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 in different fields are increasingly using the current state-of-the-art big-data-analytics softwares (e.g. Hadoop, Giraph etc) for their data-intensive HPC problems. However, understanding and designing the hardware environment that these data- and compute-intensive applications require for good performance remains challenging. With this motivation, we evaluated the performance of these bigdata-softwares over three fundamentally different distributed-cyber-infrastructures, including a traditional HPC-cluster called SuperMikeII, a regular datacenter architecture called SwatIII, and a novel microbrick based architecture 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 very good real-world example of a data- as well as compute-intensive workload.**

In this work, we address the impact of both individual hardware components as well as their overall organization (scaled-up and scaled-out) by modifying the SwatIII cluster in many different ways. Comparing with 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-IO 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 yielding almost 2-times improvement in performance per dollar in the entire benchmark test.

I. Introduction

Scientists in different fields are increasingly handling huge amount of bigdata produced by different experimental facilites. Starting from the astronomical data analysis to the coastal simulation, from the social data analysis to the genome assembly, the huge volume of data poses several new challneges, such as efficient data storage, and optimal data processing, to scientific community. 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 like MPI, etc., many HPC applications have started using the current state of the art big data analytics software, such as Hadoop, Giraph etc. for their data-intensive scientific workloads.

However, the traditional supercomputers, even with tera to peta FLOP scale processing power, are found to yield lower performance than expected, especially because of the io- and memory-bound nature of the data-intensive applications. As a result, building efficient and cost-effective hardwares 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 (eg. 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 amounts of big-data. Also, millions of dollars are being spent in programs such as XSEDE 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 very limited understanding of how the different types of hardware-architectures impact the performance of these big-data analytics softwares when applied to a real world data and compute-intensive scientific workloads. Thus, we found it extremely important to evaluate different types of distributed cyber infrastructure in the context of a 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 [1] [2]. De novo genome assembly reconstructs the entire genome from fragmented parts called short reads when no reference

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

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 40Gbps Infiniband, 2) a regular data center architecture, called SwatIII (located at Samsung, Korea) that has 128 nodes connected with 10Gbps Ethernet and 3) a new microbrick based prototype architecture, called CeresII that uses PCIe based communication (also located at Samsung, Korea).

Our performance analysis is divided into two parts as follows:

1) In the first part, 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-io rate, thus reducing the io-wait. Whereas, more memory increases the caching effect.
2) Then, in the second part we provide significant insight on efficient and cost-effective organization of different hardware components. In this part we modified 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-II describes the prior works related to our study. In Section-III we define the motivation of our study, that is, the limitations that we observed in a traditional supercomputing environment to optimally process the bigdata worloads with respect to the current state-of-the-art big-data-analytics software, in particular, Hadoop and Giraph. Section-IV 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-V, we present our performance result by comparing 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-organization and architectural-balance. Finally, in Section-VII we concluded our study.

II. Related Work

Earlier studies [3] [4], as well as our experience shows that state-of-the-art big-data analytics softwares (e.g. Hadoop, etc.) can be useful for HPC workloads involving huge amounts of bigdata. Jha [4] nicely showed the convergence between the two paradigms: the traditional HPC-softwares and the Apache Software Stack for big-data analytics. 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. [5]. Many of the traditional supercomputers also started using myHadoop [5] to provide the scientists an easy interface to configure Hadoop on-demand. Despite the growing popularity of using Hadoop and other softwares in its rich ecosystem for scientific-computing, there are very limited prior works that evaluated different distributed cyber infrastructures for these softwares 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 hardwares to accelerate the Hadoop job using the existing benchmark workloads. Vienne [6] 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 [7] found improved performance of Hadoop in traditional supercomputers due to high speed networks.

Kang [8] compared the execution time of sort, join, WordCount, Bayesian, and DFSIO workloads using SSD and HDD and obtained better performance using SSD. Wu [9] 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-performance is almost insensitive to different Hadoop performance parameter like block-size and buffer-size. Moon [10] showed a significant cost benefit by storing the intermediate Hadoop data in SSD, leaving HDDs to store Hadoop Distributed File System (HDFS [11]) data. They also used the terasort benchmark in their study. A similar result can be found in the study by Li [12] and Krish [13] 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 [10]. Tan [14] 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 like HiBench [15] or with enterprise-level analytics workloads, thus, they are unable to address the HPC aspect of Hadoop. Furthermore, very limited studies consider the in-memory graph processing frameworks like Giraph eventhough, graph analysis is a core part of many analytics workloads.

