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 applications over HPC environment. However, understanding and designing the hardware environment that these data- and compute-intensive applications require for good performance is 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 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 evaluate the impact of both individual hardware components as well as overall organization, we changed the configuration of SwatIII 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 SwatIII 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. Here, The MicroBrick-based CeresII prototype shows similar performance as SuperMikeII while giving more than 2-times improvement in performance/$ in the entire benchmark test.

I. Introduction

Since experimental facilities at large-scale sciences, such as biology, astronomy etc., have produced unprecedented amount of data, scientific communities encounter new challenges in terms of storage and optimal processing. The fundamental computation-model of these scientific applications is rapidly changing 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.

Consequently, 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, 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, and commercial cloud-vendors, to develop cost-effective, high-performance testbeds that will drive the next generation scientific research involving big data. Also, millions of dollars are being spent in programs, such as XSEDE and NSFCloud, 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 and 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 example of a HPC workload that recently made its way to the forefront of big data challenges [1] [2]. 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), 2) a regular datacenter architecture, called SwatIII (located at Samsung, Korea) and 3) a novel MicroBrick-based prototype architecture, called CeresII that uses PCIe based communication (also located at Samsung, Korea).

Our performance analysis is divided into two parts:

1) Firstly, we compare the individual impact of different hardware components over different clusters. We observed 70% improvement in the Hadoop-based data-intensive graph-construction stage 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 in many different ways to better understand the impact of different architectural balance. Here, we provide significant insight on cost-effective deployment of different hardware components, especially how to leverage SSDs in a cost effective manner. In this part, the new MicroBrick-based CeresII cluster is found to provide almost similar performance as SuperMikeII while yielding almost 2-times improvement in performance/$.

The rest of the paper is organized as follows: Section-II describes the prior works related to our study. In Section-III we define the motivation of our study, that is, the issues in running big data workloads on traditional supercomputer. Section-IV describes our evaluation methodology including the experimental testbeds, the workload and the input data that we use in this work. In Section-V, shows the individual impact of different types of network, storage, and memory architectures over different clusters. Section-VI compares the performance of PGA over different types of hardware organizations. Finally, in Section-VII we conclude our study.

II. Related Work

Earlier studies [3] [4], as well as our prior experience [5], [6] show that state of the art big data analytic software can be useful for data-intensive HPC workloads. 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. Despite the growing popularity of using these big data analytic software 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 [7] and Yu [8] evaluated the performance of Hadoop on different high speed interconnects, such as, 40GigE RoCE and InifiniBand FDR or QDR. In both the studies, the authors argued that InfiniBand produces better result than Ethernet.

Kang [9] compared the execution time of sort, join, WordCount, and DFSIO workloads using SSD and HDD and obtained better performance using SSD. Wu [10] found that Hadoop performance can be increased almost linearly with the increasing fraction of SSDs in the storage system using the TeraSort benchmark. Moon, using the same TeraSort benchmark [11] showed a significant cost benefit by storing the intermediate Hadoop data in SSD, leaving HDDs to store Hadoop Distributed File System (HDFS) data. Li [12], Krish [13] and Tan [14] also reached the same conclusion as Moon [11] for other enterprise level workloads, such as, Hive queries, HBase enabled TPC-H queries etc.

All of the above studies have been performed either with existing benchmarks (e.g., HiBench [15]) 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 [16] 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 contrary, Appuswamy [17] concluded that a single scaled up server performs better than a 8-nodes scaled out cluster for many different enterprise-level Hadoop workloads, e.g., log processing, 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 [17] 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.

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

In this section, we briefly describe the programming model of two popular big data analytic software: Hadoop and Giraph and discuss several issues that we observed on a traditional HPC 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 HDFS in the form of disjoint sets 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 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 load data into distributed memory over different compute nodes.

B. Issues: big data application over traditional-supercomputers

1) Network issues: Traditional HPC clusters use an InfiniBand interconnect with high bandwidth and low latency to deliver short size of messages. A atandard 2:1 blocking ratio is used because compute-intensive applications neither produce nor exchange much data. On the contrary, the current programming model emphasizes bandwidth. Therefore, big data applications might suffer from bottleneck problems over HPC-clusters with typical high blocking ratio networks.

