Augmenting Amdahl's Second Law: A Theoretical Model to Build 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¶, and Ling Liu†
¶School of EECS, Center for Computation and Technology, Louisiana State University, Baton Rouge, LA, USA
*Samsung Electronics Co., Ltd., Yongin-si, Gyeonggi-do, South Korea
†School of Computer Science, Georgia Institute of Technology, Atalanta, GA, USA
Email: {akdas, sgoswami, rplatania, klee, sjpark}@cct.lsu.edu
{jaeki.hong, wooseok_chang}@samsung.com
ling.liu@cc.gatech.edu

***Abstract*—High-performance analysis of big data demands more computing resources, forcing similar growth in computation cost. So, the challenge to the HPC system designers is providing not only high performance but also high performance at lower cost. For high performance yet cost effective cyberinfrastructure, we propose a new system model augmenting Amdahls second law for balanced system to optimize price-performance-ratio. We express the optimal balance among CPU-speed, I/O-bandwidth and DRAM-size (i.e., Amdahls I/O- and memory-number) in terms of application characteristics and hardware cost. Considering Xeon processor and recent hardware prices, we showed that a system needs almost 0.17GBPS I/O-bandwidth and 3GB DRAM per GHz CPU-speed to minimize the price-performance-ratio for data- and compute-intensive applications.**

To substantiate our claim, we evaluate three different cluster architectures: 1) SupermikeII, a traditional HPC cluster, 2) SwatIII, a regular datacenter, and 3) CeresII, a MicroBrick-based novel hyperscale system. CeresII with 6-Xeon-D1541 cores (2GHz/core), 1-NVMe SSD (2GBPS I/O-bandwidth) and 64GB DRAM per node, closely resembles the optimum produced by our model. Consequently, in terms of price-performance-ratio CeresII outperformed both SupermikeII (by 65-85%) and SwatIII (by 40-50%) for data- and compute-intensive Hadoop benchmarks (TeraSort and WordCount) and our own benchmark genome assembler based on Hadoop and Giraph.

I. Introduction

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 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 (i.e., minimizing price-performance-ratio).

At this inflection point of HPC landscape, system designers must consider more degrees of freedom for new cluster architecture for big data processing 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?* These complex performance and economic factors together motivates new designs of HPC infrastructure.

With this motivation, this paper makes an initial attempt to resolve the existing performance and cost conundrum by augmenting Amdahl's second law (i.e., Amdahl's I/O and memory number) for balanced system. We propose a simple additive model to optimize price-performance-ratio 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. The final outcome of this model are the two modified Amdahl's numbers which can be easily used by hardware vendors to propose cost-effective architecture for data- and compute-intensive applications.

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 0.17-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 different cluster architectures: 1) SupermikeII, a traditional HPC cluster, 2) SwatIII, a regular datacenter, and 3) CeresII, a MicroBrick-based novel hyperscale system. CeresII with 6-Xeon-D1541 cores (2GHz/core), 1-NVMe SSD (2GBPS I/O-bandwidth) and 64GB DRAM per node, closely resembles the optimum produced by our model. Consequently, it outperformed both the clusters for data- and compute-intensive Hadoop benchmarks (TeraSort and WordCount) as well as our own benchmark genome assembler based on Hadoop and Giraph. Overall, CeresII showed 65-85% and 40-50% better price-performance-ratio over SuperMikeII and SwatIII respectively.

The rest of the paper is organized as follows: Section-II describes the prior work related to our current effort. In Section-III, we discuss Amdahl's second law in detail. Section-IV describes the proposed model. Then, in Section-V we show the details of our experimental testbeds and classify those using our proposed model. Section-VI describes the evaluation

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.27

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. Finally, Section-VIII concludes the paper with possible improvement to stimulate discussion and future work.

II. Related Work

Numerous studies have been performed evaluating the performance implication of different big data analytic software (e.g., Hadoop) on different types of hardware infrastructures.

Hardware evaluation studies such as, [1], [2], [3], [4], etc. unanimously concluded that the use of SSD can accelerate the Hadoop applications with respect to standard benchmark (e.g., TeraSort, WordCount, etc.). On the other hand, [5], [6], and [7] evaluated the over all cluster architecture for different Hadoop benchmark. Although these studies provide good insight on performance of different hardware, the rapid changes in hardware technologies and cost limits their scope.

