संकुल नवप्रवर्तन केंद्र, दिल्ली विश्वविद्यालय
CLUSTER INNOVATION CENTRE, UNIVERSITY OF DELHI

Cluster Computing

Semester Long Project Report

Mentor

Dr. Sunita Negi
Assistant Professor, CIC
University of Delhi

Submitted By

Sumit Yadav & Rahul Yadav

Abstract

Computing resources are limited. Every now and then we can find people complaining about problems such as low hard-disk space or low RAM in their computer systems. This project aims to combine multiple computer systems at the Cluster Innovation Centre to utilize the computational power in that network simultaneously. Hadoop and Parallel Python Execution Server, commonly known as PP-server are the two approaches exploited to achieve the goal of harnessing computing power in unison.

Introduction

"This computer is slow!" "The RAM is not enough!" "System is showing blue screen of death!"

Such sentences were very common during our simulation of scenes with POV-Ray which is a Ray-Tracing software. There were times when the computer would show that the system is out of memory. We were using 16GB of RAM on a workstation that time. We wonder what could have been the reason- was it really the problem of lesser RAM or were we writing a code that was unnecessarily consuming huge amount of memory. In another project, an audio song was needed to process. While importing the complete song which was sampled at 44.1 kHz at once, MATLAB was out of memory! Nevertheless, now we know that Google servers have thousands of terabytes of server memory which perform huge amount of data processing in several milliseconds.

This kind of experiences triggered us to undertake the project on Clustered Computing. At first, we thought we would be creating multiple grids of computers and use their computational resources to perform tasks which were very difficult to process on a single machine. It was later that we realized not every job is parallelizable and that what we were trying to implement is known as a Cluster and not Grid and having it online doesn't make it a Cloud either. Cluster differs from Cloud and Grid in a way that a cluster is a group of computers connected by a local area network (LAN), whereas cloud and grid are more wide scale and can be geographically distributed. Another way to put it is to say that a cluster is tightly coupled, whereas a Grid or a cloud is loosely coupled. Also, clusters are made up of machines with similar hardware, whereas clouds and grids are made up of machines with possibly very different hardware configurations.

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud; the service being sold is Utility Computing. We use the term Private Cloud to refer to internal datacenters of a business or other organization, not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. People can be users or providers of SaaS, or users or providers of Utility Computing. The difference between a cloud and a grid can be expressed as below:

Resource distribution: Cloud computing is a centralized model whereas grid computing is a decentralized model where the computation could occur over many administrative domains.
Ownership: A grid is a collection of computers which is owned by multiple parties in multiple locations and connected together so that users can share the combined power of resources whereas a cloud is a collection of computers usually owned by a single party.
The examples of cloud computing are- Amazon Web Services, Google APP Engine, Microsoft Azure etc. and that of grid includes Future Grid, World Wide LHC Computing Grid (WLCG).