Impact of overall archtecticture 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 other hand, Appuswamy [17] 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 [17] can be accomodated 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 entire workflow of a genome assembly workload instead of chosing a single job, thus, working closer to the real world.

III. Motivation: Big-data-softwares on traditional supercomputers

In this section, we briefly describe the programming model of two popular big-data analytics softwares: Hadoop and Giraph followed by their general performance characteristics and the issues that we observed on a traditional supercomputing environment.

A. Programming models for big-data analytics

Hadoop and Giraph were originated as the opensource counterpart of Google's MapReduce [18] and Pregel [19] respectively. Both the softwares 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 frawork enables data- and compute-intensive applications to run large voume of distributed datasets over distributed compute nodes with local storage. On the other hand, Giraph uses the Bulk Synchronous Parallel model [20] 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 daa into distributed memories over different compute nodes.

B. Issues: big-data-workload over traditional-supercomputers

"Traditional supercomputers focused on performing calculations at blazing speeds have fallen behind when it comes to sifting through huge amounts of Big Data."[2]
In this section, we discuss the issues and limitations that we observed while running Hadoop and Giraph workload over traditional supercomputing resources.

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

1) Network: 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: 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 io-operations per second (IOPS). On the other hand, the bigdata workloads that consider data locality, typically involve a huge volume of data read/write from/to the directly attached storage (DAS) of the compute nodes. Therefore, the job might suffer from io-bottleneck. Although some variations of Hadoop are optimized to read/write large volume of data from/to other parallel file systems (e.g. Lustre or GPFS), thus taking advantage of huge amount of disk IOPS availale through the dedicated io-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-io in the entire job flow. For example, at the beginning (and the end) of a Hadoop job, all the mappers read from the HDFS (and the reducers write) a huge volume of data parallely to the HDFS which is mounted on the Directly Attached Storage (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 the data from/to the HDFS only during the initial input and the final output.

3) Memory: The traditional supercomputers, normally uses a 2GB/core memory as a standard configuration. This causes a significant tradeoff 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, there by making the job severely io-bound especially in case of HDD.

	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-Scaleup	Scaled-up	Medium-sized	Scaled-out-in-box
#Processor/workstation	2	2	2	2	2	2	1
#Physical-Cores/workstation	16	16	16	16	16	16	2
DRAM(GB)/workstation	32	32	32	256	256	64	16
#Disks(500GB each)/workstation	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/workstation ($)	3804	3829	4300	6526	SSD:9226, HDD:6545	SSD:5068, HDD:4302	879
#DataNodes (DN) used for bubmble 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: Different hardware component used in different cluster and their cost

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 the memory for iterative computation.

IV. Evaluation Methodology

A. Experimental Testbeds

Table-I shows the overview 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 to a single user. SwatIII represents a regular datacenter with 128 compute nodes (running on Ubuntu 12.0.4 LTS). In this study, we use maximum 16 nodes of SwatIII. We configure SwatIII in seven different ways to study the pros and cons of different hardware components and their architectural organization (scaled-up and Scaled-out). For the sake of convenience, we call each configurations with meaningful names as shown in Table-I. CeresII, Samsung MicroBrick-based architecture (now in prototype phase) is developed to address the problems in the existing HPC-cluster and the regular data center. In our study we evaluated it as a next generation cluster and the future direction of our study. It is to be noted, we always use homogeneous configuration for any type of cluster.

