Received November 19, 2015, accepted December 12, 2015, date of publication December 18, 2015,
date of current version December 29, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2509598

Performance Evaluation and Estimation Model Using Regression Method for Hadoop WordCount

JOSEPH A. ISSA
Department of Electrical and Computer Engineering, Notre Dame University at Louaize, Zouk Mosbeh 2501-1305, Lebanon
Corresponding author: J. Issa (joseph.issa@ndu.edu.lb)

ABSTRACT Given the rapid growth in cloud computing, it is important to analyze the performance of different Hadoop MapReduce applications and to understand the performance bottleneck in a cloud cluster that contributes to higher or lower performance. It is also important to analyze the underlying hardware in cloud cluster servers to enable the optimization of software and hardware to achieve the maximum performance possible. Hadoop is based on MapReduce, which is one of the most popular programming models for big data analysis in a parallel computing environment. In this paper, we present a detailed performance analysis, characterization, and evaluation of Hadoop MapReduce WordCount application. We also propose an estimation model based on Amdahl's law regression method to estimate performance and total processing time versus different input sizes for a given processor architecture. The estimation regression model is verified to estimate performance and run time with an error margin of $<5\%$.

INDEX TERMS Performance analysis, cloud computing, Hadoop WordCount.

I. INTRODUCTION

In recent years, the demand for cloud computing has increased exponentially. This is due to an increasing demand in storing, processing and retrieving a large amount of data in a cloud cluster. The data can be either stored to a cloud network such as scientific data (i.e. Climate modeling, Fusion, Bioinformatics...etc) or use the cloud network for data-intensive tasks such as collecting experimental data, dumping data on parallel storage systems, run large scale simulations...etc. Hadoop was introduced as a solution to handle processing, storing and retrieving Big Data in a cloud environment. It is important for processor architects to understand what processor micro-architecture parameters that affect performance. It is also important for benchmark developers to optimize the benchmark software for a given hardware to achieve maximum performance possible. Hadoop is an open-source framework with three main components: MapReduce, HBase and Hadoop Distributed File System (HDFS). HDFS is the primary storage for Hadoop; it is highly reliable and uses sockets for communications. The HDFS is horizontally scalable and each cloud cluster consists of a single NameNode and DataNode. The NameNode job acts like a master server that manages the file system namespace which in turn manages access to files by clients. The DataNode manages storage attached to a node that they run on. DataNodes process server reads, write requests, block creation, and replication. HDFS is for batch processing not serving random read or write requests. The HBase is considered the Hadoop database for distributed big data storage, but the two main components of Hadoop are HDFS which is horizontally scalable with a default setting of three copies for storage and MapReduce which is parallelized scalable for computation framework. Hadoop framework consists of several applications developed using MapReduce framework; one of these applications is WordCount. MapReduce is a programmable framework for pulling data in parallel out of a cluster. Parallel processing is when a program executes multiple code paths simultaneously for a massive amount of data. In this paper, we present a detailed performance and power sensitivity analysis for Hadoop WordCount using different processors such as Intel's ATOM D525, Xeon X5690, and AMD's BobCat E-350. The paper is organized as follows: In section II we discuss the motivation behind performance modeling using analytical approach rather than simulation approach and why we chose Hadoop WordCount application as compared to other Hadoop applications. In Section III, we present an overview of Hadoop WordCount and how it fits the MapReduce architecture. In Section IV, we discuss related work in which we compare our work to other published papers of the same topic. In section V, we present a detailed performance and power evaluation of Hadoop WordCount for different processor architecture parameters. In section VI, we present an estimation model for total processing time and performance using Amdahl's law regression method. In section VII, we conclude and discuss future work guidelines.

II. MOTIVATION

There are several important reasons to model the performance for a cloud application on a given processor architecture using analytical approach instead of the simulation 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. It can provide and early performance indicator for a given cloud workload using a given processor architecture as a baseline for the model. For example, a change in performance is a result in the increase in the number of cores, increase in core frequency or increase in cache size which reduces cache misses. This enables processor architects to estimate performance before measuring the benchmark on a future processor that is not available yet for measurement. It also enables comparing different cloud applications projected performance scores across different processors of the same architecture. Usually, we expect an increase in performance (and power) for future processors given that more hardware features and capabilities are added for a given processor roadmap. Some of these features are an increase in the number of cores, higher memory speed and capacity, increase in cache size or an increase in the core frequency. The performance estimation model proposed in this paper covers most of these features that enables processor developers to get an early indication on Hadoop WordCount workload performance for future processors. The approach for developing the model is simple so the same method can be implemented for different Hadoop application. The reason why we chose Hadoop WordCount is because it is CPU intensive (compute-bound) workload in both the map phase and the reduce phase. This puts most of the performance metric on the processor computation power rather than memory capacity/bandwidth or IO latency. Other Hadoop applications can be memory-bound or IO bound.

For CPU-bound applications, performance is gated by activity on the processor. Important performance parameters are the number of cores/threads, the core frequency, and the latency/bandwidth from processor caches. Therefore, systems are cheaper to build for CPU-bound workloads. For Memory-bound workloads, it is the opposite of CPU-bound - performance is mainly determined by off-chip events, mainly how many main memory transactions can be completed per unit time, i.e. by the bandwidth actually achieved from/to main memory. Table 1 summarizes the system resource utilization for different Hadoop applications.

III. HADOOP WordCount OVERVIEW

Hadoop WordCount is a workload that reads an input text file and counts how often a certain word exists in the text file. The input to the framework is a text file referred to in this paper as Input Size, which generates different output text files that contain a word and the count of how often this word exists in the input file. Using MapReduce, each line is fed into a mapper as an input and breaks down each line into words. It then uses a key/value pair for each word with '1' next to each word as initial count. A reducer sums the count of each word and issues a key/value pair with the word and sum for each word as shown in the example in Figure 1. Almost all data can be mapped into key/value pairs somehow and the key and value can be of any type (string, integers...etc). Hadoop WordCount is easy to parallelize, the counts on one piece of a file are independent of counts from other pieces. This leads to MapReduce implementation using key/value pairs.

