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
Louisisna State University
Baton Rouge, LA, 70801
Email: {adas7, sjpark} @lsu.edu

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

***Abstract*—The enormous growth in the amount of data that the various experimental-devices generate is rapidly changing the computational model. Recently, data and compute intensive computation frameworks, such as Hadoop and Giraph, have emerged as bigdata analytic softwares. In particular, scientists are increasingly using these softwares to efficiently handle these bigdata, deviating from traditional MPI or grid-based technologies . However, there is limited understanding that how the different types of hardware-architectures impact the performance of these bigdata analytics softwares when applied to a real world data and compute intensive scientific workload.**

In this study, we evaluated the performance of the bigdata analytics softwares over different hardware architectures including HPC clusters (i.e., SuperMikeII) and two different types of private cloud infrastructures (e.g., SwatIII based on regular data center architecture and CeresII based on new microbrick architecture) using our own benchmark software package (i.e., Parallel Genome Assembler (PGA) developed atop Hadoop and Giraph) serving as a very good real world example of data as well as compute intensive workload.

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 the entire compute clusters. In this part, the CeresII prototype-cluster is found to yield same level of performance as in SuperMikeII while yielding more than 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 which make the so called compute intensive scientific applications a severe data intensive endeavor. 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 challneges to the scientific community starting from efficiently storing and managing to optimally processing it. The fundamental model of computation involved in the scientific applications is rapidly changing in order to address these challenges. Deviating from the decade old compute intensive programming paradigm like MPI, Grid etc. many HPC aficionados have started using the current state of the art big data analytics software like Hadoop, Giraph etc. for their data intensive scientific workloads.

Consequently, the traditional supercomputers, even with tera to peta FLOP scale processing power are found to yield suboptimal performance, especially because of the io- and memory-bound nature of the data intensive applications. As a result, providing efficient and cost-effective hardwares became more challenging, however, openning 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 the three communities, including the HPC-Scientists, the hardware-manufacturers as well as the commercial cloud-service-providers to develop cost-effective, high-performance testbeds that will drive the next generation scientific research involving huge amount of bigdata. Also, millions of dollars are being spent in programs like NSFCloud[1] where several academic organizations and manufacturing companies collaborated to address the challenges involved in developing these novel distributed cyber infrastructures.

Despite of this growing interest in both the scientific as well as the industrial community, there is very limited understanding of the performance characteristics of the underlying hardwares that the current state-of-the-art bigdata analytics softwares can obtain when applied for high performance data 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 workload.

In this work, we use the large scale de novo genome assembly as one of the most challenging and complex real world example of high performance computing workload that

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

recently made its way to the forefront of bigdata challenges. De novo genome assembly reconstructs the entire genome from fragmented parts called short reads when no reference genome is available. The assembly pipeline consists of huge amount of short read analysis followed by a complex largescale graph analysis, thus, serving as a very good example of both data- as well as compute-intensive workload.

Specifically, in this paper, we compre the performance of different distributed cyber infrastructure with our own benchmark large scale parallel genome assembler, called PGA, that we developed using Hadoop and Giraph. We present the performance result of PGA atop three different types of clusters as follows. 1) A traditional HPC cluster, called SuperMikeII, located in LSU, USA. 2) A regular data center architecture, called SwatIII, located in Samsung, Korea and 3) A new microbrick based prototype architecture, called CeresII, also located in Samsung, Korea. Our performance analysis is divided into two parts as follows:

1) In the first part, comparing the individual impact of different hardware component over different clusters, we observe almost 70% improvement in the Hadoop-based graph construction stage and 35% improvement in the Giraph based 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 (the regular datacenter architecture) in many different ways to better understand the impact of different architectural balance. In this part, we observe that after a certain threshold on the number of disks per node, SSD and HDD yields similar performance because of similar amount of disk-io rate. Where the threshold is determined by the number of cores. We also observed that the new microbrick based prototype architecture, CeresII is found to yield 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 discuss the programming model offered by Hadoop and Giraph as well as provide a general overview of the expected performance characteristics of traditional supercomputing hardwares. Section-IV-B discusses the overview of our Parallel Genome Assembler followed by details about the input data in section-IV-C. Section-IV describes different types of cluster archetecture and the Hadoop configurations we used for our evaluation purpose. In section-V we compare the impact of different network, storage and memory architecture individually with the CPU-utilization, and IO-patterns in details. Finally, in section-VI we compare different architectural balance in terms of both performance as well as performance/$.