Table-II shows the hardware specification used in different clusters and their cost[3]. We calculated the cost of each workstation 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 SwatIII and CeresII, to this baseline. Each node of SuperMiekII 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. SwatIII uses DDR3 ECC REG 16GB memory modules, WesternDigital Velociraptor HDD, Samsung 840 Pro series SSD in different variants. 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 per node. 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 workstation, SwatIII-FullScaleup-HDD/SSD and SwatIII-Medium-HDD/SSD use more than one DAS (directly attached storage) device (Either HDD or SSD as the names suggest) per workstation. They also vary in terms of total amount of memory per workstation. However, the total amount of storage and memory space is almost same accross all these clusters. We use these clusters to mainly evaluate different types of hardware-organization and architectural-balance 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 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

[3]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.

the genral recommendation by [21], Cloudera, Yahoo, etc. Use of the JBOD configuration eliminates the limitation on disk-io 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 [21], 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 that is an improvement over CeresI [22]. Currently, it is in prototype phase. The architecture of CeresII is based on Samsung-MicroBricks, which dissipates extremely low power. A single MicroBricks chassis consists of 22 computation- and storage-modules. Each module consists of one intel Xeon E3-1220L V2 processor with two physical cores, 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 CeresII-modules in a single chassis are connected to a common PCIe switch to communicate with each other. The high density of compute-modules per chassis in CeresII yields 44 physical cores connected through PCIe comparing to 16 physical cores per node as in SuperMikeII and SwatIII, thus resulting in better performance. Furthermore, the use of SSD reduces the io-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 [23]. We classified the de novo assembly in two different 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): This stage consists of two different Hadoop job. The first one is a mapper-only Hadoop jobs that filters the actual short reads (i.e. the lines containing only neucleotide characters A,T,G, and C) from a standard fastq format file. The second map-reduce job that constructs the de Bruijn graph from the filtered reads is extremely compute as well as shuffle-intensive. In the map phase, each read is divided into several short fragments of length k known as k-mers. Two subsequent k-mers are emitted as intermediate key-value pair that represents a vetrtex 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): This stage consists of a series of Giraph jobs. The large scale graph data structure produced by the last map-reduce stage is analyzed in this stage, making the computation extremely memory-intensive. The 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. The subsequent jobs in the workflow do the similar computation incrementally on the previous job's output.

	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 (787619 vertices)	15	-	966GB

TABLE III: Moderate-size Bumble-bee genome assembly

In order to compress the linear chains into a single vertex, we use a randomized parallel algorithm [24]. 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 meassage 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 merge to that vertex. After the compression, all the tip-structures in the graph are removed in two supersteps. A tip refers to a vertex with very short length and is disconnected on one end. The first superstep identifies all the vertices with no outgoing edge and length is less than $2k$ as tips. The second superstep is used to delete those vertices. After the tips, all the bubble-structures in the graph 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 $5k$) 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 veertices are found similar enough, then the lower frequency vertex is removed.

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 another magnitude higher than the reads itself making the entire assembly pipe line severely data-intensive.

In this paper, we use two genome datasets: 1) a moderate size bumble bee genome data (90GB) and 2) a large scale human genome data (452GB). The corresponding graph sizes are 95GB and 3.2TB based upon the number of unique k-mers

	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 (2077438 vertices)	15	-	4.1TB

TABLE IV: Large-size Human genome assembly

(using $k = 31$ in both the cases) in the data set. The bumble bee genome is available in Genome Assembly Gold-standard Evaluation (GAGE [25]) website[4] in fastq format. The Human genome is available in NCBI website with accession number SRX016231[5]. 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 hardwares and the balance among different hardware components, 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. In order 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: We performed rigorous testing on all the clusters by launching different number of containers concurrently and reported the most optimized result.

Amount of memory per container and Java-heap-space: In each node in any cluster, we kept 10% of the memory available per node 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 this as per normal recommendation.

Total number of Reducers: We observed the job profile of our workload many times over several sizes of data and fixed the total number of reducers as double the total number of concurrently launched containers across all clusters, which was found to yield good performance.

Giraph workers: We changed the numbers of Giraph workers according to the number of Yarn-containers launched simultaneously. Each Yarn-container corresponds to a Giraph-worker. Memory per Giraph-worker is fixed similarly to the yarn containers.

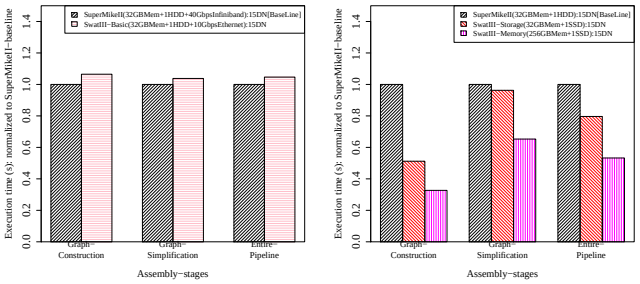
Other Giraph parameters: We always used enough memory to accomodate the graph structure in memory and always avoided using the out-of-core execution feature (and the checkpointing) of Giraph, which writes huge data to the disk.

