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

Arghya Kusum Das, Jaeki Hong, Kisung Lee, Seung-Jong Park, Wooseok Chang

***Abstract*—The I/O- and memory-bound nature of data-driven scientific applications are posing several challenges to next generation HPC system designers as they strive for system balance with respect to processor speed, I/O bandwidth and memory while also maintaining a balance between performance and economy. Augmenting Amdahl's design principles for balanced system, collectively known as *Amdahl's second law* in the context of current big data analytic software (e.g., Hadoop, Giraph, etc.) and recent advances in hardware is relevant to design cost efficient HPC infrastructure for data-intensive scientific applications. This paper proposes a simple, easy to use, general purpose, additive model for cost/performance to quantify and optimize the system balance (among CPU, I/O bandwidth and size of DRAM) in terms of both software characteristics as well as the current hardware cost. Our performance model reveals that a balanced HPC system needs almost 0.25-GBPS I/O bandwidth, and almost 4 GB of DRAM per GHz of CPU speed considering Intel Xeon micro architecture which is widely used by the HPC system designers. However, our model can be used for any given architecture. To the best of our knowledge, this is the first attempt to theoretically extend Amdahl's second law considering more degrees of freedom, such as, workload characteristics and hardware cost.**

We validated our result by evaluating three fundamentally different cluster architecture, 1) a traditional HPC cluster called SupermikeII, 2) a regular datacenter called SwatIII, and 3) a novel micro brick based hyperscale system called CeresII. CeresII has 6-Xeon cores each running at 2GHz, 1-NVMe SSD with 2-GBPS I/O bandwidth and 64GB DRAM, thus closely resembles to the optimum produced by our model. We evaluated these three clusters with respect to two widely used Hadoop-benchmark (e.g., terasort and wordcount) as well as our own genome assembler based on Hadoop and Giraph which serves as a very good real world example of I/O-, compute- and memory-intensive workload. CeresII outperformed all other architecture for all the benchmark in terms of both execution time and cost/performance while using same processing power as the other two and significantly less memory than SwatIII.

***Index Terms*—Balanced HPC System, Amdahl's Second Law, Data-driven Science, Hadoop, Giraph.**

I. Introduction

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

As a consequence, most of the existing HPC systems 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. Furthermore, the system characteristics of the data-intensive scientific applications varies tremendously, especially because of varying nature and volume of data. This workload diversity also indicates huge potential for improved efficiency via balanced system designs. Hardware vendors as well as cloud vendors (e.g., Amazon, Google, R-HPC, etc.) are investing huge amount of money for this. Millions of dollars are being spent for programs such as, NSF Cloud, XEDE, etc. where several industries and universities collaborate to find such balanced architectures to drive the next generation cloud research.

At this inflection point of HPC landscape, the next generation system designers must consider more degrees of freedom for cluster architecture 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 the performance and cost? These complex performance and economic factors together motivates new design of HPC infrastructure. However, these factors together impose 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. There is limited studies which address the challenges in developing a cost effective, balanced distributed cyber infrastructure to analyze large scale scientific big data.

With this motivation, this paper takes an initial attempt to theoretically augment Amdahl's design principles for balanced system (collectively known as Amdahl's second law) considering the workload characteristics as well as the price of hardware components. It proposes a simple additive model for cost/performance to quantify the system balance between

A. K. Das, K. Lee, and S.J. Park are with the Department of Computer Science and Electrical Engineering, Center for Computation and Technology Louisiana State University, Baton Rouge, USA

J. Hong and W. Chang are with Samsung Electronics, S. Korea

CPU speed, I/O bandwidth, and size of DRAM in terms of software application characteristics and cost of hardware. 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. However, the outcome of the model (i.e., configuration of an optimal balanced system) indicates the use of solid state drives (SSD) instead of hard disk drive (HDD) in a multicore machine. 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 0.2-GBPS I/O bandwidth (x% variation from original Amdahl's law), and almost 4-GB of DRAM (y% variation from original Amdahl's law) per GHz of CPU speed in the crrent scenario using Intel Xeon micro architecture which is widely used by the HPC system designers. The variation is coming because of the cost component which was ignored in Amdahl's second law in its pristine form.