II. Related Work

Earlier studies [1] [2] as well as our experience shows that state-of-the-art bigdata analytics software (e.g. Hadoop etc.) can be useful for HPC workloads involving huge amount of bigdata. Jha [2] in his study nicely showed the convergence between the two paradigms: the traditional HPC-softwares and the Apache Software Stack for bigdata analytics. As a consequence, a growing number of codes in several scientific areas such as bioinformatics, geoscience are currently being written using Hadoop, Giraph etc. [3]. Many of the traditional supercomputers also started using myHadoop [3] to provide the scientists an easy interface to configure Hadoop on-demand. Despite of 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 workload. In this section we provide the related works for our study.

Impact of individual hardware component on Hadoop-workload: There are several performance analysis study on using different types of hardwares to accelerate the Hadoop job using the existing benchmark workloads. Vienne [4] 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 [5] found improvedperformance of Hadoop in traditional supercomputers due to high speed networks.

Kang [6] compared the execution time of sort, join, WordCount, Bayesian, and DFSIO workloads using SSD and HDD and obtained better performance using SSD. Wu [7] found that Hadoop performance can be increased almost linearly with the increasing fraction of SSDs in the storage system. They used the tera-sort benchmark for their study. Additionally, they also showed that in an SSD-dominant cluster Hadoop-performance is almost insensitive of different Hadoop performance parameter like block-size and buffer-size. Moon [8], showed a significant cost benefit by storing the intermediate Hadoop data in SSD, leaving the HDDs to store Hadoop Distributed File System (HDFS [9]) source data. They also used the terasort benchmark in their study. Similar result can be found in the study by Li [10] and Krish [11] where SSDs are used to serve temporary data to reduce disk contention and used HDDs to store the HDFS data. They all reached the same conclusion as [8]. Tan [12] 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 [13] or for enterprise level analytics workloads, thus, unable to address the HPC aspect of Hadoop.

Furthermore, very limited studies consider the in-memory graph processing frameworks like Giraph, although, graph analysis is a core part of many analytics workloads.

Impact of overall archtecticture on Hadoop Workload Michael [14] investigated the performance characteristics of the scaled-out and scaled-up architecture for interactive queries and found better performance using a scaledout cluster. On the other hand, Appuswamy [15] reached 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 mostly focus on enterprise level Hadoop job. Our Hadoop enabled genome assembly workload is significantly different. Additionally, we cover a Giraph workload. 2) 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. 3) Existing works are limited in terms of data size. For example, the data size chosen in [15] can be accomodated in a single scaled-up server. On the contrary, we did not put such a restriction. Consequently, we evaluate the performance of scaled-up cluster with fewer nodes instead of evaluating only one scaled-up server. All of our clusters provide almost same amount of storage and memory. Our analysis 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: Bigdata-softwares on traditional supercomputers

In this section we briefly describe the programming model of two popular bigdata analytics softwares: Hadoop and Giraph followed by their general performance characteristics on a trditional supercomputing environment.

A. Programming model of Hadoop and Giraph

1) Hadoop: Hadoop was originated as the opensource counter part of Google's Map-Reduce [16]. Hadoop has two different components: Hadoop Distributed File System (HDFS) and a mapreduce programming abstarction. HDFS splits huge volume of data into small disjoint sets called blocks (typically of size 64mb to 128mb) and distributes those accross the cluster. A user defined map function is applied to each blocks parallely in order to extract information from each records in the form of key-value pair. These intermidiate key-value pairs are then partitioned on the basis of keys where each key gets a list of values. Finally, a user defined reduce function is applied to the value-list of each key independently and the final output is written to the HDFS.

2) Giraph Apache Giraph is an in-memory grpah processing framework that is implemented on top of Hadoop's mapreduce implementation. It is originated as the open-source counterpart to Google's Pregel [17]. Giraph is inspired by Bulk Synchronous Parallel model [18] where computation proceeds in supersteps. In each superstep all vertices of the graph excutes different istances of the same progam called vetrex-program simultaneosly without ineracting with other verties which is similar to map tasks of Hadoop After each superstep all the vertices send messages to other vertices normally cotaining the output of its vertex-progam-instance. Once all the messages are recieved by the intended vertices, the next supesrtep starts and the pocess iterates until all the vertices vote to halt simultaeously.

B. Hadoop/Giraph-workload over Traditional-Supercomputing hardwares

”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 provide a general overview of the performance characteristics of Hadoop and Giraph on top of traditional supercomputing resources.

1) Network: In a typical Hadoop job, the data movement is minimal early in the job flow when the mappers carfully consider the data locality. Once the mappers completed their tasks, the intermidiate data is shuffled to the reducers which results in a huge data movement across the cluster. On the other hand, Giraph is more network intensive. The computation phase of each Giraph-superstep is followed by a communication phase which sends a huge amount of messages accross all the Giraph workers. Furthermore, in a Giraph job, the number of TCP-connections increases almost exponentially with increase in 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 workflow.