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

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

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 both in 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 the SwatIII cluster incrementally in terms of storage by providing SSD (named as SwatIII-Basic-SSD) and then scaling up in terms of memory (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 io-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

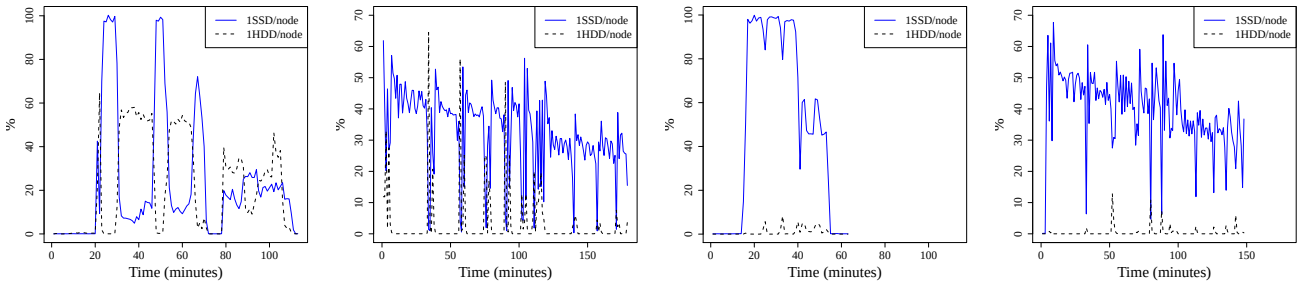
Fig. 1: Impact each individual hardware component on execution time of different stages of the assembly pipeline in 15 DataNodes

A. Effect of Network: Infiniband vs Ethernet

Figure-1a 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 although, 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-bandwith 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-1b 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



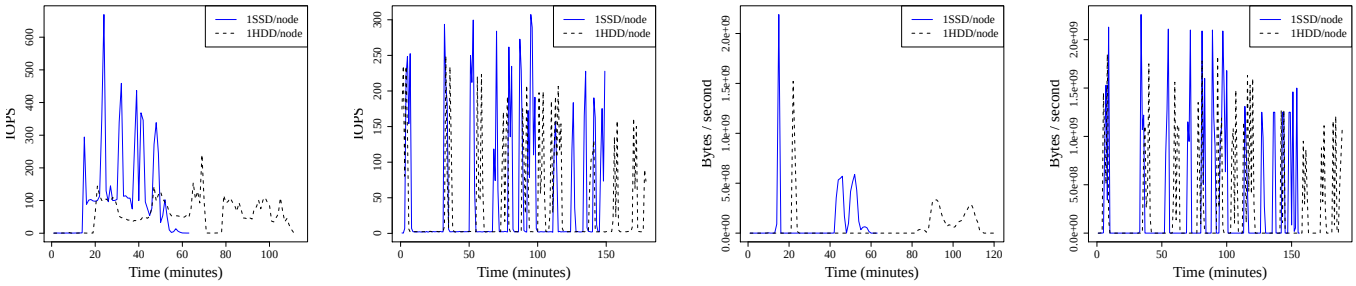
(a) Hadoop-based graph-construction in SwatIII-Basic-HDD (1HDD/node)

(b) Giraph-based graph-simplification in SwatIII-Basic-HDD (1HDD/node)

(c) Hadoop-based graph-construction in SwatIII-Basic-SSD (1SSD/node)

(d) Giraph-based graph-simplification in SwatIII-Basic-SSD (1SSD/node)

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



(a) Write-Iops in Graph Construction in SwatIII-Basic-HDD (1HDD/node) and SwatIII-Basic-SSD (1SSD/node)

(b) Write-Iops in Graph Simplification in SwatIII-Basic-HDD (1HDD/node) and SwatIII-Basic-SSD (1SSD/node)

(c) HDFS-Write-throughput in Graph Construction in SwatIII-Basic-HDD (1HDD/node) and SwatIII-Basic-SSD (1SSD/node)

(d) HDFS-Write-throughput in Graph Simplification in SwatIII-Basic-HDD (1HDD/node) and SwatIII-Basic-SSD (1SSD/node)

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

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 io-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.