Parallel Processing

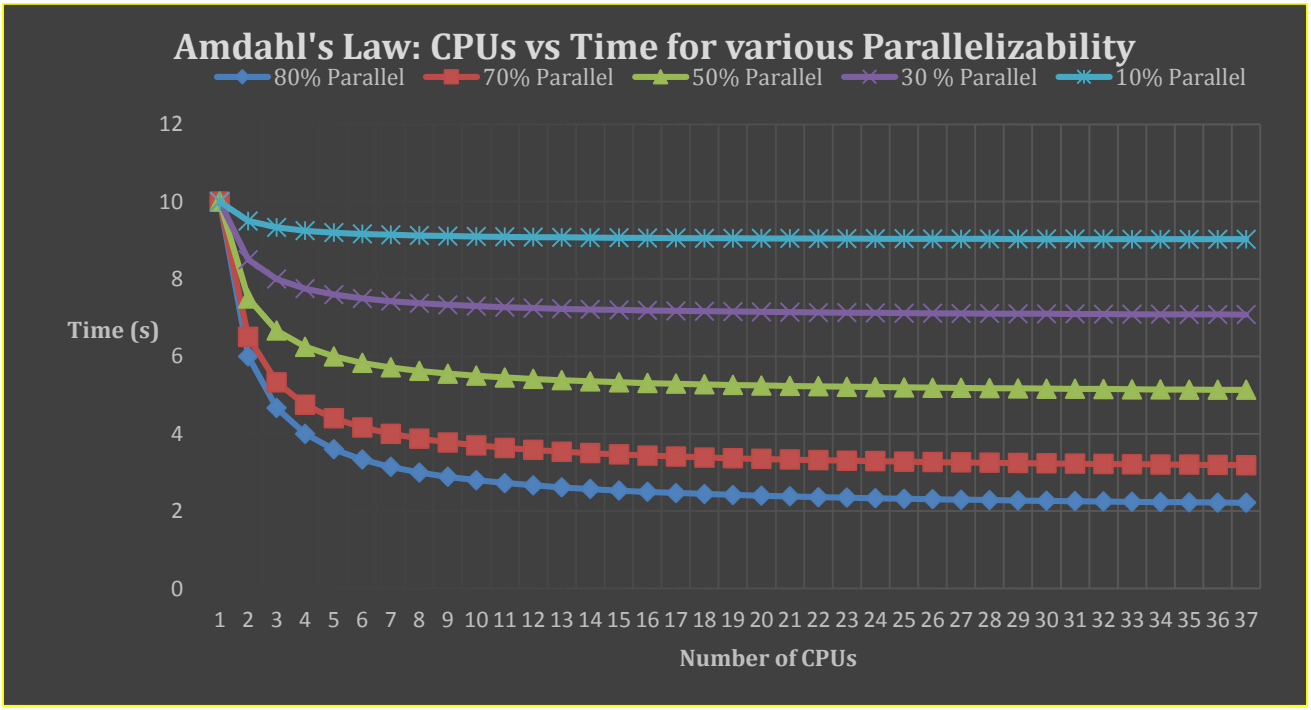
Parallel Processing is running or dividing a job into multiple jobs and running those jobs on multiple cores in order to complete the job in lesser time. But not all jobs are parallelizable and the maximum expected improvement we can achieve is thus limited and is given by Amdahl's Law which states that:

For given $n \in \mathbb{N}$, the number of threads of execution, $B \epsilon [0,1]$, the fraction of algorithm that is strictly serial, the time $T(n)$ an algorithm takes to finish when being executed on n thread(s) of execution corresponds to:

$$T(n) = T(1)\{B + \frac{1}{n}(1 - B)\}$$

Therefore, the theoretical speedup $S(n)$ that can be had by executing a given algorithm on a system capable of executing n threads of execution is:

$$S(n) = \frac{T(1)}{T(n)} = \frac{1}{B + \frac{1}{n}(1 - B)}$$



Every processor in computers available at CIC has two CPU cores. HP ProBook 4431s has 2 cores whereas Dell Optiplex 9010 has 4 cores i.e., 4 and 8 CPUs respectively. When we run a code on our computer, it is automatically assigned to one logical CPU. Since the job is not defined to run in parallel, it will continue to run on one core until it is complete. The load balancer would halt and run the job on different logical CPUs. When we run parallel programs, we direct the computer to execute the job at multiple processors at the same time which reduces the time and hence increasing efficiency of the computational resources.

MPI

MPI stands for Message Passing Interface. It is a standardized and portable message-passing system designed to function on a wide variety of parallel computers. The standard defines the syntax and semantics of library routines and allows users to write portable programs in the main scientific programming languages (FORTRAN, C, C++ or Python). Since its release, the MPI specification has become the leading standard for message-passing libraries for parallel computers.

Apache Hadoop

Apache Hadoop is an open source framework for distributed storage and distributed processing of large data sets (or Big Data) on cluster of hardware. Hadoop is a set of algorithms or framework which allows storing huge amount of data and processing it in a much more efficient and faster manner. Apache Hadoop comprises of two things: a storage part (Hadoop Distributed File System or HDFS) and a processing part (Map reduce). HDFS splits files into large blocks and distributes the blocks amongst the nodes in the cluster. For processing the data, the Hadoop Map/Reduce ships code to the nodes that have the required data and nodes then process the data in parallel, taking advantage of data locality.

TestDFSIO, a read and write test for HDFS has been performed in this project. It is widely used for benchmarking and stress testing of a Hadoop cluster. Two cluster performances are compared here, the first cluster comprises of 2 Dell OptiPlex 9010 and the other cluster comprises of single Dell Inspiron- N5050. Twice the clusters were tested, once with 1GB read and write test and the second time with 10GB read and write test. Write test is always run before read test as the data must be present for reading.

Parallel Python Execution Server

PP is a Python module which provides a simple interface to use the cores for parallel execution of an algorithm on SMP (systems with multiple processors or cores) and clusters. The most common way to write parallel applications for SMP computers is to use threads. Although, it appears that if the application is computation-bound using thread or threading, it will not allow running byte-code in parallel. This is because python interpreter uses GIL (Global Interpreter

Lock) for internal bookkeeping. This lock allows user to execute only one byte-code instruction at a time even on an SMP computer.