For example, the shuffle phase of Hadoop involves 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 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 Giraph workers use a dedicated communication path between them. It 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 suffer from I/O wait. Although some variations of Hadoop (e.g., [18]) 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. Additionally it will incur extra cost to the cluster. For simplicity, 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 of 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 graph analysis job with Giraph that loads a huge amount of data in memory for iterative computation.

IV. Evaluation Methodology

A. Experimental Testbeds

Table-I shows the overview of our experimental testbeds. SuperMikeII, the LSU HPC-cluster, offers a total 440 computing nodes. However, a maximum 128 can be allocated at a time to a single user. SwatIII is a regular datacenter with 128 compute nodes. However, we use maximum 16 nodes. We configure SwatIII in seven different ways (identified by a meaningful name as shown in Table-I) to study the pros and cons of different hardware components individually and their overall organization (scaled up and scaled out). CeresII is a novel hyperscale system based on Samsung MicroBrick. In our study, we evaluated it as a next generation cluster which is found to resolve many issues in existing supercomputers

	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. 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.

and the regular datacenter. 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. In the subsequent sections we use the term node and datanode interchangeably.

Table-II shows the hardware specification of each cluster and their cost. We calculated the cost of each node of each cluster configuration (shown in Table-I) 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 E5 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 evaluate different types architectural balances from the viewpoint of scaled out and scaled up configurations. 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 Medium, we use JBOD (Just a Bunch Of Disks) configuration as per the general recommendation by [19], Cloudera, etc. JBOD does not limit the I/O speed by the slowest disk as RAID (Redundant Array of Independent Disk) configuration. As a result, JBOD is found to perform 30% better than RAID-0 in case of HDFS write throughput [19].

The last one, CeresII, is a novel scaled out architecture based on Samsung MicroBrick [20]. It is an improvement over CeresI [21]. Currently, it is in prototype phase. A single MicroBrick chassis of CeresII consists of 22 computate server. Each server consists of one intel Xeon E3-1220L V2 processor with two physical cores, 16GB DRAM module (Samsung), and one SATA-SSD (Samsung). Each server has

several PCIe (PCI-express) ports. 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 in SuperMikeII and SwatIII. Furthermore, the use of SSD reduces the I/O wait and 8GB RAM per core improves the access parallelism.

B. Understanding the workload

De novo genome assembly problem can be interpreted as a simplified de Bruijn graph traversal problem. We classified the de novo assembly in two stages as follows: *a*) Hadoop-based de Bruijn graph construction and *b*) Giraph-based graph simplification. Following is a brief overview of our assembler.

1) Hadoop-based De Bruijn graph-construction (data- and compute-intensive workload): The map function scans through each line of the data file (written in fastq format) and filters out the lines containing only A, T, G, and C, i.e. , the nucleotide characters. These lines are called short reads which represent a small fragment of the entire genome. Then the same map function divides each read into several short fragments of length k, known as k-mers. Two adjacent 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 a key) and, finally, writes the graph structure in the HDFS in an 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): 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 and then the bubble structure (introduced by sequencing errors) in the graph. Giraph master-vertex class invokes different types of computation based on a superstep counter. The computation for compression 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 randomly labeled with either $head$ or $tail$ and send that label to its immediate predecessor. In the next superstep, all the $head$-$tail$ links are merged, that is, the $head$-$k$mer is 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 append. Tip and Bubble removal both takes two supersteps. Again, the first superstep identifies the potential tip or bubble structures based on the number of incoming and outgoing edges of each vertex and send a message with the vertex information to their predecessors. Each predecessor in the next superstep analyzes the message(s) to determine the length of its successor(s), retrieve their frequency, compute the commonality between them using edit-distance algorithm (only in case of bubble) and finally delete the erroneous successor based upon predefined criteria.