TABLE 1. Workloads based on Hadoop framework: system resource utilization.

Application	System Resource Utilization
WordSort	Sort Phase: IO-bound, Reduce Phase: Communication-bound.
Word Count	CPU-bound
TeraSort	Map Stage: CPU-Bound Reduce stage: IO-bound
NutchIndexing	IO-bound with high CPU utilizations in map stage. This workload is mainly used for web searching.
Kmeans	CPU-bound in the iteration phase, and IO-bound in the clustering phase. It is used for machine learning and data mining.

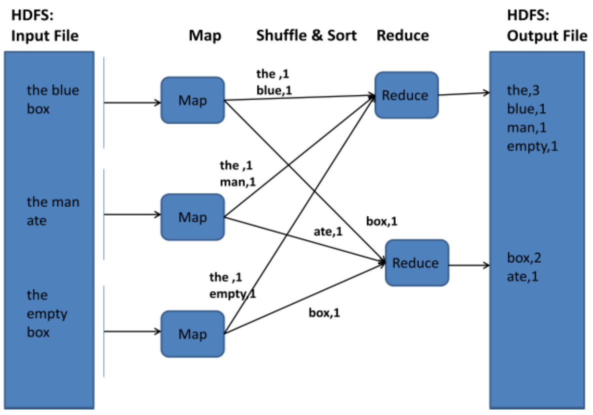

FIGURE 1. MapReduce example for Hadoop WordCount.

Figure 1 shows the role for mapper and reducer stages in parallel processing for Hadoop WordCount. An input file can be divided into multiple splits. Each line is processed by a map function. The output is a set of intermediate key/value pairs which results in the number of occurrences for each word in the input file.

IV. RELATED WORK

MapReduce is considered a simplification approach for parallel computation of large data. The MapReduce implementation depends on specific or customized cluster management that is responsible for distributing and running different tasks. Though this is not the focus of this paper, but several published papers focus on cluster management approach for improved performance. There has been considerable work on provisioning to run Hadoop computations on public clouds like Amazon published in [7] in which the author describes running Hadoop on Amazon. Our experiment was done on a private network to isolate any public network latencies and focus on the processor's computational performance. The approach is to reduce any network I/O latency for our experiments by setting a dedicated private network.

Emanuel V in [8] presents an analytical model that estimates performance for a Hadoop online prototype using job pipeline parallelism method. In comparison, the projection model proposed in this paper estimates performance and processing time using a measured baseline which is used in Amdahl's law regression method to predict the performance and execution time. Furthermore, our model is verified to predict both performance and runtime with <5% error margin for all tested cases. The performance estimation model we present in this paper is flexible and can be implemented without the need for a simulator and sampling traces.

Ibrahim et al in [8] evaluates Hadoop execution time using virtual machines. Stewart in [9] compares the performance of several data query languages. All their work is focused on different aspects for analyzing Hadoop performance. Our work complements performance analysis for Hadoop. We have published performance and power evaluation work for Hadoop frameworks in [10], in which we did a quick analysis for Hadoop WordCount and other MapReduce applications. This paper adds to what was published in [10] in which we provide an in-depth evaluation and processor competitive assessment using AMD BobCat E-350, Intel ATOM D525, and Intel Xeon X5690. Gohil et al in [11] presents Hadoop WordCount, TeraSort and other MapReduce computation workload using execution time with respect to a number of nodes. In our paper, the approach is differen; we implement comparison using different processors with performance/watt evaluation and propose an estimation method to estimate the execution time with respect to input size. Dan Wu et al in [12] evaluated Hadoop performance using HDD (magnetic drives) and SDD (Solid-State drives) in a hybrid storage system with intensive I/O access Hadoop applications. They conclude that a linear performance can be achieved by increasing the fractions of SDD drives as compared to HDD drives. This confirms our observation that SDD is a better choice to maximize the performance for Hadoop cluster as compared to HDD drives.

Krishnaprasad [21] presented methods for using Amdahl's law in different forms, but the paper did not describe how to implement Amdahl's law in a regression form as it is described in our method. Hoste et al in [22] computes the set of micro-architecture independent characteristics and weights these independent characteristics resulting in locating the application of interest in benchmark space. Performance is predicted by weighting the performance number of a benchmark in the neighborhood of application of interest. This approach for performance prediction may not fit the requirements to predict performance for a cloud computing application.

V. PERFORMANCE AND POWER EVALUATION

A. AMD's BobCat E-350 SENSITIVITY ANALYSIS

1) EXPERIMENTAL SETUP

The cluster was configured using 9 + 1 Hadoop Cluster. Hadoop uses Hadoop Data File System (HDFS) for data replication which was set to 3 (default configuration). The cluster was connected by 1Gbps switch using a private network. For hard disk, we used HDD drive set to normal spinning which is 7200 rpm. One node did only Name node and JobTracker while the other 9 nodes did DataNode and TaskTracker. This is why we refer to the cluster configuration as 9+1. The input size file used for WordCount is 50GB.

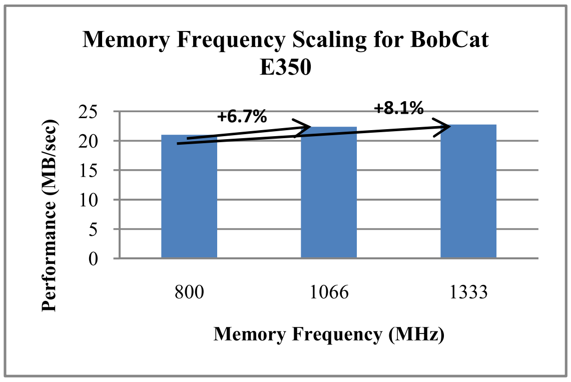

FIGURE 2. Memory performance scaling.

2) MEMORY FREQUENCY SENSITIVITY ANALYSIS

The impact of memory frequency is low for Hadoop WordCount. An increase in about 66% in higher memory frequency results into only 8% increase in performance as shown in Figure 2. As memory frequency almost doubles from 800MHz to 1333MHz, the change in performance results in only 8% change using memory frequency 800MHz as a reference baseline. This shows that WordCount is not memory-bounded which means increasing the memory speed will not have much impact on performance. Note that increasing the memory frequency will increase the power consumption while performance increase is minimum, this will result in decreasing the performance/watt which is what we want to avoid. We will show in next sections that WordCount is computer-bound (or CPU-bound) workload in map phase and reduce phase.

