Augmenting Amdahl's Second Law: A Theoretical Model for Cost-Effective Balanced HPC Infrastructure for Data-Driven Science

Journal:	*IEEE Access*
Manuscript ID	Draft
Manuscript Type:	Special Section: Theoretical Foundations for Big Data Applications: Challenges and Opportunities
Date Submitted by the Author:	n/a
Complete List of Authors:	Das, Arghya; Louisiana State University, Computer Science and Engineering; Louisiana State University Center for Computation and Technology, Hong, Jaeki; Samsung Electronics Goswami, Sayan; Louisiana State University, Computer Science and Engineering; Louisiana State University Center for Computation and Technology, Platania, Richard; Louisiana State University, Computer Science and Engineering; Louisiana State University Center for Computation and Technology, Lee, Kisung; Louisiana State University, Computer Science and Engineering; Louisiana State University Center for Computation and Technology, Chang, Wooseok; Samsung Electronics Park, Seung-Jong; Louisiana State University, Division of Computer Science and Electrical Engineering; Louisiana State University Center for Computation and Technology,
Keywords:	Clouds, High performance computing, Data analysis
Subject Category Please select at least two subject categories that best reflect the scope of your manuscript:	Computers and information processing, Biomedical Engineering
Additional Manuscript Keywords:	Big data HPC, Amdah's second law, Cost/performance, Theory for a balanced system, Hadoop and Giraph

Augmenting Amdahl's Second Law: A Theoretical Model for Cost-Effective Balanced HPC Infrastructure for Data-Driven Science

Arghya Kusum Das, Jaeki Hong, Sayan Goswami, Richard Platania,
Kisung Lee, Wooseok Chang, Seung-Jong Park
Division of Computer Science and Electrical Engineering, Center for Computation and Technology, Louisiana State University, Baton Rouge, LA, USA, 70803
Samsung Electronics Co., Ltd., 95, Samsung 2-ro, Giheung-gu, Yongin-si, Gyeonggi-do, S. Korea, 446711

***Abstract*—Recent compute technologies for analyzing enterprise and scientific big data have started demanding more processors, storage, and memory resources, forcing a similar growth in the total computation cost. Consequently, HPC system designers are facing relentless pressure to reduce the system cost while also providing expected performance for data- and compute-intensive applications. With this motivation, this paper proposes a simple, easy to use, additive model to optimize cost/performance by quantifying the system balance among CPU, I/O bandwidth, and size of DRAM in terms of both application characteristics and hardware cost. The proposed performance model theoretically augments Amdahl's second law for a balanced system (Amdahl's I/O and memory number) and reveals that a balanced system needs almost 0.17GBPS I/O bandwidth, and almost 3GB of DRAM per GHz of CPU speed, considering Intel Xeon processing architecture and current price trend in processor, storage and memory.**

To substantiate our claim, we evaluate three fundamentally different cluster architectures, 1) a traditional HPC cluster called SupermikeII, 2) a regular datacenter called SwatIII, and 3) a novel MicroBrick based hyperscale system called CeresII. CeresII has 6-Xeon cores each running at 2GHz, 1-NVMe SSD with 2GBPS I/O bandwidth and 64GB DRAM, which closely resembling the optimum produced by our model. We evaluated these three clusters with respect to two widely used Hadoop-benchmarks (TeraSort and WordCount) as well as our own genome assembler based on Hadoop and Giraph, which serves as a real world example of a data- and compute-intensive workload. CeresII outperformed all other cluster architectures for all the benchmarks in terms of both execution time and cost/performance. For a large human genome assembly, CeresII showed more than 85% improvement over SuperMikeII and almost 40% improvement over SwatIII in terms of cost/performance.

***Index Terms*—Big data HPC, Amdah's second law, Cost/performance, Theory for a balanced system, Hadoop and Giraph.**

I. Introduction

CURRENT compute technologies for enterprise and scientific big data analysis are demanding more compute cycles per processor, with extreme I/O performance also required. At the same time, the hardware cost for storing and processing this big data is increasing linearly with data volume. Consequently, today's system designers are facing relentless pressure to reduce the system cost while also providing the expected level of performance. Complicating the scenario, the exponential growth of big data is rapidly changing the fundamental model of computation involved in high performance computing. A growing number of codes in both enterprise and scientific domain are being written using state of the art big data analytic software, such as Hadoop and Giraph which carefully consider data locality.

As a consequence, most of the existing high performance computing (HPC) clusters focusing only on tera to peta FLOP scale processing speed are found to be unbalanced for data-intensive scientific applications either in terms of performance, or economy, or both ([1], [2]). The existing data center architectures also need to be reexamined. Although the current state of the art big data analytic software were initially developed to work well on scale-out clusters of cheap commodity hardware, recently published results showed that scale-up can also provide better result for different data-intensive applications both in terms of performance and cost ([3]) .

As a matter of fact, system characteristics for data-intensive applications vary tremendously, especially because of varying nature and volume of data, leading to no single solution in terms of underlying cluster architecture. The problem becomes more complicated in a public cloud environment or an HPC cluster where the application characteristics are not familiar beforehand. In these scenarios, when the application characteristics are unfamiliar, it makes sense to invest in a balanced cluster as an initial approach. Hardware vendors, as well as cloud vendors (e.g., Amazon, Google, R-HPC, etc.) are investing a huge amount of money towards this. Millions of dollars are being spent for programs such as NSF Cloud[1] or XSEDE[2], where several industries and universities collaborate to find such balanced architectures to drive next generation cloud research.

A. K. Das, S. Goswami, R. Platania, K. Lee, and S.J. Park are with the Division of Computer Science and Electrical Engineering, Center for Computation and Technology Louisiana State University, Baton Rouge, USA. Email: [adas7, sgoswa1, rplata1, klee76, sjpark] @lsu.edu

J. Hong and W. Chang are with Samsung Electronics, Gyeonggi-do, S. Korea. Email: [jaeki.hong, wooseok_chang] @samsung.com

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

At this inflection point of HPC landscape, system designers must consider more degrees of freedom for new cluster architecture for big data than for existing HPC clusters which focus only on doing calculation at blazing speed. They must address such questions as: *how much I/O bandwidth is required per processing core? How much memory is required to optimize performance and cost?* Similar questions can be asked for scale-up and scale-out architectures also, as there is no concrete definition available for these. These complex performance and economic factors together motivate a new design of HPC infrastructure. However, they pose several challenges to the system designers who are striving for system balance in terms of processing speed, I/O bandwidth, memory, and the cost of the infrastructure.

To summarize, as the scientific research is becoming more data-driven in nature, it is obvious that providing more resources to an HPC cluster (processing speed, I/O bandwidth, DRAM) will obviously improve the performance of data-intensive scientific applications. But at what cost? So, the major challenge to the system designers nowadays is not in providing only high performance but in providing expected performance in reduced cost. There is limited analytical studies that address the challenges in developing a cost-effective, balanced distributed-cyber-infrastructure to analyze large-scale scientific big data.

With this motivation, this paper makes an initial attempt to resolve the existing performance and cost conundrum by theoretically augmenting Amdahl's second law for balanced system. This paper proposes a simple additive model to optimize cost/performance by quantifying the system balance between CPU speed, I/O bandwidth, and size of DRAM in terms of software application characteristics and current trend in hardware cost. Our model does not assume any specific software framework or hardware technologies, rather, it provides an easy to use, general guideline for designing a balanced system that can provide good performance in reduced cost. However, the outcome of the model (i.e., configuration of an optimal balanced system) suggests to use solid state drives (SSD) instead of hard disk drive (HDD) in a multicore machine to maintain the system balance. Assuming an equal distribution of I/O and compute work in a data-intensive application, our model suggests that a balanced HPC system needs almost 0.19-GBPS I/O bandwidth, and almost 3-GB of DRAM per GHz of CPU speed using Intel Xeon processor and current price trend of different hardware.

To substantiate our claim, we evaluate three fundamentally different cluster architectures as follows: 1) a traditional HPC cluster called SuperMikeII (located at LSU, USA), that offers 382 computing nodes, each with 16Xeon cores, 1HDD, and 32GB DRAM, 2) a regular datacenter architecture called SwatIII (located at Samsung, Korea), that has 128 nodes, each with 16Xeon cores, 4HDD, and 256GB DRAM and 3) a new MicroBrick-based architecture called CeresII (also located at Samsung, Korea). Each CeresII node has 6Xeon cores, 1NVMe SSD with 2GBPS I/O bandwidth and 6GB DRAM per node, thus closely resembling the optimum produced by our model. We evaluated these three architectures with respect to three different benchmark applications. Two of them are widely used benchmark Hadoop applications (TeraSort and WordCount), and the other one is our own benchmark genome assembler developed atop Hadoop and Giraph, which serves as a good real-world example of a data-, compute- and memory-intensive workload. For all three benchmark evaluations, CeresII with the most optimal configuration, outperformed the others in terms of both performance and cost/performance. For a large human genome assembly, CeresII showed more than 85% improvement over SuperMikeII and almost 40% improvement over SwatIII in terms od cost/performance

The rest of the paper is organized as follows: Section-II describes the prior work relate to our current effort. In Section-III, we discuss Amdahl's second law in detail. Section-IV describes the proposed model. Then, in Secton-V we show the details of our experimental testbeds and classify those using our proposed model. Section-VI describes the evaluation methodology for these clusters (i.e., the details of the software and benchmark that we used in our evaluation). In Section-VII we discuss the experimental results. Section-**??** discuses the limitations in the current model to stimulate discussion and future work. Finally, Section-IX concludes the paper.

II. Related Work

Numerous studies have been performed evaluating the performance implication of different big data analytic software on different types of hardware infrastructures. We categorize these existing studies into four different classes: 1) experimental evaluation of different cluster architectures for big data software, 2) simulation of big data analytic software, 3) analytical performance modeling of big data analytic software, and 4) system characterization using Amdahl's second law, that is the most relevant to our current effort.

A. Experimental evaluation of different cluster architecture for big data software.

Michael [4] investigated the performance characteristics of scale-out and scale-up architectures for interactive queries and found better performance using a scale-out cluster than a scale-up. On the other hand, Appuswamy [3] reached an entirely different conclusion in their study. They observed a single scale-up server to perform better than an 8-node scale-out cluster for eleven different enterprise-level Hadoop applications, including log-processing, sorting, and Mahout machine learning. Several studies have also been performed evaluating the impact of different storage media (SSD and HDD) or network configuration on Hadoop. For example, Kang [5] and Wu [6] argued that SSD improved the performance of Hadoop clusters with repect to different Hadoop benchmarks such as WordCount, DFSIO, TeraSort, etc. Moon [7] showed a significant cost benefit in TeraSort by storing the intermediate Hadoop data in SSD, leaving HDDs to store Hadoop Distributed File System (HDFS [8]) data. A similar result can be found in the study by Li [9] and Krish [10], where SSDs are used to store temporary data to reduce disk contention, and HDDs are used to store the HDFS data.

