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.

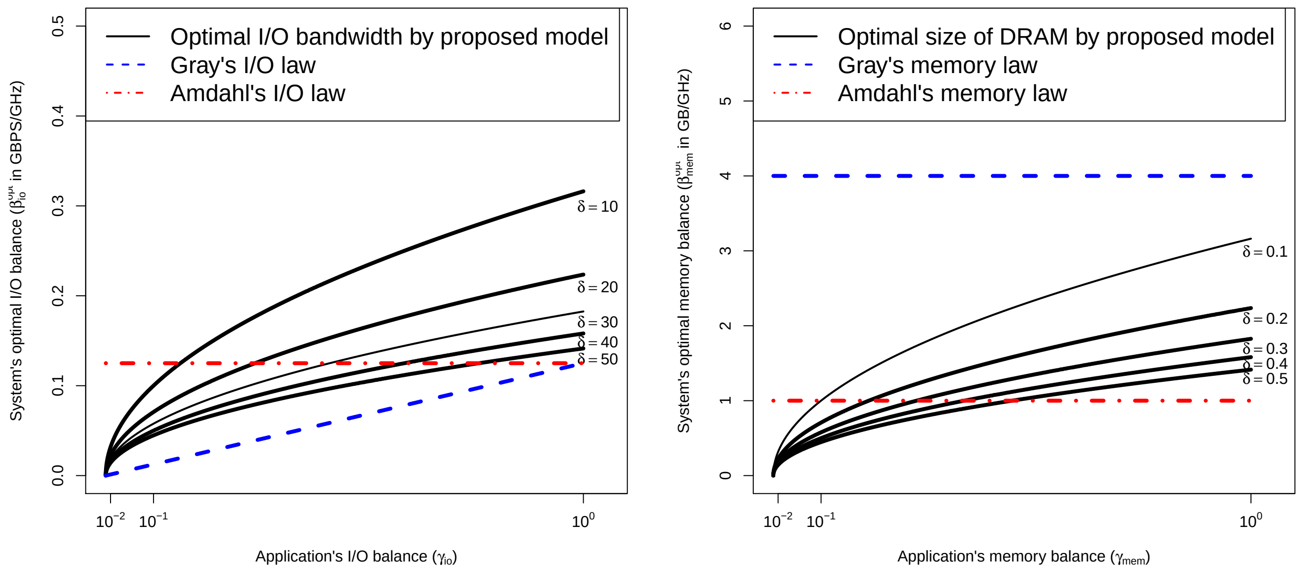
IV. Proposed Model for System Balance

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:

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

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

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

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 Bandwidth /Disk (GBPS)	Total Seq I/O Bandwidth (GBPS) /node	DRAM /node (GB)	Maximum #Nodes available	β_{io}	β_{mem}
SuperMikeII (Traditional Supercomputer)	Two 8-core SandyBridge Xeon	2.6	16	41.6	1-HDD (SATA)	0.15	0.15	32GB	128	0.003	0.77
SwatIII (Regular 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

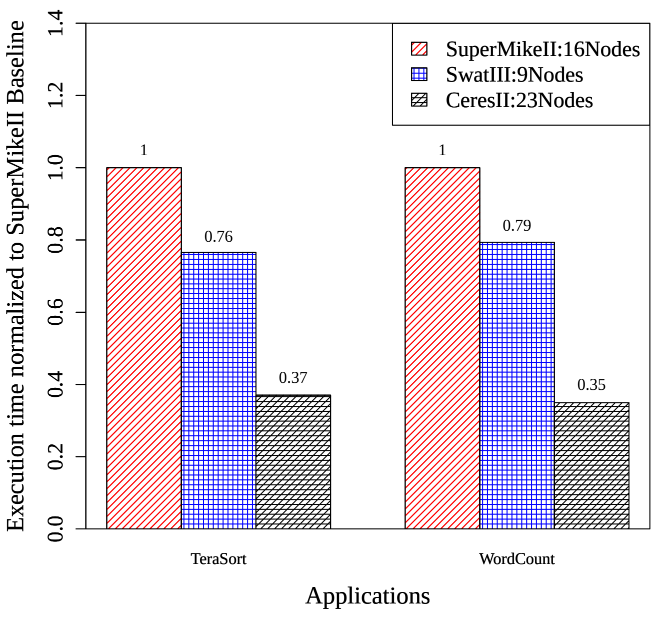
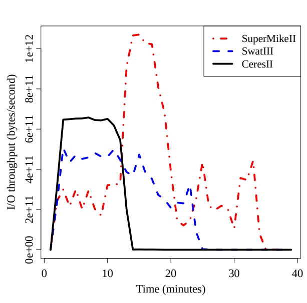


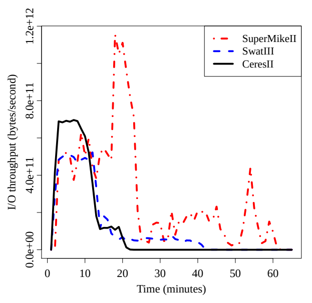
Fig. 2: Execution time of TeraSort and WordCount over different cluster architecture keeping the total cost of each cluster same

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:

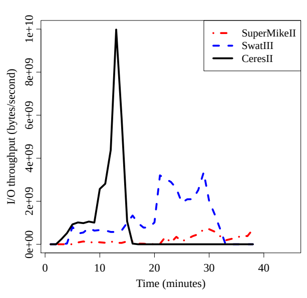
1) Comparing to the SuperMikeII baseline, for both TeraSort and WordCount benchmark application, CeresII shows almost 65% improvement.
2) Comparing to SwatIII, CeresII shows almost 50% improvement in execution time for TeraSort and WordCount.



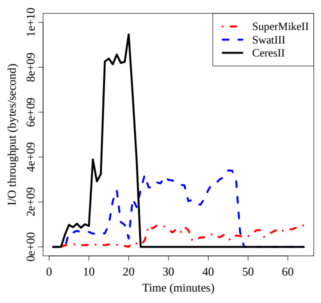
(a) TeraSort IPS



(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.

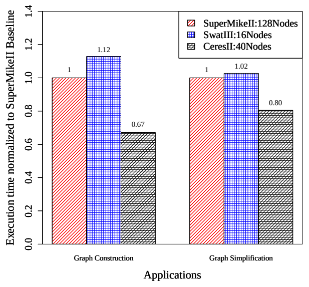
For both TeraSort and WordCount, the peak IPS of SuperMikeII is significantly higher than both SwatIII and CeresII. However, the average I/O bandwidth of SuperMikeII is significantly less than the other two, leading to significantly high I/O wait that results in extremely lowe average IPS (lower than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

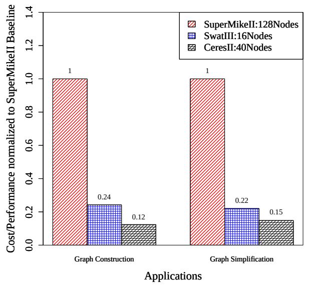
CeresII cluster is optimally balnced in terms of CPU speed and I/O bandwidth with a β_{io} value of 0.17. Figure-3a and 3b shows that CeresII achieves higher Peak IPS than SwatIII but less than SuperMikeII for both the TeraSort and WordCount benchmark. On the other hand, CeresII achieves significantly 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.