Simulation studies such as, SimMR [8], MRSim[9], MR-Perf [10], etc. reduced the hardware cost by creating virtual Hadoop job over simulated hardware environment. Although simulation is more cost-effective than real hardware, it comes with lots of software overhead. Also, the process of finding alternative architecture is mostly driven by trial-and-error method and prior experiences. Considering the broad range of available hardware alternatives with more than 200 Hadoop parameters, it is challenging to provide optimal hardware configuration and may suffer from reliability issues [11].

Analytical models such as [11], [12], [13], etc. abstract away several performance parameters and predict the performance of a Hadoop job mainly using single- or multi-layer queuing networks. Although no overhead is involved in analytical approach, like simulation it is hard to find an optimally balanced architecture in the vast range of hardware alternatives.

To date, the most practical approach to design a balanced system is to follow Amdahl's second law. Computer scientist Gene Amdahl postulated that 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. Amdahl's second law (with Gray's amendment) can be used for system characterization and proposing a balanced system. For example, Bell and Gray [14] classified the existing supercomputers based upon Amdahl's second law to clarify the future road map of the HPC architecture. Cohen [15] applied Amdahl's second law to the datacenter cluster to study the interplay between processor architecture and network interconnect in a datacenter. Chang [16] used Amdahl's second law to better understand the performance of hardware design implications of data analytic systems. Szalay [17], 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.

Unlike these studies, we consider a balanced system as one that optimizes both cost and performance. Hence, we did not consider the optimal balance of the system (i.e., the I/O and memory ratio to the processor speed) as constants as in original Amdahl's I/O and memory number. Instead, we propose an additive model to express the optimal system balance as a function of both application characteristics and hardware price.

III. Background

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, $\beta_{io}^{opt} = 0.125$ and $\beta_{mem}^{opt} = 1$.

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 I/O 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 simply put forward the statistics reflecting the contemporary state of the cluster architecture.

Using the notations in Table I Gray's laws can be expressed as, $\beta_{io}^{opt} = \gamma_{io}$ and $\beta_{mem}^{opt} = 4$

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, cannot satisfy the different resource balance ratios for a collection of analytic applications.

Gray's law is more realistic in the sense that it considers the impact of application balance on the cluster architecture. However, it is limiting to reflect the interplay between application and cost balance. The cost of hardware components

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- or CPU-intensive. Lower value means more CPU-intensive, higher 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- or CPU-intensive. Lower value means more CPU-intensive, higher 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)

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 expressed as, $\beta_{io}^{opt} = f_1(\gamma_{io}, \delta_{io})$ and $\beta_{mem}^{opt} = f_2(\gamma_{mem}, \delta_{mem})$

B. Model Assumptions

For simplicity of calculation and better usability, the model first ignores the CPU micro architecture as in [17]. 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.

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

$$T_{total} = T_{cpu} + T_{io} + T_{mem} \tag{1}$$

$$\implies T_{total} = \frac{W_{cpu}}{R_{cpu}} + \frac{W_{io}}{R_{io}} + \frac{W_{mem}}{R_{mem}} \tag{2}$$

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:

$$C_{total} = C_{cpu} + C_{io} + C_{mem} \tag{3}$$

$$\implies C_{total} = P_{cpu}R_{cpu} + P_{io}R_{io} + P_{mem}R_{mems} \tag{4}$$

C. Model Derivation

In this section our main objective is to minimize the price-performance-ratio (denoted as f_{cp}). Assuming the performance as the inverse of the execution time, f_{cp} can be expressed as:

$$f_{cp} = C_{total} \times T_{total} \tag{5}$$

$$\begin{aligned}\implies f_{cp} = (C_{cpu} + C_{io} + C_{mem}) \times \\ (T_{cpu} + T_{io} + T_{mem})\end{aligned} \tag{6}$$

$$\begin{aligned}\implies f_{cp} &= C_{cpu}T_{cpu} + C_{cpu}T_{io} + C_{cpu}T_{mem} \\ &+ C_{io}T_{cpu} + C_{io}T_{io} + C_{io}T_{mem} \\ &+ C_{mem}T_{cpu} + C_{mem}T_{io} + C_{mem}T_{mem}\end{aligned} \tag{7}$$