We validated our model by evaluating three fundamentally different cluster architecture as follows: 1) a traditional HPC cluster, called SuperMikeII (located at LSU, USA) that offers 382 computing nodes each with 16-Xeon cores, 1-HDD, and 32-GB DRAM, 2) a regular datacenter architecture, called SwatIII (located at Samsung, Korea) that has 128 nodes each with 16-Xeon cores, 4-HDD, and 256-GB DRAM and 3) a new MicroBrick-based architecture, called CeresII (also located at Samsung, Korea). Each CeresII node has 6-Xeon cores, 1-NVMe SSD with 2-GBPS I/O bandwidth and 6-GB DRAM per node, thus closely resembles to 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 (e.g., 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 the three benchmark evaluation, CeresII, with the most optimum configuration outperformed the others in terms of both performance and cost/performance when using same amount of processing cores as SuperMikeII and SwatIII across the cluster, and significantly less (almost half) DRAM than SwatIII.

II. Related Work

Numerous studies have been performed evaluating the performance implication of different big data analytic software on different types of hardware infrastructure. We categorize these existing studies into three different classes: First, experimental evaluation of different cluster architecture for big data software. Second, simulation and analytical performance modeling of big data software. Third, system characterization using Amdahl's second law which is the most relevant to our current effort. In this section we discuss the previous works done in these three categories.

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

Kang [1] compared the execution time of sort, join, WordCount, Bayesian, and DFSIO workloads using SSD and HDD and obtained better performance using SSD. Wu [2] found that Hadoop performance can be increased almost linearly with the increasing fraction of SSDs in the storage system. They used the TeraSort benchmark for their study. Additionally, they also showed that in an SSD-dominant cluster, Hadoop's performance is almost insensitive to different Hadoop performance parameters such as block-size and buffer-size. Moon [3] showed a significant cost benefit by storing the intermediate Hadoop data in SSD, leaving HDDs to store Hadoop Distributed File System (HDFS [4]) data. They also used the TeraSort benchmark in their study. A similar result can be found in the study by Li [5] and Krish [6] where SSDs are used to store temporary data to reduce disk contention and HDDs are used to store the HDFS data. They all reached the same conclusion as Moon [3]. Tan [7] also reached the similar conclusion for two other workloads including a Hive workload and an HBase workload.

Michael [8] investigated the performance characteristics of the scaled out and scaled up architecture for interactive queries and found better performance using a scaled out cluster. On the other hand, Appuswamy [9] reached an entirely different conclusion in their study. They observed a single scaled up server to perform better than a 8-nodes scaled out cluster for eleven different enterprise-level Hadoop workloads including log-processing, sorting, Mahout machine learning, etc. However, these studies do not provide any quantitative measure for scale up or scale out.

However, an older study by Kung [10] can shed light on feasibility of the scaling up approach. Kung analyzed several HPC problems and theoretically showed that the memory needs to be increased exponentially to restore the balance of the system when the number of processors is increased by a certain factor. For example, best known external sorting algorithm needs the local memory to be increased by an exponent of α when number of processors are increased by a factor of α (i.e., $M_{new} = M_{old}^{\alpha}$) to restore the balance of the system. Although this study is not directly related to this paper, some of the problems considered in [10] constitutes the core part of today's big data analytic framework. For example, Hadoop uses the similar external sorting in its shuffle phase. Hence, from the perspective of balanced system design arbitrary scaling up the size of DRAM is possibly not a feasible option. Rather, the amount of RAM should be decided based upon cost.