Figure-2 compares the CPU-utilization and IO-wait characteristics for 1-HDD and 1-SSD per node. The shuffle phase of the Hadoop job experiences the maximum io-wait, when a huge number of io-threads work concurrently on huge data-spilling (write to disk) and merging in the mapper side and subsequently huge data-copying (read from disk) to the reducer side. A Giraph job becomes io-bound when it reads/writes a large graph from/to HDFS as shown in Figure-2b. As shown in Figure-2c and 2d io-wait is significantly reduced using SSD 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 io-wait.

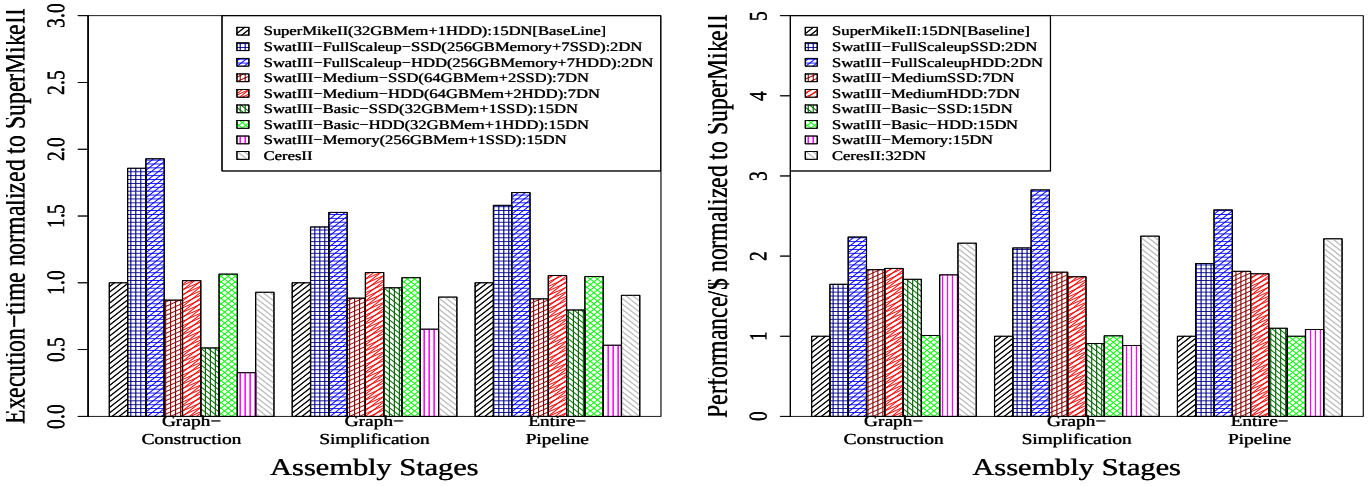
Basically, an SSD increases the disk IOPS by 7 to 8 times than an HDD especially in case of the shuffle phase of Hadoop which writes huge amount of data to the local file system as shown in Figure-3a. In case of Giraph, which writes data only to HDFS, the corresponding improvement is 2-times as shown in Figure-3b. Considering the write throughput to HDFS, we observed 2-times improvement in case of SSD for both Hadoop and Giraph as shown in Figure-3c and 3d .

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 and 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-Organization

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 atleast 3 runs of the assembler on each different hardware configuration.