Assuming a CPU core cannot perform disk and memory operation at the same time, $C_{io}T_{mem}$ (term-6) and $C_{mem}T_{io}$ (term-8) depicts the depreciation of one component when the other in use. That is, when the memory is in use the cost of disk is depreciated to zero and vice versa. Hence, term-6 and 8 of Equation 7 are practically insignificant and should be eliminated from Equation 7.

Then, by expanding all the time (T) and cost (C) terms using Equation 2 and 4 respectively and then substituting with the notation used for system balance in Table I (i.e., β_{io} and β_{mem}), Equation 7 can be rewritten as:

$$\begin{aligned}f_{cp} = P_{cpu}W_{cpu} + \frac{1}{\beta_{io}}P_{cpu}W_{io} + \frac{1}{\beta_{mem}}P_{cpu}W_{mem} \\ + \beta_{io}P_{io}W_{cpu} + P_{io}W_{io} \\ + \beta_{mem}P_{mem}W_{cpu} + P_{mem}W_{mem}\end{aligned} \tag{8}$$

Again, 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:

$$\frac{\partial f_{cp}}{\partial \beta_{io}} = -\frac{1}{\beta_{io}^2}P_{cpu}W_{io} + P_{io}W_{cpu} \tag{9}$$

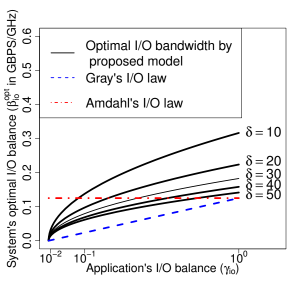
For the optimal balance (β_{io}^{opt}) between CPU speed and I/O bandwidth, Equation 9 should equal to 0. Then, solving for β_{io} and replacing it with the workload and technology-cost balance terms mentioned in Table I we get:

$$\beta_{io}^{opt} = \sqrt{\frac{\gamma_{io}}{\delta_{io}}} \tag{10}$$

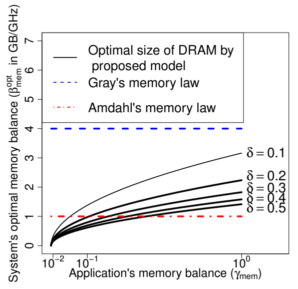
Similarly, the optimum balance (β_{mem}^{opt}) between CPU speed and size of DRAM can be derived as:

$$\beta_{mem}^{opt} = \sqrt{\frac{\gamma_{mem}}{\delta_{mem}}} \tag{11}$$

Equation 10 and 11 show the contribution of application balance and cost balance towards optimal system balance.



(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}).

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 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 suggests 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 should 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. That is, both scaled out and scaled up architecture can produce optimal price-performance-ratio if the proper balance is maintained.

E. Notes on Different types of I/O

Since the model is generalized for different types of I/O, system designers should be careful about the following issues to maintain the system balance (i.e., to achieve the recommended value of β_{io}):

1) Sequential vs Random I/O: System designers should be careful about the application characteristics. An application with frequent random I/O (e.g., shuffle phase of a Hadoop job) will get benefit from SSD.

TABLE II: Cost of different hardware components

Hardware components	Cost($)[1]	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

2) Disk Controller: Aggregate I/O bandwidth of all the disks attached to a compute node should be less than or equal to the bandwidth of the disk controllers. Otherwise, disk controller will be saturated and the application cannot utilize the full I/O bandwidth of all the disks. [7] demonstrated this issue.

3) Network I/O: The network I/O can be modeled similarly as the disk I/O model. However, system designers should be careful about network topology (e.g., FatTree, Clos, etc.) and the blocking ratio which can be changed manually affecting the cost per network bandwidth. Hence, to apply the proposed model for network I/O the impact of these factors should be eliminated. That is, given same topology and blocking ratio system designers can easily calculate optimum ratio between CPU speed and network bandwidth using the proposed model.