	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 (1.5-billion vertices)	3.8GB (3-million vertices)	15	-	4.1TB

Table IV: Large-size human genome assembly

C. Input Data

In this paper, we use two real genome data sets produced by Illumina Genome AnalyzerII, a high throughput next generation sequencing machine. The data sets are as follows: 1) a moderate size bumble bee genome (90GB) and 2) a large size human genome (452GB). The corresponding graph sizes are 95GB and 3.2TB (using $k = 31$ in both the cases). The bumble bee genome is available in GAGE website[1] (Genome Assembly Gold-standard Evaluation [22]). The Human genome is available in NCBI website with accession number SRX016231[2]. Table-III and IV show the details of the data size in the assembly pipeline for both the genomes.

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

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

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

study and use the Cloudera-Manager-5.0.0 for monitoring the system behavior. In this section we provide our evaluation methodology in details. It is worthy to mention here, although we use our benchmark genome assembler (PGA) for the evaluation purpose, our systematic analysis can be easily applied to other data-intensive applications without any modification. A brief description of the Hadoop-parameters that we configured are as follows.

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

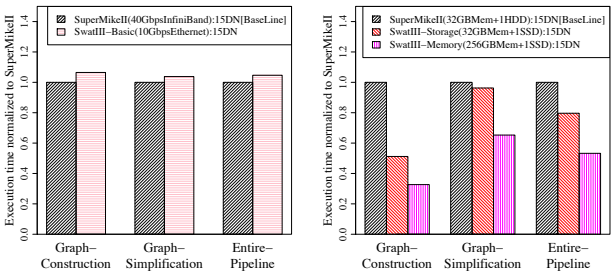
Amount of memory per container and Java-heap-space: In each node in any node of any cluster, we kept 10% of the memory for the system use. The rest of the memory is equally divided among the launched containers. The Java heap space per worker is always set to lower than memory per container as per normal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

Hadoop buffer size: For 1-HDD/DN cases we optimize the Hadoop performance by increasing the buffer size to 500MB, thus reducing data spilling to disk. However, given enough I/O throughput Hadoop performance was found to be insensitive to this parameter.

Total number of Reducers: By analyzing several job profiles we observed that 2-times of reducers than number of concurrent containers produce good performance in general.

Giraph workers: The number of Giraph workers is set according to the number of concurrent YARN containers.

V. Impact of Different Hardware Component



(a) Effect of network (InfiniBand vs Ethernet).

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

Figure 1: Impact of each individual hardware component on execution time of the assembly pipeline in 15-DN

In this section, we compare the individual impact of each hardware component, 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). The execution-times reported in this section are the means of at least 3 runs of the assembler on each different hardware configuration.

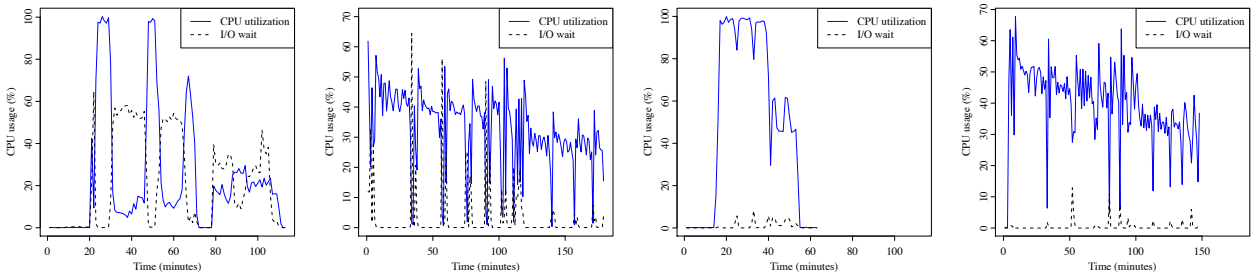
A. Effect of Network: InfiniBand vs Ethernet

Figure-1a compares the impact of network interconnect on PGA 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 and 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 SuperMikeII nodes is 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