In our previous study [11], we aggregated many of these efforts together and reported the performance result for individual hardware components (network, storage, and memory)

as well as their overall organization. Considering 9 different cluster configurations, we provided significant insight on how to deploy SSDs better in Hadoop cluster. We also showed the imbalance between performance and economy that exists in scale-up and scale-out architectures. Although we experimentally showed that this imbalance was eliminated in a Samsung MicroBrick-based prototype cluster, we did not provide any quantitative analysis, for scale-up/out or for balanced system that can improve performance while staying within the budget. In fact, none of the prior efforts mentioned in this section provide such quantitative analysis which is extremely important to propose next-generation high-performance, balanced, distributed-cyber-infrastructures for data- and compute-intensive applications.

B. Simulation of big data software

Simulation is widely used for predicting performance of different big data software (mainly Hadoop) and analyzing design trade-offs in different hardware.

At the border level, all simulators generate virtual Hadoop MapReduce jobs and tune different hardware and software parameters in a simulated environment, mostly using prior experiences or trial-and-error to optimize the performance of some parts of the Hadoop framework or a given Hadoop job. For example, HSim [12] focuses on the Hadoop job parameters to optimize the performance of the entire job. SimMR [13], on the other hand, focuses on simulating the Hadoop task scheduling algorithms. MRSim [14] and MRPerf [15] unified all these aspects in a single simulator. They simulate the hardware environment using discrete event simulator packages such as, SimJava, GridSim, NS2, etc. Then they execute the fake Hadoop job on a small subset of data to predict the performance. Given prior experience, simulators can predict architecture alternatives, thus enabling huge cost and effort savings comparing to experimental evaluation on real hardware discussed before.

However, most of these simulators come with lots of overhead. They are time and resource consuming and often fail to assess the real system characteristics in the presence of huge volumes of big data. Furthermore, the trial-and-error method of performance optimization, considering the broad range of available hardware alternatives with more than 200 Hadoop parameters, is challenging and most of the time unreliable.

C. Analytical performance modeling of big data software

Analytical models abstract away several performance parameters and predict the performance of a Hadoop job mainly using single- or multi-layer queuing networks. For example, Viana [16] model the pipeline parallelism of a Hadoop job using a queuing network to predict the performance of the job. Wu [17] proposed a layered queue network model to predict the performance of a Hadoop job in a cloud environment. In a recent work, Ahn [18] proposed a queuing theory model to predict the performance of Hadoop job atop different storage technologies (i.e, HDD and SSD). Krevat [19] computed I/O complexity of Hadoop jobs for data-intensive applications and proposed a model to quantify the hardware resource wasted by Hadoop.

Unlike simulation, there is no overhead involved in an analytical model approach. However, the available alternatives for individual hardware components are huge in number. It is hard to find an optimally balanced architecture in this vast range of alternatives. Thus, both simulation and analytical approaches that model the performance of big data software may work well in selecting alternatives between two or more existing hardware infrastructures (in a small finite search space), however, their capability is limited in proposing a new architecture.

D. System characterization using Amdahl's second law

To date, the most practical approach to design a balanced system is to follow Amdahl's second law. Computer scientist Gene Amdahl postulated several design principles for a balanced system, collectively known as Amdahl's Second Law. He stated a balanced system needs 1bit of sequential I/O per second (Amdahl's I/O number) and 1byte of memory (Amdahl's memory number) per CPU instruction per second. These design principles for balanced system still hold true, even after 50 years with some ammendments proposed by Jim Gray in early 2000 [20]. Gray observed that Amdahl's I/O number still holds true but should be measured with respect to the relevant workload. He also observed that Amdahl's memory number is rising from 1 to 4. Gray's amendments are based on contemporary observation and are not formalized. We will discuss both Amdahl's second law and Gray's amendment in more details as well as formalize Gray's amendment later in Section-III.

Amdahl's second law (with Gray's ammendment) can be used for system characterization and proposing a balanced system. For example, Bell and Gray [21] classified the existing supercomputers based upon Amdahl's second law to clarify the future road map of the HPC architecture. Cohen [22] applied Amdahl's second law to the datacenter cluster to study the interplay between processor architecture and network interconnect in a datacenter. Chang [1] used Amdahl's second law to better understand the design implications of data analytic systems by quantifying workload requirements. Szalay [2], using Amdahl's second law, proposed a new cluster architecture based on SSD and low power processors (such as, Intel Atom, Zotac etc) to achieve a balance between performance and energy efficiency. In this study, the authors ignored the CPU micro architecture and simplified the Amdahl's I/O and Memory number by dividing the I/O bandwidth (bit/s) and size of DRAM (GB) by CPU-speed (GHz) instead of using instructions per second (IPS). The cluster architecture was balanced as the simplified Amdahl's I/O number was close to 1bit/s/IPS. In a recent study [23] Zheng evaluated a similar architecture as Amdahl's balanced blade for Hadoop applications and showed Atom processor is the system's bottleneck.

Unlike Szalay, we consider a balanced system as one that optimizes both cost and performance [24]. Hence, we did not consider the optimal balance of the system (i.e., the system's I/O and memory ratio to the processor speed) as constants as

in Amdahl's I/O and memory number. Instead, we propose an additive model express the optimal system balance as a function of both application characteristics and hardware price.

III. Amdahl's Second law, Gray's Ammendment, and Their Limitations

A. Original Form of Amdahl's Second Law

Computer scientist Gene Amdahl postulated several design principles in the late 1960 for a balanced system. As mentioned earlier, these design principals are collectively known as Amdahl's second law, which are as follows:

1) *Amdahl's I/O law:* A balanced computer system needs one bit of sequential I/O per second per instruction per second. From this point we will mention this law as Amdahl's I/O number. Alternatively, Amdahl's I/O number of a balanced system can be expressed as 0.125 GBPS/GIPS (by changing in conventional units).
2) *Amdahl's memory law:* A balanced computer system needs one byte of memory per instruction per second. From this point, we will mention this law as Amdahl's memory number.

Using the notations in Table-I, Amdahl's I/O and memory numbers can be expressed as:

$$\beta_{io}^{opt} = 0.125 \tag{1}$$

$$\beta_{mem}^{opt} = 1 \tag{2}$$

B. Gray's Amendment to Amdahl's Second law

Computer scientist Jim Gray reevaluated and amended Amdahl's second law in the context of modern data engineering. These amendments are collectively known as Gray's law. The revised laws are as follows:

1) *Gray's I/O law:* A system needs 8 MIPS/MBPS I/O (same as Amdahl's I/O number, but in different unit), but the instruction rate and IO rate must be measured on the relevant workload.
2) *Gray's memory law:* The MB/MIPS (alternatively, GB/GIPS) ratio is rising from 1 to 4. This trend will likely continue.

The underlying implication of Gray's I/O Law is that it aims for systems whose Amdahl's I/O number matches the Amdahl's I/O numbers of the applications (i.e., application balance) that run on that system. In the memory law, Gray put forward statistics reflecting the contemporary state of the cluster architecture.

Using the notations in Table-I Gray's laws can be expressed as follows:

$$\beta_{io}^{opt} = \gamma_{io} \tag{3}$$

$$\beta_{mem}^{opt} = 4 \tag{4}$$

TABLE I: Notations used in the model and their meaning

R_{cpu}	CPU speed of a given system S (GHz)
R_{io}	I/O bandwidth of system S (GBPS)
R_{mem}	DRAM size of system S (GB)
W_{cpu}	Fraction of work done by the CPU for a given application W
W_{io}	Fraction of work done by the disk(s) for W
W_{mem}	Fraction of work done by DRAM for W
P_{cpu}	Price per GHz of CPU speed
P_{io}	Price per GBPS of I/O bandwidth
P_{mem}	Price per GB of DRAM
β_{io}	System balance between I/O bandwidth and CPU speed for system S ($= R_{io}/Rcpu$)
β_{mem}	System balance between DRAM size and CPU speed for system S ($= R_{mem}/Rcpu$)
γ_{io}	Application balance between CPU and I/O bandwidth for application W ($= W_{io}/Wcpu$). This term quantifies to what extent the application is I/O-intensive or CPU-intensive. Lower value means more CPU-intensive, higher value means I/O-intensive. 1 represents both I/O- and CPU-intensive
γ_{mem}	Application balance between CPU and DRAM size for application W ($= W_{mem}/Wcpu$). This term quantifies to what extent the application is memory-intensive or CPU-intensive. Lower value means more CPU-intensive, higher value means memory-intensive. 1 represents both memory- and CPU-intensive
δ_{io}	Cost balance between CPU and I/O bandwidth for system S ($= P_{io}/Pcpu$)
δ_{mem}	Cost balance between CPU and DRAM for system S ($= P_{mem}/Pcpu$)
β_{io}^{opt}	Optimal system balance between I/O bandwidth and CPU speed (The problem under consideration)
β_{mem}^{opt}	Optimal system balance between DRAM size and CPU speed (The problem under consideration)

C. Limitations of Existing Laws

Amdahl's second law for balanced systems does not consider the impact of application balance (or applications' resource requirement). Because of the diverse resource requirements, a one-size-fit-all design as suggested in the original law (expressed as the constants in the right hand side of Equation-1 and 2), cannot satisfy the different resource balance ratios for a collection of analytic applications.

Gray's law is more realistic in the sense that it consider the impact of application balance on the cluster architecture. However, it is limiting to reflect the interplay between application balance and cost balance. The cost of hardware components has already changed the performance point and will keep on changing it as the technology continues to advance.

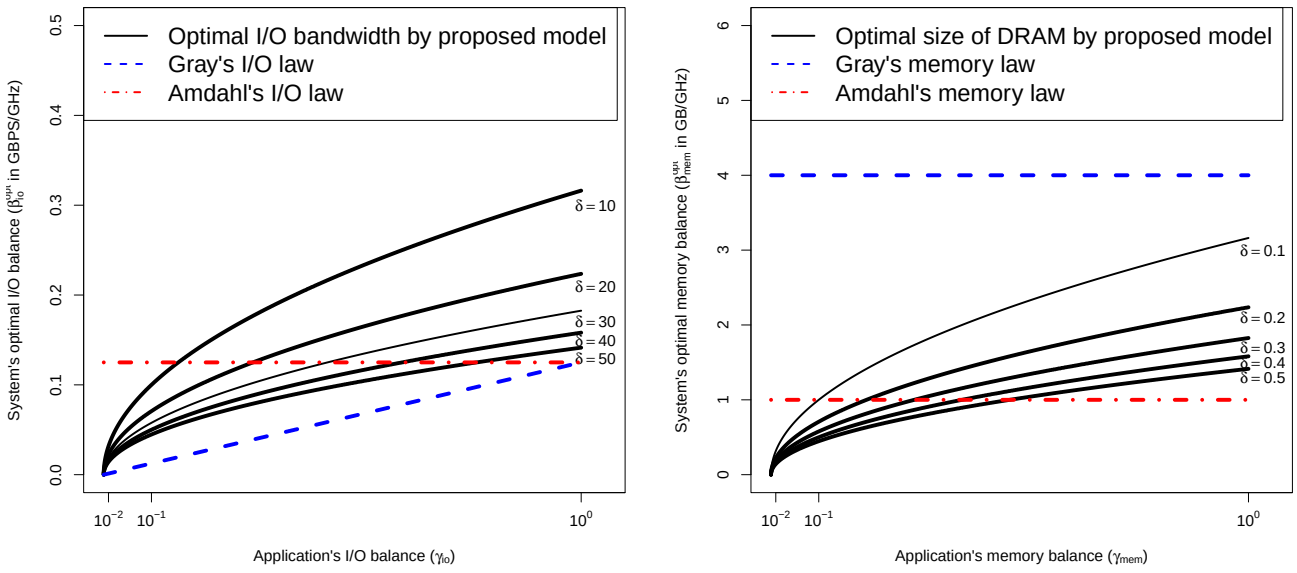
IV. Proposed Model for System Balance

A. Problem Definition

Using the notations described in Table-I, the optimal system balance (i.e., β_{io}^{opt} and β_{mem}^{opt}) needs to be expressed as a function of application balance (i.e., γ_{io} and γ_{mem}) and cost balance (i.e., δ_{io} and δ_{mem}). Mathematically it can be written as:

$$\beta_{io}^{opt} = f_1(\gamma_{io}, \delta_{io}) \tag{5}$$

$$\beta_{mem}^{opt} = f_2(\gamma_{mem}, \delta_{mem}) \tag{6}$$



(a) I/O balance (b) Memory balance

Fig. 1: Change in system's optimum I/O balance (β_{io}^{opt}) and memory balance (β_{mem}^{opt}) as a function of application balance (γ_{io} and γ_{mem}) for different cost balance (δ_{io} and δ_{mem}). For calculation of δ_{io} and δ_{mem} refer to Section-IV-F

B. Model Assumptions

For simplicity of calculation and better usability, the model first ignores the CPU micro architecture. That is, we consider the number of instruction executed per cycle (IPC) as proportional to CPU core frequency. Hence, express the balance between I/O and CPU in terms of GBPS/GHz, and balance between DRAM size and CPU in terms of GB/GHz. The practical implication of this assumption is that, the system designers may need to use this model repeatedly for different micro architectures based upon their corresponding IPC. It is feasible as micro architecture does not change as frequently as number of cores increases.

Second, for simplicity, we assume that the model is additive. That is, we ignore the overlap between work done by CPU, I/O, and memory subsystem. This way, the total execution time (T_{total}) of an application can be written as:

$$T_{total} = T_{cpu} + T_{io} + T_{mem} \tag{7}$$

$$\implies T_{total} = \frac{W_{cpu}}{R_{cpu}} + \frac{W_{io}}{R_{io}} + \frac{W_{mem}}{R_{mem}} \tag{8}$$

Third, we assume the total cost of the system as the summation of individual cost of CPU, I/O, and memory subsystems only. We ignore several constant components such as base cost, service cost, etc. This way, the total system cost (C_{total}) can be written as:

$$C_{total} = C_{cpu} + C_{io} + C_{mem} \tag{9}$$

$$\implies C_{total} = P_{cpu}R_{cpu} + P_{io}R_{io} + P_{mem}R_{mems} \tag{10}$$

C. Model Derivation

Assuming the performance as the inverse of the total execution time, the cost/performance (denoted as f_{cp}) can be expressed as:

$$f_{cp} = C_{total} \times T_{total} \tag{11}$$

$$\implies f_{cp} = (C_{cpu} + C_{io} + C_{mem}) \times (T_{cpu} + T_{io} + T_{mem}) \tag{12}$$

$$\begin{aligned} \implies f_{cp} = {} & C_{cpu}T_{cpu} + C_{cpu}T_{io} + C_{cpu}T_{mem} \\ & + C_{io}T_{cpu} + C_{io}T_{io} + C_{io}T_{mem} \\ & + C_{mem}T_{cpu} + C_{mem}T_{io} + C_{mem}T_{mem} \end{aligned} \tag{13}$$

By expanding all the time (T) and cost (C) terms using Equation-8 and 10 respectively and then substituting with the notation used for system balance in Table-I (i.e., β_{io} and β_{mem}), Equation-13 can be rewritten as:

$$\begin{aligned} f_{cp} = {} & P_{cpu}W_{cpu} + \frac{1}{\beta_{io}}P_{cpu}W_{cpu} + \frac{1}{\beta_{mem}}P_{cpu}W_{mem} \\ & + \beta_{io}P_{io}W_{cpu} + P_{io}W_{io} + P_{io}W_{mem} \\ & + \beta_{mem}P_{mem}W_{cpu} + P_{mem}W_{io} + P_{mem}W_{mem} \end{aligned} \tag{14}$$

Assuming a CPU core cannot perform disk and memory operation at the same time, partial differentiation with respect to β_{io} and β_{mem} can separately lead us to the optimum system balance in terms of I/O bandwidth and DRAM size respectively, with respect to processing speed.

Partially differentiating with respect to β_{io}, we get:

$$\frac{\partial f_{cp}}{\partial \beta_{io}} = -\frac{1}{\beta_{io}^2}P_{cpu}W_{io} + P_{io}W_{cpu} \tag{15}$$

For the optimal balance (β_{io}^{opt}) between CPU speed and I/O bandwidth, Equation-15 should equal to 0. Then, solving for β_{io} and replacing it with the workload and technology-cost balance terms mentioned in Table-I we get:

$$\beta_{io}^{opt} = \sqrt{\frac{\gamma_{io}}{\delta_{io}}} \tag{16}$$

Similarly, the optimum balance (β_{mem}^{opt}) between CPU speed and size of DRAM can be derived as:

$$\beta_{mem}^{opt} = \sqrt{\frac{\gamma_{mem}}{\delta_{mem}}} \tag{17}$$

Equation-16 and 17 show the contribution of application balance and cost balance towards optimal system balance.

D. Observations and Inferences

Gray's law is a special case of our model when the cost balance equals the inverse of the application balance (i.e., the application's CPU I/O and, memory requirement exactly balance the contemporary cost of the hardware).

Figure-1 shows the optimal system balance as a function of application balance and cost balance as proposed by our model. It also compares our model with Amdahl's second law and Gray's law. The horizontal X-axis shows the different types of application balance. value of $\gamma_{io} = 1$ presents an I/O as well as CPU-intensive application where Gray's I/O law and Amdahl's I/O law both intersect. Likewise, $\gamma_{mem} = 1$ presents a memory and CPU-intensive applications.

It can be observed that our model sugests to use less DRAM than suggested by Gray's memory law. However, our model yields higher values for system's I/O bandwidth comparing to Gray's I/O law. This is because current price of magnetic disk or SSD is much lower than that of DRAM.

It can be noticed that lower balance ratio between per-GBPS-I/O-cost to per-GHz-CPU-cost leads to higher balance ratio between system-I/O-GBPS to system-CPU-GHz. One straight forward interpretation for this observation is that if per-GBPS-I/O-cost starts decreasing faster than the per-GHz-CPU-cost, designers need to increase the system-I/O-GBPS of a single server to achieve the new I/O balance ratio (GBPS/GHz). However, Instead of scaling up a single server in terms of storage, designers can scale out in terms of processor. That is, reduce the number of processor in a single server and add more servers with the same new system balance ratio. Theoretically, both these scale-up and scale-out cluster are optimally balanced, i.e., both scale-out and scale-up can provide optimal cost/performance.

A small example scenario can clarify the above concept. Let us consider a fictitious cluster (say, $Cluster^1$). Each node of this cluster has R_{cpu}^1 CPU-speed with R_{io}^1 I/O bandwidth. Each node of $Cluster^1$ is optimally balanced so that it can minimize cost/performance. Let us consider, the optimal I/O balance ratio equals β_{io}^1. Since, each node of $Cluster1$ is optimally balanced, we can say, $\beta_{io}^1 = \frac{R_{io}^1}{R_{cpu}^1}$. Now, let us consider for faster price fall for I/O bandwidth than for processor speed, the new optimal system balance is raised by n times. That is, the current optimal system balance is $n\beta_{io}^1$. To maintain this new optimal system balance, designers have two alternatives:

- They can straight forward scale up in terms of storage. That is, each node in the new cluster has R_{cpu}^1 CPU speed with nR_{io}^1 I/O bandwidth. Let us denote this cluster as $Cluster_{scale-up}^1$
- Alternatively, they can scale out in terms of processor. That is, each node now has $\frac{R_{cpu}^1}{n}$ CPU speed and there are n more nodes in the cluster. Let us denote this cluster as $Cluster_{scale-out}^1$

It is worth noticing that both $Cluster_{scale-up}^1$ and $Cluster_{scale-out}^1$ now have same total aggregated I/O bandwidth, same amount of processing speed. At the same time, Both of them are theoretically maintaining the new optimal system balance ratio ($n\beta_{io}^1$) to maximize the cost/performance.

A Similar example can be given for the system's memory balance also. Hence, an implication of our model is that both scale-out and scale-up can theoretically optimize the cost/performance if the cluster is properly balanced in the absence of any network bottleneck.

E. Notes on Different types of I/O

1) Sequential vs Random I/O: It should be noted that the model does not consider sequential and random I/O bandwidth separately. Designers need to be careful about the application characteristics. An application with frequent random I/O may find it beneficial to use SSD instead of HDD to align with the optimum I/O balance (i.e., β_{io}) produced by the model. For example, Hadoop involves a huge amount of random I/O in its shuffle phase. To provide such a huge random I/O bandwidth, SSD can be used.

An underlying assumption of the model is that the aggregate I/O bandwidth of all the disks attached to a compute node is less than or equal to the bandwidth of the disk controllers. Otherwise, the disk controllers can be saturated (an observation by Szalay [2] and our previous work [11]). Because of lower I/O bandwidth of HDD, traditionally, disk controllers have never been the source of a bottleneck. However, the demand for I/O bandwidth is increasing for big data applications. To meet this demand, it is common to use multiple disks per node. Furthermore, the recent SSD technologies started producing multiple GBPS of I/O bandwidth. It is the designer's responsibility to choose correct hardware so that disk and the disk controller is balanced (i.e., the disk controller is neither overprovisioned nor underprovisioned).