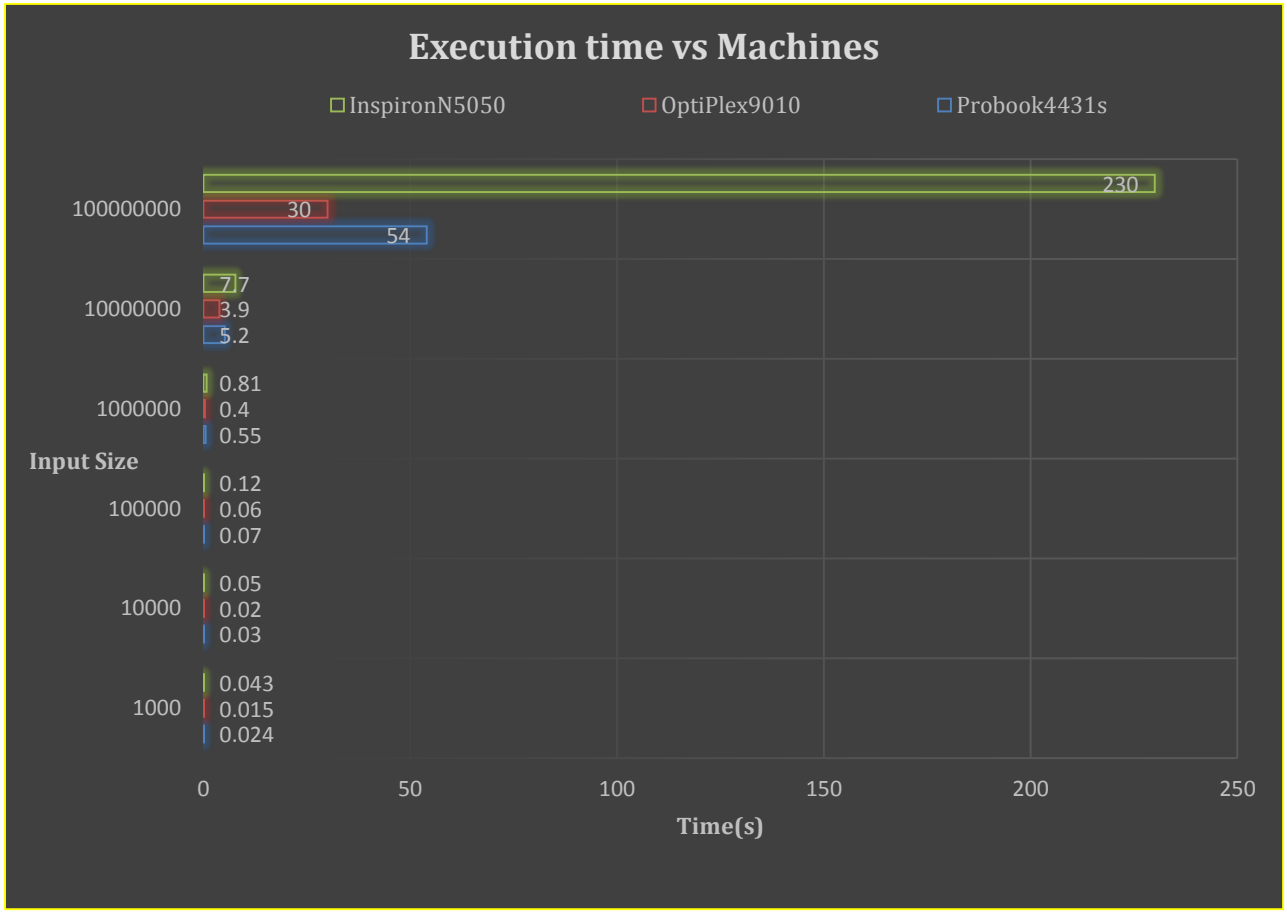
Parallel python module overcomes this limitation and provides a simple way to write parallel python applications. Internally ppsmp uses processes and IPC (Inter Process Communications) to organize parallel computations. Most of the complexities of the latter are taken care of, and our application just submits jobs and retrieves their results thus making it the easiest way to write parallel applications.