3) POWER SAVING IMPACT ON PERFORMANCE

Next, we tested power savings impact on performance using for BobCat E350 using Hadoop WordCount. The power saving setting was enabled in BIOS; this feature has no impact in performance as it did not change by enabling and disabling the power savings feature. We conclude there is no impact on performance when power savings is enabled as shown in Figure 3.

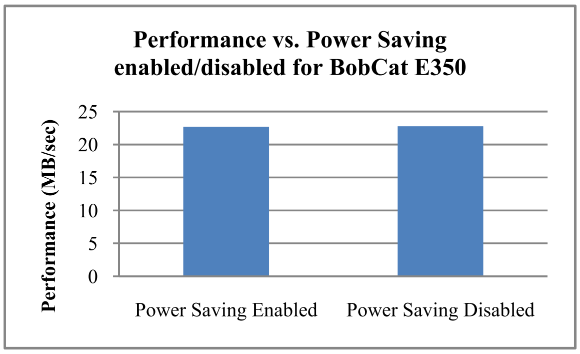

FIGURE 3. Performance vs. power saving enabled/disabled.

In summary, the performance impact as memory frequency increases has no impact on Hadoop WordCount performance. The power saving setting has no impact on performance as well. The cluster (9 + 1) consumes about 412Watts at maximum performance. This is total power consumption measured out of wall outlet. Comparing performance/watt for BobCat E350 and ATOM D525, we found that Bobcat has a lower performance/watt (55.2 KB/sec/Watt) compared to ATOM D525 with Hyper-Threading (HT) enabled (59KB/sec/Watt). However, the power consumption for BobCat when idle (Cool&Quiet enabled) is 332 Watts better than ATOM D525 369 Watts. BobCat power consumption without Cool&Quiet setting enabled is about 336Watts.

B. ATOM D-525 VERSUS BobCat E-350 SENSITIVITY ANALYSIS

1) EXPERIMENTAL SETUP

The following setup and configurations were used for ATOM and BobCat clusters. For both systems, a 9 + 1 Hadoop cluster was used. The HDFS replication was set to 3 and the cluster was connected by 1Gbps switch on a private network. We used HDD disk drives for both systems with normal spinning disk were used (7200 rpm). One node did only NameNode and JobTracker, while the other 9 nodes did Data node and TaskTracker. For ATOM D525, we used 800MHz memory speed (the maximum core frequency it can support) as compared to BobCat we used 1333MHz memory speed. The core frequency for atom was set at 1.8GHz (the maximum core frequency it can support), with 4GB memory size at 800MHz with HT on. The core frequency for BobCat was set at 1.6GHz (maximum core frequency it can support), with 4GB memory size at1333MHz. The input size used for both clusters was 50GB.

2) PERFORMANCE COMPARISON FOR BOTH ATOM AND BOBCAT CLUSTERS

Comparing the performance for Hadoop WordCount for BobCat and ATOM clusters, we conclude that ATOM cluster performance by about 6% better as compared to BobCat cluster as shown in Figure 4.

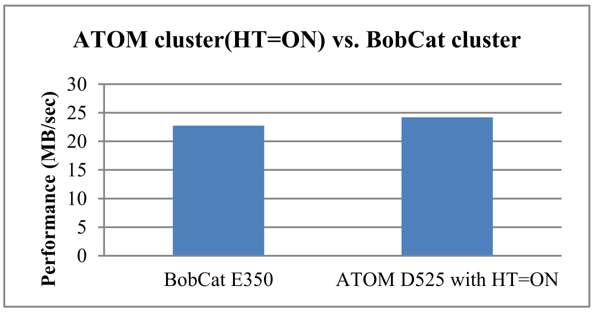

FIGURE 4. Performance for BobCat cluster vs. ATOM cluster with HT = ON.

When turning HT off for ATOM cluster, BobCat cluster shows better performance by about 36% increase as shown in Figure 5. This shows that performance for Hadoop WordCount is very sensitive to HT on ATOM cluster.

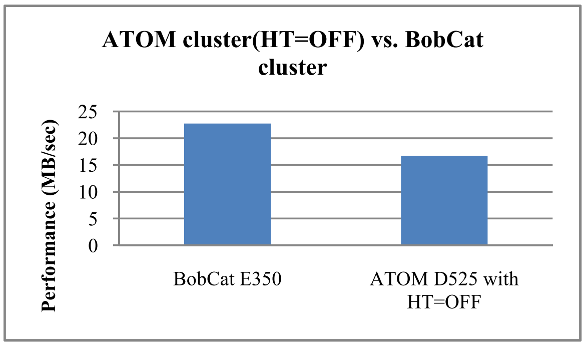

FIGURE 5. Performance for BobCat cluster vs. ATOM cluster with HT = OFF.

We also measured performance for both clusters at the same RAM frequency of 800MHz with HT off for ATOM. The performance for Hadoop WordCount for ATOM cluster is 16.69 MB/sec compared to BobCat which is 21.04 MB/sec. This shows that there is about 26% increase in performance for BobCat at the same RAM frequency of 800MHz with HT off for ATOM cluster.

We conclude in this section that ATOM outperforms BobCat with the help of Hyper-Threading. Without Hyper-Threading, ATOM at 1.8GHz core frequency has a lower performance compared to BobCat at 1.6GHz core frequency. If we assume there is a linear scaling relative to performance for BobCat at 1.8GHz predicted performance, this means that both systems now are at 1.8GHz core frequency, we can estimate that BobCat can beat ATOM at 1.8GHz with HT on. Using the linear scaling for core frequency, we estimate BobCat to achieve a performance of 25.6 MB/sec at 1.8GHz as compared to 24.21 MB/sec for ATOM @ 1.8GHz with HT on.