2) Network I/O: The model does not consider the impact of network interconnect. Hence, we find it important to provide some qualitative information regarding current state of the network architecture in the HPC cluster, source of architectural imbalance, and possible solution. Traditional HPC clusters use a hierarchy of InfiniBand switches with a fat tree model of connectivity. This approach typically uses a layered architecture. The hosts or servers are connected to the bottom layer, called access layer, which is then aggregated to an intermediate layer of switches, called edge layer or leaf. The edge layer switches are then connected to the top layer switches, called core layer or spine, where the actual bandwidth of the network is scaled. To prevent over subscription, the link speeds get progressively higher from access layer to leaves

TABLE II: Cost of different hardware components

Hardware components	Cost($)[3]	Unit Price
Intel Xeon 64bit 2.6 GHz E5 series (8-cores) processor	1760	$42/GHz
Intel Xeon D-1541	650	$54/GHz
Western Digital RE4 HDD (120MBPS), 500GB	132	$2252.80/GBPS/TB
Western Digital VelociRaptor HDD (150MBPS), 600GB	167	$1900.09/GBPS/TB
Samsung 840Pro Series SATAIII SSD (400MBPS), 512GB	450	$2250.00/GBPS/TB
Samsung NVMe SSD PM953 (2GBPS), 950GB	450	$242.52/GBPS/TB
Samsung DDR3 16GB memory module	159	$10/GB
32GB 1600MHz RAM (decided by Dell)	140	$4.37/GB

to spine starting from only few Mbps to several Gbps. This architecture may suffer from bottleneck issues, introducing architectural imbalance as the bandwidth of the host adapter is increasing at an outstanding pace (an observation by Cohen [22]). Furthermore, to accomodate many servers with fewer switches (thus, reducing the cost of the network), a blocking is used, which again limits the performance of big data genomic applications which demand more bandwidth. As an alternative, the simple Clos-based architecture is gaining popularity, where all the lower layer switches (i.e., leaf) are connected to all the top layer switches (i.e., spine) using a full mesh topology, thus achieving a non-blocking model using inexpensive devices. The data-intensive applications based on Hadoop or Giraph, which transmit huge amounts of data over network in different phases of computation, can use more bandwidth from this all-to-all connection model.

F. An Illustrative Example of Applying the Model to Build a Balanced Cluster

In this section, we demonstrate an example of how to apply our model to build a cost-effective, balanced cluster. To reflect today's data-, compute-, and memory-intensive scientific applications, we consider the work done by the CPU, I/O and memory subsystem (i.e., W_{cpu}, W_{io}, and W_{mem}) are equal for that application. That is, using the notation in Table-I, the application balance can be written as:

$$\gamma_{io} = \gamma_{mem} = 1 \tag{18}$$

Table-II shows a sample list of different hardware components and their corresponding cost[4]. Some of these hardware components are actually used in our experimental testbeds discussed in Section-V. Hence, the optimal system balance derived in this section are always referred as a real optimum value throughout the paper.

The *Unit Price* shows the current price trend for different processor, storage and memory alternatives in their corresponding unit. We consider Intel Xeon processor. As it can be seen in Table-II, the cost per MBPS of sequential I/O for both HDD and SATA-SSD is almost similar irrespective of change in storage technology provided the same storage space per disk. Whereas, the I/O bandwidth cost started reducing significantly with NVMe SSD. The cost per GB of DRAM is increased almost double from DDR2 to DDR3.

We first calculated the average cost of each hardware component from the available list (Table-II) in terms of their corresponding unit price. For example, we have two different Xeon processors, E5-series and D-series as shown in Table-II. Their rspective unit prices are $42/GHz and $54/GHz. Hence, in this example we selected average unit cost of the processor as $48/GHz. Similarly the cost of DRAM is calculated as $7.20/GB. However, calculating the average I/O bandwidth cost is not as simple as with processor or memory. This is because, the price of the storage devices is often directed by the capacity (i.e., GB), whereas, the model needs the cost per bandwidth (i.e., GBPS) which again depends on the number of disks. For simplicity, we calculated the unit price of a disk in terms of cost per GBPS per TB. That is, we calculated the cost per GBPS using the same storage capacity (1TB) for each disk. Finally, we calculated the average cost of I/O bandwidth (again assuming 1TB storage) for all the disks listed. For the disk drives shown in Table-II, the average I/O bandwidth cost is $1661.35/GBPS.

Next, we divided the I/O cost per GBPS and DRAM cost per GB by the CPU cost per GHz to get the cost balance for I/O and memory respectively. Using the notation in Table-I, cost balance for this example can be written as:

$$\delta_{io} = 34.61 \tag{19}$$

$$\delta_{mem} = 0.15 \tag{20}$$

Using Equation-16 and replacing the value of γ_{io} and δ_{io} from Equation-18 and 19 respectively, we can get the system's I/O balance in terms of GBPS/GHz as:

$$\beta_{io}^{opt} = 0.17 \tag{21}$$

Similarly, using Equation-17, 18, and 20, we can get the system's DRAM balance in terms of GBPS/GHz as:

$$\beta_{mem}^{opt} = 2.7 \tag{22}$$

V. Experimental Testbeds: Cluster Characterization Using Proposed Model

As mentioned earlier, we evaluate three different cluster architectures. Table-III shows the overview of our experimental testbeds. The first one, SuperMikeII represents a traditional HPC cluster. This LSU HPC cluster offers a total of 440 computing nodes. However, a maximum 128 can be allocated at a time to a single user. SwatIII represents a regular datacenter. This Samsung datacenter has 128 nodes. However, we used maximum 16 nodes for our experiments. The last one, CeresII, is a novel hyperscale system, based on Samsung MicroBrick with a maximum of 40 nodes available to us.

[3]Sample costs are collected from amazon.com, newegg.com, ark.intel.com.

[4]A detail analysis of cost trend is beyond the scope of this paper. However, the samples can reflect the current trend well. Also, the process described in this example can be used for any range of hardware components

TABLE III: Experimental Testbeds

Cluster name	Processor	CPU Core Speed (GHz)	#Cores /node	CPU Speed (GHz) /node	#Disks /node and type	Seq I/O Band-width /Disk (GBPS)	Total Seq I/O Band-width (GBPS) /node	DRAM /node (GB)	Maximum #Nodes avail-able	β_{io}	β_{mem}
SuperMikeII (Traditional Su-percomputer)	Two 8-core SandyBridge Xeon	2.6	16	41.6	1-HDD (SATA)	0.15	0.15	32GB	128	0.003	0.77
SwatIII (Regu-lar Datacenter)	Two 8-core SandyBridge Xeon	2.6	16	41.6	4-HDD (SATA)	0.15	0.60	256	16	0.015	6.15
CeresII (MicroBrick-based Hyperscale System)	One 6-core Xeon	2	6	12	1-SSD (NVMe)	2	2	64	40	0.166	5.33

A. Cluster Characterization System's I/O and Memory Balance

1) SuperMikeII (Traditional HPC cluster): SuperMikeII has two 8-core Intel Xeon E5 series processor per node thus offering huge processing power. However, each SuperMikeII node is equipped with only one HDD (Western Digital RE4), thus limited in terms of I/O bandwidth. Also, each SuperMikeII node has only 32GB DRAM (Dell). As a result, SuperMikeII has $\beta_{io} = 0.003$ and $\beta_{mem} = 0.77$, both a magnitude smalller than the optimum produced by our model for a data-, compute- and memory-intensive application as shown in Equation-16 and Equation-17. Using the plot shown in Figure-1 (or using Equation-16 and 17 with SuperMikeII hardware cost shown in Table-II) SuperMikeII provides optimal cost/performance for those applications where $\gamma_{io} = 0.0005$ or $\gamma_{mem} = .06$. Hence, it can be said SuperMikeII can provide cost-optimized performance for traditioal compute-intensive applications such as supercomputing simulations, astrophysics calculations where γ_{io} is in the order of 10^{-3}.

2) SwatIII (Existing Datacenter): Unlike SuperMikeII which has only one HDD per node, SwatIII uses 4HDDs (Western Digital VelociRaptor) per node using JBOD (Just a Bunch of Disk) configuration while using the same processors (i.e. two 8core Intel Xeon E5 series) as SuperMikeII. Since the I/O throughput increases linearly with the number of disks, SwatIII's $\beta_{io} = 0.015$ is higher than SuperMikeII but lower than the optimum produced by our model for an I/O- and compute-intensive application (Equation-16). On the other hand, each SwatIII node has 256GB DRAM (Samsung DDR3), thus achieving a very high value for $\beta_{mem} = 6.15$. It is worth noticing that β_{mem} of SwatIII is even higher than the optimum produced by the model (Equation-17). Using Figure-1 (or using Equation-16 and 17 with SuperMikeII hardware cost shown in Table-II), we can show SwatIII can produce cost optimized performance when $\gamma_{io} = 0.01$ and $\gamma_{mem} = 1.47$. That is, SwatIII can be a good choice for moderately I/O-intensive applications and for memory-intensive applications such as in-memory NoSQL. However, SwatIII may show worse cost/performance for many of the modern I/O-intensive big data applications.

3) CeresII (MicroBrick-based Hyperscale System): The last one, CeresII, is a novel hyperscale system based on Samsung MicroBricks. Each MicroBrick (or simply a compute server) of CeresII consists of a 6core Intel Xeon D-1541 processor with a core frequency of 2GHz, one NVMe-SSD (Samsung PM953) with an I/O bandwidth of 2GBPS, and 64GB DRAM (Samsung DDR3). β_{io} of CeresII is 0.17 which is same as the optimum calculated by our model in Equation-16. On the other hand, β_{mem} of each CeresII module is 5.33. Although, it is higher than the optimal, it is less than SwatIII. Thus, CeresII is the most balanced cluster among all the available resources and we expect to get the best cost to performance for today's I/O-, compute- and memory-intensive applications.

VI. Cluster Evaluation Methodology

A. Software platform: Hadoop and Giraph

To evaluate the relative merits of the clusters with respect to data-intensive scientific applications, we first need a big data analytic platform. The obvious selection is Hadoop, which is nowadays the de facto standard for big data analysis and rapidly gaining popularity in scientific computing. We also evaluate the cluster hardware with respect to Giraph, which is a large scale graph processing framework developed atop Hadoop. We use Clouera-Hadoop-2.3.0 and Giraph-1.1.0 for the entire study.

B. Hadoop Configurations Overview

For each cluster configuration, one of the nodes is always used as Hadoop NameNode. All other nodes are used as DataNodes. Since our goal is to evaluate the the architectural balance of different hardware components, we avoid any unnecessary changes in the source code of Hadoop or Giraph. To evaluate the balance between the hardware components, we set up the Hadoop parameters such that the application can make use of the maximum available processing speed, I/O bandwidth and DRAM. However, it is worthy to mention here that due to limited I/O bandwidth in SuperMikeII (traditional HPC cluster with only 1HDD per node) we could launch only 8 concurrent YARN containers (i.e., 8 concurrent map/reduce tasks), which is half of the total number of cores to get the optimal performance. Appendix-A shows a brief description of our Hadoop configurations.

TABLE IV: Data size for different benchmark applications

Job name	Job Type	Input	Final output	# jobs	Shuffled data	HDFS Data	Application Characteristics
Terasort	Hadoop	1TB	1TB	1	1TB	1TB	Map: CPU-intensive, Reduce: I/O-intensive
Wordcount	Hadoop	1TB	1TB	1	1TB	1TB	Map and Reduce: CPU-Intensive
Graph Construction (large size human genome)	Hadoop	452GB (2-billion reads)	3TB	2	9.9TB	3.2TB	Map and Reduce: CPU- and I/O-intensive
Graph Simplification (large size human genome)	Series of Giraph jobs	3.2TB (1483246722 vertices)	3.8GB (3032297 vertices)	15	-	4.1TB	Memory-Intensive

C. Benchmark Application Characteristics

Table-IV summarizes the details of the benchmark applications, the data size handled by each of these applications and their corresponding characteristics. We used three benchmark applications for our evaluation: TeraSort, WordCount, and a real world genome assembly application. A brief description of each of these applications are as follows:

1) TeraSort: The first one is TeraSort, a standard benchmark for any data processing framework. The Hadoop version of TeraSort samples and partitions the input data in its map phase based upon only first few characters. The reduce phase then uses the quicksort algorithm to sort each of the partitions locally. The map phase of TeraSort is CPU-intensive as it reads only the first few characters of each row in the input dataset to generate the key (called sample-key) for each data. However, based upon the data size, the reduce phase can be severely I/O-intensive. The entire dataset needs to be transferred to different reducers, and each reducer needs to read each row at least once to sort the data set. Again, for huge volumes of data partitions, that cannot fit in memory, an external sort may be applied making the reduce phase severely I/O intensive.

