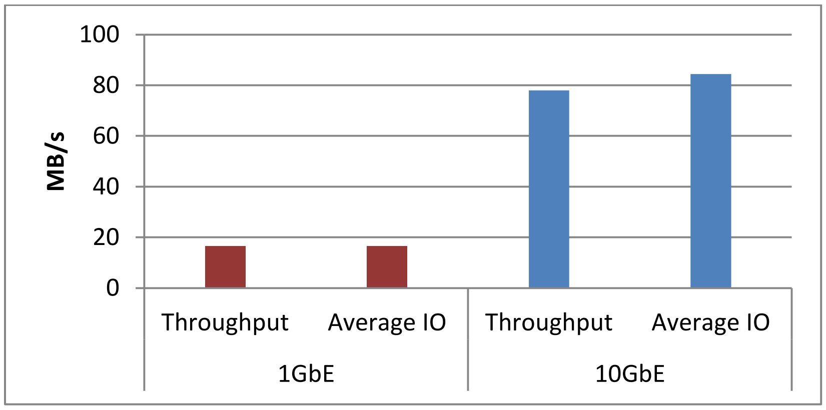

Figure 3. 10Gb adapter offers 3.4x more throughput and average bandwidth

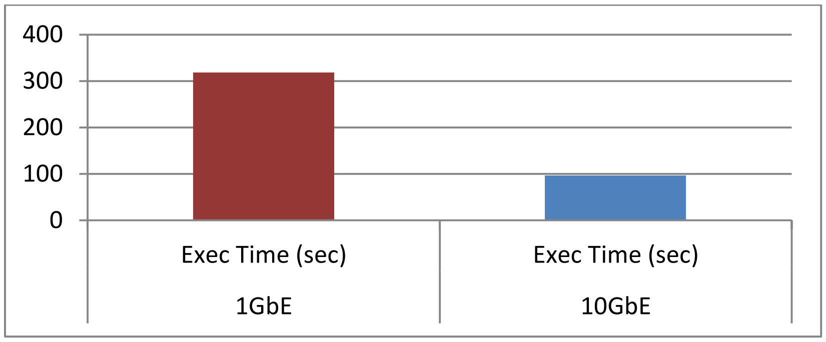

Figure 4. 10Gb interface reduces execution time by 3.2x even with background replication

TestDFSIO Write

Writing of data in HDFS introduces replication overhead on the network, severely bottlenecking the gigabit interface. A 10Gb interface provides nearly 5x the amount of bandwidth which offers significantly reduced execution time as seen in the graph below. As the cluster scales, the top-of-the-rack switch can sustain 5 to 6 times more traffic than traditional gigabit Ethernet allowing more data to flow through the compute cluster, providing a dramatic reduction in execution time.

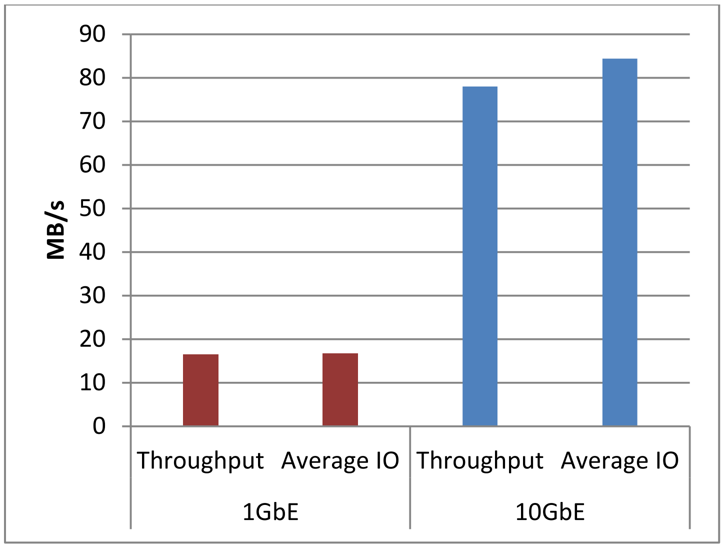

Figure 5. The write bandwidth of 10GbE interface is 5 to 6 times greater than 1GbE