To make things even better, the software written with parallel python works in parallel even on many computers connected via local network or Internet. Cross-platform portability and dynamic load-balancing allows PP to parallelize computations efficiently even on heterogeneous and multi-platform clusters, which is also verified in this project.

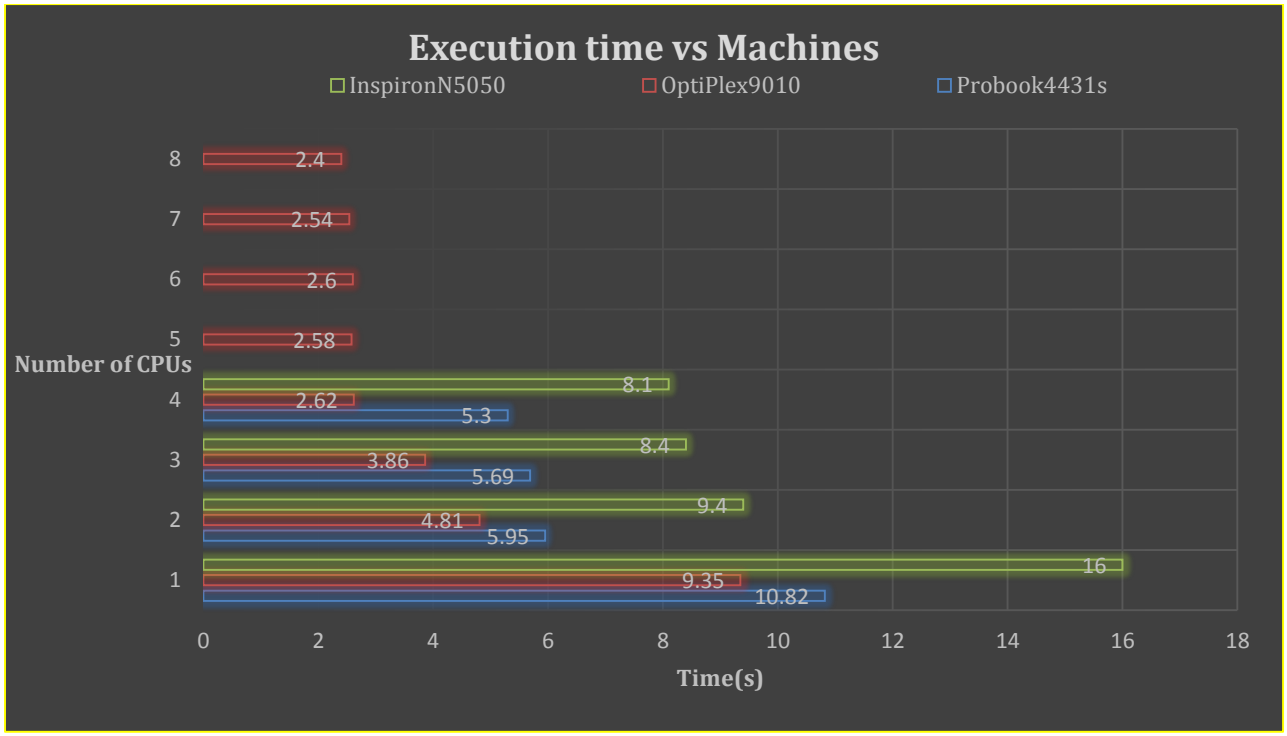
Results

[PP Server] Problem Statement: To generate random integer X between two bounds B1 and B2 repeated N times and compute the sum of primes from below X.