2) WordCount: The second one is WordCount, another widely used Hadoop benchmark. The map phase of WordCount parses the input dataset line by line to extract each word. Each word is emitted with a 1 as its initial count, which is then summed up in the reduce phase to output its frequency. Since both the map and reduce phase read the entire data set sequentially just once, both phases of WordCount is CPU-intensive. However, it should not be compared with the traditional compute-intensive applications such as simulations. Because of the huge volume of data, WordCount also needs to spend a significant amount of time for I/O.

3) Genome Assembly: The third one is a genome assembly application that we developed based on Hadoop and Giraph to address the challenges in large-scale genome assembly, which recently made its way to the forefront of big data challenges. The first phase uses Hadoop to scan the genome data set to filter the lines containing only neucleotide characters (i.e., A, T, G, and C) and then divides each of those lines into smaller fragments of length k, known as k-mer. The map phase emits each two subsequent k-mers as a key-value pair representing a vertex and an edge from it. The reduce phase then aggregates all the edges emitted from each vertex to write the entire graph structure to HDFS. The second phase of the assembly application uses Giraph to to compress each of the linear chains in the graph structure to one vertex and remove the tip and bubble structures in the graph (included because of sequencing error). After removing the tips and bubbles the graph may have some new linear chains which are compressed again. Similarly, new tip and bubble structures are removed again. So, the process continues incrementally as a series of Giraph jobs until there are no tips or bubbles in the graph. Appendix-B shows the overview of the algorithms used in our genome assembly application.

D. Data Size

For both TeraSort and WordCount, we generated 1TB random dataset as the input for these benchmarks. Both TeraSort and WordCount generate almost a similar amount data in the intermediate shuffle phase. The output data size of TeraSort is also same as its input (i.e., 1TB in this work), whereas the output of WordCount may vary based upon the frequency of different words in the randomly generated dataset. However, it is closed to 1TB.

For the genome assembly benchmark application, we use a large size human genome dataset (452GB). The corresponding graph size is 3.2TB. The Hadoop stage of the assembly application is severely shuffle intensive. The temporary shuffle data is almost 21-times more than the input size. The Human genome is openly available in NCBI website with accession number SRX016231[5].

VII. Result and Discussion

A. Evaluating the Cost/Performance of Different Clusters with TeraSort and WordCount

To show the balance between performance and economy, we keep the total cost[6] the same across all the clusters. Table-II shows the available resources for all three cluster architectures available to us (i.e. SuperMikeII, SwatIII and CeresII) while keeping the total cost same across all the clusters. We used the cost of 16 nodes of SuperMikeII as the baseline. Then, we divided this baseline-cost by the cost of each node in SwatIII and CeresII to count the number of nodes to be used in these two different clusters.

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

[6]The cost information of each hardware component can be found in Table-II

TABLE V: Resources available for each cluster architecture when scaling with respect to cost. The cost of 16 nodes of SuperMikeII is used as baseline. This is used for TeraSort and WordCount benchmark

Cluster Configurations	SuperMikeII	SwatIII	CeresII
Total cost ($)	60864	60864	60864
Cost/nodes ($)	3804	6911	2700
#Nodes	16	9	23
Total processing speed (GHz)	665.60	374.4	276.00
Total I/O bandwidth (GBPS)	2.4	5.40	46.00
Total storage space (TB)	8.00	21.60	21.85
Total DRAM Size (TB)	0.50	2.34	1.47

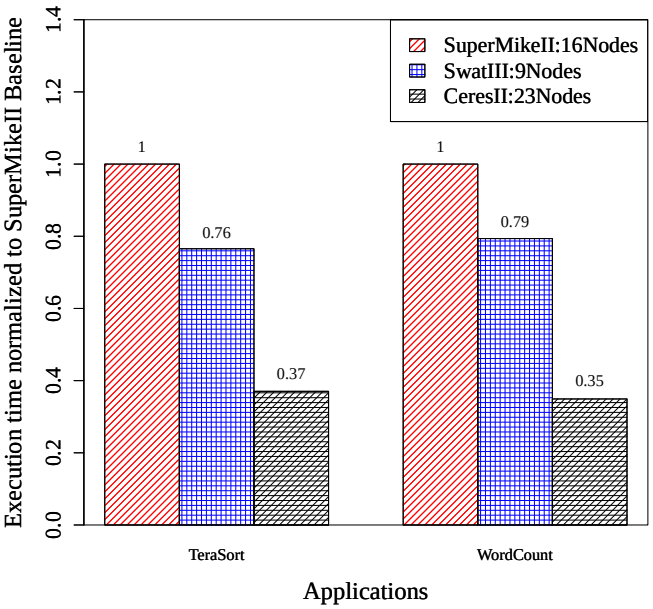
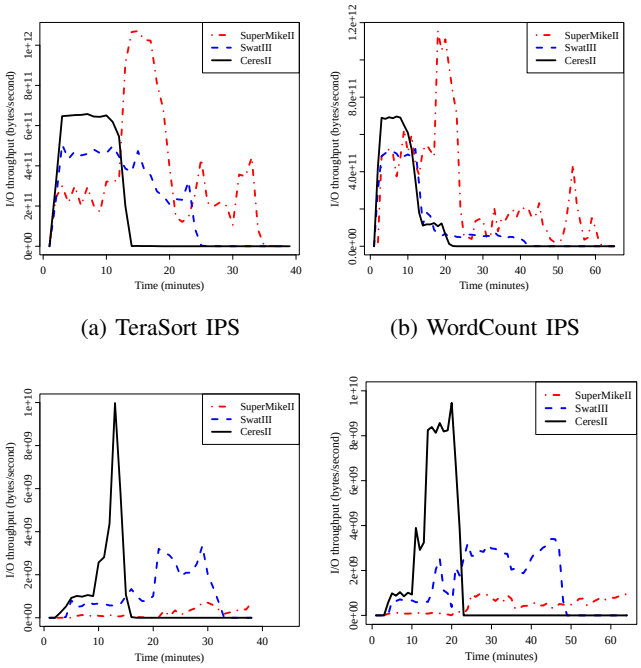


Fig. 2: Execution time of TeraSort and WordCount over different cluster architecture keeping the total cost of each cluster same

We ran the first two applications described in Table-IV in all three cluster configurations and measured their execution time. All results are the means of at least 3 runs of each application on each configuration. Figure-VI shows the results normalized to the SuperMikeII baseline. That is, the execution time of SuperMikeII is always 1. We see that CeresII, being closer to the optimum produced by our model performs surprisingly well:

1) Comparing to the SuperMikeII baseline, for both TeraSort and WordCount benchmark application, CeresII shows almost 65% improvement.
2) Comparing to SwatIII, CeresII shows almost 50% improvement in execution time for TeraSort and WordCount.



(a) TeraSort IPS (b) WordCount IPS

(c) TeraSort I/O throughput (d) WordCount I/O throughput

Fig. 3: Average #CPU instruction per second (IPS) and average I/O bandwidth per node of different cluster architectures for TeraSort and WordCount benchmark

1) System Characteristics: To monitor the system characteristics we used Perf tool and the Cloudera-Manager-5.0.0. Figure-3 shows the total number of CPU instructions per second (IPS) and I/O bandwidth across different cluster architectures while keeping the total cost of the cluster the same.

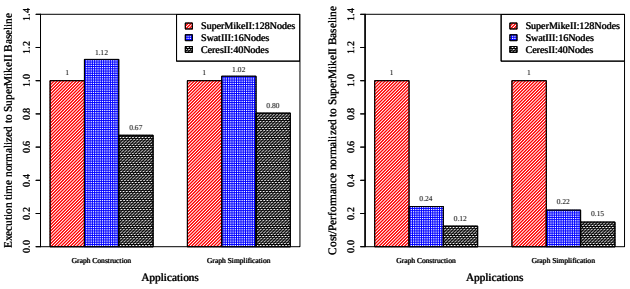
For both TeraSort and WordCount, the peak IPS of SuperMikeII is significantly higher than both SwatIII and CeresII. However, the average I/O bandwidth of SuperMikeII is significantly less than the other two, leading to significantly high I/O wait that results in extremely lowe average IPS (lower than the others). This low average IPS results in longer execution time.

SwatIII cluster shows a better balance between CPU and I/O than SuperMikeII. However, because of its high size of DRAM, it again gives up I/O bandwidth with respect to CeresII when the total cost of the cluster is same.

CeresII cluster is optimally balnced in terms of CPU speed and I/O bandwidth with a β_{io} value of 0.17. Figure-3a and 3b shows that CeresII achieves higher Peak IPS than SwatIII but less than SuperMikeII for both the TeraSort and WordCount benchmark. On the other hand, CeresII achieves significantly higher I/O bandwidth than both clusters, keeping the peak and average IPS of the application similar, leading to the lowest execution time for any of the applications.

TABLE VI: Maximum available resources in each cluster architecture (used for large human genome assembly)

Cluster Configurations	SuperMikeII	SwatIII	CeresII
Total cost ($)	486912	110576	108000
Cost/node ($)	3804	6911	2700
#Nodes	128	16	40
Total processing speed (GHz)	5324.8	665.6	480.00
Total I/O bandwidth (GBPS)	19.2	9.60	80.00
Total storage space (TB)	64.00	38.40	40.00
Total DRAM Size (TB)	4	4	2.56



(a) Execution time (b) Cost/performance

Fig. 4: Performance of different cluster for large size human genome assembly

B. Evaluating Different Cluster for Large Size Human Genome Assembly

To assemble the large human genome (452GB), we used the maximum available resources in each of the clusters to accommodate the huge amount of shuffled data (9.9TB) as well as the graph data (3.2TB). That is, we used all 128 nodes of SuperMikeII, 16 nodes of SwatIII, and 40 nodes of CeresII for this application. Figure-4a and 4b shows the execution time and the cost/performance respectively for the Hadoop and Giraph stages of the human genome assembly pipeline. As shown in the model, we consider the performance as the inverse of the execution time and multiplied the execution time with the total cost of each cluster to get the corresponding cost/performance. All data is again normalized to SuperMikeII baseline. The results are as follows:

1) CeresII, even with almost 90% less processing speed than SuperMikeII across the cluster, outperformed SuperMikeII by more than 30% in the Hadoop stage of the assembly. In the Giraph stage, the performance gain is almost 20%. In terms of cost/performance, the corresponding gains are 88% and 85% respectively.
2) Comparing to SwatIII, the processing power of CeresII is 72%. However, due to the optimal architectural balance, CeresII outperforms SwatIII by almost 50% in the Hadoop stage. The corresponding improvement in the Giraph stage is 20%. In terms of cost/performance, CeresII shows almost 50% and 30% improvement over SwatIII for the Hadoop and Giraph stages respectively.