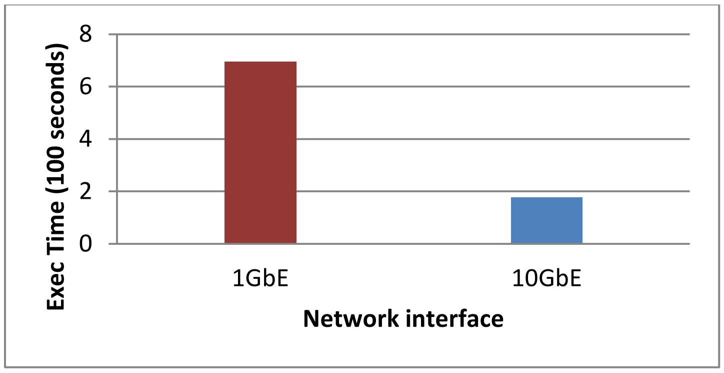

Figure 6. The same write task takes 3.68x less time on 10GbE

TeraSort

Terasort is a resource intensive benchmark that benefits from high network bandwidth. Teragen is mostly sequential write whereas TeraValidate is a single map task with sequential read. However the output variability makes direct comparison with TestDFSIO difficult. The time taken for each phase of the TeraSort benchmark to sort a 1TB file is presented below. Replication factor has been maintained at 3 to preserve in rack network traffic. As seen in the below chart, the generation phase is sequential write and benefits the most from the available network bandwidth. The Sort phase of the benchmark is IO bound and hence execution time is almost comparable. TeraValidate forcefully sets the number of reducers to 1 and hence it may not observe the same performance benefits as the other tests. Overall cluster performance of the cluster is significantly better for 10GbE due to the log effect (total execution time of a job is dominated by the longest running map and reduce task).

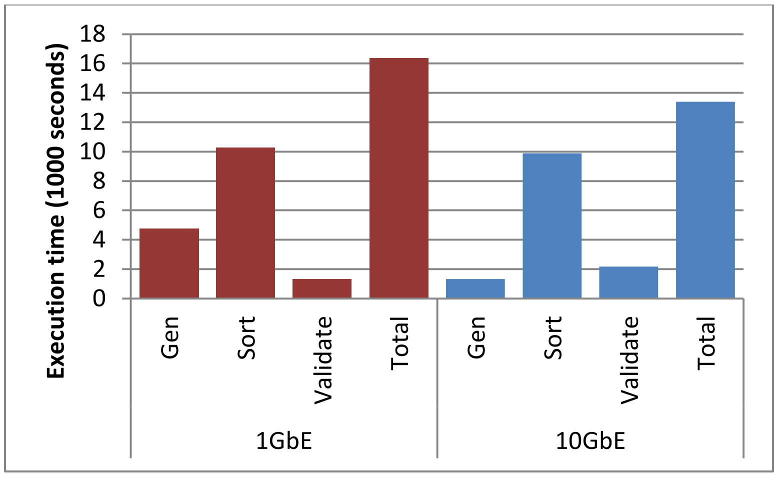

Figure 7. Generation state benefits the most due to its parallel nature

Conclusion

Traditional Hadoop workloads are IO and network intensive and performance of a Hadoop cluster greatly benefits from availability of increased network bandwidth. As the amount of data grows, the 10Gb interface scales much better than grouped 1Gb interfaces, offering significant reduction in execution time. While MapReduce offers an attractive solution to part of the Big Data puzzle, each application has unique resource usage patterns and fine grained tuning of parameters is required to get optimum performance out of the compute infrastructure. Before deploying the application for production usage, benchmarking the cluster with test and real application workloads can offer invaluable insights into the inner workings of Hadoop.

Bibliography

1. **Facebook.** [Online] http://www.facebook.com.

2. **LinkedIn.** [Online] http://www.linkedin.com.

3. **McKinsey.** Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute.* [Online] May 2011. http://www.mckinsey.com/~/media/McKinsey/dotcom/Insights%20and%20pubs/MGI/Research/Technology%20and%20Innovation/Big%20Data/MGI_big_data_exec_summary.ashx.

4. **IDC.** International Data Corporation. *International Data Corporation Press Release.* [Online] December 1, 2011. http://www.idc.com/getdoc.jsp?containerId=prUS23177411.

5. **insideHPC.** ORNL researcher proposes to use supers to identify health care fraud. [Online] January 11, 2010. http://insidehpc.com/2010/01/11/ornl-researcher-proposes-to-use-supers-to-identify-health-care-fraud/.

6. *MapReduce: simplified data processing on large clusters.* **Jeffrey Dean, Sanjay Ghemawat.** s.l. : ACM Digital Library, 2004, Vol. 6.

7. *The Google file system.* **Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung.** 5, New York : ACM, 2003, Vol. 37.

8. **Amax Engineering Corporation.** [Online] 2012. http://www.amax.com/enterprise/phatdata.asp.

9. **Noll, Michael.** Benchmarking and Stress Testing an Hadoop Cluster with TeraSort, TestDFSIO & Co. [Online] 4 9, 2011. http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/.

10. **Intel.** Optimizing Hadoop Deployments. [Online] 2010. http://software.intel.com/file/31124.

11. **AMD Blogs.** AMD Blogs. [Online] April 13, 2011. http://blogs.amd.com/work/2011/04/13/big-data-in-hpc-back-to-the-future/.