B. Simulation and analytical performance modeling of big data software

Simulation is widely used for predicting performance of different big data software (mainly Hadoop) and analyzing design trade-offs in a large number of hardware domains. At the border level, all these simulators generate fake Hadoop MapReduce jobs and tune different hardware and software parameters in a simulated environment mostly using prior experiences or trial-and-error to obtain optimize the performance of some parts of the Hadoop framework or a given Hadoop job. For example, HSim [11] mainly focused on the Hadoop job parameters to optimize the performance of the entire job. SimMR [12] on the other hand focus on simulating the Hadoop task scheduling algorithms. MRSim [13] and MRPerf [14] 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 hardware alternative, thus capable to save huge cost comparing to experimental evaluation on real hardware discussed before. However, most of these simulators comes 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 broad range of available hardware infrastructure with more 200 Hadoop parameters is challenging and most of the time unreliable.

Analytical (or, theoretical) models are faster than simulation as they abstract away several performance tuning parameters both at software and hardware level. Thus, analytical model can work as early performance indicator of a given big data analytic application and can be used to evaluate alternative hardware configurations. Most of these anlaytical model use single or multi layer queuing network to predict the performance of a given Hadoop job. For example, Viana [15] model the pipeline parallelism of a Hadoop job using queuing network to predict the performance of the job. Wu [16] proposed a layered queue network model to predict the performance of a Hadoop job in a cloud environment. In a recent work Ahn [17] proposed a queuing theory model to predict the performance of Hadoop job atop different storage technologies, i.e, HDD and SSD. Krevart [18] computed the I/O complexity of the Hadoop job for data-intensive applications and proposed a model to quantify the hardware resource wasted by Hadoop. They (Krevart) shouted for improving the big data analytic software.

C. System characterization using Amdahl's second law

Computer scientist Gene Amdahl postulated several design principles for a balanced system design collectively known as Amdahl's Second Law. He stated a balanced system needs 1-bit of sequential I/O per second (Amdahl's I/O number), and 1-byte of memory (Amdahl's memory number) per CPU instruction per second. Jim Gray [19] in 2000 revised Amdahl's second law by suggesting to measure the instruction rate and I/O rate on the relevant workload. We will discuss it in more details later in Section-IV.

Bell and Gray [20] also classified the existing supercomputers based upon Amdahl's second law to clarify the future roadmap of the HPC architecture. Chang [21] used Amdahl's second law to better understand the design implications of data analytic systems by quantifying workload requirements. Cohen [22] applied Amdahls second law to study the interplay between processor architecture and network interconnect in a datacenter.

Szalay [23], used Amdahl's I/O and memory numbers to propose an energy efficient balanced cluster architecture Amdahls bananced blades. [23] used SSD and low power processors (such as, Intel Atom, Zotac etc) to achieve the system balance. 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) respectively by CPU-speed (GHz). The cluster architecture was balanced as the simplified Amdahl's I/O number was close to unity. In a recent study [24] Zheng evaluated a similar architecture as Amdahl's balanced blade for Hadoop applications and showed Atom processor is the system's bottleneck

III. Motivation: Architectural (Im)balance in Traditional HPC Cluster and Recent trend in Hardware

A. Hadoop and Giraph Programming Model

Hadoop was originated as the open-source counterpart of Googles MapReduce [18]. Hadoop read the input data from the underlying Hadoop Distributed File System (HDFS) in theform of disjoint sets or blocks 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. According to the programming model, users need to provide only the map and reduce function. The data flow characteristics is similar for all other phases across all jobs.

B. Network Architecture

Traditional HPC clusters uses a hierarchy of switches with a fat tree model of connectivity. This approach typically uses layered architecture. The hosts or servers are connected to the bottom layer, called access layer which is then aggregated to an intermidiate 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 got progressively higher from access layer to leaves to spine starting from only few Mbps to several Gbps. This architecture may suffer from bottleneck issues thereby introduces architectural imbalance in the modern datacenter or cloud infrastructures as the bandwidth of the host adapter is increasing in an outstanding pace (an observation by Cohen [24]). 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 of Hadoop or Giraph, which transmits huge amount of data over network in different phases of computation, can use more bandwidth from this all-to-all connection model.