VIII. Limitations of the Model

As an initial approach, our main focus was on simplicity. To keep the model simple, we gave up several details and micro-architecture in CPU, storage, and memory subsystems. It is always possible to include more subtle parameters, such as CPU multi-threading, number of I/O operations per second and I/O latency, network interconnect across cluster, memory-bandwidth and latency, etc., to the existing additive model to come up with better and more accurate equation for system balance. We also ignored the overlap between work done by CPU and disk which severely depends on the software framework and application's I/O characteristics and its nature of the parallelism (partitioned and pipelined). Building a better, accurate model is the part of our future work.

IX. Conclusion and Future Work

Big data needs big resources. As the nature of scientific computing is changing from compute-centric to data-centric, it is obvious that providing more resources (more processing speed, I/O bandwidth, DRAM, etc.) will provide more performance. So, the major challenge is now in providing expected performance in reduced cost. This paper makes an initial attempt to analytically resolve the performance and cost conundrum prevalent in big data research.

We propose a simple additive model to express the optimal system balance in terms of application characteristics and hardware price to minimize the cost/performance. We also verified our claim experimentally that the cluster that resembles the optimal produced by our model, (i.e., CeresII) outperforms an existing HPC cluster and a regular data center architecture by several magnitudes for different types and sizes of applications.

The proposed model also has implications for the way current big data analytic clusters or data center clusters are provisioned. We theoretically showed that if proper balance is maintained between different hardware components, both scale-out and scale-up clusters can provide optimal the cost/performance. However, because of resource limitations, we could not validate this claim and, thus, this is obviously one of the future directions of our research.

Appendix A
Hadoop and Giraph: Programming Model and Configuration

This part is similar to our previous work [11].

A. Programming model of Hadoop and Giraph

Hadoop and Giraph were originated as the open-source counterpart of Google's MapReduce [25] and Pregel [26] respectively. Both the software read the input data from the underlying Hadoop Distributed File System (HDFS) in the form of disjoint sets or partitions of records. Then, in the MapReduce abstraction, a user-defined map function is applied to each disjoint set concurrently to extract information from each record in the form of intermediate key-value pairs. These key-value pairs are then grouped by the unique keys

and shuffled to the reducers. Finally, a user-defined reduce function is applied to the value-set of each key, and the final output is written to the HDFS. The MapReduce framework enables data- and compute-intensive applications to run large volume of distributed data sets over distributed compute nodes with local storage. On the other hand, Giraph uses the Bulk Synchronous Parallel model [27] where computation proceeds in supersteps. In the first phase of a superstep, Giraph leverages Hadoop-mappers when a user-defined vertex-program is applied to all the vertices concurrently. In the end of each superstep, each vertex can send a message to other vertices to initiate the next superstep. Alternatively, each vertex can vote to halt. The computation stops when all the vertices vote to halt unanimously in the same superstep. Giraph enables memory- and compute-intensive applications to upload data into distributed memories over different compute nodes.

B. Hadoop configuration and optimization

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

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

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

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

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

APPENDIX B
GENOME ASSEMBLY ALGORITHMS

A similar description of the assembly application can be found in our previous work [11]. Also, the paper describing the genome assembly algorithms in detail is already accepted in 8th international proceedings of BICoB 2016. However yet to be published [28].

The motivation behind selecting genome assembly application is that, the high throughput next generation DNA sequencing machines (e.g., Illumina Genome Analyzer) has outpaced Moore's law and started producing a huge amount of short read sequences typically in the scale of several Gigabytes to Terabytes. Furthermore, the size of the de Bruijn graph built from these vast amount of short reads may be a magnitude higher than the reads itself making the entire assembly pipe line severely data-intensive.

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 [29]. We classified the de novo assembly in two stages as follows: *a*) Hadoop-based de Bruijn graph-construction and *b*) Giraph-based graph-simplification.

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

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

We compress the linear chains into a single vertex using a randomized parallel list ranking algorithm of [30] implented with Giraph. The computation proceeds in rounds of two supersteps until a user defined $limit$ is reached. In one superstep, each compressible vertex with only one incoming and outgoing edge is labeled with either $head$ or $tail$ randomly with equal probability and send a message containing the tag to the immediate predecessor. In the next superstep, all the $head$-$tail$ links are merged, that is, the $head$-$k$mer is extended (or, appended) with the last character of the $tail$-$k$mer and the $tail$ vertex is removed. Each vertex also maintain a frequency counter which increments after each extension to that vertex.After the compression, all the tip-structures in the graph are removed in two supersteps. The first superstep identifies all the vertices with very short length (less than a threshold) and no outgoing edge as tips which are removed in the second superstep.Finally, the bubbles-structures are resolved in another two supersteps. In the first superstep, the vertices with same predecessor and successor as well as very short length (less than a threshold) are identified as bubbles.

They send a message containing their id, value, and frequency to their corresponding predecessors. The predecessor employs a Levenshtein-like edit distance algorithm. If the vertices are found similar enough, then the lower frequency vertex is removed.

ACKNOWLEDGMENT

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

REFERENCES

[1] J. Chang, K. T. Lim, J. Byrne, L. Ramirez, and P. Ranganathan, "Workload diversity and dynamics in big data analytics: implications to system designers," in *Proceedings of the 2nd Workshop on Architectures and Systems for Big Data*. ACM, 2012, pp. 21–26.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] A. K. Das, S.-J. Park, J. Hong, and W. Chang, "Evaluating different distributed-cyber-infrastructure for data and compute intensive scientific application," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 134–143.

[12] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "Hsim: a mapreduce simulator in enabling cloud computing," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 300–308, 2013.

[13] A. Verma, L. Cherkasova, and R. H. Campbell, "Play it again, simmr!" in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*. IEEE, 2011, pp. 253–261.

[14] S. Hammoud, M. Li, Y. Liu, N. K. Alham, and Z. Liu, "Mrsim: A discrete event based mapreduce simulator," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 6. IEEE, 2010, pp. 2993–2997.

[15] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups." in *MASCOTS*, vol. 9. Citeseer, 2009, pp. 1–11.

[16] E. Vianna, G. Comarela, T. Pontes, J. Almeida, V. Almeida, K. Wilkinson, H. Kuno, and U. Dayal, "Analytical performance models for mapreduce workloads," *International Journal of Parallel Programming*, vol. 41, no. 4, pp. 495–525, 2013.

[17] X. Wu, Y. Liu, and I. Gorton, "Exploring performance models of hadoop applications on cloud architecture," in *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 2015, pp. 93–101.

[18] S. Ahn and S. Park, "An analytical approach to evaluation of ssd effects under mapreduce workloads," *JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE*, vol. 15, no. 5, pp. 511–518, 2015.

[19] E. Krevat, T. Shiran, E. Anderson, J. Tucek, J. J. Wylie, and G. R. Ganger, "Applying performance models to understand data-intensive computing efficiency," DTIC Document, Tech. Rep., 2010.

[20] J. Gray and P. Shenoy, "Rules of thumb in data engineering," in *Data Engineering, 2000. Proceedings. 16th International Conference on*. IEEE, 2000, pp. 3–10.

[21] G. Bell, J. Gray, and A. Szalay, "Petascale computations systems: Balanced cyberinfrastructure in a data-centric world," 2005.

[22] D. Cohen, F. Petrini, M. D. Day, M. Ben-Yehuda, S. W. Hunter, and U. Cummings, "Applying amdahl's other law to the data center," *IBM Journal of Research and Development*, vol. 53, no. 5, pp. 5–1, 2009.

[23] D. Zheng, A. Szalay, and A. Terzis, "Hadoop in low-power processors," *arXiv preprint arXiv:1408.2284*, 2014.

[24] J. D. McCalpin, "System performance balance, system cost balance, application balance and spec cpu2000/cpu2006 benchmarks," http://www.cs.virginia.edu/m̃ccalpin/SPEC_Balance_2007-06-20.pdf, 2007.

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

[26] 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.

[27] 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.

[28] P. K. Kondikoppa, A. K. Das, S. Goswami, R. Platania, and S.-J. Park, "Giga:giraph-based genome assembler for gigabase scale genomes," in *Paper accepted in proceedings of the 8th International BICob Conference*. ISCA, 2016, p. To be published.

[29] 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.

[30] 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.

Arghya Kusum Das is a PhD student in the Division of Computer Science and Engineering at Louisiana State University.
He received his bachelor's in computer science from West Bengal University of Technology, India.
His primary research focus is in the intersection of high performance big data applications and distributed-cyber-infrastructure.

Jae-ki Hong is Senior Engineer of Software R&D Center in Device Solution Business of Samsung Electronics. Since he joined Samsung in 2007 as a system software engineer, he has worked on various R&D projects from firmware to bigdata framework. Before he joined Software R&D Center in 2012, he developed the firmware of HDDs(Hard Disk Drives) for 5 years. He is currently a member of Next Generation Software Lab in Software R&D Center, focusing on Bigdata framework in bioinformatics.

Sayan Goswami is doctoral student in the Division of Computer Science at the School of Electrical Engineering & Computer Science at Louisiana State University. His primary research interests include frameworks for large scale genome assembly.
He received his B.Tech in Computer Science & Engineering from National Institute of Durgapur, India

Richard Platania is a Ph.D. student in the Division of Computer Science and Engineering at Louisiana State University. He is researching under Dr. Seung-Jong Park, with a focus in data- and compute-intensive scientific applications involving Big Data. His previous research experience includes large-scale molecular dynamics simulations over distributed cyberinfrastructures, de novo genome assembly, and deep learning. Prior to his Ph.D. studies, he received his B.S. degree in Computer Science at Louisiana State University.

Kisung Lee is an assistant professor in the Division of Computer Science and Engineering at Louisiana State University.
He received his bachelor's and master's degrees in computer science from KAIST and doctoral degree in computer science from the Georgia Institute of Technology.
His research interests lie in the intersection of big data and distributed computing systems.
He is also working on research problems in spatial data management and social network analytics.
He has been a recipient of the best paper awards of IEEE Cloud 2012 and MobiQuitous 2014.

Wooseok Chang is Vice President of Software R&D Center in Device Solution Business of Samsung Electronics. Since he joined Samsung in 1995 as a system software engineer, he has worked on various R&D projects from embedded systems to high-end servers. Before he joined Software R&D Center in 2012, he led SSD(Solid State Drive) Software Development group for three years, for which Samsung successfully launched a series of client SSDs. He currently leads Next Generation Software Lab in Software R&D Center, focusing on hyperscale converged solutions, big data analytics, and bio-processor platform.