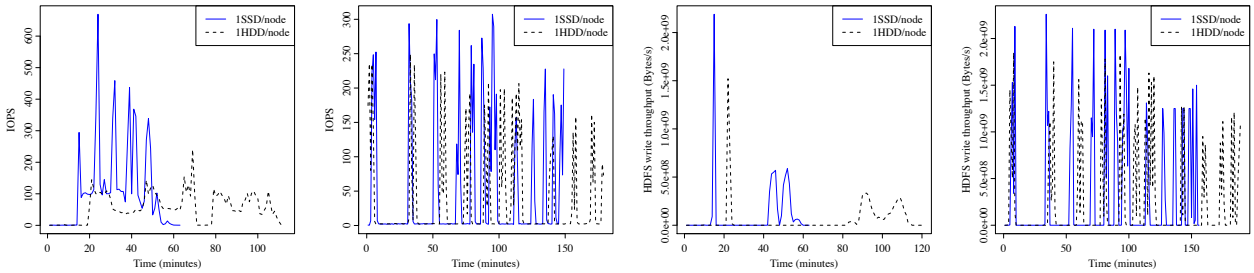
Figure-1b compares the execution time of SwatIII-Basic-SSD to the SuperMikeII-baseline. The second column of each stage of the assembler in Figure-1b shows the impact of using SSD in that stage of the assembly. We observed almost 50% improvement in the shuffle intensive graph-construction stage because 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 using SSD.

Actually, the shuffle phase of Hadoop experiences maximum I/O wait when a many I/O threads work concurrently to spill and subsequently read huge data to/from the disk. This I/O wait is significantly reduced using SSD (Figure-2c and -2d) resulting in remarkable performance improvement in Hadoop. However, for Giraph we did not observe any notable 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 improving the shuffle phase's CPU utilization as shown in Figure-3a. In case of Giraph, the corresponding improvement is 1.5-times as shown in Figure-3b. 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-3c and -3d. 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, for Hadoop, these charac-



(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)

Figure 2: CPU-Utilization and I/O Wait characteristics in SwatIII-Basic-HDD and SwatIII-Basic-SSD



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

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

teristics are different as it writes lots of data to local file system during shuffle phase.

C. *Effect of size of DRAM*

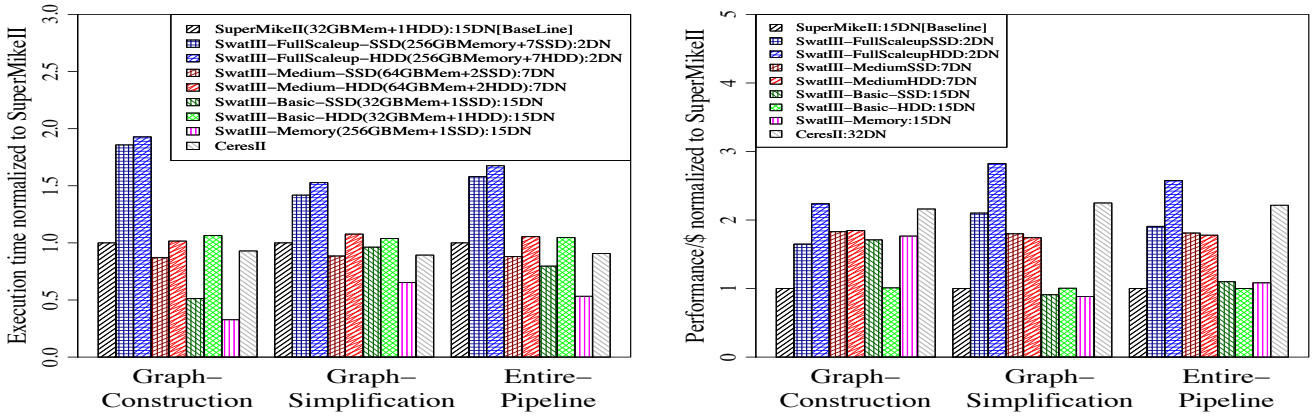
The third columns of Figure-1b shows the impact of increasing the amount of memory per node. We observed almost 20% improvement in the initial graph-construction phase from SwatIII-Basic-SSD, i.e., almost 70% improvement to the baseline. In the Giraph phase, the corresponding improvement is 35%. The improvement is mostly because of the caching. Especially, in case of Giraph, where computation proceeds in iterative supersteps, a huge amount of data is cached and is fetched upon requirement during the next compute-superstep.