F. An Illustrative Example to Build a Balanced Cluster

In this section, we demonstrate a real 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, $\gamma_{io} = \gamma_{mem} = 1$

The *Unit Price* in Table II shows the current price trend for different processor, storage and memory alternatives in their corresponding unit. We consider Intel Xeon processor. As it can 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 respective unit prices are \$42/GHz and \$54/GHz. Hence, in this example we selected average unit cost of the processor

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

TABLE III: Experimental Testbeds

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

as \$48/GHz. Similarly the cost of DRAM is calculated as \$7.20/GB. To eliminate the impact of storage amount while calculating the I/O bandwidth cost 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. This way, 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: $\delta_{io} = 34.61$ and $\delta_{mem} = 0.15$.

Finally, for $\gamma_{io} = \gamma_{mem} = 1$ (i.e., CPU-, I/O- and memory-intensive application), using Equation 10 and 11 we get $\beta_{io}^{opt} = 0.17$ and $\beta_{mem}^{opt} = 2.7$.

V. Experimental Testbeds

1) SuperMikeII (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. 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 smaller than the optimum produced by our model for a data-, compute- and memory-intensive application as shown in Equation 10 and Equation 11. Using the plot shown in Figure 1 (or using Equation 10 and 11 with SuperMikeII hardware cost shown in Table II) SuperMikeII provides optimal price-performance-ratio for those applications where $\gamma_{io} = 0.0005$ or $\gamma_{mem} = .06$. Hence, it can be said SuperMikeII can provide cost-optimized performance for traditional compute-intensive applications such as supercomputing simulations, astrophysics calculations where γ_{io} has the order of 10^{-3} [17].

2) SwatIII (Existing Datacenter): This Samsung datacenter has 128 nodes. However, we used maximum 16 nodes for our experiments. 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 10). 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 11). Using Figure 1 (or using Equation 10 and 11 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 price-performance-ratio for many of the modern I/O-intensive big data applications.

3) CeresII (MicroBrick-based Hyperscale System): CeresII, is a novel hyperscale system, based on Samsung MicroBrick with a maximum of 40 nodes available to us. 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 10. 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.

4) The relationship between cluster-balance and the cluster-capability: The I/O and memory balance terms (β_{io} and β_{mem}) indicates the level of contention for I/O devices and memory subsystem respectively. The ratio between β_{io} (or, β_{mem}) of two different clusters can indicate their relative level of contention in I/O subsystem (or memory subsystem).

As a concrete example, let us consider CeresII (6 cores, 1 NVMe SSD per node, and $\beta_{io} = 0.166$) and SuperMikeII (16 cores, 1 HDD per node, and $\beta_{io} = 0.003$). For their corresponding value of β_{io} we can say CeresII is almost 55 ($0.166/0.003$) times more balanced, or more powerful than SuperMikeII. It can be interpreted as, CeresII has 55 times less I/O contention compared to SuperMikeII. Consequently, to achieve an optimized performance as CeresII, SuperMikeII needs almost 55 HDDs per node if those same 16 cores are used. This corollary of the proposed model can also be verified with other well known cluster architectures such as, GrayWulf [18] which won the storage challenge in SC-08 with 8 cores and 30 HDDs per node (comparing to 16 cores and 55 HDDs per node as predicted by our model).

β_{io} and β_{mem} can also be used to determine the scalability across different cluster architectures more accurately. The speed up can be determined using Universal Scalability Law [19] by Neil Gunther, $S(p) = \frac{p}{1+c_1((p-1)+c_2(p-1))}$ Where, p is the total number of processors. $S(p)$ is the speed up. c_1 is the level of contention and c_2 is the coherency delay. In absence of any coherency delay (i.e., $c_2 = 0$), the relative contention of two systems can be easily derived using the ratio of their corresponding β_{io} or, β_{mem} to get a better estimate of scalability. However, the detailed analysis of scalability of a data-intensive application, and modification of the corresponding laws should possibly be the focus of a different paper.