C. Storage Architecture

The performance of any system is traditioanlly constrained by the I/O subsystem. So is the traditional HPC cluster which are normally equipped with only one hard disk drive. Over the span of last 10 years the the hard disk speed is improved by only twice from 7000RPM to 15000RPM. Consequently, the I/O throughput increased from 80MB/s to 150MB/s. At this rate, to read just a 1TB hard disk takes almost two hours. In real world scenario, where a huge amount of data is read/write from/to the local disk the performance may be even worse. So, in many state of the art data center cluster and cloud infrastructure the data is stripped across multiple smaller disks to improve the I/O throughput.

Alternatively, the SSDs can be used. It uses the similar flash memory that is used in memory subsystem, thus increases the per-disk I/O throughput by almost 4-times than an HDD. Current SSDs offer almost 550MB/s I/O throughput in a moderate cost while maintaining considerable storage capacity. Due to the high bandwidth of SSDs, the most intuitive approach to build a high performance cluster is to use multiple SSDs with high end processors. However, the disk controllers are found to

D. Memory Architecture

With the introduction of DDR (Double Data Rate) technology the memory bandwidth has been improved significantly because it reads one word of data during the positive edge and one word during the negative edge of the processor clock pulse. Most of the current HPC cluster as well as the computation clusters use either DDR2 or DDR3 RAM (random access memory or, simply main memory) modules. There is hardly any difference among current clusters (including traditional HPC clusters and datacenters) in terms of memory architecture or processormemory communication model.

However, for improved performance, suffcient memory should be provided to the cluster that can hold the required computation in conjunction with the big data. Furthermore, more memory improves the caching effect which again improves the applications performance. However, the main memory is costly which makes the job of building the balanced system complicated. Again, the SSD is nowadays increasingly considered as a new layer between memory (Since they use similar flash arrays as RAM) and storage due to its increased I/O throughput which may change the designers approcah towards a balanced system in near future. There are several important reasons to model the performance for a cloud application on a given processor architecture using analytical approach. The most common one is evaluating alternative hardware configurations parameters to estimate performance for a given cloud application without simulation. The performance estimation model aids processor architects in the designing and fine-tuning of future processors.

IV. Amdahl's Second law, Gray's Ammendment, and Limitations

A. Original Form of Amdahl's Second Law

Computer scientist Gene Amdahl postulated several design procniple in late 1960 to make a balanced system. As mentioned earlier, these design principals are collectively known as Amdahls second law which are as follows:

1) Amdahls 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 Amdahls 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) Amdahls memory Law: A balanced computer system needs one byte of memory per instruction per second. IFrom this point we will mention this law as Amdahls memory number.

Using the notations in Table-I the original Amdahl's I/O and meory number can be expressed as:

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

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

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

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

1) Revised Amdahls I/O law: A system needs 8 MIPS/MBpsIO, but the instruction rate and IO rate must be measured on the relevant workload. (Sequential workloads tend to have low CPI (clocks per instruction), while random workloads tend to have higher CPI.)
2) Revised Amdahls memory law: Alpha (the MB/MIPS ratio) is rising from 1 to 4. This trend will likely continue.

The underlying implication of the Gray's first ammendment (i.e., Revised Amdahl's I/O Law) is that, it aims for systems with high Amdahl I/O numbers at a given performance level that match the Amdahl I/O numbers of the applications. Regarding Amdahl memory number, in stead of any theoretical justification, Gray put forward a statistics reflecting the contemporary state of cluster architecture.