VI. Impact of Different Hardware Organization

In this section, we compare different cluster architecture in terms of execution time and performance-to-price. Again, 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-4a 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. The observations are as follows: *1*) **SwatIII-Basic (16-DNs):** As discussed earlier in Section-V, for Hadoop, the SSD variant of this scaled out cluster shows 2x speedup over the baseline whereas the HDD variant performs similar to the baseline. For Giraph, both of them perform similar to the baseline. *2*) **SwatIII-FullScaleup (2-DNs):** This scaled up small sized cluster takes the maximum time for any workload because of least number of processing cores. Observe that for the Hadoop job, both SSD and HDD variants of this scaled up configuration perform similarly, which is in contrast with scaled out SwatIII-Basic. We discuss it in more detail in Section-VI-D. *3*) **SwatIII-Medium (7-DNs):** The HDD variant of this cluster perform similar 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 (16-DNs):** It is no surprise that this configuration shows the



(a) Execution time (Lower execution time means better performance) (b) Performance-to-Price (Higher performance per dollar is better)

Figure 4: Comparison among different type of cluster architectures in terms of execution time and performance-to-price

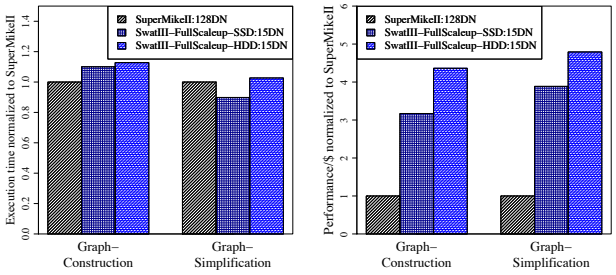
lowest execution time among all because of the maximum resource availability.

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/$. Because of similar total memory and storage sapce across clusters none of them gets price benefit over other in terms of storage and memory. 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-4b 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. *2*) **SwatIII-FullScaleup:** Although this small scaled up cluster takes shows maximum 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.The HDD variant obviously shows better performance/$ than the SSD as their execution time is similar. 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 2-times benefit to the baseline. However, once SSD is used as the underlying storage more memory do not add any advantage in terms of performance/$ (comparing to SwatIII-Basic-SSD). For Giraph, it does not have any impact on performance/$ comparing to the baseline.

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



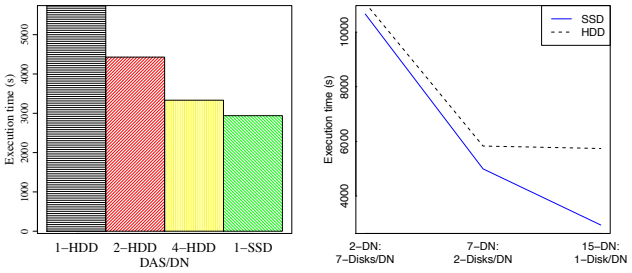
(a) Execution time (Lower execution time is better). (b) Performance/$ (Higher performance/$ is better).

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

The large human genome (452GB) produces huge amount of shuffled data (9.9TB) as well as the graph data (3.2TB). We use 127 DataNodes in SuperMikeII and 15 DataNodes in the SwatIII-Full-Scaleup for this. Figure-5a and 5b shows the execution time and the performance/$ respectively for the human genome assembly pipeline. The observations are as follows: *1*) For Hadoop, the 127-DNs of SuperMikeII (2032-cores) show only 15-17% better performance than 15-DNs (240 cores) of SwatIII-FullScaleup cluster (any variant) while using 8.5-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-III.