VI. Cluster Evaluation Methodology

A. Hadoop configuration overview

To evaluate different clusters we use Clouera-Hadoop-2.3.0 and Giraph-1.1.0. We set up the Hadoop and Yarn parameters such that the application can make use of maximum available processing speed, I/O bandwidth, and DRAM. For example, we used all the compute cores for YARN where each Hadoop (and Giraph) worker uses one core. On the other hand, keeping 10% of DRAM for system use, we divided the rest among all Hadoop workers equally. For Giraph, we used enough DRAM to accommodate the entire graph structure in memory and always avoided out-of-core execution. We spread the HDFS data (with replication factor 1) and the shuffled data among all the disk(s) in the node using default scheduling of Hadoop.

B. Benchmark application characteristics

Table IV summarizes the details of the benchmark applications. We used three benchmark applications for our evaluation: TeraSort, WordCount, and a real world genome assembly application described as follows:

1) TeraSort: The map phase of TeraSort samples and partitions the input data 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.

2) WordCount: 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.

3) Genome assembly: Accounting the scarcity of good big data HPC benchmark and relevant big dataset we use our own genome assembler [20] built atop Hadoop and Giraph as a real use case which represents many HPC/Datacenter applications. 1) The first stage of the assembler is a shuffle intensive Hadoop job representing an I/O-bound application. It scans through all genomic short reads (lines containing A, T, G or C only) and

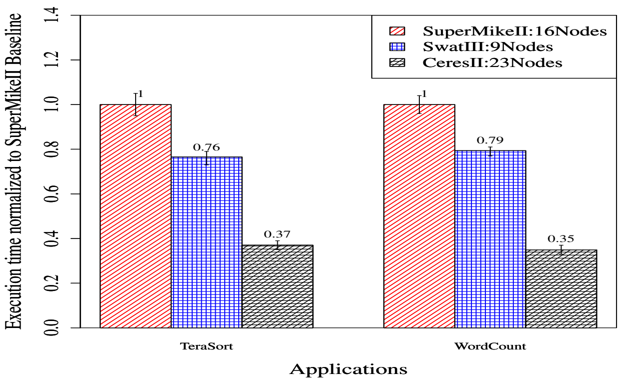


Fig. 2: Execution time of TeraSort and WordCount over different cluster architectures keeping the total cost of each cluster same. All the execution-times are normalized to the execution time in 16 nodes of SuperMikeII (Lower is better).

divides them to smaller fragment of length k, called k-mer. The map phase emits two consecutive k-mers as intermediate key-value pairs representing a vertex and an edge from it. Reduce phase aggregate all edges from each vertex to write the entire graph structure on HDFS. 2) The second stage of our genome assembler represents terabyte-scale graph processing which is the core part of many HPC problem. In this phase, we compress each of the linear chains in the graph structure to one vertex. Then we remove all the tip and buble structure of the graph as those are introduced because of sequencing error.

C. Data Size

For both TeraSort and WordCount, we generated 1TB random dataset as the input. The shuffle and output data size of TeraSort is also same as its input (i.e., 1TB in this work). 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 human genome dataset (452GB), openly available in NCBI website (http://www.ncbi.nlm.nih.gov) with accession number SRX016231. 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.

VII. Results and Discussion

A. Evaluation Results of TeraSort and WordCount

Figure 2 compares the relative merit of all the three cluster architectures (i.e. SuperMikeII, SwatIII and CeresII). To show the balance between performance and economy, we use that many resources in each cluster to keep the total cost same across all the clusters. Table II shows the available resources for all three clusters 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

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 (human genome)	Hadoop	452GB (2B reads)	3TB	2	9.9TB	3.2TB	Map and Reduce: CPU- and I/O-intensive
Graph Simplification (human genome)	Series of Giraph jobs	3.2TB (1.5B vertices)	3.8GB (3M vertices)	15	-	4.1TB	Memory-Intensive

TABLE V: Resources in each cluster architecture used for TeraSort and WordCount

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

of each node in SwatIII and CeresII to count the number of nodes to be used in these two different clusters.

