Evaluating Different Distributed-Cyber-Infrastructure for Data and Compute Intensive Scientific Application

Arghya Kusum Das, Seung-Jong Park
School of Electrical Engineering and Computer Science
Center for Computation and Technology
Louisiana State University, Baton Rouge, LA, 70803
Email: {adas7, sjpark} @lsu.edu

Jaeki Hong, Wooseok Chang
Samsung Electronics Co., Ltd.
95, Samsung 2-ro
Giheung-gu, Yongin-si, Gyeonggi-do, 446711
Email: {jaeki.hong, wooseok_chang} @samsung.com

***Abstract*—Scientists are increasingly using the current state of the art big data analytic software (e.g., Hadoop, Giraph, etc.) for their data-intensive scientific applications over HPC environment. However, understanding and designing the hardware environment that these data- and compute-intensive applications require for good performance remain challenging. With this motivation, we evaluated the performance of big data software over three different distributed-cyber-infrastructures, including a traditional HPC-cluster called SuperMikeII, a regular datacenter called SwatIII, and a novel MicroBrick-based hyperscale system called CeresII, using our own benchmark software package, i.e., Parallel Genome Assembler (PGA). PGA is developed atop Hadoop and Giraph and serves as a good real-world example of a data- as well as compute-intensive workload.**

To address the impact of both individual hardware components as well as overall organization (scaled up and scaled out), we changed the configuration of the SwatIII cluster in different ways. Comparing the individual impact of different hardware components (e.g., network, storage and memory) over different clusters, we observed 70% improvement in the Hadoop-workload and almost 35% improvement in the Giraph-workload in the SwatIII cluster over SuperMikeII by using SSD (thus, increasing the disk I/O rate) and scaling it up in terms of memory (which increases the caching). Then, we provide significant insight on efficient and cost-effective organization of these hardware components. In this part, the MicroBrick-based CeresII prototype shows same level of performance as in SuperMikeII while giving more than 2-times improvement in performance per dollar in the entire benchmark test.

I. Introduction

Since experimental facilities at large-scale sciences, such as astronomy, coastal science, biology, chemistry, physics, etc., have produced unprecedented amount of data, scientific communities encounter new challenges, such as how to store data efficiently, how to process data optimally, etc. The fundamental model of computation involved in the scientific applications is rapidly changing in order to address these challenges. Deviating from the traditional compute intensive programming paradigm, e.g., MPI, etc., many HPC applications have started using the current state of the art big data analytic software, such as Hadoop, Giraph, etc., for their data-intensive scientific workloads.

However, the traditional supercomputers, even with tera to peta FLOPs scale processing power, are found to yield lower performance than expected, especially because of the I/O- and memory-bound nature of the data-intensive applications. As a result, building efficient and cost-effective hardware infrastructure became more challenging. However, this started opening new opportunities for the hardware-manufacturers. Furthermore, in the last few years, an increasing number of data-intensive HPC applications started shifting towards the pay as you go cloud infrastructure (e.g., Amazon Web Service, Penguin, R-HPC etc.) especially because of the elasticity of resources and reduced setup-time and cost.

As a consequence, there is a growing interest in all three communities, including HPC-scientists, hardware-manufacturers, as well as commercial cloud-service-providers, to develop cost-effective, high-performance testbeds that will drive the next generation scientific research involving huge amount of big data. Also, millions of dollars are being spent in programs, such as XSEDE and NSFCloud[1], where several academic organizations and manufacturing companies collaborated to address the challenges involved in developing novel distributed-cyber-infrastructures.

Despite this growing interest in both the scientific as well as the industrial community, there is a limited understanding of how the different types of hardware architectures impact the performance of these big data analytic software when applied to real world data and compute-intensive scientific workloads. Therefore, it is critical to evaluate different types of distributed-cyber-infrastructure in the context of real world, data-intensive, high performance, scientific workloads.

In this work, we use a large-scale de novo genome assembly as one of the most challenging and complex real world examples of a high performance computing workload that recently made its way to the forefront of big data challenges [**?**] [**?**]. De novo genome assembly reconstructs the entire genome from fragmented parts called short reads when no reference genome is available. The assembly pipeline of our Parallel Genome Assembler (PGA) involves a terabyte scale short read data analysis in a Hadoop job followed by a complex large-scale graph analysis with Giraph, thus, serving as a very good example of both data- as well as compute-intensive workload.

In this paper, we present the performance result of PGA atop three different types of clusters as follows: 1) a traditional HPC cluster, called SuperMikeII (located at LSU, USA) that offers 382 computing nodes connected with a 40-Gbps InfiniBand, 2) a regular datacenter architecture, called SwatIII (located at Samsung, Korea) that has 128 nodes connected with 10-Gbps Ethernet and 3) a new MicroBrick-based prototype ar-

[1] https://www.chameleoncloud.org/nsf-cloud-workshop/

chitecture, called CeresII that uses PCIe based communication (also located at Samsung, Korea).

Our performance analysis is divided into two parts as follows:

1) Firstly, we compare the individual impact of different hardware components over different clusters. We observed almost 70% improvement in the data-intensive graph-construction stage based on Hadoop and 35% improvement in the Giraph-based, memory-intensive graph-simplification stage in the SwatIII cluster over SuperMikeII by using SSD and scaling it up in terms of memory. SSD increases the disk I/O rate, thus reducing the I/O wait. Whereas, more memory increases the caching effect.
2) Secondly, we provide significant insight on efficient and cost-effective organization of different hardware components by modifying the underlying hardware organization of SwatIII cluster (the regular datacenter architecture) in many different ways to better understand the impact of different architectural balance. Here, we provide significant insight on cost-effective deployment of both scaled out and scaled up cluster, especially how to leverage SSDs in a cost effective manner. In this part, the new MicroBrick-based prototype architecture, CeresII is found to provide almost similar performance as SuperMikeII while yielding almost 2-times improvement in performance per dollar.

The rest of the paper is organized as follows: Section-**??** describes the prior works related to our study. In Section-**??** we define the motivation of our study, that is, the issues in a traditional supercomputer to process the big data workloads with respect to the current state of the art big data analytic software, in particular, Hadoop and Giraph. Section-**??** describes our evaluation methodology where we shed light on the experimental testbeds, the workload and the input data that we use in this work. In Section-**??**, we present our performance result by comparing the individual impact of different types of network, storage, and memory architectures over different clusters. Section-**??** compares the performance of PGA over different types of hardware organizations and architectural balances. Finally, in Section-**??** we conclude our study.