(a) Execution-time (Lower is better)

(b) Performance/$ (Higher is better)

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

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 entirity 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 follows: *1*) **SwatIII-Basic:** As discussed earlier in Section-V, SSD variant of this scaled-out configuration (32GB-RAM & 1-disks/DN & 16DN) shows 2x speedup over the baseline and the HDD variant performs similar to the baseline. For giraph, both the variants perfom similarly. *2*) **SwatIII-Full-Scaleup:** This scaled-up (256GB-RAM & 7-disks/DN) small-sized cluster (only 2-DN) takes the maximum time. For Hadoop, both the SSD and HDD variant shows $1/2$ speedup over the baseline. Observe its contrast with the the scaled-out SwatIII-Basic. Here, Hadoop performs similar in both SSD and HDD. *3*) **SwatIII-Medium:** Both the HDD and SSD variant of this cluster perform almost similarly to the baseline. However, the SSD variant shows slightly better result. *4*) **SwatIII-Memory:** This is also discussed in Section-V. It is no surprise that this configuration (256GB-RAM & 1SSD/DN & 16DN) takes the lowest lowest execution-time among all the configuration because of the huge amount of memory availabile across the cluster.

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

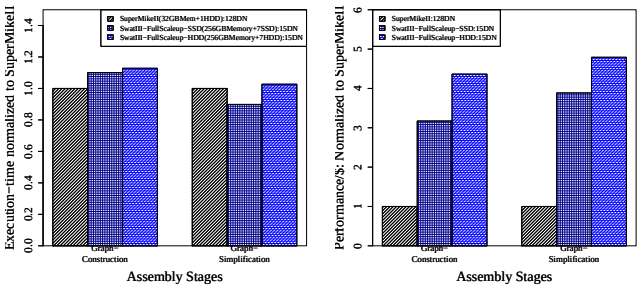
We consider the performance as the inverse of the execution-time and divided it to the total cost of the cluster to get the performance/$. Since all the cluster has same amount of storage and memory space (except SwatIII-Memory), none of the clusters get any price benefit over the other because of the 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-4b compares the performance/$ metric among all the clusters. The observations are as follows: *1*) **SwatIII-Basic:** The SSD variant shows 2-times better performance/$ for Hadoop. However, for Giraph, it performs almost similar to the baseline. The HDD variant, as expected shows similar result as the basline as expected because there similarity in the configuration. *2*) **SwatIII-FullScaleup:** Although it takes the longest execution-time, it shows very good performance/$ for both Haoop and Giraph. For Hadoop, the SSD and HDD shows 1.5 and 2.5 times benefit to the baseline. For Giraph, the benefit is 2 and 3 times respectively for SSD and HDD. Observe its co-trast with the scaled-out (SwatIII-Basic) case. Here, the HDD variant shows better result than SSD for Hadoop. *3*) **SwatIII-Medium:** Both the HDD and SSD variant of this configuration shows similar result, almost 2-times better than the baseline for any workload. Considering both performance and the price, it is the most optimal configuration. *4*) **SwatIII-Memory**: For Hadoop, it shows almost 2-times benefit over the baseline. However, once SSD is used (comparing to SwatIII-Basic-SSD) more memory did not add any advantage in performance/$. For Giraph, although it improves the execution-time significantly over the baseline, there is hardly any impact on performance/$

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 (256GB-RAM + 7disks/node) and 15 datanodes in the SwatIII-Full-Scaleup-(256GB-RAM + 7disks/node) HDD and SSD. Figure-5a and 5b shows the execution time and the performance/$ respectively for the Hadoop and Giraph stage of the human genome assembly pipline. The observations are as follows: *1*) The 127 datanodes of SuperMikeII (2032-cores) show only 15-17% better performance than 15-datanodes of any variant of SwatIII-FullScaleup cluster (240 physical cores) while using almost 9-times more cores in the Hadoop-based graph-construction stage. For this large data also, both the SSD and HDD variants are found to perform similarly as observed in Section-VI-A. Consequently, SwatIII-FullScaleup-HDD shows almost 3-times improvement in performance/$ for this data and compute-intensive Hadoop workload. As discussed earlier in Section-III and -V, the reason behind the suboptimal performance in SuperMikeII is two folded: first, the huge io-bottleneck caused by only one HDD per node, and second, the lower network-bandwidth between compute-nodes because of the 2:1 blocking ratio in the switch while the resource is shared among many users. *2*) The Giraph-based graph-simplification stage takes almost similar time both in the SuperMikeII-baseline and the HDD variant of SwatIII-FullScaleup. The SSD variant, on the other hand, shows little bit better result to the baseline in terms of execution time. In terms of performance/$, the corresponding gain is almost 5-times. The impact of the network is more severe in case of Giraph, which transmits huge amount of messages after each superstep. *3*) Observe also that for this large volume of data also, both the HDD and SSD variants of SwatIII-FullScaleup perform almost similarly as observed in section-VI-A.