High performance scientific applications running in a supercomputing environment traditionally use an Infiniband interconnect with high performance and low latency. But, Hadoop and Giraph were developed to work atop cheap clusters of commodity hardware based on an Ethernet network. The java based network communication in both Hadoop and Giraph can hardly take the advantage of the Infiniband.

2) Storage: A typical Hadoop job involves a huge amount of disk-io in different phases. First, in the beginning of the map phase, the input data is read from a distributed filesystem parallely by all the mappers. Normally, the HDFS is used for this purpose which is mounted on the Directly Attached Storage (DAS) device(s) in each of the compute nodes. Then, during the shuffle phase, a huge amount of data (intermidiate key-value pair) is written by the mappers and subsequently read by the reducers to/from the local file system which is again mounted on the Directly Attached Storage (DAS) device(s) of each compute node. Finally, at the end of the job, the reducers write the final output on the underlying parallel distributed file system. Giaph, on the other hand, is an in-memory framework. It reads/writes the data from the disk only twice. First, in the beginning of the job when it reads the graph data structure from the dfs. And finally, after the completion of the entire computation it writes the final output to the HDFS.

In a traditional supercomputing environment, each node is normally attached with only one HDD. This configuration puts a practical limitation in terms of total number of disk-io operations per second (IOPS). Some variations of Hadoop (eg. MyHadoop etc.) are capable to read/write the data from/to other parallel file system like Lustre or GPFS which are mounted on dadicated io-servers in an HPC environment. Although these versions of Hadoop can be well optimized to take the advantage of huge amount of disk-IOPS availale through the dadicated io-servers (that are used to mount the Luster or GPFS), the performance can be severely constrained by the available network bandwidth which is shared among many users, consequently generating huge network-traffic. Furthermore, distributing the Shuffled data across dadicated io-servers needs complicated partitioning on the parallel file system which is hardly available in a traditional supercoputing environment. In this paper, we consider only HDFS to evaluate the underlying hardwares.

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

3) Memory: The performance of a Hadoop job can be improved by providing more memory per node in the compute-cluster. At the end of the map-phase, each map task spills a huge amount of data onto the DAS. Providing more memory with properly tuned buffer-size (io.sort.mb and io.sort.factor) reduce the amount of spilling to the disk, thus, improve the performance of a Hadoop job significantly especially in case of HDD where disk-io creates huge performance-bottleneck. Furthermore, increasing memory in each node improves the caching effect during the computation which is extremely beneficial for iterative computation in Giraph. Also, more the memory modules per node, more is the number of memory channels, thus increasing the access parallelism of the processors in each node which can also improve the overall performance.

In a traditional Supercomputing environment, normally, 2GB per core is used as a standard configuration which obviously poses a tradeoff between the number of concurrently running mappers and the amount of buffers used by each of them before spilling the output into the disk. Furthermore, for a memory-intensive job, like graph analysis with Giraph that loads a huge amount of data in the memory for iterative computation, the low amount of memory per node hinders the caching. Hence, results in lower performance.

IV. Evaluation Methodology

A. Experimental Testbeds

Table-I shows the overview of the experimental testbeds that we use in our study. We use the configuration of LSU supercomputing resource, SuperMikeII as the baseline and compare all the performance result of our private cloud infrastructures, SwatIII and CeresII to this baseline. Each node in any of the SwatIII variant has the same number of processors and cores as in SuperMikeII. In particular, each SuperMikeII- and SwatIII-node uses two Intel SandyBridge Xeon 64bit Ep series processor. Each of the processor has 8-cores, thus yielding 16 physical cores per node. The first three variant of SwatIII: SwatIII-Basic, SwatIII-Storage and SwatIII-Memory is used to evaluate the impact of each individual component of a compute cluster, i.e. network-interconnect, storage type and amount of memory per node. SwatIII-basic is similar in every aspect of SuperMikeII except it uses 10-Gbps Ethernet instead of 40-Gbps Infiniband as in SuperMikeII. SwatIII-Storage, as the name suggests, is storage-optimized and use one SSD per node instead of one HDD as in SuperMikeII. 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 SuperMikeII.

Unlike SuperMikeII or SwatIII-Basic/Storage/Memory which use only one DAS device per workstation, SwatIII-FullScaleup-HDD/SSD and SwatIII-Medium-HDD/SSD use more than one DAS device (Either HDD or SSD as the names suggest) per workstation. They also vary in terms of total amount of memory per worstation. 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 in terms of raw execution time as well as performance/$. In either of SwatIII-FullScaleup and SwatIII-Medium, we use JBOD (Just a Bunch Of Disks) configuration as per the genral recommendation by [23], 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 [23], JBOD is found to perform 30% better than RAID-0 in case of HDFS write throughput.