Dr. Seung-Jong Park is an associate professor and a graduate advisor in the Division of Computer Science at the School of Electrical Engineering & Computer Science and the Center for Computation & Technology of Louisiana State University.
Dr. Park has studied interdisciplinary research areas Interdisciplinary research involving (1) distributed computing ranging from cloud computing over high speed optical networks to mobile computing over wireless networks; and (2) data intensive sciences analyzing Big Data at Gene Sequence Alignment, De Novo Assembly, and large-scale molecular simulation with cloud computing and high speed networking technologies. He has worked several major projects funded by NSF, GENI Project Office, Dept. of Defense (Office of Naval Research), Air Force Research Lab and Louisiana Board of Regents to develop cyber-infrastructure, network protocols and network architectures.

He received his Ph.D. from the School of Electrical and Computer Engineering at Georgia Institute of Technology, 2004. Prior to that, he had also received a B.S. degree in Computer Science at Korea University, Korea and a M.S. degree in Computer Science from KAIST (Korea Advanced Institute of Science and Technology), Korea in 1993 and 1995, respectively.

2015 IEEE International Conference on Big Data (Big Data)

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

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

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

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

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

I. Introduction

Since experimental facilities at large-scale sciences, such as biology, astronomy etc., have produced unprecedented amount of data, scientific communities encounter new challenges in terms of storage and optimal processing. The fundamental computation-model of these scientific applications is rapidly changing to address these challenges. Deviating from the traditional compute-intensive programming paradigm, e.g., MPI, etc., many HPC applications have started using the current state of the art big data analytic software, such as, Hadoop, Giraph, etc.

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

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

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

In this work, we use a large-scale de novo genome assembly as one of the most challenging and complex real world example of a HPC workload that recently made its way to the forefront of big data challenges [1] [2]. De novo genome assembly reconstructs the entire genome from fragmented parts called short reads when no reference genome is available. The assembly pipeline of our Parallel Genome Assembler (PGA) involves a terabyte scale short read data analysis in a Hadoop job followed by a complex large-scale graph analysis with Giraph, thus, serving as a very good example of both data- as well as compute-intensive workload.

In this paper, we present the performance result of PGA

978-1-4799-9926-2/15/$31.00 ©2015 IEEE

atop three different types of clusters as follows: 1) a traditional HPC cluster, called SuperMikeII (located at LSU, USA), 2) a regular datacenter architecture, called SwatIII (located at Samsung, Korea) and 3) a novel MicroBrick-based prototype architecture, called CeresII that uses PCIe based communication (also located at Samsung, Korea).

Our performance analysis is divided into two parts:

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

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

II. Related Work

Earlier studies [3] [4], as well as our prior experience [5], [6] show that state of the art big data analytic software can be useful for data-intensive HPC workloads. As a consequence, a growing number of codes in several scientific areas, such as bioinformatics, geoscience, etc., are currently being written using Hadoop, Giraph, etc. Despite the growing popularity of using these big data analytic software for scientific-computing, there are very limited prior works that evaluated different distributed-cyber-infrastructures for these software when applied for data-intensive scientific workloads.

Impact of individual hardware component on Hadoop-workload: There are several performance analysis studies on using different types of hardware to accelerate the Hadoop job using the existing benchmark workloads. Vienne [7] and Yu [8] evaluated the performance of Hadoop on different high speed interconnects, such as, 40GigE RoCE and InifiniBand FDR or QDR. In both the studies, the authors argued that InfiniBand produces better result than Ethernet.

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

All of the above studies have been performed either with existing benchmarks (e.g., HiBench [15]) or with enterprise-level analytic workloads, thus, they are unable to address the HPC aspect of Hadoop. Furthermore, very limited studies consider the in-memory graph processing frameworks (e.g., Giraph, etc.) even though, graph analysis is a core part of many analytic workloads.

Impact of overall architecture on Hadoop Workload: Michael [16] investigated the performance characteristics of the scaled out and scaled up architecture for interactive queries and found better performance using a scaled out cluster. On the contrary, Appuswamy [17] concluded that a single scaled up server performs better than a 8-nodes scaled out cluster for many different enterprise-level Hadoop workloads, e.g., log processing, machine learning, etc. Our study is significantly different in the following aspects. 1) Existing works focus on the data-intensive, enterprise-level Hadoop jobs (e.g., log processing, query processing, etc.). On the contrary, genome assembly is severely data- and a magnitude more compute-intensive. Additionally, it involves a large graph analysis which is extremely memory-intensive. 2) Existing works are limited in terms of job size. For example, the data size chosen in [17] can be accommodated in a single scaled up server. We did not put such a restriction on storage space or memory. Consequently, our performance comparison is more generic and realistic in a sense that, most of the time the choice of the cluster-size is driven by the data size rather than the performance.

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

In this section, we briefly describe the programming model of two popular big data analytic software: Hadoop and Giraph and discuss several issues that we observed on a traditional HPC environment while running these workload.

A. Programming models for big data analytic software

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

B. Issues: big data application over traditional-supercomputers

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

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

2) Storage issues: In a traditional supercomputing environment, each node is normally attached with only one HDD. This configuration puts a practical limitation on total number of disk I/O operations per second (IOPS). On the other hand, the big data applications that consider data locality, typically involve a huge volume of data read/write from/to the Direct-Attached-Storage (DAS) of the compute nodes. Therefore, the applications might suffer from I/O wait. Although some variations of Hadoop (e.g., [18]) are optimized to read/write large volume of data from/to other parallel file systems (e.g., Lustre and GPFS), thus taking advantage of huge amount of IOPS available through the dedicated I/O servers, the performance can be severely constrained by the network bottleneck. Additionally it will incur extra cost to the cluster. For simplicity, in this work, we use the HDFS as the distributed file system and use the local file system for the shuffled data.

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

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

IV. Evaluation Methodology

A. Experimental Testbeds

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

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

Table I: Experimental testbeds with different configurations

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

Table II: Hardware components used in different cluster configurations, and their cost. Price information is collected from http://www.newegg.com and http://www.amazon.com. The minimum listed price is considered as per Jun 17, 2015.

and the regular datacenter. We always use homogeneous configuration across any cluster. We reported the performance and the price of different clusters in terms of the Hadoop datanodes (DN) only. In the subsequent sections we use the term node and datanode interchangeably.

Table-II shows the hardware specification of each cluster and their cost. We calculated the cost of each node of each cluster configuration (shown in Table-I) based upon this. We use the hardware configuration of SuperMikeII as the baseline and compare all the performance results of SwatIII and CeresII to this baseline. Each node of SuperMikeII and any SwatIII variants has the same number of processors and cores, in particular, 2 8-core Intel SandyBridge Xeon 64bit E5 series processors. To do a fair comparison, we disabled the HyperThreading in the SwatIII as SuperMikeII does not have it.

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

Unlike SuperMikeII or SwatIII-Basic and -Memory which use only one DAS device per node, SwatIII-FullScaleup-HDD/SSD and SwatIII-Medium-HDD/SSD use more than one DAS device (Either HDD or SSD as the names suggest) per node. They also vary in terms of total amount of memory per node. However, the total amount of storage and memory space is almost same across all these clusters. We use these clusters to evaluate different types architectural balances from the viewpoint of scaled out and scaled up configurations. In case of SwatIII, we use the term scaled up and out in terms of amount of memory and number of disks. The numbers of cores per node is always same. In either of SwatIII-FullScaleup and Medium, we use JBOD (Just a Bunch Of Disks) configuration as per the general recommendation by [19], Cloudera, etc. JBOD does not limit the I/O speed by the slowest disk as RAID (Redundant Array of Independent Disk) configuration. As a result, JBOD is found to perform 30% better than RAID-0 in case of HDFS write throughput [19].

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

several PCIe (PCI-express) ports. All the compute servers of CeresII in a single chassis are connected to a common PCIe switch to communicate with each other. The highly dense servers per chassis in CeresII have a total 44 physical cores connected through PCIe comparing to 16 physical cores per node in SuperMikeII and SwatIII. Furthermore, the use of SSD reduces the I/O wait and 8GB RAM per core improves the access parallelism.

B. Understanding the workload

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

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

2) Giraph-based Graph Simplification (memory- and compute-intensive workload): This stage consists of a series of memory-intensive Giraph jobs. Each Giraph job consists of three different types of computation: compress linear chains of vertices followed by removing the tip and then the bubble structure (introduced by sequencing errors) in the graph. Giraph master-vertex class invokes different types of computation based on a superstep counter. The computation for compression proceeds in rounds of two supersteps until a user defined $limit$ is reached. In one superstep, each compressible vertex with only one incoming and outgoing edge is randomly labeled with either $head$ or $tail$ and send that label to its immediate predecessor. In the next superstep, all the $head$-$tail$ links are merged, that is, the $head$-$k$mer is appended with the last character of the $tail$-$k$mer and the $tail$ vertex is removed. Each vertex also maintain a frequency counter which increments after each append. Tip and Bubble removal both takes two supersteps. Again, the first superstep identifies the potential tip or bubble structures based on the number of incoming and outgoing edges of each vertex and send a message with the vertex information to their predecessors. Each predecessor in the next superstep analyzes the message(s) to determine the length of its successor(s), retrieve their frequency, compute the commonality between them using edit-distance algorithm (only in case of bubble) and finally delete the erroneous successor based upon predefined criteria.

	Job Type	Input	Final out-put	# jobs	Shuffled data	HDFS Data
Graph Con-struction	Hadoop	90GB (500-million reads)	95GB	2	2TB	136GB
Graph Simpli-fication	Series of Gi-raph jobs	95GB (71581898 vertices)	640MB (62158 ver-tices)	15	-	966GB

Table III: Moderate-size bumble bee genome assembly

	Job Type	Input	Final out-put	# jobs	Shuffled data	HDFS Data
Graph Con-struction	Hadoop	452GB (2-billion reads)	3TB	2	9.9TB	3.2TB
Graph Simpli-fication	Series of Gi-raph jobs	3.2TB (1.5-billion vertices)	3.8GB (3-million ver-tices)	15	-	4.1TB

Table IV: Large-size human genome assembly

C. Input Data

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

D. Hadoop configurations and optimizations

Since our goal is to evaluate the underlying hardware components and their organizations, we avoid any unnecessary change in the source code of Hadoop or Giraph. We use Clouera-Hadoop-2.3.0 and Giraph-1.1.0 for the entire

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

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

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

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

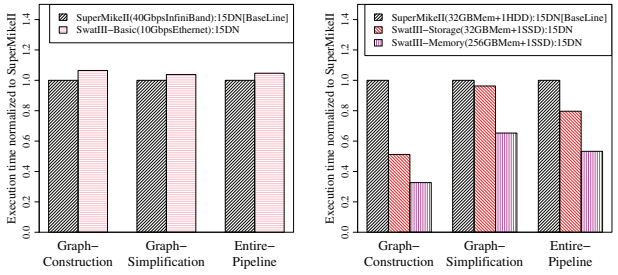
Amount of memory per container and Java-heap-space: In each node in any node of any cluster, we kept 10% of the memory for the system use. The rest of the memory is equally divided among the launched containers. The Java heap space per worker is always set to lower than memory per container as per normal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

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

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

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

V. Impact of Different Hardware Component



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

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

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

In this section, we compare the individual impact of each hardware component, such as, network, storage, and memory individually on our benchmark genome assembler. To do that, we use 16 nodes in both SuperMikeII and SwatIII. Each node in both the clusters has 16 processing cores. We started with comparing the impact of the network between SuperMikeII and SwatIII-Basic-HDD. Then, we further optimized those 16 nodes of SwatIII cluster incrementally in terms of storage by providing SSD (named as SwatIII-Basic-SSD) and then providing more memory to each node (named as SwatIII-Memory). The execution-times reported in this section are the means of at least 3 runs of the assembler on each different hardware configuration.