II. Related Work

Earlier studies [**?**] [**?**], as well as our experience show that state of the art big data analytic software can be useful for HPC workloads involving huge amount of big data. Jha [**?**] nicely showed the convergence between the two paradigms: the traditional HPC-software and the Apache Software Stack for big data analytic. As a consequence, a growing number of codes in several scientific areas, such as bioinformatics, geoscience, etc., are currently being written using Hadoop, Giraph, etc. [**?**]. Many of the traditional supercomputers also started using myHadoop [**?**] to provide the scientists an easy interface to configure Hadoop on-demand. Despite the growing popularity of using Hadoop and other software in its rich ecosystem for scientific-computing, there are very limited prior works that evaluated different distributed-cyber-infrastructures for these software when applied for data-intensive scientific workloads.

Impact of individual hardware component on Hadoop-workload: There are several performance analysis studies on using different types of hardware to accelerate the Hadoop job using the existing benchmark workloads. Vienne [**?**] evaluated the performance of Hadoop on different high speed interconnects such as 40GigE RoCE and InifiniBand FDR and found InfiniBand FDR, yields the best performance for HPC as well as cloud computing applications. Similarly, Yu [**?**] found improved performance of Hadoop in traditional supercomputers due to high speed networks.

Kang [**?**] compared the execution time of sort, join, WordCount, Bayesian, and DFSIO workloads using SSD and HDD and obtained better performance using SSD. Wu [**?**] found that Hadoop performance can be increased almost linearly with the increasing fraction of SSDs in the storage system. They used the TeraSort benchmark for their study. Additionally, they also showed that in an SSD-dominant cluster, Hadoop's performance is almost insensitive to different Hadoop performance parameters such as block-size and buffer-size. Moon [**?**] showed a significant cost benefit by storing the intermediate Hadoop data in SSD, leaving HDDs to store Hadoop Distributed File System (HDFS [**?**]) data. They also used the TeraSort benchmark in their study. A similar result can be found in the study by Li [**?**] and Krish [**?**] where SSDs are used to store temporary data to reduce disk contention and HDDs are used to store the HDFS data. They all reached the same conclusion as Moon [**?**]. Tan [**?**] also reached the similar conclusion for two other workloads including a Hive workload and an HBase workload.

All of the above studies have been performed either with existing benchmarks (e.g., HiBench [**?**]) or with enterprise-level analytic workloads, thus, they are unable to address the HPC aspect of Hadoop. Furthermore, very limited studies consider the in-memory graph processing frameworks (e.g., Giraph, etc.) even though, graph analysis is a core part of many analytic workloads.

Impact of overall architecture on Hadoop Workload: Michael [**?**] investigated the performance characteristics of the scaled out and scaled up architecture for interactive queries and found better performance using a scaled out cluster. On the other hand, Appuswamy [**?**] reached an entirely different conclusion in their study. They observed a single scaled up server to perform better than a 8-nodes scaled out cluster for eleven different enterprise-level Hadoop workloads including log-processing, sorting, Mahout machine learning, etc. Our study is significantly different in the following aspects. 1) Existing works focus on the data-intensive, enterprise-level Hadoop jobs (e.g., log-processing, query-processing, etc.). On the contrary, genome assembly is severely data- and a magnitude more compute-intensive. Additionally, it involves a large graph analysis which is extremely memory-intensive. 2) Existing works are limited in terms of job size. For example, the data size chosen in [**?**] can be accommodated in a single scaled up server. We did not put such a restriction on storage space or memory. Consequently, our performance comparison is more generic and realistic in a sense that, most of the time the choice of the cluster-size is driven by the data size rather than the performance. 3) Unlike the existing works, we consider the genome assembly workflow instead of choosing a single job, thus, working closer to the real world.

III. Motivation: Issue in running big data applications on traditional supercomputers

"Traditional supercomputers focused on performing calculations at blazing speeds have fallen behind when it comes to sifting through huge amount of Big Data."[2]
In this section, we briefly describe the programming model of two popular big data analytic software: Hadoop and Giraph followed by several issues that we observed on a traditional supercomputing environment while running these workload.

A. Programming models for big data analytic software

Hadoop and Giraph were originated as the open-source counterpart of Google's MapReduce [**?**] and Pregel [**?**] respectively. Both the software read the input data from the underlying Hadoop Distributed File System (HDFS) in the form of disjoint sets or partitions of records. Then, in the MapReduce abstraction, a user-defined map function is applied to each disjoint set concurrently to extract information from each record in the form of intermediate key-value pairs. These key-value pairs are then grouped by the unique keys and shuffled to the reducers. Finally, a user-defined reduce function is applied to the value-set of each key, and the final output is written to the HDFS. The MapReduce framework enables data- and compute-intensive applications to run large volume of distributed data sets over distributed compute nodes with local storage. On the other hand, Giraph uses the Bulk Synchronous Parallel model [**?**] where computation proceeds in supersteps. In the first phase of a superstep, Giraph leverages Hadoop-mappers when a user-defined vertex-program is applied to all the vertices concurrently. In the end of each superstep, each vertex can send a message to other vertices to initiate the next superstep. Alternatively, each vertex can vote to halt. The computation stops when all the vertices vote to halt unanimously in the same superstep. Giraph enables memory- and compute-intensive applications to upload data into distributed memories over different compute nodes.

B. Issues: big data application over traditional-supercomputers

In this section, we discuss the issues and limitations that we observed while running Hadoop and Giraph workload over traditional supercomputing resources.

1) Network issues: Traditional HPC clusters use an InfiniBand interconnect with high bandwidth and low latency to deliver short size of messages. In addition, InfiniBand-based networks use a standard 2:1 blocking ratio because compute intensive applications neither produce nor exchange much of data. However, Hadoop and Giraph were developed to work atop inexpensive clusters of commodity hardware based on Ethernet network to exchange large volume of data. Therefore, big data applications might suffer from bottleneck problems over HPC-clusters with typical high blocking ration networks

For example, during the shuffle phase of a Hadoop job there is a huge data movement across the cluster. However, in other phases the data movement is minimal in the network when mappers and reducers carefully consider the data locality. On the other hand, Giraph is more network-intensive. At the end of each superstep a huge amount of messages are passed across all the Giraph workers. Furthermore, every pair of workers uses a dedicated communication path between them that results in an exponential growth in the number of TCP connections with increase in the number of workers. At these points, the data network is a critical path, and its performance and latency directly impact the execution time of the entire job-flow.