The last one: CeresII is an improvement over CeresI [24] and is in prototype phase. The architecture of CeresII is based on Samsung-MicroBricks. A single MicroBricks chassis consists of 22 computation- and storage-module. 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-module 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 reduce the io-wait and 8GB RAM per physical cores 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 [20]. We classified the de novo assembly in two different stages as follows: *a*) Hadoop based de Bruijn graph construction. *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: This stage consists of two different Hadoop job. The first one is a mapper-only Hadoop job 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-mer. Two subsequent k-mers are emitted as intermediate key-value pair that represents a vetrtex and an edge (emmited 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 emmited 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 input.

	Super MikeII	SwatIII-Basic	SwatIII-Storage	SwatIII-Memory	SwatIII-FullScaleup-HDD/SSD	SwatIII-Medium-HDD/SSD	CeresII
#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
#DataNodes used for bubmble bee genome (90GB)	15	15	15	15	4	2	31
#DataNodes used for human genome (452GB)	128	-	-	-	16	-	-

TABLE I: Experimental Tetbeds

2) Giraph-based Graph Simplification: 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 (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 last job's output.

In order to compress the linear chains into a single vertices we use a randomized parallel algorithm [21]. The computation is 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 immidiate 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 legth 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 low frequency vertex is removed.

C. Input Data

High throughput next generation DNA sequencing machines like Illumina Genome Analyzer produce 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.

	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 II: Bumble-bee genome assembly

	Job Type	Input	Final output	# jobs	Shuffled data	HDFS Data
Graph Construction	Hadoop	452GB (2-billion reads)	3TB	2	9.9TB	3.2TB
Graph Simplification	Series of Giraph jobs	3.2TB (1483246722 vertices)	3.8GB (2077438 vertices)	15	-	4.1TB

TABLE III: Human genome assembly

In this paper, we use two genome dataset, 1) a moderate size bumble bee genome data (90GB) and 2) a large scale human genome data (452GB). The corresponding graph size is 95GB and 3.2TB based upon the number of unique k-mers in the data set The bumble bee genome is available in Genome Assembly Gold-standard Evaluation (GAGE [22]) website[3] in fastq format. The Human genome is available in NCBI website with accession number SRX016231[4]. Table-II and III shows 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. In order to evaluate the relative merits of dif-

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

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

ferent 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 harware infrastructure in SwatIII cluster to optimize the perfromance in each configuration. A brief description of the Hadoop-parameters that we changed are as follows.

Number of concurrent Yarn containers: We performed regorous testing on any of 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 both for the system use. 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 seevral size of data and fixed the total number of reducers as double the total number of concurrently launched containers always across all clusters which was found to yield good performance.

Slow Restart: It is always set to 1 i.e. the reducers will start only when all the mappers are finished. Although performance is found to drop little bit for this, it helps us separating the impact of network as well as observe the io-characteristics of map and reduce phase separately.

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

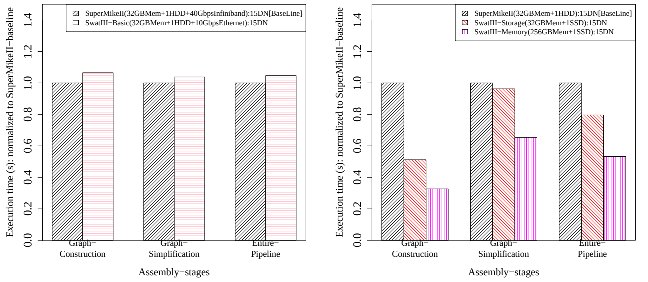
Other Giraph parameters: We always use enough memory to accomodate the graph structure in memory and always avoided using the out-of-core execution feature (and the check-pointing) of Giraph which writed huge data to the disk.

V. Individual impact of different hardware optimization

In this section, we compare the impact of each hardware component: network, storage and memory individually on our benchmark genome assembler. In order 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 network between SuperMikeII and SwatIII-Basic. Then, we further optimized the SwatIII cluster incrementally in terms of storage (named as SwatIII-Storage) and then memory (named as SwatIII-Memory) and compare the performance of each component in each stage of our assembler.

A. *Performance in SuperMikeII*

Since each node of SuperMikeII is equipped with only 1-HDD, there is a practical limitation on number of concurrently running yarn containers (hence, mappers and reducers) per node. More the number of mappers (or reducers) running simultaneously, more is the parallel disk-io especially during the shuffle phase. With only 1-HDD per node, the performance of Hadoop is advarsely affected because of huge amout of io-wait. As a consequence, even if each node of SuperMikeII has 16 processing cores we observed the best performance for our entire assembly pipeline by running only 8 yarn-containers concurrently in each node, i.e. only half of the number of cores per node.