We ran TeraSort and WordCount in all three cluster configurations and measured their execution time. All results are the means of at least three runs of each application on each configuration. Figure 2 shows the results normalized to the SuperMikeII baseline. That is, the execution time of SuperMikeII is always 1. CeresII, being closer to the optimum produced by our model performs significantly well:

1) Comparing to the SuperMikeII baseline, for both TeraSort and WordCount, CeresII shows almost 65% improvement.

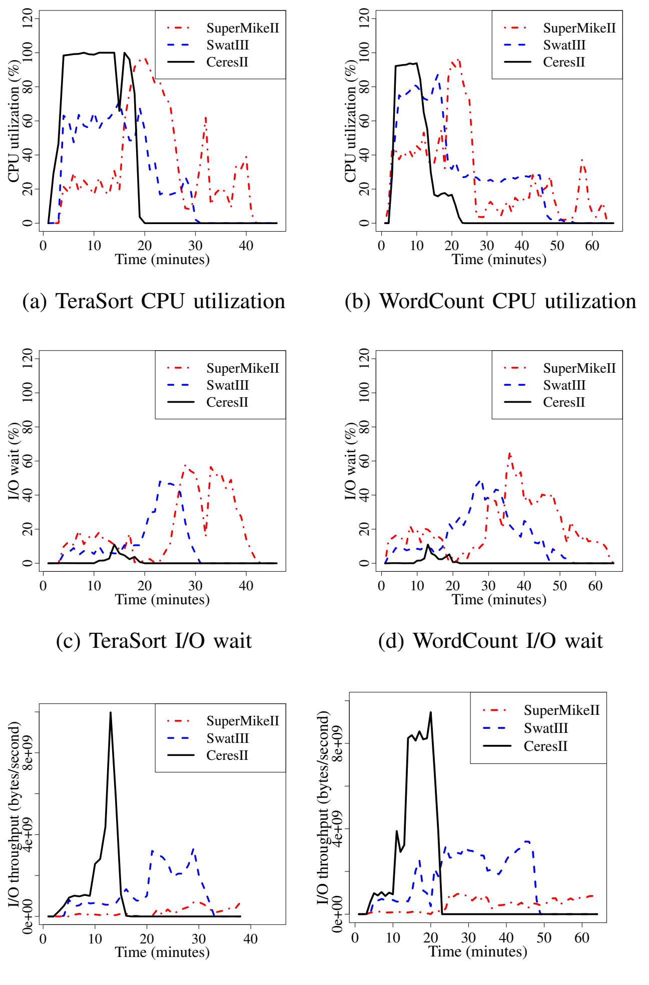
2) Comparing to SwatIII, CeresII shows almost 50% improvement in execution time for TeraSort and WordCount.

B. System Characteristics

To monitor the system characteristics we used the Cloudera-Manager-5.0.0. The observations are as follows:

1) As shown in Figure 3a and 3b, for the compute-intensive WordCount application, on an average CeresII shows 20% better CPU utilization compared to SuperMikeII as our goal is to perform the task as soon as possible resulting in better price-performance-ratio. Whereas for TeraSort with an I/O intensive reduce phase results in almost 50% better CPU utilization in CeresII than SuperMikeII. Comparing to SwatIII, CeresII shows almost 15 to 40% better CPU utilization for TeraSort and WordCount respectively. Since, our goal is to perform the task as soon as possible (to better utilize the system's cost), CeresII is considered as the best architecture.

2) Figure 3c and 3d shows the reason behind the better CPU utilization of CeresII. Since the architectural balance of CeresII closely resembles to the optimal β_{io} produced by our model, CeresII shows almost negligible I/O wait (less than 2%) for any of the applications. On the other extreme, SuperMikeII with only one HDD results in extremely low β_{io}, consequently shows highest I/O wait among all the cluster architecture. SwatIII with four HDD per node shows an I/O wait lower than SuperMikeII but significantly higher than CeresII.