Again, observe that for this large data set also, both SwatIII-FullScaleup-HDD and SSD perform similarly. *2*) For the Giraph-based graph-simplification stage also, SuperMikeII did not show the expected performance mainly because of the low network bandwidth resulted by 2:1 blocking. *3*) In terms of performance/$ the scaled up configuration shows huge gain over the baseline. For Hadoop, the gain is 3 to 5 times based upon the storage media. For Giraph, the corresponding gain is almost 4 to 5 times. Again, because of the similar execution time, the scaled up HDD variant shows better performance/$ than the SSD variant.

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



(a) Hadoop performance trend using 1, 2 and 4 HDD(s) and 1-SSD per node using 15 datanodes in the cluster.

(b) Hadoop performance trend for SSD and HDD using 1, 2, 7 disks per node in 15, 7 and 2 datanodes in the cluster.

Figure 6: Performance trend using HDD and SSD in Hadoop. SSD shows better performance and sclability in a scaled out environment

Storage optimized, scaled up cloud instances frequently come with 4 to 8-SSDs per instance to improve the performance, consequently incurring high setup-cost and service-charge. For example, AWS-i2.8xlarge offers 8-SSDs per instance at a rate of $6.82/hour. But, is it the effective way to deploy the SSDs? The disk controllers saturate after a certain I/O throughput (an observation by Szalay [23]). In this section, we compare the performance characteristics of HDD and SSD in both scaled out and scaled up environment.

Figure-6a compares the performance of a single SSD and increasing number of HDDs per node for the Hadoop stage of the bumble bee genome assembly using 15-DNs. The performance improves almost linearly by increasing the number of HDDs per node 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 I/O thoughput the disk controller saturates, and adding more disk(s) to the nodes does not improve the performance. Both Figure-6b as well as Figure-5a substantiate our claim for moderate-size bumble bee and large-size human genome data. For both the data, both the HDD and SSD variant of SwatIII-FullScaleup perform similarly where each DN 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.

E. Performance of CeresII: Samsung MicroBrick with PCIe communication

In this section, we evaluate a Samsung MicroBrick-based novel CeresII architecture. Currently CeresII is in prototype phase. As mentioned before, CeresII uses 2 physical cores, 1-SSD and 16GB memory per compute server and uses a PCIe-based interface to communicate among high-density compute-servers in a chassis. The PCIe based communication improves efficiency of the cluster enormously by reducing the communication overhead. At the same time, the SSD based MicroBricks enable efficient resource utilization in a significantly low cost.

To assemble the 95GB bumble bee genome we use 32 compute-servers of CeresII as Hadoop DNs. The last columns in Figure-4a and Figure-4b show the execution-time and performance/$ respectively for CeresII for different stages of the assembly. CeresII always shows the similar execution time to the baseline while producing more than 2-times improvement in performance/$.

From the performance comparison between SuperMikeII and SwatIII, we noticed a huge trade-off between the execution time and the performance/$. For example, even though the full scaled up small-sized clusters (2-DNs cases) show low performance, they show a magnitude higher performance/$. We also concluded that the medium sized clusters (7-DNs) are well balanced considering both performance and cost. CeresII always shows similar execution time as the medium sized clusters and better performance/$. Moreover, the Samsung MicroBrick consumes less power and space [20], [21]. Hence, we conclude that CeresII shows the maximum benefit in terms of TCO (total cost of ownership).

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 HPC cluster, both, in individual node layer (e.g., memory and storage) as well as network interconnect layer. The novel MicroBrick-based CeresII-cluster with low-power but high-density compute nodes connected with PCIe-based communication interface is a good future direction to alleviate many of the existing architectural limitations.

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 traditional 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.

ACKNOWLEDGMENT

This work has been supported in part by the NSF CC-NIE grant (NSF award #1341008).

REFERENCES

[1] E. Georganas, A. Buluç, J. Chapman, L. Oliker, D. Rokhsar, and K. Yelick, "Parallel de bruijn graph construction and traversal for de novo genome assembly," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*. IEEE, 2014, pp. 437–448.

[2] Y. Li, P. Kamousi, F. Han, S. Yang, X. Yan, and S. Suri, "Memory efficient minimum substring partitioning," in *Proceedings of the VLDB Endowment*, vol. 6, no. 3. VLDB Endowment, 2013, pp. 169–180.