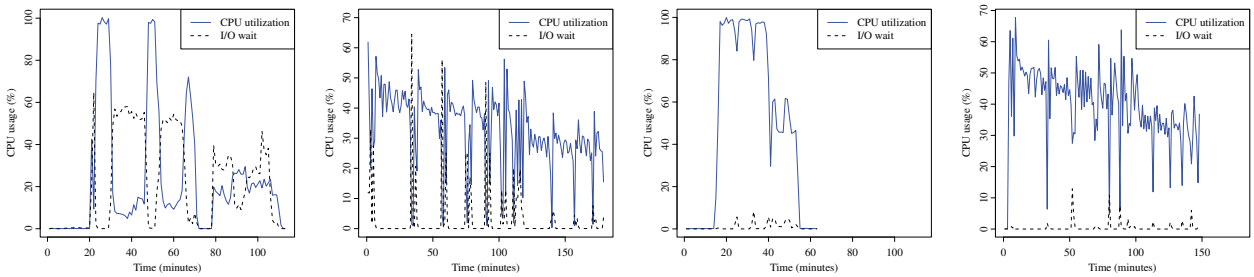
A. Effect of Network: InfiniBand vs Ethernet

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

B. Effect of local storage device: HDD vs SSD

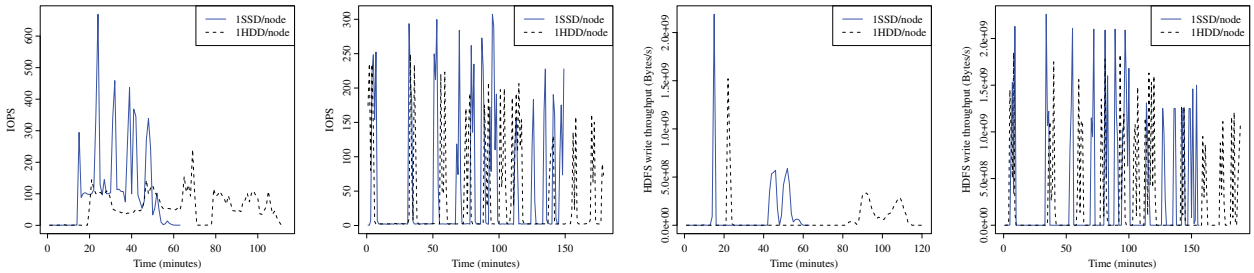
Figure-1b compares the execution time of SwatIII-Basic-SSD to the SuperMikeII-baseline. The second column of each stage of the assembler in Figure-1b shows the impact of using SSD in that stage of the assembly. We observed almost 50% improvement in the shuffle intensive graph-construction stage because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

Actually, the shuffle phase of Hadoop experiences maximum I/O wait when a many I/O threads work concurrently to spill and subsequently read huge data to/from the disk. This I/O wait is significantly reduced using SSD (Figure-2c and -2d) resulting in remarkable performance improvement in Hadoop. However, for Giraph we did not observe any notable improvement using SSD because of very less I/O wait. Basically, an SSD increases the disk IOPS per DataNode by 7 to 8 times than an HDD improving the shuffle phase's CPU utilization as shown in Figure-3a. In case of Giraph, the corresponding improvement is 1.5-times as shown in Figure-3b. Considering the I/O throughput to HDFS, we also observed 1.5-times improvement in case of SSD for both Hadoop and Giraph as shown in Figure-3c and -3d. Giraph, which writes data to the HDFS only, shows the similar I/O characteristics for both IOPS per DataNode (DN) and HDFS I/O throughput because they are related by the equation: $HDFS\_IO\_Throughput = IOPS\_per\_DN \times Bytes\_per\_IO \times \#DN$, where $Bytes\_per\_IO$ is the characteristics of the disk. However, for Hadoop, these charac-



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

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



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

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

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

C. *Effect of size of DRAM*

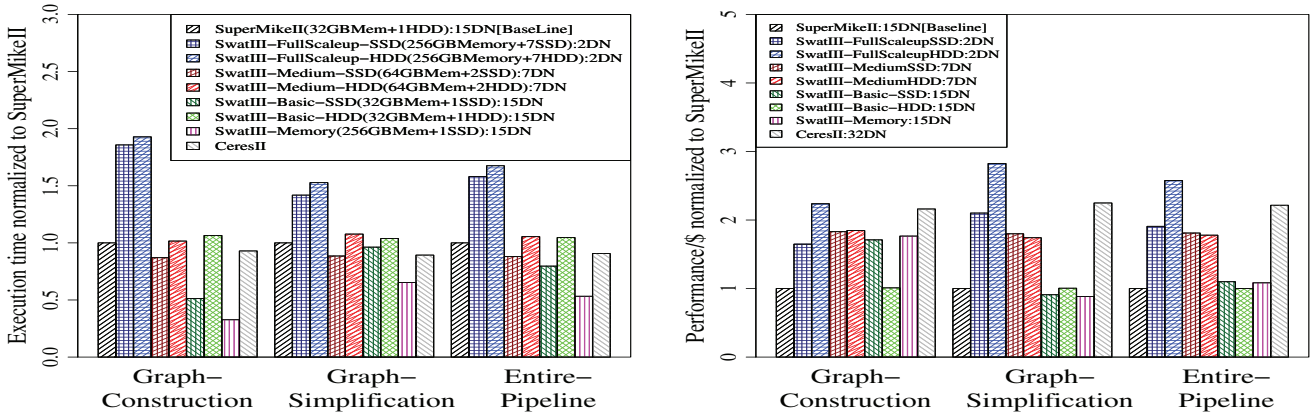
The third columns of Figure-1b shows the impact of increasing the amount of memory per node. We observed almost 20% improvement in the initial graph-construction phase from SwatIII-Basic-SSD, i.e., almost 70% improvement to the baseline. In the Giraph phase, the corresponding improvement is 35%. The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

VI. Impact of Different Hardware Organization

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

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

Figure-4a shows the relative merits of different cluster architectures in terms of raw execution time. Observe that we always keep the total aggregated storage and memory space almost same across all the clusters (Except the SwatIII-Memory). The basic assumption behind this experimental setup is that the total amount of data should be held in its entirety in any of the cluster. The observations are as follows: *1*) **SwatIII-Basic (16-DNs):** As discussed earlier in Section-V, for Hadoop, the SSD variant of this scaled out cluster shows 2x speedup over the baseline whereas the HDD variant performs similar to the baseline. For Giraph, both of them perform similar to the baseline. *2*) **SwatIII-FullScaleup (2-DNs):** This scaled up small sized cluster takes the maximum time for any workload because of least number of processing cores. Observe that for the Hadoop job, both SSD and HDD variants of this scaled up configuration perform similarly, which is in contrast with scaled out SwatIII-Basic. We discuss it in more detail in Section-VI-D. *3*) **SwatIII-Medium (7-DNs):** The HDD variant of this cluster perform similar to the baseline for both Hadoop and Giraph even though the total number of cores in the cluster is half of the baseline. It is because 2-HDDs and 64GB RAM per node increase the IOPS and the caching respectively. The SSD variant shows slightly better result than the HDD because of further increase in IOPS. *4*) **SwatIII-Memory (16-DNs):** It is no surprise that this configuration shows the



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

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

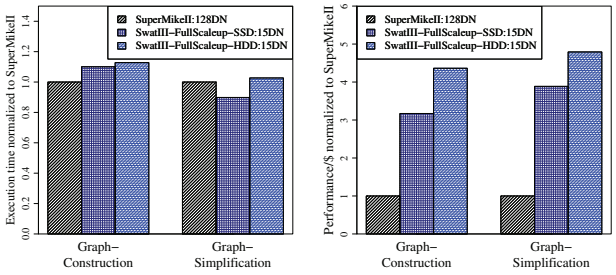
lowest execution time among all because of the maximum resource availability.

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

We consider the performance as the inverse of the execution time and divided it by the total cost of the cluster to get the performance/$. Because of similar total memory and storage space across clusters none of them gets price benefit over other in terms of storage and memory. Rather, we compare the performance to price from the view point of a proper architectural balance among number of cores, number of disks, and amount of memory per node. We did not consider the cost of network for a fair comparison with SuperMikeII, the public HPC cluster that is shared among many users.

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

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



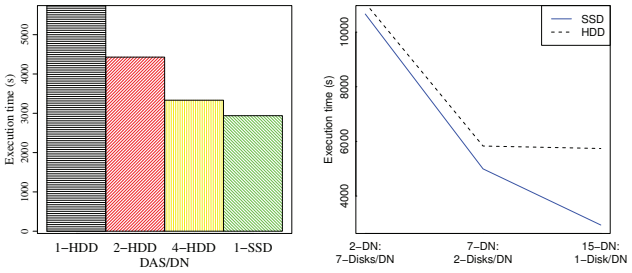
(a) Execution time (Lower execution time is better). (b) Performance/$ (Higher performance/$ is better).

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

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

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

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



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

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

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

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

Figure-6a compares the performance of a single SSD and increasing number of HDDs per node for the Hadoop stage of the bumble bee genome assembly using 15-DNs. The performance improves almost linearly by increasing the number of HDDs per node in the cluster. On the other hand, 4-HDDs per node shows similar performance (only 5% variation) with a single-SSD per node. At this I/O thoughput the disk controller saturates, and adding more disk(s) to the nodes does not improve the performance. Both Figure-6b as well as Figure-5a substantiate our claim for moderate-size bumble bee and large-size human genome data. For both the data, both the HDD and SSD variant of SwatIII-FullScaleup perform similarly where each DN is equipped with 7-disks. However, SSD showed significantly better performance and scalability than HDD when we scaled out by adding more compute nodes to the cluster.

E. Performance of CeresII: Samsung MicroBrick with PCIe communication

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

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

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

VII. Conclusion and Future-work

In this paper, we analyze the performance characteristics of two popular state of the art big data analytic software, Hadoop and Giraph, on top of different distributed-cyber-infrastructures with respect to a real world data- and compute-intensive HPC workload. We pointed out several limitations in a traditional HPC cluster, both, in individual node layer (e.g., memory and storage) as well as network interconnect layer. The novel MicroBrick-based CeresII-cluster with low-power but high-density compute nodes connected with PCIe-based communication interface is a good future direction to alleviate many of the existing architectural limitations.

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

ACKNOWLEDGMENT

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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