Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH **Open Access**

Hadoop and memcached: Performance and power characterization and analysis

Joseph Issa* and Silvia Figueira

* Correspondence: joseph_issa@yahoo.com
Department of Computer Engineering, Santa Clara University, Santa Clara, CA 95053-0566, USA

Abstract

Given the rapid expansion in cloud computing in the past few years, there is a driving necessity of having cloud workloads running on a backend servers analyzed and characterized for performance and power consumption. In this research, we focus on Hadoop framework and Memcached, which are distributed model frameworks for processing large scale data intensive applications for different purposes. Hadoop is used for short jobs requiring low response time; it is a popular open source implementation of MapReduce for the analysis of large datasets, while Memcached is a high performance distributed memory object caching system that could speed up throughput of web applications by reducing the effect of bottlenecks on database load. In this paper, we characterize different workloads running on Hadoop framework and Memcached for different processor configurations and microarchitecture parameters. We implement an analytical estimation model for performance and power using different server processor microarchitecture parameters. The proposed analytical estimation model uses analytical method to scale different processor microarchitecture parameters such as CPI with respect to processor core frequency. We also propose an analytical model to estimate power consumption scaling for different processor core frequency. The combination of both performance and power consumption analytical models enables the estimation of performance per watt for different cloud benchmarks. The proposed estimation models are verified to estimate power and performance with less than 10% error deviation.

Keywords: Performance estimation, Performance analysis, Power analysis, Power estimation, Cloud computing, Hadoop, Memcached

Introduction

With the continuing growth of web services, more servers are being added to data centers, also known as backend servers, to keep up with demand for cloud computing. These systems are scalable, manageable, and reliable in performing data-intensive requests. In this paper, we present performance and power characterizations and predictions for different cloud computing frameworks and workloads. We experiment with Memcached and Hadoop [1], which are mainly used by Google, Amazon, Yahoo, among others.. The main performance metric for these workloads is the latency to get a computation operation completed over a cloud network. Given the infrastructure of cloud networks, there are many factors contributing to the latency between clients and servers. The latency is related to the processor latency, Internet access latency, disk IO latency, and latency associated with moving data within a given cluster. In this paper, we characterize all of these latency factors and their contribution to the overall latency, and we compare different architectures, such as ATOM, Nehalem (NHM), and Westmere (WSM) Xeon processors.

In this paper, we also propose a performance and performance-per-watt analytical projection model. The model is verified to project performance and performance-per-watt with <10% error deviation between the measured and the projected data. The projection model is based on previous work published in [2,3], and we added the power factor in the regression model for the performance-per-watt projection. The latency associated with executing these workloads can be divided into three categories. The first category is related to workload characteristics such as data block size to be processed and threads requested by the client to the server. The second category is related to the processor microarchitectures, such as Cycle-per-Instruction (CPI), number of cores, number of threads, and memory latency due to Last Level Cache (LLC) misses, core frequency, and processor efficiency.

The performance and power projection models are based on the overall latency related to the backend server's processors. Performance-per-watt is defined as the rate of computation such as performance score, for every watt consumed. The power consumed by a computer is converted into heat, so the higher the wattage, the more cooling is required, which increases the cost for maintaining a given operating temperature. The objective is to achieve a higher performance per watt for a given workload.

The remaining sections of this paper are organized as follows: Section 2 is related work in which we review other published papers related to cloud performance and power characterization and evaluation. Section 3 is an overview and characterization of Hadoop framework and different workloads running on Hadoop MapReduce architecture in which we characterize performance-per-watt and performance-per-$. Then in section 4, we similarly characterize memcached workload on different server processor architectures. In Section 5, we present performance-per-watt characterization and analysis for disk IO. In Section 6, we present a detailed performance-per-watt projection analytical model and conclude in section 7.

Related work

Several papers on cloud workload characterization and optimization have been published. There are a few published papers on cloud computing performance prediction model, which is mainly related to the research presented in this paper. For instance, Vianna in [4] proposed an analytical model to predict performance for a Hadoop online prototype using intra-job pipeline parallelism with no reference to power consumption. In comparison with our analytical model, we project performance and performance-per-watt for Hadoop and Memcached from a measured baseline while changing one microarchitecture variable (e.g., core frequency and Cycles per Instruction (CPI). Our model predicts with <10% error deviation from measured numbers in all tested cases. It can be simply implemented without the need for a simulator or traces.

Xie in [5] focuses on the optimization of the MapReduce performance in heterogeneous Hadoop clusters. The paper shows performance improvements for placing data across multiple nodes so that each node has a balanced data processing performance.