2) Storage issues: In a traditional supercomputing environment, each node is normally attached with only one HDD. This configuration puts a practical limitation on total number of disk I/O operations per second (IOPS). On the other hand, the big data applications that consider data locality, typically involve a huge volume of data read/write from/to the Direct-Attached-Storage (DAS) of the compute nodes. Therefore, the applications might have variations of Hadoop (e.g., MyHadoop etc.) are capable to read/write the data from/to other parallel file system such as Lustre or GPFS which are mounted on dedicated I/O servers in an HPC environment. Although some variations of Hadoop are optimized to read/write large volume of data from/to other parallel file systems (e.g., Lustre and GPFS), thus taking advantage of huge amount of IOPS available through the dedicated I/O servers, the performance can be severely constrained by the network bottleneck. In this work, we use the HDFS as the distributed file system and use the local file system for the shuffled data.

Hadoop involves a huge amount of disk I/O in the entire job flow. For example, at the beginning (and the end) of a Hadoop job, all the mappers read (and the reducers write) a huge volume of data in parallel from (to) the HDFS which is mounted on the DAS device(s) of the compute nodes. Again, in the shuffle phase, a huge volume of intermediate key-value pairs is written by the mappers and subsequently read by the reducers to/from the local file system which is again mounted on the same DAS. Giraph, on the other hand, is an in-memory framework. It reads/writes a huge volume of data from/to the HDFS only during the initial input and the final output.

3) Memory issues: The traditional supercomputers, normally uses a 2GB/core memory as a standard configuration. This causes a significant trade-off between the number of concurrently running workers (mappers or reducers), and the memory used by each of them. Lower memory per worker (lower java heap space) can significantly increase the garbage collection frequency based upon the data size handled by each worker. Also, in case of Hadoop, smaller memory per worker puts a practical limitation on its buffer size resulting in a huge amount of data spilling to the disk in the shuffle phase, thereby making the job severely I/O-bound especially in case of HDD. Furthermore, the lower memory per node hinders the caching especially for a memory-intensive job, like graph analysis with Giraph that loads a huge amount of data in memory for iterative computation.

IV. Evaluation Methodology

A. Experimental Testbeds

Table-**??** shows the overview of our experimental testbeds. SuperMikeII, the LSU HPC-cluster, offers a total 440 computing nodes (running Red Hat Enterprise Linux 6 operating system). However, a maximum 128 can be allocated at a time

[2]http://spectrum.ieee.org/tech-talk/computing/hardware/ibm-redesigned-supercomputers-to-solve-big-data-problems

	SuperMikeII (Traditional Supercomputer)	SwatIII-Basic-HDD (Regular Datacenter)	SwatIII-Basic-SSD (Regular Datacenter)	SwatIII-Memory (Regular Datacenter)	SwatIII-FullScaleup-HDD/SSD (Regular Datacenter)	SwatIII-Medium-HDD/SSD (Regular Datacenter)	CeresII (Samsung MicroBrick with PCIe-communication)
Cluster category	HPC-cluster	Scaled out	Scaled out	Memory optimized	Scaled up	Medium-sized	Hyperscale
#Physical-Cores/node	16	16	16	16	16	16	2
DRAM(GB)/node	32	32	32	256	256	64	16
#Disks/node	1-HDD	1-HDD	1-SSD	1-SSD	7-HDD/SSD	2-HDD/SSD	1-SSD
Network	40-Gbps QDR InfiniBand (2:1 blocking)	10-Gbps Ethernet	10-Gbps Ethernet	10-Gbps Ethernet	10-Gbps Ethernet	10-Gbps Ethernet	10-Gbps Virtual Ethernet
Cost/node ($)	3804	4007	4300	6526	SSD:9226, HDD:7175	SSD:5068, HDD:4482	879
#DataNodes (DN) used for bumble bee genome (90GB)	15	15	15	15	4	2	31
#DataNodes (DN) used for human genome (452GB)	127	-	-	-	15	-	-

TABLE I: Experimental testbeds with different configurations

Hardware component	Used in	Cost ($)
Intel SandyBridge Xeon 64bit Ep series (8-cores) processor	SuperMikeII, SwatIII	1766
Intel Xeon E3-1220L V2 (2-cores) processor	CeresII	384
Western Digital RE4 HDD	SuperMikeII	132
Western Digital VelociRaptor HDD, 500GB	SwatIII HDD-variants	157
Samsung 840Pro Series SATAIII SSD, 500GB	SwatIII SSD-variants	450
Samsung 840Pro Series SATAIII SSD, 250GB	CeresII	258
Samsung DDR3 16GB memory module	SwatIII, CeresII	159
32GB 1600MHz RAM (decided by Dell)	SuperMikeII	140 (Average)

TABLE II: Hardware components used in different cluster configurations, and their cost

to a single user. SwatIII is a regular datacenter with 128 compute nodes (running on Ubuntu 12.0.4 LTS). However, we use maximum 16 nodes. We configure SwatIII in seven different ways to study the pros and cons of different hardware components individually and from the viewpoint of their overall organization in scaled out and scaled up environment. For the sake of convenience, we call each configuration with meaningful name as shown in Table-**??**. CeresII is a novel hyperscale (”the ability of an architecture to scale appropriately as increased demand is added” [3]) system based on Samsung MicroBrick. In our study, we evaluated it as a next generation cluster which is found to resolve many of the problems in the existing HPC-cluster and the regular datacenter. It is to be noted, we always use homogeneous configuration across any cluster. We reported the performance and the price of different clusters in terms of the Hadoop datanodes (DN) only. For the masternode, we always use a minimal configuration based upon the resources in the cluster. In the subsequent sections we use the term node and datanode interchangeably.

Table-**??** shows the hardware specification used in different clusters and their cost[4]. We calculated the cost of each node of each cluster configuration (shown in Table-**??**) based upon this. We use the hardware configuration of SuperMikeII as the baseline and compare all the performance results of SwatIII and CeresII, to this baseline. Each node of SuperMikeII and any SwatIII variants has the same number of processors and cores, in particular, 2 8-core Intel SandyBridge Xeon 64bit Ep series processors. To do a fair comparison, we disabled the HyperThreading in the SwatIII as SuperMikeII does not have it.

The first three variants of SwatIII, SwatIII-Basic-HDD, SwatIII-Basic-SSD and SwatIII-Memory, are used to evaluate the impact of each individual component of a compute cluster i.e., network, storage and the memory. SwatIII-Basic-HDD is similar in every aspect to SuperMikeII except it uses 10-Gbps Ethernet instead of 40-Gbps InfiniBand as in SuperMikeII. SwatIII-Basic-SSD, as the name suggests, is storage optimized and uses one SSD per node instead of one HDD as in SuperMikeII and SwatIII-Basic-HDD. On the other hand, SwatIII-Memory is both memory and storage optimized, i.e., it uses 1-SSD as well as 256GB memory per node instead of 32GB as in the previous three clusters.