[3] Z. Fadika, M. Govindaraju, R. Canon, and L. Ramakrishnan, "Evaluating hadoop for data-intensive scientific operations," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 67–74.

[4] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, "A tale of two data-intensive paradigms: Applications, abstractions, and architectures," in *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE, 2014, pp. 645–652.

[5] U. C. Satish, P. Kondikoppa, S. Park, M. Patil, and R. Shah, "Mapreduce based parallel suffix tree construction for human genome," in *20th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2014, Hsinchu, Taiwan, December 16-19, 2014*, 2014, pp. 664–670.

[6] P. Kondikoppa, C.-H. Chiu, C. Cui, L. Xue, and S.-J. Park, "Network-aware scheduling of mapreduce framework ondistributed clusters over high speed networks," in *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*. ACM, 2012, pp. 39–44.

[7] J. Vienne, J. Chen, M. Wasi-Ur-Rahman, N. S. Islam, H. Subramoni, and D. K. Panda, "Performance analysis and evaluation of infiniband fdr and 40gige roce on hpc and cloud computing systems," in *High-Performance Interconnects (HOTI), 2012 IEEE 20th Annual Symposium on*. IEEE, 2012, pp. 48–55.

[8] J. Yu, G. Liu, W. Hu, W. Dong, and W. Zhang, "Mechanisms of optimizing mapreduce framework on high performance computer," in *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE, 2013, pp. 708–713.

[9] Y. Kang, Y.-s. Kee, E. L. Miller, and C. Park, "Enabling cost-effective data processing with smart ssd," in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*. IEEE, 2013, pp. 1–12.

[10] D. Wu, W. Luo, W. Xie, X. Ji, J. He, and D. Wu, "Understanding the impacts of solid-state storage on the hadoop performance," in *Advanced Cloud and Big Data (CBD), 2013 International Conference on*. IEEE, 2013, pp. 125–130.

[11] S. Moon, J. Lee, and Y. S. Kee, "Introducing ssds to the hadoop mapreduce framework," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 272–279.

[12] B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "A platform for scalable one-pass analytics using mapreduce," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 985–996.

[13] K. Krish, A. Khasymski, G. Wang, A. R. Butt, and G. Makkar, "On the use of shared storage in shared-nothing environments," in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 313–318.

[14] W. Tan, L. Fong, and Y. Liu, "Effectiveness assessment of solid-state drive used in big data services," in *Web Services (ICWS), 2014 IEEE International Conference on*. IEEE, 2014, pp. 393–400.

[15] S. Huang, J. Huang, Y. Liu, L. Yi, and J. Dai, "Hibench: A representative and comprehensive hadoop benchmark suite," in *Proc. ICDE Workshops*, 2010.

[16] M. Michael, J. E. Moreira, D. Shiloach, and R. W. Wisniewski, "Scale-up x scale-out: A case study using nutch/lucene," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–8.

[17] R. Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron, "Scale-up vs scale-out for hadoop: Time to rethink?" in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 20.

[18] S. Krishnan, M. Tatineni, and C. Baru, "myhadoop-hadoop-on-demand on traditional hpc resources," *San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego*, 2011.

[19] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[20] J. Min, J. Min, K. La, K. Roh, and J. Kim, "Microbrick: A flexible storage building block for cloud storage systems," in *Poster Session*. USENIX FAST, 2014.

[21] J. Min, H. Ryu, K. La, and J. Kim, "Abc: dynamic configuration management for microbrick-based cloud computing systems," in *Proceedings of the Posters & Demos Session*. ACM, 2014, pp. 25–26.

[22] S. L. Salzberg, A. M. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T. J. Treangen, M. C. Schatz, A. L. Delcher, M. Roberts *et al.*, "Gage: A critical evaluation of genome assemblies and assembly algorithms," *Genome research*, vol. 22, no. 3, pp. 557–567, 2012.

[23] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White, "Low-power amdahl-balanced blades for data intensive computing," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 71–75, 2010.