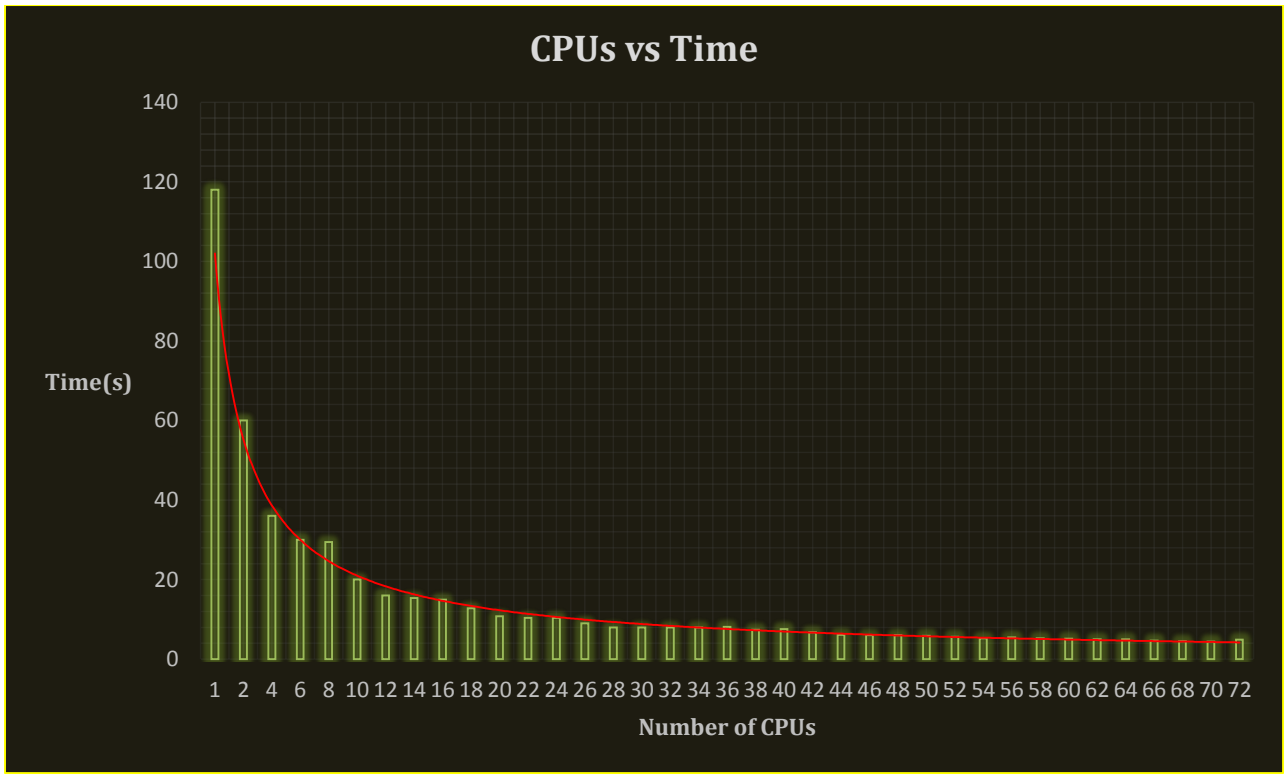
1. Execution on stand-alone machines with two cores (4 logical CPUs):