Unlike SuperMikeII or SwatIII-Basic and -Memory which use only one DAS device per node, SwatIII-FullScaleup-HDD/SSD and SwatIII-Medium-HDD/SSD use more than one DAS device (Either HDD or SSD as the names suggest) per node. They also vary in terms of total amount of memory per node. However, the total amount of storage and memory space is almost same across all these clusters. We use these clusters to mainly evaluate different types of hardware organizations and architectural balances from the viewpoint of scaled out and scaled up configurations. It is to be noted in case of SwatIII, we use the term scaled up and out in terms of amount of memory and number of disks. The numbers of cores per node is always same. In either of SwatIII-FullScaleup and SwatIII-Medium, we use JBOD (Just a Bunch Of Disks) configuration as per the general recommendation by [**?**], Cloudera, Hortonworks, Yahoo, etc. Use of the JBOD configuration eliminates the limitation on disk I/O speed, which is constrained by the speed of the slowest disk in case of a RAID (Redundant Array of Independent Disk) configuration. As mentioned in [**?**], JBOD is found to perform 30% better than RAID-0 in case of HDFS write throughput.

The last one, CeresII, is a novel scaled out architecture based on Samsung MicroBrick. It is an improvement over CeresI [**?**]. Currently, it is in prototype phase. A single MicroBrick chassis of CeresII consists of 22 computation-modules (or, compute servers). Each module consists of one intel Xeon E3-1220L V2 processor with two physical cores,

[3]Source: https://en.wikipedia.org/wiki/Hyperscale

[4]Price information is collected from http://www.newegg.com and http://www.amazon.com. The minimum listed price is considered as per Jun 17, 2015.

16GB DRAM module (Samsung), and one SATA-SSD (Samsung). Each module has several PCI-express (PCIe) ports. Unlike SuperMikeII (traditional supercomputer) and SwatIII (regular datacenter), all the compute servers of CeresII in a single chassis are connected to a common PCIe switch to communicate with each other. The highly dense servers per chassis in CeresII have a total 44 physical cores connected through PCIe comparing to 16 physical cores per node as in SuperMikeII and SwatIII. Furthermore, the use of SSD reduces the I/O wait and 8GB RAM per physical core improves the access parallelism.

B. Understanding the workload

De novo genome assembly refers to the construction of an entire genome sequence from a huge amount of small, overlapping and erroneous fragments called short read sequences while no reference genome is available. The problem can be mapped as a simplified de Bruijn graph traversal [**?**]. We classified the de novo assembly in two stages as follows: *a*) Hadoop-based de Bruijn graph-construction and *b*) Giraph-based graph-simplification. In this section, we provide a brief overview of each stage of the assembler.

1) Hadoop-based De Bruijn graph-construction (data- and compute-intensive workload): After filtering the actual short reads (i.e., the line containing only nucleotide characters A, T, G, and C) from a standard fastq file, an extremely shuffle-intensive Hadoop job creates the graph from these reads. the Hadoop map task divides each read into several short fragments of length k known as k-mers. Two subsequent k-mers are emitted as an intermediate key-value pair that represents a vertex and an edge (emitted from that vertex) in the de Bruijn graph. The reduce function aggregates the edges (i.e the value-list) of each vertex (i.e., the k-mer emitted as key) and, finally, writes the graph structure in the HDFS in the adjacency-list format. Based upon the value of k (determined by biological characteristics of the species), the job produces huge amount of shuffled data. For example, for a read-length of 100 and k of 31 the shuffled data size is found to be 20-times than the original fastq input. On the other hand, based upon the number of unique k-mers, the final output (i.e., the graph) can vary from 1 to 10 times of the size of the input.

2) Giraph-based Graph Simplification (memory- and compute-intensive workload): The large scale graph data structure produced by the last MapReduce stage is analyzed here. This stage consists of a series of memory-intensive Giraph jobs. Each Giraph job consists of three different types of computation: compress linear chains of vertices followed by removing the tip-structure and then the bubble-structure (introduced due to sequencing errors) in the graph. Giraph can maintain a counter on the number of supersteps and the master-vertex class invokes each type of computation based on that.

We compress the linear chains into a single vertex using a randomized parallel list ranking algorithm of [**?**] implented with Giraph. The computation proceeds in rounds of two supersteps until a user defined $limit$ is reached. In one superstep, each compressible vertex with only one incoming and outgoing edge is labeled with either $head$ or $tail$ randomly with equal probability and send a message containing the tag to the immediate predecessor. In the next superstep, all the $head$-$tail$ links are merged, that is, the $head$-$k$mer is extended (or, appended) with the last character of the $tail$-$k$mer and the $tail$ vertex is removed. Each vertex also maintain a frequency counter which increments after each extension to that vertex. After the compression, all the tip-structures in the graph are removed in two supersteps. The first superstep identifies all the vertices with very short length (less than a threshold) and no outgoing edge as tips which are removed in the second superstep. Finally, the bubbles-structures are resolved in another two supersteps. In the first superstep, the vertices with same predecessor and successor as well as very short length (less than a threshold) are identified as bubbles. They send a message containing their id, value, and frequency to their corresponding predecessors. The predecessor employs a Levenshtein-like edit distance algorithm. If the vertices are found similar enough, then the lower frequency vertex is removed.

	Job Type	Input	Final output	# jobs	Shuffled data	HDFS Data
Graph Construction	Hadoop	90GB (500-million reads)	95GB	2	2TB	136GB
Graph Simplification	Series of Giraph jobs	95GB (71581898 vertices)	640MB (62158 vertices)	15	-	966GB

TABLE III: Moderate-size bumble bee genome assembly

	Job Type	Input	Final output	# jobs	Shuffled data	HDFS Data
Graph Construction	Hadoop	452GB (2-billion reads)	3TB	2	9.9TB	3.2TB
Graph Simplification	Series of Giraph jobs	3.2TB (1483246722 vertices)	3.8GB (3032297 vertices)	15	-	4.1TB

TABLE IV: Large-size human genome assembly

C. Input Data

High throughput next generation DNA sequencing machines (e.g., Illumina Genome Analyzer) produce a huge amount of short read sequences typically in the scale of several Gigabytes to Terabytes. Furthermore, the size of the de Bruijn graph built from these vast amount of short reads may be a magnitude higher than the reads itself making the entire assembly pipe line severely data-intensive.