3) PERFORMANCE/WATT COMPARISON FOR ATOM AND BobCat CLUSTERS

It is important to analyze the power consumption for each cluster relative to performance which leads to a very important term known as performance/watt. No matter how much higher performance a cluster can achieve, it is important to keep the power consumption checked. The ideal situation for comparing both clusters is by comparing the highest performance/watt possible. For ATOM cluster in idle mode, the total power consumption was 369 watts. With HT set to OFF, the power increases to 400 watts which result in a performance/watt of 42 KB/sec/Watt. With HT set to ON, the power increases to 414 Watts which results in an increase in performance/watt to 59 KB/sec/Watt. We can conclude that if HT is enabled the power consumption for ATOM cluster will increase by 3.5%. For Bobcat cluster, the idle power (with Quick&Quiet enabled in BIOS) is 332 Watts, this is the power the wall socket. At maximum performance, the power consumption increases to 412 Watts which results in a performance/watt equal to 55KB/sec/Watt. Comparing both clusters, we conclude that ATOM cluster can achieve a higher performance/watt compared to BobCat cluster at maximum performance.

In summary, for ATOM with HT is ON is more power efficient than when HT is OFF. With ATOM HT is ON leads to more power efficient cluster than BobCat cluster. Without HT, ATOM becomes less power efficient compared to BobCat. In idle mode, BobCat consumes less power than ATOM (332 Watts vs. 369 Watts). Atom in idle mode saves 11% power compared to full load (maximum performance) while Bobcat saves 19% in idle mode compared to full load.

C. HADOOP WordCount PERFORMANCE ANALYSIS FOR XEON X5690 CLUSTER

The cluster setup used for Xeon X5690 cluster is similar to ATOM and BobCat cluster with some changes. We used Solid State Drives (SSD) for hard drives instead of spinning HDD, with 50GB input file size. All other configurations are the same as ATOM and BobCat. The core frequency set to 3.4GHz with 48GB RAM size.

1) PERFORMANCE EVALUATION WITH RESPECT TO NUMBER OF CORES USING 2 PROCESSOR SOCKETS

The performance for Hadoop WordCount scales approximately linearly (not perfectly linear) with respect to an increase in number for cores as shown in Figure 6.

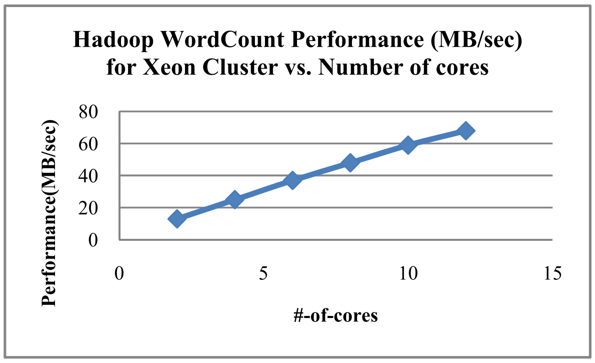

FIGURE 6. Core count performance scaling for Hadoop WordCount using Xeon Cluster.

This means that the rate of performance increase in MB/sec is proportional to the rate of change in the number of cores. For higher number of cores that cannot be measured on Xeon X5690, we can estimate a linear increase in performance which we will discuss in the estimation model in section V.

Analyzing the performance with respect to performance/#-of-cores, we noticed that the ratio of performance/# of cores drops as the number of cores increases as shown in Figure 7. That decrease is slight, but it is worth to note that this is because what is shown in Figure 6 is not a perfect linear line. If it was perfectly linear, the ratio of performance/#-of-cores with respect to #-of-cores would result in a flat constant line.

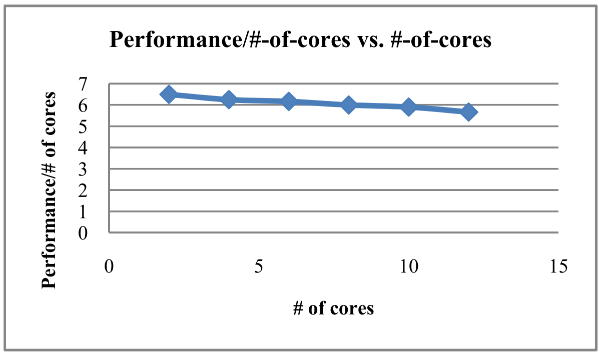

FIGURE 7. Performance/#-of-cores vs. #-of-cores.

For one socket, the performance per core slightly decreases when increasing the core count. So this applies to both one and two sockets processor configuration. In summary, WordCount scales ok on Xeon X5690 cluster as the number of cores increases, but the scaling is not perfectly linear. As we increase the number of cores from 2 to 12, the performance increases 5.25 times instead of a factor of 6 which is ideal scaling. This non-perfectly linear curve results in a slight decrease in performance/core with respect to an increase in the number of cores. There is a ∼12% decrease in performance/core from 2 to 12 cores.

2) PERFORMANCE EVALUATION WITH RESPECT TO MEMORY SPEED AND PRE-FETCHER SETTINGS

Setting pre-fetcher to ON will enable the processor to request instruction from memory before it actually needs it. Once it's received from memory, it will be placed in the cache for faster access later. Enabling Pre-Fetch will increase the Hadoop WordCount performance by about 3.8% for both memory speeds 1066MHz and 1333MHz as shown in Figure 8. This will result into fewer wait states given the processor speed are much faster than the memory speed. We also noticed that performance does not change with the increase in memory speed, so this confirms that Hadoop WordCount is not sensitive to memory speed (not memory-bound) regardless of processor architecture.

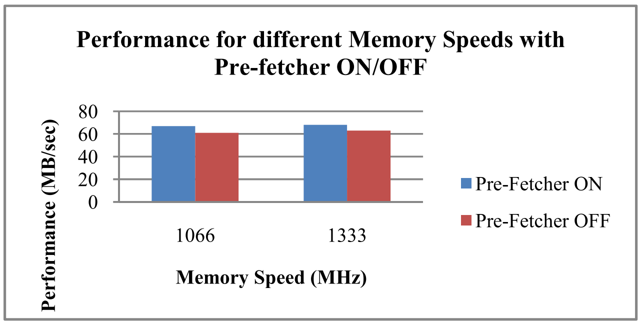

FIGURE 8. Hadoop WordCount performance evaluation vs. memory speed and different pre-fetch settings.