Using the notations in Table-I Gray's ammendments to original Amdahl's second law can be expressed as follows:

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

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

C. Limitations of Existing Laws and Current Problem Definition

Amdahl's second law for balanced system does not consider the impact of application balance (or, applications' resource requirement) on cluster architecture. Because of the diverse resource requirements, one-size-fit-all design as suggested in the original law (expressed as the constants in the right hand

TABLE I: Notations and their meaning

R_{cpu}	CPU speed of a system S (GHz)
R_{io}	I/O bandwidth of the system S (GBPS)
R_{mem}	DRAM size of the system S (GB)
W_{cpu}	Fraction 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, compute-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, compute-intensive
δ_{io}	Technology-Cost balance of between CPU and I/O bandwidth for system S $(= P_{io}/Pcpu)$
δ_{mem}	Technology-Cost balance between CPU and DRAM for system S $(= P_{mem}/Pcpu)$

side of Equation-1 and 2) cannot satisfy the different resource balance ratios for a collection of analytic applications.

Gray's ammendment to Amdahl's second law is more realistic in the sense that it consider the impact of application balance (or, applications' resource requirement) on the cluster architecture. However, it is limiting to reflect the interplay between application balance and technology-cost balance. The cost of hardware components has already changed the performance point and will keep on changing as the technology advances. Hence, we need a theoretical model for balanced cluster architecture considering both the application characteristics for optimal performance as well as the economy.

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 technology-cost balance (i.e., δ_{io} and δ_{mem} in Table-I). Mathematically it can be written as:

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

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

V. Proposed Model for System Balance: Augmenting Amdahl's Second Law

A. Model Assumptions

The model first assumes the same CPU micro architecture. The practical implication of this assumption is that, the optimum system balance (i.e., β_{io} and β_{mem}) may vary with different CPU microarchitectures such as, Intel Atom and Xeon, etc. However, the system designers can use this model repeatatively for different micro architectures without any loss of generality. On the other hand, the processor micro architecture does not change as frequently as their speed (refer to Intel's *Tick* and *Tock* model for release of new processor technology). Thus, the proposed model significantly reduces the search space for the designers of balanced system.

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

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

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

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

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

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

B. Model Derivation

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

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

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

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

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:

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

Partially differentiating with respect to β_{io},

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

To find the optimum balance (β_{io}^{opt}) between CPU speed and I/O bandwidth, Equation-15 should equal to 0. Then, solving for β_{io} and replacing with the workload and 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 reveals a convincing physical interpretation for contribubutions of application balance and technology-cost balance to system balance. Given a CPU micro architecture, these two equations provide the required I/O bandwidth and size of DRAM respectively to maintain the system balance for different types of applications while considering the price trend of different hardware components, such as, CPU, disk drives and DRAM.

C. Some Useful Implications

For an improved processor micro architecture operated at same core speed (such as, Intel Haswell comparing to Intel Xeon), the cost is likely to be higher. That is, δ_{io} and δ_{mem} will decrease with a corresponding increase in β_{io} and β_{mem}, thus driving the need for more I/O bandwidth and DRAM size. However, the growth is limited by the square root.

It should be noticed that the model did not consider sequential and random I/O bandwidth separately. Designers need to be careful about the application characteristics. An application with many 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 number of random I/O in its shuffle phase. To provide such a huge random I/O bandwidth, SSD can be used.

D. 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 do that, we chose Intel Xeon as the processor micro architecture. To reflect today's data-, compute-, and memory-intensive scientific application, we consider the work done by CPU, I/O and memory subsystem (i.e., W_{cpu}, W_{io}, and W_{mem}) is 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 the price of different hardware components for different hardware vendors. Column *Unit Price* shows the current price trend for mostly used processor, storage and memory technologies in their corresponding unit. For this example, we consider Intel Xeon micro architecture. As it can can be seen in Table-II, the cost per MBPS sequential I/O for both SSD and HDD is almost similar irrespective of change in storage technology provided the same storage space per disk. Whereas, the cost per GB of DRAM is increased almost double from DDR2 to DDR3. We calculated the average cost per GBPS of I/O bandwidth, and average cost per GB of DRAM respectively and divided it with cost per GHz of CPU Speed to get the technology-cost balance. Using the notation in Table-I technology-cost balance can be written as:

$$\delta_{io} = 13.57 \tag{19}$$

$$\delta_{mem} = 0.08 \tag{20}$$

TABLE II: Cost of different hardware components

Hardware components	Cost($)	Unit Price
Intel SandyBridge Xeon 64bit Ep series (8-cores) processor	1766	$82/GHz
Western Digital RE4 HDD (120MBps)	132	$1.1/MBPS
Western Digital VelociRaptor HDD (150MBPS), 500GB	157	$1.04/MBPS
Samsung 840Pro Series SATAIII SSD (400MBPS), 500GB	450	$1.12/MBPS
Samsung DDR3 16GB memory module	159	$10/GB
32GB 1600MHz RAM (decided by Dell)	140	$4.37/GB

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:

$$\beta_{io}^{opt} = 0.24 \tag{21}$$

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

$$\beta_{mem}^{opt} = 3.75 \tag{22}$$

VI. Experimental Evaluation

A. Experimental Testbeds

As mentioned earlier, in our study 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. We use SuperMikeII as the baseline configuration and compare the performance of other clusters with respect to this. 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. For the masternode, we always use a minimal configuration based upon the resources in the cluster. In the subsequent sections we use the term node and datanode interchangeably.

B. Cluster Characterization using γ and β_{io}

Figure-1a and 1b shows the change in system's optimum I/O and memory balance respectively with respect to Intel Xeon processor architecture for varying application balance (i.e., application's resource requirement). Based upon this figure characterize each of our experimental testbeds on the basis of β_{io} and β_{mem} and discuss the major pros and cons of each different architecture.

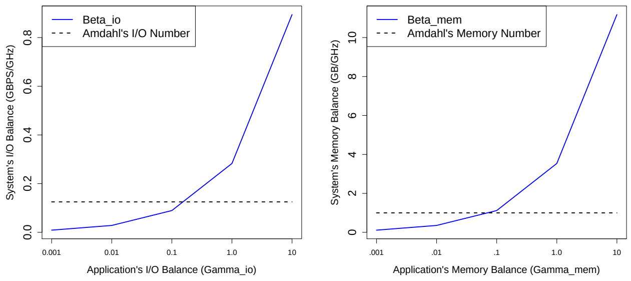
1) SuperMikeII (Traditional HPC cluster): SuperMikeII has two 8-core Intel Sandybridge Xeon processor per node thus offering huge processing power. However, each SuperMikeII node is equipped with only one HDD, thus, limited in terms of I/O bandwidth. Also, each SuperMikeII node has only 32GB DRAM. Thus, SuperMikeII has both, β_{io}

TABLE III: Experimental Testbeds

Cluster	Processor	CPU Core Speed (GHz)	#Cores /DN	CPU Speed (GHz)	Seq I/O Band-width /Disk (GBPS)	#Disks /DN	Total Seq I/O Band-width (GBPS)	DRAM /DN (GB)	β_{io}	β_{mem}
SuperMikeII (Traditional Supercomputer)	Two 8-core SandyBridge Xeon	2.6	16	41.6	0.15	1	0.15	32GB	0.003	0.77
SwatIII (Regular Datacenter)	Two 8-core SandyBridge Xeon	2.6	16	41.6	0.15	4	0.60	256	0.015	6.15
CeresII (MicroBrick-based Hyperscale System)	One 6-core Xeon	2	6	12	2	1	2	64	0.16	5.33

TABLE IV: Experimental Testbeds

Cluster	Total Cost ($)	Cost /DN ($)	#DN	Total Processing Speed (GHz)	Total Storage Space	Total DRAM (TB)	Execution Time
SuperMikeII	57060	3804	15	624.00	7.50	0.48	0
SwatIII	57060	6911	8	330.83	16.00	2.00	0
CeresII	57060	2282	25	300.00	25.00	1.60	0