In this paper, we use two genome data sets: 1) a moderate size bumble bee genome data (90GB) and 2) a large size human genome data (452GB). The corresponding graph sizes are 95GB and 3.2TB based upon the number of unique k-mers (using $k = 31$ in both the cases) in the data set. The bumble bee genome is available in Genome Assembly Gold-standard Evaluation (GAGE [**?**]) website[5] in fastq format. The Human genome is available in NCBI website with accession number SRX016231[6]. Table-**??** and **??** show the details of the data size in the assembly pipeline for both the genomes.

[5]http://gage.cbcb.umd.edu/

[6]http://www.ncbi.nlm.nih.gov/sra/SRX016231[accn]

D. Hadoop configurations and optimizations

Since our goal is to evaluate the underlying hardware components and their organizations, we avoid any unnecessary change in the source code of Hadoop or Giraph. We use Clouera-Hadoop-2.3.0 and Giraph-1.1.0 for the entire study and use the Cloudera-Manager-5.0.0 for monitoring the system behavior. In this section we provide our evaluation methodology in details. To evaluate the relative merits of different clusters, we started with tuning and optimizing different Hadoop parameters to the baseline, that is a traditional supercomputing environment, SuperMikeII. Then, we further modified the parameters with change in the underlying hardware infrastructure in SwatIII cluster to optimize the performance in each configuration. A brief description of the Hadoop-parameters that we changed are as follows.

Number of concurrent YARN containers: After performing rigorous testing, we observed, that for SuperMikeII and SwatIII-Basic-HDD (1-HDD/DN cases), 8-conainers/DN (i.e., half of total cores/node) show the best result. For any other cluster, number of concurrent containers per datanode is kept equal to the number of cores per node.

Amount of memory per container and Java-heap-space: In each node in any node of any cluster, we kept 10% of the memory for the system use. The rest of the memory is equally divided among the launched containers. The Java heap space per worker is always set to lower than memory per container as per normal recommendation.

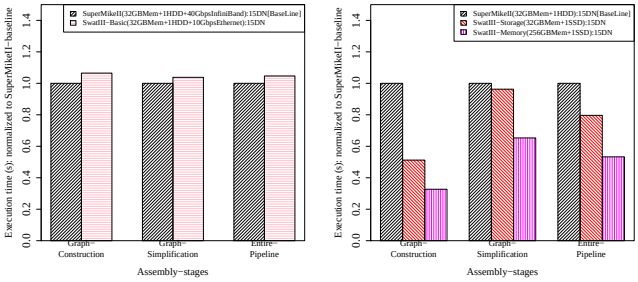
Total number of Reducers: Based on the observation of job profiles, we observed that 2-times of reducers than number of concurrent containers produce good performance in general cases.

Giraph workers: We setup the number of Giraph workers according to the number of concurrently running YARN containers.

Other Giraph parameters: We always used enough memory to accommodate the graph structure in memory and always avoided using the out-of-core execution feature of Giraph, which writes huge data to the disk. We also avoided using the checkpointing feature for the same reason.

V. Individual Impact of Different Hardware Component

In this section, we compare the impact of hardware components, such as, network, storage, and memory individually on our benchmark genome assembler. To do that, we use 16 nodes in both SuperMikeII and SwatIII. Each node in both the clusters has 16 processing cores. We started with comparing the impact of the network between SuperMikeII and SwatIII-Basic-HDD. Then, we further optimized those 16 nodes of SwatIII cluster incrementally in terms of storage by providing SSD (named as SwatIII-Basic-SSD) and then providing more memory to each node (named as SwatIII-Memory). It is worthy to mention here that 1-HDD/node puts a practical limit on the number of concurrently running Yarn-containers due to I/O bottleneck. As a consequence, both in SuperMikeII and SwatIII-Basic-HDD, we observed the best performance by using only 8 Yarn-containers concurrently in each node, i.e., only half of the number of cores per node. The execution-times reported in this section are the means of at least 3 runs of the assembler on each different hardware configuration.



(a) Effect of network (InfiniBand vs Ethernet)

(b) Effect of local storage (HDD vs SSD) and size of DRAM

Fig. 1: Impact each individual hardware component on execution time of different stages of the assembly pipeline in 15-DN

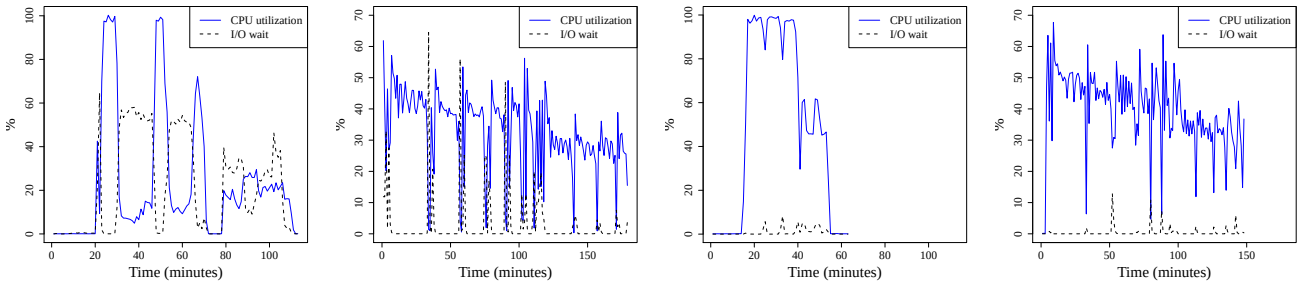
A. Effect of Network: InfiniBand vs Ethernet

Figure-**??** compares the impact of network interconnect on each stage of PGA's genome assembly pipeline while assembling a 90GB bumble bee genome. The execution time is normalized to the SuperMikeII-baseline. We did not find any visible performance difference (less than 2%) on any of the stages of our assembly pipeline even though SuperMikeII uses 40-Gbps QDR InfiniBand whereas SwatIII-Basic-HDD uses a 10-Gbps ethernet. The reason is as follows: although the average latency in SuperMikeII is almost $1/14$ of that in SwatIII (0.014ms in SuperMikeII compare to 0.2ms in SwatIII), the average effective bandwidth between any two compute nodes of SuperMikeII was found to be almost 10-times lower than that of SwatIII (954Mbit/s in SuperMikeII, whereas 9.2Gbit/s in SwatIII) because of the $2:1$ blocking ratio in the InfiniBand network.