2. Execution on stand-alone machines with varying number of logical CPUs:

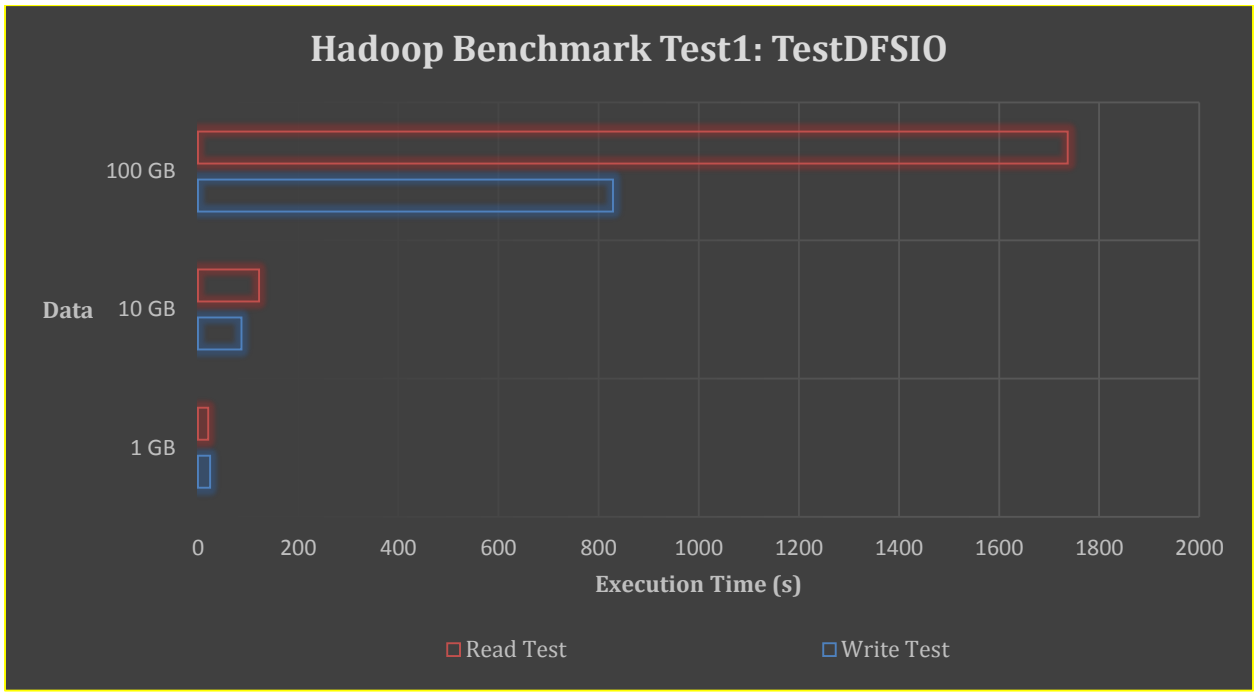


3. Execution on cluster of machines with varying number of logical CPUs:

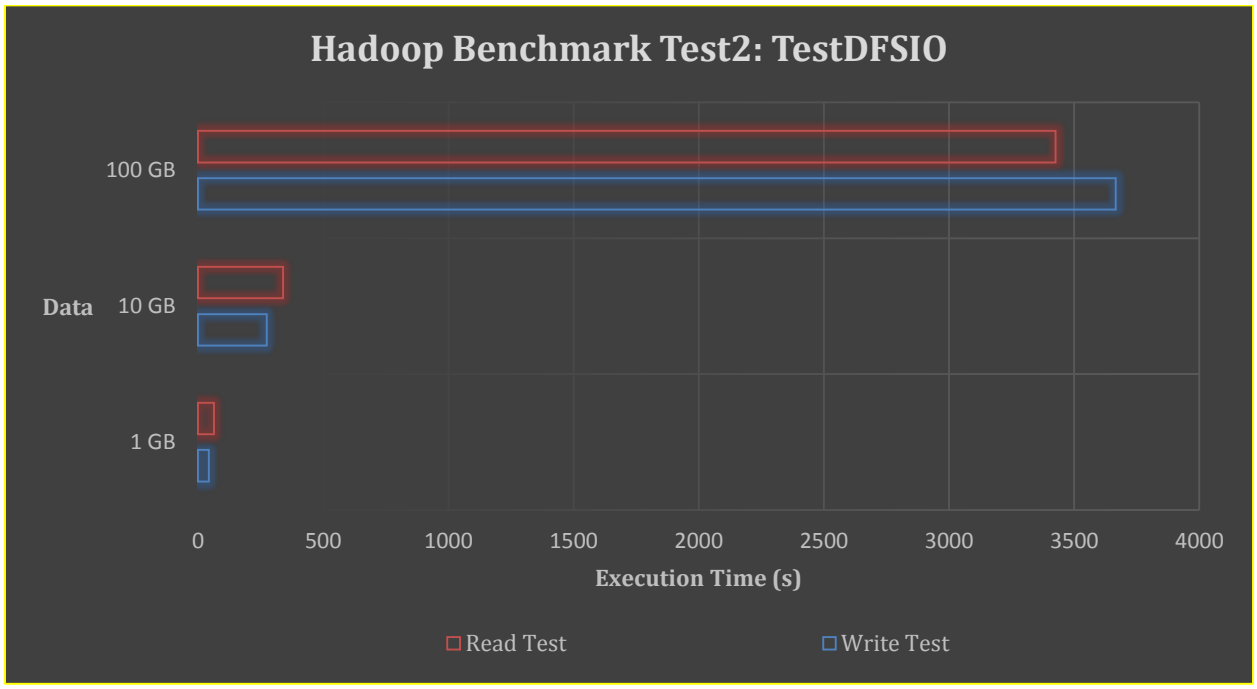


Hadoop Cluster Testing: Running a benchmark test (TestDFSIO) for stress testing and cluster performance of Hadoop Cluster.

1. Dell OptiPlex9010



2. Dell Inspiron N5050



Conclusion

As evident from the results, parallel processing amazingly scales down the time required to perform a parallelizable job. Hundreds of Gigabytes of data can be processed within minutes using Hadoop. Cluster made of multiple computer systems can have better utilization of available resources. Computer systems can be easily combined to form a cluster to perform scientific computing even with wireless network.

References

1. Hadoop: http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

2. Installation Guide: http://cis.csuohio.edu/~sschung/cis612/Setting-up-Hadoop-made-easy.pdf

3. Linux Command Reference: http://www.computerhope.com

4. Michael Noll Hadoop Tutorials: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster

5. Parallel Python: http://www.parallelpython.com/