(a) Effect of network (b) Effect of SSD and DRAM

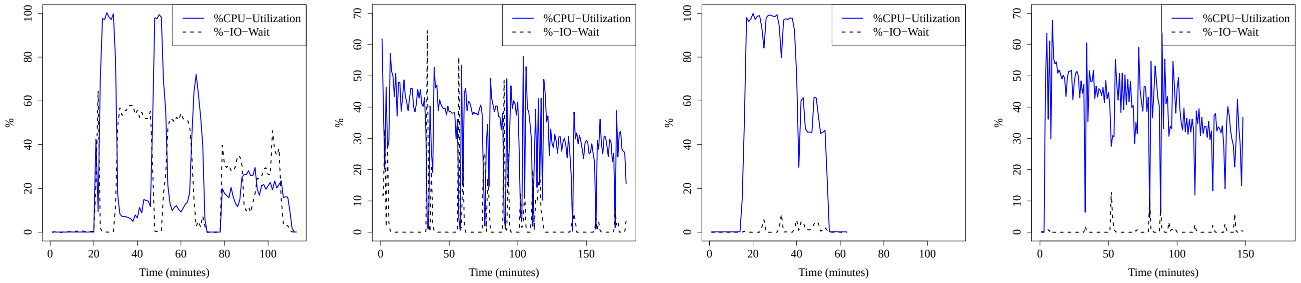
Fig. 1: Impact each individual hardware component on execution time of different stages of the assembly pipeline in 15 DataNodes

B. *Effect of Network*

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. That is, the execution time on SuperMikeII for different stages of the asembler always have the value 1. SuperMikeII uses a 40-Gbps QDR Infiniband whereas SwatIII-basic uses 10-Gbps ethernet. However, we did not find any visible performance difference (less than 2%) on any of the stages of our assembly pipeline due to change in the network. In order to investigate the reson in details, we measure the avaerage latency and the effective network-bandwith between two arbitrary compute nodes in SwatIII and SuperMikeII. Since SuperMikeII uses an Infiniband connection the average latency was found to be more than 10-times lower compare to SwatIII which uses ethernet. The avarage latency in SuperMikeII was found to be .014ms whereas in SwatIII, it is almost 0.2ms. However, due to the resource sharing among many users, the average effective-bandwith of SuperMikeII was found to be almost 10-times lower than that of SwatIII: 949Mbit/s in SuperMikeII, whereas 9.1Gbit/s in SwatIII. Furthermore, the java-based APIs of Hadoop or Giraph is not optimized to take the advantage of Infiniband. Hence, in any of the Hadoop or Giraph workloads in the moderate size bumble bee genome assembly pipeline we did not observe major performance difference using same number of nodes in SwatIII.

C. *Effect of SSD*

Figure-1b compares the execution time of SwatIII-Storage (1-SSD/node) to the SuperMikeII-baseline (1-HDD/node). The second column of each stage of the assembler in Figure-1 shows the impact of using SSD in that stage of the assembly. We observed almost 50% improvement in the shuffle intensive



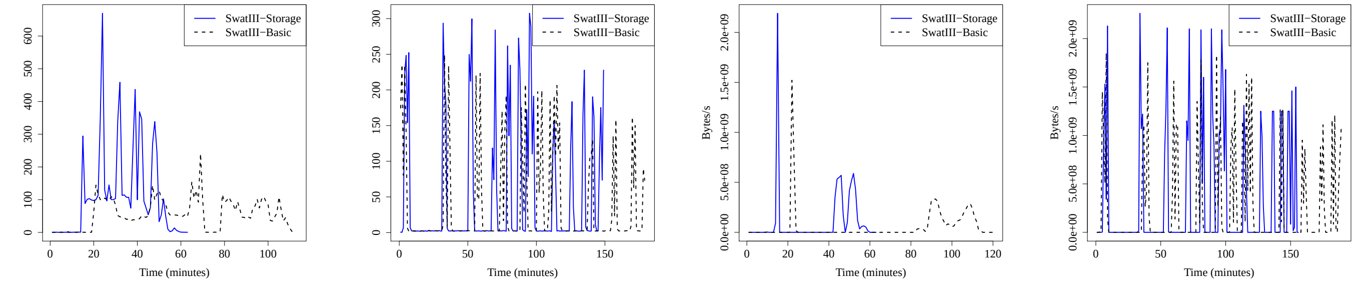
(a) Graph Construction in SwatIII-Basic

(b) Graph Simplification in SwatIII-Basic

(c) Graph Construction in SwatIII-Storage

(d) Graph Simplification in SwatIII-Storage

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



(a) Write-Iops in Graph Construction in SwatIII-Basic and SwatIII-Storage

(b) Write-Iops in Graph Simplification in SwatIII-Basic and SwatIII-Storage

(c) HDFS-Write-Rate in Graph Construction in SwatIII-Basic and SwatIII-Storage

(d) HDFS-Write-Rate in Graph Simplification in SwatIII-Basic and SwatIII-Storage

Fig. 3: Comparison of rate of disk write on Local File System (of one datanode) and HDFS (across all datanodes) SwatIII-Basic (1-HDD/node) and SwatIII-Storage (1-SSD/node)

graph-construction stage because of reduced io-wait. However, graph-simplification, being a series of in-memory Giraph jobs is not affected much (less than 5%) by storage optimization with SSD.