B. Effect of local storage device: HDD vs SSD

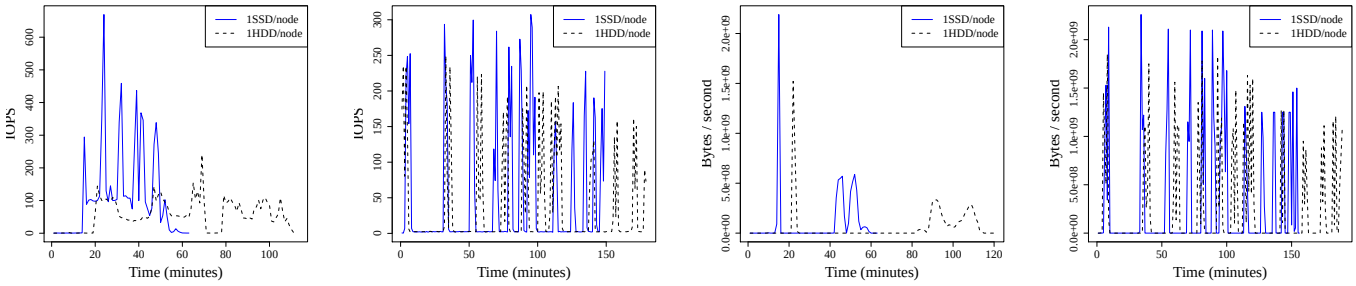
Figure-**??** compares the execution time of SwatIII-Basic-SSD (1-SSD/node) to the SuperMikeII-baseline (1-HDD/node). The second column of each stage of the assembler in Figure-**??** shows the impact of using SSD in that stage of the assembly. We observed almost 50% improvement in the shuffle intensive graph-construction stage because of reduced I/O wait. However, graph-simplification, a series of in-memory Giraph jobs (that read/write data only to the HDFS), is not affected much (less than 3%) by storage optimization with SSD.

Actually, the shuffle phase of the Hadoop job experiences a huge amount of I/O wait. A huge number of I/O threads work concurrently to spill a huge amount of data to the disk (by the mappers) and subsequently read by the reducers. Giraph shows I/O wait only when it reads/writes a large graph from/to HDFS (Figure-**??**). I/O wait is significantly reduced using SSD (Figure-**??** and -**??**) instead of HDD which improves the Hadoop performance significantly. However, for Giraph we did not observe any significant performance improvement using SSD because of very less I/O wait. Basically, an SSD increases the disk IOPS per datanode by 7 to 8 times than an HDD especially during the shuffle phase of Hadoop which writes huge



(a) Hadoop-based graph-construction in SwatIII-Basic-HDD (1-HDD/node)
(b) Giraph-based graph-simplification in SwatIII-Basic-HDD (1-HDD/node)
(c) Hadoop-based graph-construction in SwatIII-Basic-SSD (1-SSD/node)
(d) Giraph-based graph-simplification in SwatIII-Basic-SSD (1-SSD/node)

Fig. 2: CPU-Utilization and I/O Wait characteristics in SwatIII-Basic-HDD (1-HDD/node) and SwatIII-Basic-SSD (1-SSD/node)



(a) Write-IOPS/DN in graph-construction in SwatIII-Basic-HDD (1-HDD/node) and SwatIII-Basic-SSD (1-SSD/node)
(b) Write-IOPS/DN in graph-simplification in SwatIII-Basic-HDD (1-HDD/node) and SwatIII-Basic-SSD (1-SSD/node)
(c) HDFS-Write-throughput in graph-construction in SwatIII-Basic-HDD (1-HDD/node) and SwatIII-Basic-SSD (1-SSD/node)
(d) HDFS-Write-throughput in graph-simplification in SwatIII-Basic-HDD (1-HDD/node) and SwatIII-Basic-SSD (1-SSD/node)

Fig. 3: Comparison of disk-write IOPS (write) on Local File System (of each datanode) and I/O throughput for HDFS-write (across all datanodes) for HDD and SSD. The shuffle phase of Hadoop gains maximum from SSD.

amount of data to the local file system as shown in Figure-**??**. In case of Giraph, the corresponding improvement is 1.5-times as shown in Figure-**??**. Considering the I/O throughput to HDFS, we also observed 1.5-times improvement in case of SSD for both Hadoop and Giraph as shown in Figure-**??** and -**??**. Giraph, which writes data to the HDFS only, shows the similar I/O characteristics for both IOPS per datanode (DN) and HDFS I/O throughput because they are related by the equation: $HDFS\_IO\_Throughput = IOPS\_per\_DN \times Bytes\_per\_IO \times \#DN$, where $Bytes\_per\_IO$ is the characteristics of the disk. However, the HDFS throughput characteristics across all the DN varies significantly from the IOPS per DN in case of shuffle-intensive Hadoop job, which writes lots of data to the local file system.

C. Effect of size of DRAM

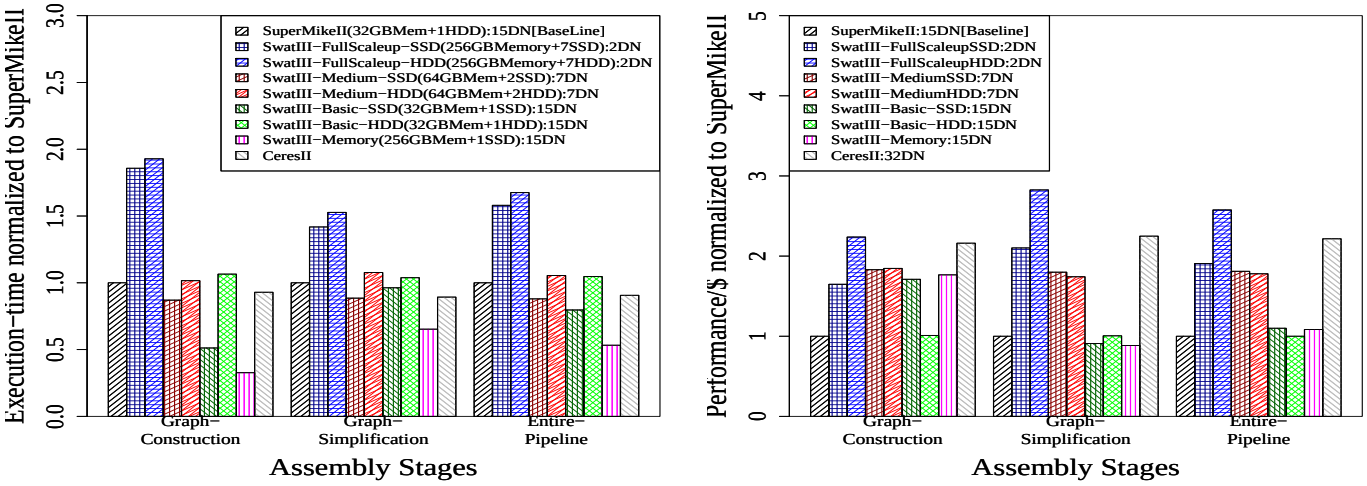
The third columns of Figure-**??** shows the impact of increasing the amount of memory per node. We observed almost 20% improvement in the initial graph-construction phase from SwatIII-Basic-SSD, consequently, almost 70% improvement to the baseline. In the Giraph phase, the corresponding improvement is almost 35%. The improvement is because of the caching especially in case of Giraph, where computation proceeds in iterative supersteps. A huge amount of data is kept in cache and is fetched upon requirement during the next compute-superstep.