3) PERFORMANCE EVALUATION WITH RESPECT TO HYPER-THREADING

Hyper-Threading is an Intel's feature used to improve parallelization of computations in X86 processors. Enabling Hyper-Threading for Xeon cluster will increase the Hadoop WordCount performance by ~30% as shown in Figure 9. This is because when Hyper-Threading is enabled for a given processor, the technology will enable two logical processors per physical core.

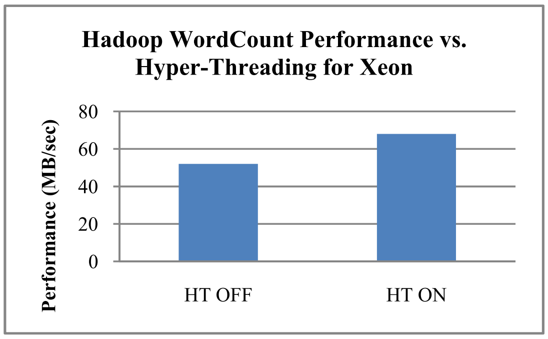

FIGURE 9. Performance vs. hyper-threading for Xeon.

There is only ~1.5% performance increase from 1066MHz to 1333MHz memory speed. Hadoop WordCount is batch-oriented which means it hides the RAM access speed very well. The cache performance is well given the CPU is very little affected by RAM frequency. The Pre-Fetch does not affect the performance significantly only by ~3.5% with pre-fetch enabled compared to Pre-Fetch disabled. The most important factor that impact Hadoop WordCount is Hyper-Threading which results in about 30% increase in performance when enabled on Xeon cluster.

4) PERFORMANCE EVALUATION WITH RESPECT TO NUMBER OF XEON PROCESSOR SOCKETS

Adding a second processor will almost double the performance for Hadoop WordCount as shown in Figure 10. This shows that Hadoop WordCount is compute-bound (or processor-bound) in both map and reduce phases rather than being memory-bound or IO-bound. This means that the performance doubles as the number of processors is doubled. Also, increasing memory speed and memory capacity does not impact much the performance as discussed earlier.

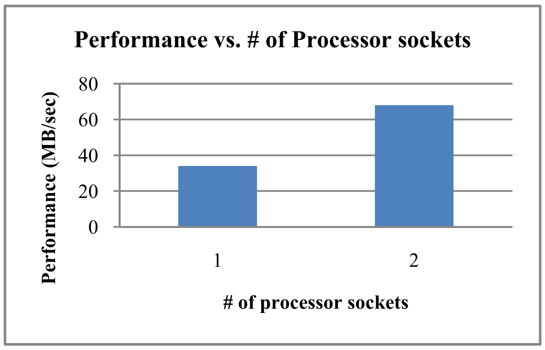

FIGURE 10. Performance vs. # of sockets for Xeon.

In summary, Xeon processors based clusters have potential for scaling performance by increasing the number of sockets which will increase the number of cores. The memory does not impact much the performance, so Hadoop WordCount is not a memory-bound workload; it's a compute-bound workload in both map and reduce phases. Hyper-Threading is the parameter with the highest impact on performance. For peak performance, Xeon needs two to three workers for each thread (72 maps plus 32 reducers for 24 threads). It can process about 68MB/sec of input data when running WordCount at 100% CPU utilization. Solid State Drives (SSD) provides enough Input/Output for WordCount, so spinning hard disks cannot feed 1 Xeon socket to be 100% utilized, this is why we used SSD instead of HDD spinning hard disks. Given the price differences between ATOM system (~$300) and Xeon X5690 (~$5000) and using the performance data we have for both systems, the performance/dollar for Xeon is about 11.3KB/sec/dollar while the performance/dollar for ATOM D525 is about 9.3KB/sec/dollar. This shows that Xeon-based systems are much more expensive compared to ATOM D525 and provides a higher performance/dollar ratio.

5) PERFORMANCE AND PROCESSING TIME EVALUATION WITH RESPECT TO INPUT SIZE

In this section, we evaluate performance and total processing time with respect to an increase in input size request. The measured performance for WordCount in Figure 11 and Figure 12 show the performance in MB/sec versus input size in GB for both ATOM and Xeon processors.

For both processors, the performance initially is linear with respect to change in the input size. It then tends to level off as input size request increases. In the case of ATOM, the performance is almost flat with little increase in performance when input size increases beyond 4GB. In the case for Xeon, the performance is almost flat when input size increases beyond 13GB. This shows that at some point no matter how much input size is increased the performance will stay almost flat with little increase. This also shows that an increase in the number of processing cores will increase

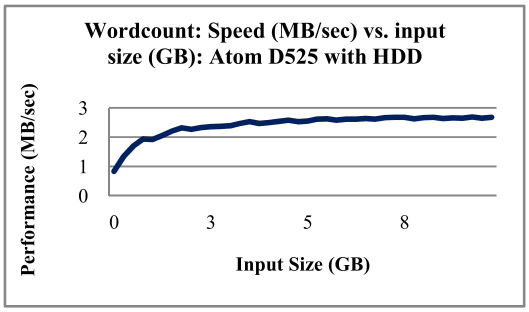

FIGURE 11. Performance evaluation vs. input size for ATOM D525 with HDD.

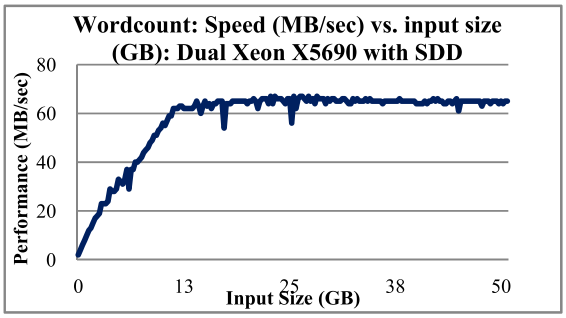

FIGURE 12. Performance evaluation vs. input size for Xeon X5690 with SDD.