Figure-2 compares the CPU-utilization and IO-wait characteristics for 1-HDD and 1-SSD per node. The performance of a shuffle-intensive Hadoop job becomes io-bound mainly in two places. First, at the end of each mapper-wave when many mappers write intermediate shuffle data parallely to the local file system. Second, when this shuffled data is read by the reducers. 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 reduced by using solid state drive (SSD) instead of HDD.

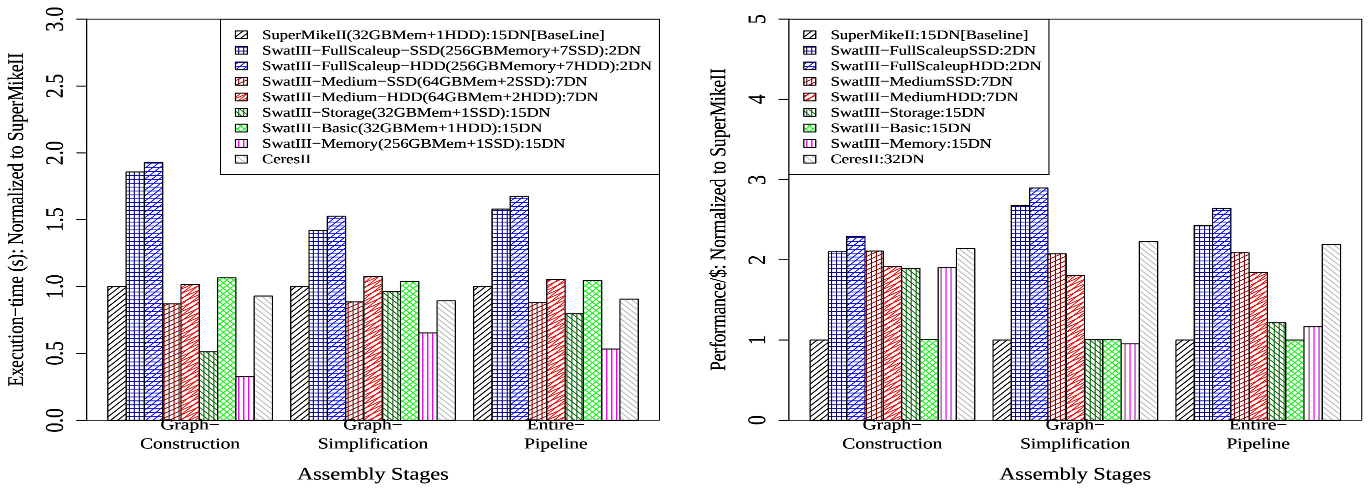
Basically the SSD increases the disk io-rate tremendously especially in case of the local file system that is the shuffle phase of Hadoop. Figure-3a shows that SSD improves the peak IOPS 7 to 8 times in one of the datanodes than corresponding HDD case during the shuffle phase of the Hadoop build-graph job. On the other hand, figure-3b compares the rate of local disk-write of the same datanode for a series of Giraph jobs which read/write only to the HDFS. Whereas, figure-3c and 3d shows the write rate over HDFS across all datanodes. From the later three cases, the SSD is found to yield 2-times improvement in the peak HDFS read/write.

D. *Effect of DRAM*

The third columns of Figure-1b shows the impact of memory in different stages of our assemblers in SwatIII-Memory normalized to the SuperMikeII-baseline. We observed almost 20% improvement in the initial graph-construction phase from SwatIII-Storage and almost 70% improvement to the baseline. In the Giraph phase, the corresponding improvement is almost 35%. The improvement is because of the caching and increase in cpu-memory access parallelism. 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 between Different Architectural-Balance

In this section we compare the performance of different cluster architecture in terms of raw execution time as well as performance per dollar.