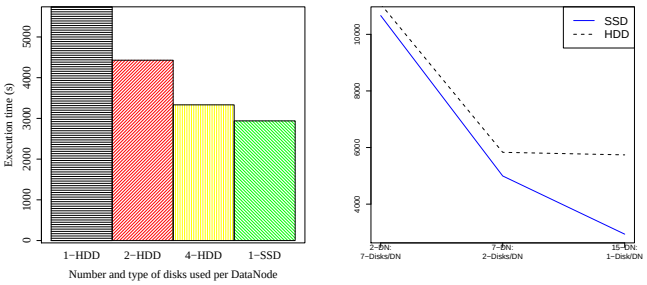


(a) Execution-time (Lower is better) (b) Performance/$ (Higher is better)

Fig. 5: Compare different type of cluster architeture for human genome assembly pipeline

D. Performance of SSD in scaled-up and scaled-out architecture

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[6] 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 the SSDs show tremendous performance gain over the HDDs in a scaled-out scenario, how much effective it is in a scaled-up cluster? In this section, we compare the performance characteristics of HDD and SSD from the perspective of scaled-up and scaled-out configuration.



(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

Figure-6a 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. 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 point the job is CPU-bound, and adding more disk to the datanodes does not improve the performance. Consequently, we did not expect any significant performance improvement after this point. Both Figure-6b as well as Figure-5a 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 HDD and SSD perform similarly in SwatIII-FullScaleup where each datanode is equipped with 7-disks. However, SSD showed significantly better performnce and scalability than HDD when we scaled out by adding more compute nodes to the cluster and reducing the number of disks per node.

E. Performance of CeresII: Samsung-MicroBricks with PCIe communication

In this section, we evaluate a significantly low-power consuming, Samsung-MicroBricks-based novel prototype-architecture, called CeresII which is an improvement over CeresI [22]. As mentioned before, CeresII uses 2 physical cores, 1 SSD and 16GB memory per module (or compute-server) and uses a PCIe-based interface to communicate among high-density compute-servers in a chassis.

In order to assemble the 95GB bumble bee genome we use 32 such modules of this cluster as Hadoop-datanodes. The last columns in Figure-4a show the execution time of CeresII for different stages of the bumble bee genome assembly. Whereas, the last column of Figure-4b show the corresponding performance/$. CeresII shows the similar execution-time to the baseline in every stage of the assembly pipeline while giving almost 2-times improvement in performance/$.

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

From the performance-study between SuperMikeII and SwatIII, we noticed a huge tradeoff between the execution time and the performance/$. For example, eventhough the full-scaledup small-sized clusters (2-DNs cases) show extremely low performance, they show a magnitude higher performance/$. Considering both performance and cost, we can say the medium sized clusters (7-DNs) are well balanced. On the other hand, CeresII shows similar performance both in terms of execution time as well as performance/$ to the medium sized cluster (7DNs). Moreover, the Samsung MicroBricks-based architecture consumes less power, and obviously, occupies less space. Hence, CeresII shows more 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 analytics softwares, 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 organization. Although the network usage and traffic in a public cluster puts a significant bottleneck on the performance of big-data analytics 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 significant the tradeoff between the performance and the price that the data-, as well as memory-intensive HPC applications experience with the existing hardware infrastructures. The existing hardware 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.

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, vol. 11, no. 2007, p. 21, 2007.

[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] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[19] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 135–146.

[20] T. Cheatham, A. Fahmy, D. Stefanescu, and L. Valiant, "Bulk synchronous parallel computinga paradigm for transportable software," in *Tools and Environments for Parallel and Distributed Systems*. Springer, 1996, pp. 61–76.

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

[22] 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.

[23] P. A. Pevzner, H. Tang, and M. S. Waterman, "An eulerian path approach to dna fragment assembly," *Proceedings of the National Academy of Sciences*, vol. 98, no. 17, pp. 9748–9753, 2001.

[24] U. Vishkin, "Randomized speed-ups in parallel computation," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM, 1984, pp. 230–239.

[25] 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.