(a) β_{io} vs γ_{io}

(b) β_{mem} vs γ_{mem}

Fig. 1: Change in system's optimum I/O and memory balance (β_{io} and β_{mem}) with respect to Intel Xeon processor ($\delta_{io} = 13.57$ and $\delta_{mem} = 0.08$) for different types of applications (varying values of γ_{io} and γ_{mem})

($= 0.003$) and β_{mem} ($= 0.77$) a magnitude smalller than the optimum produced by our model (as shown in Equation-16 and Equation-17) for a data-, compute- and memory-intensive application. According to Figure-1 SuperMikeII can provide optimal performance only for compute-intensive applications (precisely for those application with $\gamma_{io} = x$ and $\gamma_{mem} = y$).

2) SwatIII (Existing Datacenter): Unlike SuperMikeII which use only one HDD per node, SwatIII uses 4-HDDs per node using JBOD (Just a Bunch of Disk) conguration while using the same processor (i.e. two 8-core Intel SandyBridge Xeon) as SuperMikeII. Since the I/O throughput increases linearly with number of disks, SwatIII's β_{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, thus achieve very high value for β_{mem} ($= 6.15$). It is to be noticed that β_{mem} of SwatIII is even higher than the optimum produced by the model (Equation-17). Thus, according to Figure-1 SwatIII can produce optimal performance for moderately I/O-intensive applications and for memory-intensive applications (precisely when $\gamma_{io} = x$ and $\gamma_{mem} = y$)

3) CeresII (MicroBrick-based Hyperscale System): The last one, CeresII is a novel hyperscale system based on Samsung MicroBricks. It uses one 6-core Intel Xeon processor with a core frequency of 2-GHz. Each MicroBrick (or,simply the computation module) of CeresII consists of a 6-core Intel Xeon processor with a core frequency of 2-GHz, two NVMe-SSD (Samsung) each with an I/O bandwidth of 2GBPS, and 64-GB DRAM. β_{io} of CeresII is 0.32 which is closer to the optimum produced by our model (as shown 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 significantly less than SwatIII. Thus, CeresII is the most balanced cluster among all the avalilable resources and we expect to get the best cost to performance for today's I/O-, compute- and memory-intensive applications for which $\gamma_{io} = \gamma_{mem} = 1$.

C. Understanding the Workload

1) Terasort:

2) Wordcount:

3) Genome Assembly with Hadoop and Giraph: 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 [25]. We classified the de novo assembly in two stages as follows: *a*) Hadoop-based de Bruijn graph-construction and *b*) Giraph-based graph-simplification. In this section, we provide a brief overview of each stage of the assembler.

Hadoop-based De Bruijn graph-construction (I/O- 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.

Benchmark Specification	Job name	Job Type	Input	Final output	# jobs	Shuffled data	HDFS Data
HiBench	Terasort	Hadoop	1TB	1TB	1	1TB	1TB
HiBench	Wordcount	Hadoop	1TB	1TB	1	1TB	1TB
Moderate size Genome Assembly	Graph Construction	Hadoop	90GB (500-million reads)	95GB	2	2TB	136GB
Moderate size Genome Assembly	Graph Simplification	Series of Giraph jobs	95GB (71581898 vertices)	640MB (62158 vertices)	15	-	966GB
Large size Genome Assembly	Graph Construction	Hadoop	452GB (2-billion reads)	3TB	2	9.9TB	3.2TB
Large size Genome Assembly	Graph Simplification	Series of Giraph jobs	3.2TB (1483246722 vertices)	3.8GB (3032297 vertices)	15	-	4.1TB

TABLE V: Data size for different benchmark applications

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.

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.

VII. Input Data

VIII. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] H. Kung, “Memory requirements for balanced computer architectures,” in *ACM SIGARCH Computer Architecture News*, vol. 14, no. 2. IEEE Computer Society Press, 1986, pp. 49–54.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[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] 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.

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

[25] 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.