(a) Execution-time (Lower is better)

(b) Performance/$ (Higher is better)

Fig. 4: Compare different type of cluster architeture for bumble bee genome assembly pipeline

A. Performance comparison of SuperMikeII and SwatIII with bumble bee genome

Figure-4a shows the relative merits of different cluster architecture in terms of raw execution time. The execution time is normalized to the baseline i.e. SuperMikeII. Observe, we always keep the total aggregated storage and memory space almost same across all the clusters. 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. 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) The SwatIII-Memory, i.e. the 15-DataNode cluster with 256GB memory and one SSD per node performs the best in terms of raw execution time for any type of workload due to high resource availability. 2) Because of the io-bound nature of the Hadoop job, SSD shows better scalabality than HDD in the graph-construction stage with increase in number of nodes in the cluster, thus increasing the number of cores. There is almost no performance improvement in SwatIII-Medium-HDD (7-DN with 2-HDD each) and SwatIII-basic (15-DN with 1-HDD each). However, there is almost 50% improvement in the corresponding SSD case (i.e. SwatIII-medium-SSD and SwatIII-Storage). 3) Number of cores plays a critical role in case of Giraph. We observed the optimum performance in graph simplification stage in SwatIII-Medium (8-nodes) cluster. 4) Although the use of SSD is beneficial for Hadoop when there is only one disk per node (as shown in SuperMikeII and SwatIII-Storage), in a full scaled-up environment with multiple disks per node, Hadoop shows similar performance with both HDD and SSD as shown in SwatIII-FullScaleup-SSD and SwatIII-FullScaleup-HDD.

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

Table-IV shows the cost[5] of each hardware component used in different clusters. As mentioned earlier, the total aggregated storage and memory space is kept almost same across all the clusters except SwatIII-Memory which has a huge amount of total aggregated memory. Which means, in our experiments, none of the cluster gets 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. Since SuperMikeII resources are shared among many users whereas the SwatIII and CeresII are private cluster we did not consider the cost of network for a fair comparison. Figure-4b shows the performance to price comparison among all the clusters. The observations are as follows: 1) For a shuffle intensive Hadoop job, SuperMikeII and SwatIII-basic yields the lowest performance/$ because of huge performance bottleneck caused by the io-wait. 2) Use of more memory per node as in SwatIII-Memory does not show any better result in terms of performance/$ once SSD is used as underlying storage as in SwatIII-Storage for Hadoop. 3) Although the use of huge memory space in SwatIII-Memory shows the maximum performance in case of Giraph, it does not have much impact on performance/$ when compared to low memory cluster (e.g. SwatIII-Storage or SwatIII-Basic) given the fact that, the graph fit in the total memory space. 5) For Giraph, the performance/$ decreases almost linearly with the increase in the number of nodes. Whereas, Hadoop-performance/$ shows very small variation with increasing the number of nodes, given there is no io-bottleneck (as in the SSD variant of SwatIII-FullScaleup-/Medium/Storage). This is because each computation-core analyzes the same amount of data at a certain point of time in case of Hadoop (determined

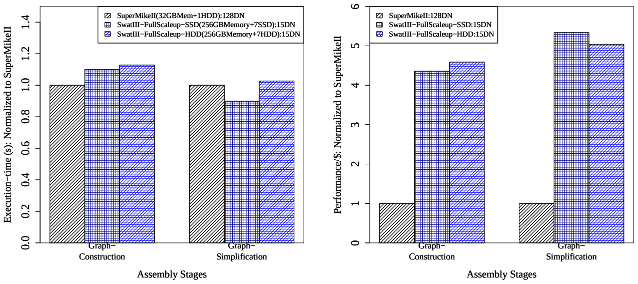
[5] Price information is collected from http://www.newegg.com/ and http://www.amazon.com/

Component	Cost ($)
Intel SandyBridge Xeon 64bit Ep series (8-cores): used in SuperMikeII and Each variant of SwatIII	1520
Intel Xeon E3-1220L V2 (2-cores): used in CeresII	389
1-HDD (Western Digital, 500GB)	67
1-SSD (Samsung, 500GB)	177
Dell Poweredge 16GB meory module	139

TABLE IV: Cost of each hardware component

by the HDFS block size), resulting in full CPU-utilization. Whereas, in a Giraph job, more the number of nodes less is the amount of data handled by each core at a certain point of time, thus improving the performance in the cost of lower CPU utilization.

C. Comparing SuperMikeII and SwatIII with Large human-genome



(a) Execution-time (Lower is better)

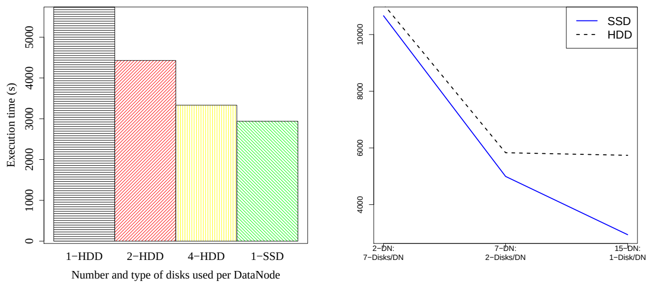
(b) Performance/$ (Higher is better)