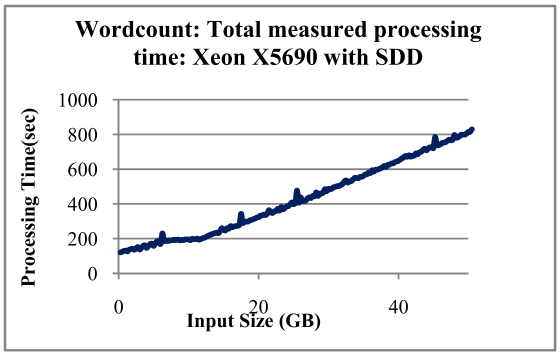

FIGURE 13. Processing time (seconds) vs. input size (GB) for Xeon X5690 with SDD.

the performance linearly as input size request increase. In Figure 13 and Figure 14 we measured the total processing time with respect to an increase in the input size. As expected, the total processing time increases almost linearly with increase in input size for both ATOM and Xeon processors. The rate of increase differs between the two processors given the difference in computing power, for example, to process a 5GB of data; it takes approximately 2000 seconds for ATOM compared to about 160 seconds for Xeon. An increase in computation power will reduce the total processing time for a given input size.

VI. ESTIMATION MODEL USING AMDAHL's LAW METHOD

The measured data for total processing time versus input size (GB) shows a linear relation between both

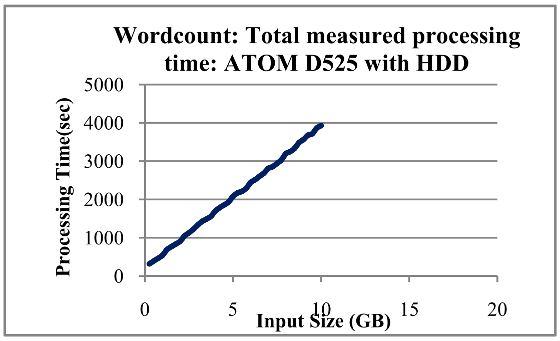

FIGURE 14. Processing time (seconds) vs. input size (GB) for ATOM D525 with HDD.

factors for ATOM D525and Xeon X5690 as shown in Figure 13 and Figure 14. However, the performance versus input size shows a logarithmic curve. Given the difference between performance and processing time versus input size curve characteristics, the estimation method we are going to use for this analysis is based on Amdahl's law regression method. The method is implemented to estimate performance or processing time while scaling one variable which is the input size. Amdahl's law is defined as the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used, which means that a system's overall performance increase is limited by the fraction of the system that cannot take advantage of the enhanced performance. The estimation model requires a measured baseline which is formed using two measured data points or more. This baseline is measured for a processor of similar architecture (i.e. ATOM) for the one to which we are estimating processing time (or performance). For example, if the measured baseline is based on ATOM D525 with two processing time data points at two different input sizes, we can use the same baseline to estimate processing time for the same ATOM architecture processor but at higher input size. In case we have to estimate performance or processing time for a different processor architecture family, a new measured baseline must be used for that same architecture.

Based on Amdahl's law definition we discussed before, the performance of a given processor can be divided into two parts, the part which increases with the performance improvement and is said to scale is defined as variable *a*, and the other part which does not improve due to the performance enhancement and is said to not scale is defined as variable *b*. The *a* and *b* variables can be derived using the basic definition of Amdahl's law which can be written in the form of:

$$T = T_o + (T_1 - T_o) \times \frac{I_1}{I} \tag{1}$$

where T1 is the measured execution time at a given input size I1, and To be the non-scale execution time. We can write To in terms of a second measurement T2 at I2:

$$T_o = \frac{T \times I_2 - T_1 \times I_1}{(I_2 - I_1)} \tag{2}$$

When we substitute Eq(2) for To in Eq(1) we obtain Amdahl's law in terms of two specific measurements without reference to To:

$$T = a + b \times \frac{1}{I} \tag{3}$$

where

$$a = \frac{I_2 \times T_2 - I_1 \times T_1}{(I_2 - I_1)} \tag{4}$$

and

$$b = \frac{(I_1 \times I_2)(T_2 - T_1)}{(I_2 - I_1)} \tag{5}$$

The variables a and b can be transformed to the performance instead of the time domain by using P = 1/T. This will give us a and b variables in terms of performance and input size as shown in Eq(6) and Eq(7).

$$a = \frac{P_1 \times I_2 - P_2 \times I_1}{(P_1 \times P_2)(I_2 - I_1)} \tag{6}$$

and

$$b = \frac{(I_1 \times I_2)(P_2 - P_1)}{(P_1 \times P_2)(I_2 - I_1)} \tag{7}$$

For two data points, we will have (I_1, P_1) and (I_2, P_2), and for n data points, we will have $(I_1, P_1), \ldots, (I_n, P_n)$. We expect these points to satisfy an equation of the form (except for noise):

$$P_i = \frac{I_i}{a \times I_i + b} \tag{8}$$

Because of noise, we cannot expect to find values for a and b that produce equality for each point i. In this case, we resort to the theory of linear least-squares estimation to obtain best estimates for a and b. In particular, given a and b, we take the error in our estimate for P_i in terms of I_i to be the difference between the measured and estimated value for P_i:

$$e_i = \left(\frac{1}{P_i} - \left(a + b \cdot \frac{1}{P_i}\right)\right) \tag{9}$$

The best estimates for$a$ and b are those that minimize the sum of the squares of these errors:

$$E = \sum_{i=1}^{n} 2e^2 \tag{10}$$

The estimates for a and b are those at which the values of the partial derivatives $\partial E/\partial a$ and $\partial E/\partial b$ are simultaneously zero. By computing these derivatives explicitly, we obtain equations satisfied by the best choices for a and b, which is the best functional fit to the measured data.

$$\frac{\partial E}{\partial a} = \sum_{i=1}^{n} 2e_i \frac{\partial}{\partial a}\left(\frac{1}{P_i} - \left(a + b \cdot \frac{1}{I_i}\right)\right) \tag{11}$$