(a) TeraSort CPU utilization (b) WordCount CPU utilization

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

(e) TeraSort I/O throughput (f) WordCount I/O throughput

Fig. 3: CPU and I/O characteristics of each node of different cluster architectures for TeraSort and WordCount benchmark

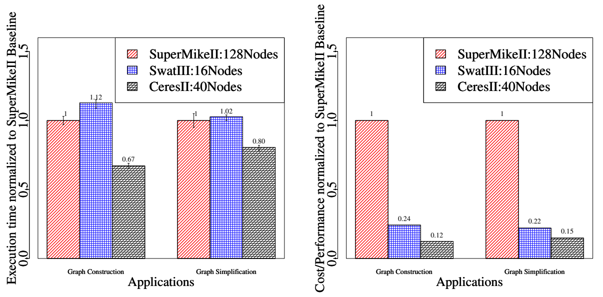
3) Figure 3e and 3f compare the I/O throughput of all the clusters. CeresII with the most optimal β_{io} shows the highest I/O throughput and SuperMikeII with the lowest β_{io} shows the lowest I/O throughput among all the clusters. SwatIII lies in between these two extremes as its β_{io} lies between them.

C. Evaluation Results of Large-Size Human Genome Assembly

To assemble the large human genome (452GB), we used the maximum available resources in each of the clusters to

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) price-performance-ratio

Fig. 4: Performance of different cluster for large size human genome assembly (normalized to 128 nodes of SuperMikeII).

accommodate the huge amount of shuffled data (9.9TB) and 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 show the execution time and the price-performance-ratio (i.e., cost $\times$ execution-time) respectively for the Hadoop and Giraph stages of the human genome assembly. The results are as follows:

1) CeresII, even with almost 90% less processing speed than SuperMikeII across the cluster, outperformed it by almost 88% in terms of price-performance-ratio. In Giraph stage the corresponding gain in 85%. In terms of execution time CeresII gains almost 30% and 20% respectively over SuperMikeII.

2) Comparing to SwatIII, the processing power of CeresII is 72%. However, due to the optimal architectural balance, CeresII shows almost 50% and 30% improvements over SwatIII in terms of price-performance-ratio for Hadoop and Giraph respectively. For execution time, those gains are 50% and 20%.

VIII. Conclusion and Future Work

Big data needs big resources. With increasing popularity of data-driven research, it is obvious that providing more resources (CPU, I/O bandwidth, DRAM, etc.) will provide more performance. So, the major challenge is now in providing expected performance in reduced cost. We make an initial attempt to analytically resolve the performance and cost conundrum prevalent in big data cyberinfrastructure.

The model also provides a new metric to show the capacity of the HPC clusters. For I/O and memory-bound applications, β_{io} and β_{mem} can provide a better and easy-to-use alternative of FLOPS which shows only the CPU capacity.

In this initial attempt we focus on simplicity of the model to make it useful in practical purposes (e.g., investing for a new cluster with limited information on application characteristics). However, more subtle parameters (e.g., CPU multi-threading, I/O latency, etc.) can be added to improve its accuracy.

Acknowledgment

This work was supported in part by NIH-P20GM103424, NSF-MRI-1338051, NSF-CC-NIE-1341008, NSF-IBSS-L-1620451, and LA BoR LEQSF(2016-19)-RD-A-08.

References

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

[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 *International Conference on Advanced Cloud and Big Data*, 2013.

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

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

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

[6] 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.

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

[8] A. Verma, L. Cherkasova, and R. H. Campbell, "Play it again, simmr!" in *IEEE International Conference on Cluster Computing*, 2011.

[9] S. Hammoud, M. Li, Y. Liu, N. K. Alham, and Z. Liu, "Mrsim: A discrete event based mapreduce simulator," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE, 2010.

[10] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups." in *MASCOTS*, 2009.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] L. Dobos, I. Csabai, A. S. Szalay, T. Budavári, and N. Li, "Graywulf: A platform for federated scientific databases and services," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. ACM, 2013, p. 30.

[19] N. J. Gunther, "A simple capacity model of massively parallel transaction systems," in *CMG-CONFERENCE-*. COMPSCER MEASUREMENT GROUP INC, 1993, pp. 1035–1035.

[20] P. K. Kondikoppa, A. K. Das, S. Goswami, R. Platania, and S.-J. Park, "Giga:giraph-based genome assembler for gigabase scale genomes," in *Proceedings of the 8th International BICob Conference*. ISCA, 2016.