VI. Comparison among Different Hardware-Organizations

In this section, we compare the performance of different cluster architecture in terms of raw execution time as well as performance-to-price. As mentioned earlier, the execution-times are the means of at least 3 runs of the assembler on each different hardware configuration.

A. Execution time comparison between SuperMikeII and SwatIII variants (with moderate-size bumble bee genome)

Figure-**??** shows the relative merits of different cluster architectures in terms of raw execution time. Observe that we always keep the total aggregated storage and memory space almost same across all the clusters (Except the SwatIII-Memory). The basic assumption behind this experimental setup is that the total amount of data should be held in its entirety in any of the cluster, hence we did not compromise this with the storage or memory space. Furthermore, the choice of the cluster in a cloud scenario is often driven by the sheer volume of the data rather than the performance. The observations are as



(a) Execution time (Lower execution time means better performance)

(b) Performance-to-Price (Higher performance per dollar is better)

Fig. 4: Performance comparison among different type of cluster architectures in terms of normalized execution time and performance-to-price

follows: *1*) **SwatIII-Basic:** As discussed earlier in Section-**??**, for Hadoop, the SSD variant of this scaled out cluster (32GB-RAM + 1-disk/DN & 16-DN) shows 2x speedup over the baseline whereas the HDD variant performs similar to the baseline. For Giraph, both of them perform almost similar to the baseline. *2*) **SwatIII-FullScaleup:** This scaled up (256GB-RAM + 7-disks/DN) small sized cluster (only 2-DN) takes the maximum time for any workload because of least number of processing cores among all. Observe that for the Hadoop job, both SSD and HDD variants of this scaled up cluster perform similarly, which is in contrast with scaled out cluser (SwatIII-Basic). We discuss it in more detail in Section-**??**. *3*) **SwatIII-Medium:** The HDD variant of this cluster (64GB-RAM + 2-disks/DN & 7-DN) perform almost similarly to the baseline for both Hadoop and Giraph even though the total number of cores in the cluster is half of the baseline. It is because 2-HDDs and 64GB RAM per node increase the IOPS and the caching respectively. The SSD variant shows slightly better result than the HDD because of further increase in IOPS. *4*) **SwatIII-Memory:** The performace characteristics of this cluster is discussed earlier in Section-**??**. It is no surprise that this configuration (256GB-RAM + 1-SSD/DN & 16DN) shows the lowest execution time among all because of the huge amount of memory available across the cluster.

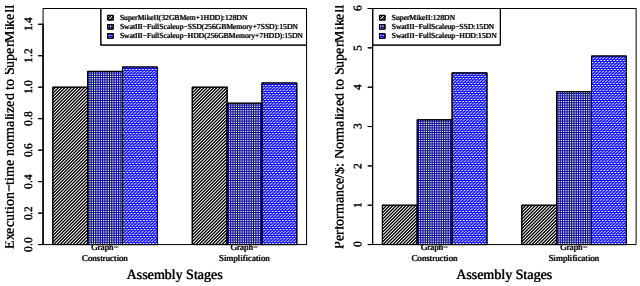
B. Performance-to-Price comparison between SuperMikeII and SwatIII variants (with the bumble bee genome)

We consider the performance as the inverse of the execution time and divided it by the total cost of the cluster to get the performance/$. Since all the clusters have same amount of storage and memory space (except SwatIII-Memory), none of them get any price benefit over another because of the total storage or memory space. Rather, we compare the performance to price from the view point of a proper architectural balance among number of cores, number of disks, and amount of memory per node. We did not consider the cost of network for a fair comparison with SuperMikeII, the public HPC-cluster that is shared among many users.

Figure-**??** compares the performance/$ metric among all the clusters. The observations are as follows: *1*) **SwatIII-Basic:** For Hadoop, the SSD variant of this scaled out cluster shows 2-times better performance/$ comparing to the baseline as well as to its HDD variant. However, for Giraph, it does not add any benifit. The HDD variant, as expected, shows similar result as the baseline. *2*) **SwatIII-FullScaleup:** Although the performance of this scaled up cluster is the lowest in terms of execution time, it shows high performance/$ for both Hadoop and Giraph. For Hadoop, the SSD and HDD variants show 1.5 and 2.5 times benefit to the baseline respectively. For Giraph, the corresponding benefit is 2 and 3 times respectively for SSD and HDD. Due to the similar execution time in both HDD and SSD variant of this scaled up cluster, the HDD variant obviously shows better performance/$ than the SSD variant. This is again in contrast with the scaled out (SwatIII-Basic) case *3*) **SwatIII-Medium:** Both the HDD and SSD variant of this configuration shows similar result, almost 2-times better than the baseline for both Hadoop and Giraph. Considering both performance and the price, it is the most optimal configuration in our evaluation. *4*) **SwatIII-Memory**: For Hadoop, it shows almost 2-times benefit to the baseline. However, once SSD is used as the underlying storage (comparing to SwatIII-Basic-SSD) more memory do not add any advantage in terms of performance/$. On the other hand, for Giraph, it does not have any impact on performance/$ comparing to the baseline.

C. Comparing SuperMikeII and SwatIII (with large human-genome)

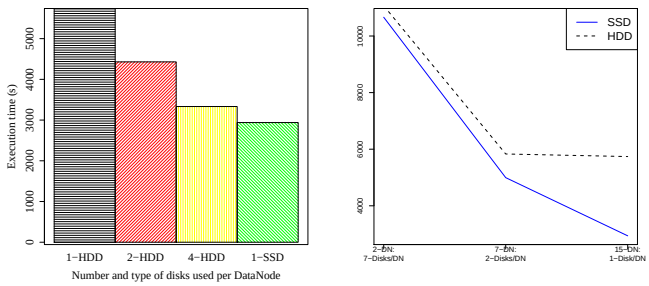
The large human genome (452GB) produces huge amount of shuffled data (9.9TB) as well as the graph data (3.2TB). We did a stress-testing with this huge data with the maximum available resources. We use 127 datanodes in SuperMikeII (32GB-RAM + 1-disk/DN) and 15 datanodes in the SwatIII-



(a) Execution time (Lower execution time is better)

(b) Performance-to-Price (Higher performance per dollar is better)

Fig. 5: Comparison of different types of cluster architecture for human genome assembly pipeline.