Thus, ∂E/∂a = 0 implies

$$\left(\sum_{i=1}^{n} \frac{1}{P_i}\right) = n \cdot a + \left(\sum_{i=1}^{n} \frac{1}{I_i}\right) \cdot b, \tag{12}$$

and

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{n} 2e_i \frac{\partial}{\partial b}\left(\frac{1}{P_i} - \left(a + b \cdot \frac{1}{I_i}\right)\right) \tag{13}$$

Thus, ∂E/∂b = 0 implies

$$\left(\sum_{i=1}^{n} \frac{1}{P_i}\frac{1}{I_i}\right) = \left(\sum_{i=1}^{n} \frac{1}{I_i}\right) \cdot a + \left(\sum_{i=1}^{n} \frac{1}{I_i^2}\right) \cdot b. \tag{14}$$

Thus, the best values for a and b satisfy the equations

$$(n) \cdot a + \left(\sum_{i=1}^{n} \frac{1}{I_i}\right) \cdot b = \left(\sum_{i=1}^{n} \frac{1}{P_i}\right), \tag{15}$$

and

$$\left(\sum_{i=1}^{n} \frac{1}{I_i}\right) \cdot a + \left(\sum_{i=1}^{n} \frac{1}{I_i^2}\right) \cdot b = \left(\sum_{i=1}^{n} \frac{1}{P_i}\frac{1}{I_i}\right). \tag{16}$$

This is just a conventional linear system with solution

$$a = \frac{c_{22}d_1 - c_{12}d_2}{c_{11}c_{22} - c_{12}c_{21}}, \quad b = \frac{c_{11}d_2 - c_{21}d_1}{c_{11}c_{22} - c_{12}c_{21}}, \tag{17}$$

where

$$c_{11} \equiv n, \tag{18}$$

$$c_{12} \equiv c_{21} \equiv \sum_{i=1}^{n} \frac{1}{I_i}, \tag{19}$$

$$c_{22} \equiv \sum_{i=1}^{n} \frac{1}{I_i^2}, \tag{20}$$

$$d_1 \equiv \sum_{i=1}^{n} \frac{1}{P_i}, \tag{21}$$

$$d_2 \equiv \sum_{i=1}^{n} \frac{1}{P_i}\frac{1}{I_i}. \tag{22}$$

Given these best estimates for a and b in terms of $(I_1, P_1), \ldots, (I_n, P_n)$, we have the following best estimate for P in terms of I:

$$P \approx \frac{I}{a \cdot I + b}. \tag{23}$$

where

$$a = \frac{\left(\sum_{i=1}^{n} \frac{1}{I_i^2} \cdot \sum_{i=1}^{n} \frac{1}{P_i}\right) - \left(\sum_{i=1}^{n} \frac{1}{fi} \cdot \sum_{i=1}^{n} \frac{1}{P_i \cdot I_i}\right)}{\left(n \cdot \sum_{i=1}^{n} \frac{1}{I_i^2}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i}\right)^2},$$

$$b = \frac{\left(n \cdot \sum_{i=1}^{n} \frac{1}{P_i \cdot I_i}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i} \cdot \sum_{i=1}^{n} \frac{1}{P_i}\right)}{\left(n \cdot \sum_{i=1}^{n} \frac{1}{I_i^2}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i}\right)^2}, \tag{24}$$

If the data points are in the Time Domain $(I_1, T_1), \ldots, (I_n, T_n)$, then, we can use the relationship of score to time,

$$P \propto \frac{1}{T} \tag{25}$$

And determine the best estimate for T in terms of I:

$$T \approx a + \frac{b}{I}. \tag{26}$$

$$a = \frac{\left(\sum_{i=1}^{n} \frac{1}{I_i^2} \cdot \sum_{i=1}^{n} T_i\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i} \cdot \sum_{i=1}^{n} \frac{T_i}{I_i}\right)}{\left(n \cdot \sum_{i=1}^{n} \frac{1}{I_i^2}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i}\right)^2},$$

$$b = \frac{\left(n \cdot \sum_{i=1}^{n} \frac{T_i}{I_i}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i} \cdot \sum_{i=1}^{n} T_i\right)}{\left(n \cdot \sum_{i=1}^{n} \frac{1}{I_i^2}\right) - \left(\sum_{i=1}^{n} \frac{1}{I_i}\right)^2}, \tag{27}$$

In Figure 15, we show the estimated performance curve compared to the measured curve for ATOM D525. The error margin between measured and estimated is <5% for all tested cases. This enables performance estimation for larger input sizes for ATOM D525. The same implementation can be used for Xeon processor using a new Xeon measured baseline.

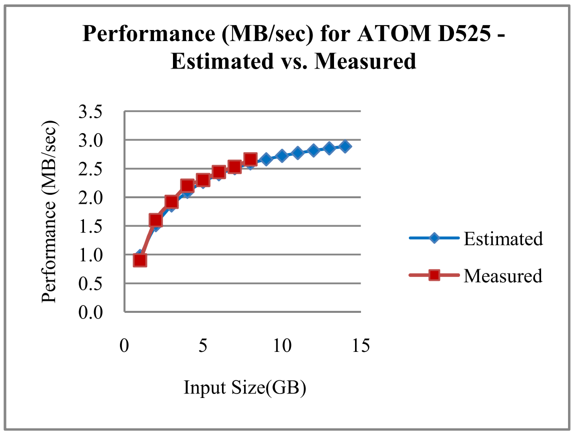

FIGURE 15. Performance estimation for ATOM D525.

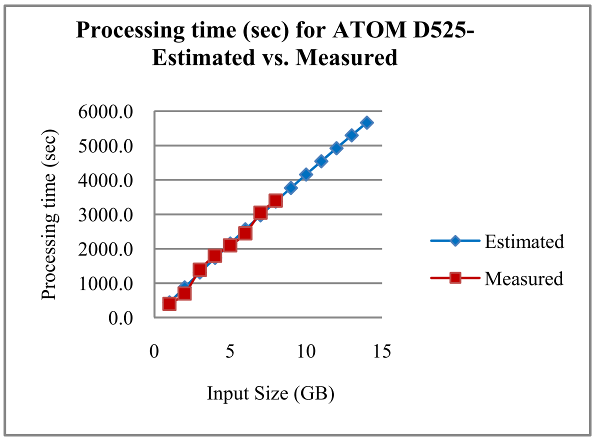

FIGURE 16. Processing time estimation for ATOM D525.