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

In order to evaluate the cumulative impact of all the hardware components (i.e. network, cpu-cores, storage and memory) on the entire cluster we summarize our study with a stress testing of SuperMikeII and SwatIII clusters using the large scale 452GB human genome that produces 3.2TB of graph(refer to table-III). In this part we utilize the maximum amount of resources that are available in any of the compute-clusters. We used 127-datanodes of SuperMikeII to accomodate the huge volume of data: either stored on disk (HDFS data or shuffled data) or the huge graph that is loaded in memory. On the other hand, we use 15-datanodes in the SwatIII-Full-Scaleup cluster (both HDD and SSD variant) which yields almost same amount of storage space as well as memory. As mentioned before, due to the low value of maximum allowed file descriptor in SuperMikeII ($ulimit - n$ is set to 1024) we could not run the scaffolding stage in this cluster. Hence, remove it from the comparison.

Figure-5a and 5b shows the execution time and the performance/$ respectively for the Hadoop-based graph-construction and Giraph-based graph-simplification stage of human genome assembly on three different cluster architectures. The observations are as follows: 1) The 128nodes of SuperMikeII (2032-cores) shows only 15-17% better performance than 15-datanodes of any variant of SwatIII-Full-Scaleup cluster (240 physical cores) while using almost 9-times more cores in the Hadoop-based graph-construction stage. The reason behind the suboptimal performance in superMikeII is two folded: first, the huge amount of io-wait resulted by only one HDD per node as discussed in section-V-C. Second, the availability of less effective-network-bandwidth between compute-nodes because of resource sharing among many users as discussed in section-V-B. However, Hadoop depends on network only once in the entire job: during data transfer tot he reducers 2) The Giraph-based graph-simplification stage performs 10% better in SwatIII-FullScaleup-SSD than SuperMikeII. Since Giraph is more network intensive, the higher effective-network-bandwidth between compute-nodes of SwatIII yields better performance although it has very less amount of cores than SuperMikeII. 3) Both HDD and SSD variants of SwatIII-FullScaleup shows almost similar performance for both Hadoop and Giraph (less than 5% variation). The effect is similar as discussed in section-VI-D. 4) In terms of performance/$ any of the SwatIII-FullScaleup cluster is found to yield almost 4.5-times better result for the Hadoop stage and almost 5-times better result for the Giraph stage.

D. Scaledup cluster and SSD



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

Cloud service providers already started using SSDs as the elemental feature in their cloud infrastructure. Furthermore, many of the available cloud instances provide more SSDs per compute-node in order to improve the performance. Consequently, the setup-cost as well as the pricing of these scaledup instances increase. For example AWS i2.8xlarge storage-optimized instance offers 8SSDs per compute-node with 32 v-cores per node in a rate of $6.82 per hour which is one of the high-cost AWS-EC2-instances [6]. In this section, we analyze how to leverage SSDs in a cost-effective manner. In particular, we point out the scaledup scenario where HDDs and SSDs yield similar level of performance.

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

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

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. We did not observe any more significant improvement by providing more SSDs per node, concluding the disk-io rate reaches a saturation point. As a consequence, we did not find any significant performance-difference between SwatIII-FullScaleup-SSD and -HDD as shown in figure-6b while assembling the same bumble bee genome using 2-datanodes each with 7-disks. However, SSD shows significantly better performnce as well as scalability than HDD when we scale out by adding more compute nodes to the cluster (thus, increasing the total number of cores) and reducing the number of disks per node (thus, keeping the total storage space almost same in the cluster).

E. CeresII: Samsung-MicroBricks for shared nothing paradigm

In this section, we evaluate a Samsung-MicroBricks based novel prototype-architecture called CeresII which is an improvement over CeresI [24]. As mentioned before, CeresII uses 2 physical cores, 1 SSD and 16GB memory per computation module. In order to assemble the 95GB bumblebee genome we use 32 such modules of this cluster as Hadoop-datanodes. The last columns of different stages of the assembly in Figure-4a shows the execution time of CeresII. As it can be seen, CeresII performs almost similar in every stage of the assembly pipeline and shows almost 2-times improvement in performance/$.

From the performance-study between SuperMikeII and SwatIII, we noticed a huge tradeoff between the execution time and the performance/$. For example, the full-scaledup small-sized clusters (2-DNs cases), even though shows lower performance than the SuperMikeII-baseline, it shows a magnitude higher Perf/$. Again, 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 is expected to consume 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

Acknowledgment

The authors would like to thank...

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in *eScience, 2008. eScience'08. IEEE Fourth International Conference on*. IEEE, 2008, pp. 222–229.

[20] 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.

[21] 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.

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

[24] 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.