(a) Performance trend using 1, 2 and 4 HDD(s) and 1-SSD per node using 15 datanodes

(b) Performance trend for SSD and HDD using 1, 2, 7 disks per node in 15, 7 and 2 datanodes

Fig. 6: Performance trend using HDD and SSD in Hadoop

Full-Scaleup (256GB-RAM + 7-disks/DN) HDD and SSD. Figure-**??** and **??** shows the execution time and the performance/$ respectively for the Hadoop and Giraph stage of the human genome assembly pipeline. The observations are as follows: *1*) For Hadoop, the 127-DN of SuperMikeII (2032-cores) show only 15-17% better performance than 15-DN (240 cores) of SwatIII-FullScaleup cluster (any variant) while using almost 9-times more cores. The reasons behind the lower performance in SuperMikeII are both the huge I/O and network bottleneck as discussed earlier in Section-**??** and -**??**. Again, observe that for this large data set also, both the SSD and HDD variants of SwatIII-FullScaleup perform similarly as observed in Section-**??**. *2*) For the Giraph-based graph-simplification stage also, SuperMikeII did not show the expected performance mainly because of the network bottleneck. To anlyze the terabyte scale graph, Giraph passes a huge amount of messages across the workers, which experience significant bottleneck by the 2:1 blocking ratio in the InfiniBand network and the lower effective bandwidth between compute nodes of SuperMikeII. *3*) In terms of performance per dollar the scaled up configuration shows huge benefit over the baseline. Again, because of the similar execution time, the HDD variant of SwatIII-FullScaleup shows better performance per dollar than the SSD variant. Comparing to the baseline, SwatIII-FullScaleup-SSD and -HDD shows almost 3 and 4.5 times better performance per dollar respectively for the data- and compute-intensive Hadoop workload. For the memory- and compute-intensive Giraph workload, the corresponding benefit is 4 and 5 times respectively.

D. Performance of SSD in scaled out and scaled up cluster

Storage optimized, scaled up cloud instances frequently come with 4 to 8-SSDs per instance to improve the performance, consequently incurring high setup-cost as well as service-charge. For example, AWS-i2.8xlarge[7] offers 8-SSDs per instance at a rate of $6.82/hour, which is one of the high-cost AWS-EC2-instances. But, is it the effective way to leverage the SSDs? Although SSDs shows huge performance gain over HDDs in a scaled out scenario, our observations (on both moderate and large size data) earlier in this section obviously raise the question that how much effective SSD is in a scaled up cluster? In this section, we compare the performance characteristics of HDD and SSD from the perspective of scaled out and scaled up configuration.

Figure-**??** compares the performance of a single SSD and increasing number of HDDs per node for the Hadoop-based graph-construction stage of the bumble bee genome assembly pipeline while using a total of 15-DN. The performance improves almost linearly by increasing the number of HDDs per datanode in the cluster. On the other hand, 4-HDDs per node shows similar performance (only 5% variation) with a single-SSD per node. At this point the job is CPU-bound, and adding more disk(s) to the datanodes does not improve the performance. Consequently, we did not expect any significant performance improvement after this point (that is, more than 4-disks per node). Both Figure-**??** as well as Figure-**??** substantiate our claim for moderate-size bumble bee and large-size human genome data. It can be clearly observed that for both the data, both the HDD and SSD variant of SwatIII-FullScaleup perform similarly where each datanode is equipped with 7-disks. However, SSD showed significantly better performance and scalability than HDD when we scaled out by adding more compute nodes to the cluster because lower number of HDDs make the job I/O bound as shown in Figure-**??**.

E. Performance of CeresII: Samsung MicroBrick with PCIe communication

In this section, we evaluate a Samsung MicroBrick-based novel architecture, called CeresII. CeresII is in prototype phase and is an improvement over CeresI [**?**]. As mentioned before, CeresII uses 2 physical cores, 1-SSD and 16GB memory per computation module (or compute-server) and uses a PCIe-based interface to communicate among high-density compute-servers in a chassis. For communication between different chassis it uses 10-Gbps Ethernet.

To assemble the 95GB bumble bee genome we use 32 compute-servers of CeresII as Hadoop datanodes. The last columns in Figure-**??** and Figure-**??** show the execution-time and performance per dollar respectively for CeresII for different stages of the 90GB bumble bee genome assembly. CeresII shows the similar execution time to the baseline in every stage of the assembly pipeline while giving almost 2-times improvement in performance/$.

From the performance-study between SuperMikeII and

[7]http://aws.amazon.com/ec2/pricing/

SwatIII, we noticed a huge trade-off between the execution time and the performance/$. For example, even though the full scaled up small-sized clusters (2-DNs cases) show extremely low performance, they show a magnitude higher performance/$. We also concluded that the medium sized clusters (7-DNs) are well balanced considering both performance and cost. On the other hand, CeresII shows similar execution time as the medium sized clusters (which is eventually same as the baseline) and better performance per dollar than this. Moreover, the Samsung MicroBrick based architecture consumes less power, and obviously, less space [**?**]. Hence, we conclude that CeresII shows the maximum benefit in terms of TCO (total cost of ownership) among all the clusters.

VII. Conclusion and Future-work

In this paper, we analyze the performance characteristics of two popular state of the art big data analytic software, Hadoop and Giraph, on top of different distributed-cyber-infrastructures with respect to a real world data- and compute-intensive HPC workload. We pointed out several limitations in a traditional supercomputing environment both in terms of the individual hardware components as well as their overall organizations. Although the network usage and traffic in a public cluster puts a significant bottleneck on the performance of big data analytic workloads, it can be significantly improved by changing the underlying storage and memory organization. Also, the novel MicroBrick-based CeresII-architecture that uses a PCIe based communication interface between highly dense compute-servers in a single chassis can be a good future direction to alleviate the network bottleneck.

We also pointed out the huge trade-off between the performance and the price that the data- and memory-intensive HPC applications experience with the taditional deployment of the existing hardware components. The existing distributed-cyber-infrastructures should be modified significantly in order to provide good performance while staying within the budget. It is indeed the future direction of our work. CeresII, from that perspective also provides a very good initial starting point.