Using the same Amdahl's law regression model, we established a new processing time measured baseline for ATOM D525 and used Amdahl's law regression method to estimate processing time at higher input size value. The estimated line curve is linear as shown in Figure 16. An interesting finding is that running times of the Hadoop job nicely fit a least-square model of empirical data using Amdahl's Law regression method. The error margin between measured and estimated is <5% for all tested cases.

VII. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we presented a detailed performance evaluation and analysis for Hadoop WordCount workload using different processors such as Intel's ATOM D525, Xeon X5690, and AMD's BobCat E350. Our analysis shows that Hadoop WordCount is compute-bound workload in both map phase and reduce phase. The results show that enabling HT and increasing the number of sockets have a high impact on the Hadoop WordCount performance while memory speed and capacity does not affect performance significantly. We also conclude that the Intel's ATOM cluster can achieve a higher performance/watt compared to AMD's BobCat cluster at maximum performance. Comparing Intel's ATOM to Intel's Xeon X5690, the performance/dollar for Xeon is higher compared to the performance/dollar for ATOM. We also presented an estimation model that can estimate the total execution time with respect to input size change using Amdahl's law regression method. The estimation model average error was less than 5% compared to a measured data line. The method can be modified by establishing a new measured baseline and estimate the processing time (or performance) from that baseline to a different processor of the same architecture (i.e. different ATOM processor or different Xeon processor). The estimation method presented in this paper does not require a trace-based simulator, but it does require a code. Some future work guidelines are: implement the same evaluation for different cloud computing workloads that is in between the compute-bound and memory-bound so we can analyze the impact of memory speed and memory capacity on performance and execution time.

REFERENCES

[1] A. Baratloo, M. Karaul, Z. Kedem, and P. Wyckoff, "Charlotte: Metacomputing on the Web," in *Proc. 9th Int. Conf. Parallel Distrib. Comput. Syst.*, 1996, pp. 1–13.

[2] J. Bent, D. Thain, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Explicit control in the batch-aware distributed file system," in *Proc. 1st USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Mar. 2004, pp. 365–378.

[3] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier, "Cluster-based scalable network services," in *Proc. 16th ACM Symp. Oper. Syst. Principles*, Saint-Malo, France, 1997, pp. 78–91.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th Symp. Oper. Syst. Principles*, New York, NY, USA, 2003, pp. 29–43.

[5] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi, "Evaluating MapReduce on virtual machines: The Hadoop case," in *Proc. Int. Conf. Cloud Comput.*, vol. 5931. 2009, pp. 519–528.

[6] J. Issa and S. Figueira, "Graphics performance analysis using Amdahl's law: IEEE/SCS SPECTS," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst.*, Ottawa, ON, Canada, 2010, pp. 127–232.

[7] T. White. (2007). *Running Hadoop MapReduce on Amazon EC2 and Amazon S3*. [Online]. Available: http://developer.amazonwebservices.com/connect/entry.jspa?externalID=873&categoryID=112

[8] E. Vianna *et al.*, "Modeling the performance of the hadoop online prototype," in *Proc. 23rd Int. Symp. Comput. Archit. High Perform. Comput. (SBACPAD)*, Oct. 2011, pp.152–159.
[9] R. Stewart, "Comparing high level map reduce query languages," in *Proc. APPT*, 2011. [Online]. Available: http://www.macs.hw.ac.uk/~rs46/papers/appt2011/RobertStewart_APPT2011.pdf
[10] J. Issa and S. Figueira, "Hadoop and memcached: Performance and power characterization and analysis," *J. Cloud Comput., Adv. Syst. Appl.*, vol. 1, no. 1, pp. 1–20, Dec. 2012.
[11] P. Gohil, D. Garg, and B. Panchal, "A performance analysis of MapReduce applications on big data in cloud based Hadoop," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2014, pp. 1–6.
[12] 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 *Proc. Int. Conf. Adv. Cloud Big Data*, Dec. 2013, pp. 125–130.
[13] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun. (HPCC)*, Dalian, China, Sep. 2008, pp. 5–13.
[14] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The performance of MapReduce: An in-depth study," in *Proc. LDB Endow*, 2010, vol. 3. nos. 1–2, pp. 472–483.
[15] *Managing Hadoop Cluster*. [Online]. Available: https://developer.yahoo.com/hadoop/tutorial/module7.html, accessed Dec. 2015.
[16] K. R. Jackson *et al.*, "Performance analysis of high performance computing applications on the Amazon Web services cloud," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Washington, DC, USA, 2010, pp. 159–168.
[17] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: A survey," *ACM SIGMOD Rec.*, vol. 40, no. 4, pp. 11–20, Dec. 2011.
[18] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of Hadoop clusters," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 61–65, Jan. 2010.
[19] B. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini, "An energy case for hybrid datacenters," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 76–80, Jan. 2010.
[20] J. Dejun, G. Pierre, and C.-H. Chi, "EC2 performance analysis for resource provisioning of service-oriented applications," in *Proc. Int. Conf. Service-Oriented Comput.*, 2009, pp. 197–207.
[21] S. Krishnaprasad, "Uses and abuses of Amdahl's law," *J. Comput. Sci. Colleges*, vol. 17, no. 2, pp. 288–293, Dec. 2001.
[22] K. Hoste, L. Eeckhout, and H. Blockeel, "Analyzing commercial processor performance numbers for predicting performance of applications of interest," in *Proc. SIGMETRIC*, 2007, pp. 375–376.

JOSEPH A. ISSA received the B.Comp. degree in computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 1997, the M.S. degree in electrical engineering from San Jose State University, San Jose, CA, in 2000, and the Ph.D. degree in computer engineering from Santa Clara University, Santa Clara, CA, in 2012. Since 2013, he has been an Assistant Professor with the Computer and Electrical Engineering Department, Notre Dame University at Louaize, Lebanon. His research interests include processor architecture and performance modeling.