The paper does not provide a prediction model to verify and estimate performance variations for different disks and processor architectures. The paper also does not analyze disk IO latency variation for different patterns, nor does it show any improvement in the power consumption associated with the proposed optimized data placing method.

Other work related to Hadoop performance includes Dejun and Chi. [6], who characterize response time and I/O performance. Ibrahim et al. [7] analyze Hadoop execution on virtual machines. Stewart [8] compares performance of several data query languages for Hadoop. All of these works focus on different aspects and approaches for performance analyses. Our work complements these previous works, as we also present a power analysis as well as a prediction method for performance and performance-per-watt, which is the focus of the research presented in this paper.

Leverich and Kozyrakis [9] presented a power model estimate for Hadoop cluster based on a linear interpolation of CPU utilization. Our power model is based on a regression prediction method. In addition, we present performance-per-watt to understand the ratio of performance relative to power for a given processor architecture.

Wiktor in [10] presented a comprehensive study related to Hadoop configuration parameters affecting query performance focusing on data size, number of nodes, number of reducers and other configuration variables. This study complements our characterization for various cloud workloads running on Hadoop framework. Our focus in this paper is performance, performance-per-watt and performance-per-$ characterization for different back-end server processors. We also propose a prediction method to project performance and performance-per-watt for different processor microarchitecture variables.

Jiang in [11] conducted an in-depth performance analysis for MapdReduce. The research presented optimization methods to improve performance. The research does not present the impact of this improvement with respect to performance-per-watt and performance-per-$.

Hadoop overview and characterization

Hadoop overview

Hadoop is a framework used to process large data sets in a distributed computing environment. The underlying architecture of Hadoop is HDFS (Hadoop Distributed File System). It provides fault-tolerance by replicating data blocks. The NameNode in Hadoop architecture stores information on data blocks, the DataNodes stores data blocks, and host Map-Reduce computation, and JobTracker is used to track jobs and detects failure. Hadoop is based on Google's MapReduce in which an application can break into small parts or blocks that can be run on any node so that applications can run on systems with thousands on nodes. Hadoop framework includes several benchmarks such as Sort, Word Count, Terasort, Kmeans iterations, and NutchIndexing. These benchmarks are based on distributed computing and storage. Apache Hadoop has an architecture that is similar to the MapReduce runtime used by Google. Apache Hadoop runs on the Linux operating system. Hadoop accesses data via HDFS (Hadoop Distributed File System), which maps all the local disks of the computing nodes to a single file-system hierarchy, allowing the data to be dispersed across all the data/computing nodes. HDFS also replicates the data on multiple nodes so that failures of nodes containing a portion of the data will not affect the computations that use that data.

The resource utilization for the benchmarks is categorized as IO-bound, CPU-bound, or in between. Table 1 summarizes the system resource utilization for each workload.

The disk I/O bandwidth limits the performance for the IO-bound benchmarks, so adding more disks may benefit performance. In addition, memory can be a performance-limiting factor for computation-bound workloads, such as Terasort, so adding more memory will increase memory buffers and will reduce the amount of data being moved back to disk. In addition, Memory is a limiting factor for Memcached, which we will discuss in a later section. There is a big split between CPU-bound versus memory-bound workloads. The most important characteristic affecting performance of any workload on any system is the number of main-memory transactions it does.

For CPU-bound workloads, performance is gated by activity on the processor. Important performance parameters are core frequency latency and 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.

In a Hadoop cluster, a master node controls a group of slave nodes by assigning tasks to the slave nodes based on their availability. In this section, we characterize the Hadoop framework based benchmarks performance and power on ATOM and Xeon-based systems:

ATOM D510:

- Core Frequency = 1.66 GHz, # of cores = 2, Threads/core =2, L2 cache size = 1 M, DDR2-667/800, Memory BW = 6.4 GB/s.

Xeon X5570:

- Core Frequency = 2.93 GHz, # of cores = 4, Threads/core = 2, Memory BW = 32 GB/s.

For a performance metric analysis, we consider the latency, i.e., completion time, as well as the throughput (tasks completed per unit time). For system power, we consider total average power during the entire execution. The basic configuration consists of 8 slaves and 1 master system all connected to one switch as shown in Figure 1. The slaves run TaskTracker and DataNode, while the master runs JobTracker and NameNode.

Each server in a Hadoop cluster can be configured to handle a specific capacity. JobTracker performs a specific task assignment, while NameNode maintains the HDFS, which requires high RAM capacity. TaskTracker performs the map-reduce task and

Table 1 Workloads based on Hadoop framework: System Resource Utilization

Workloads	System resource utilization
Wordsort	Sort Phase: IO-bound, Reduce Phase: Communication-bound.
Word Count	CPU-bound
Terasort	Map Stage: CPU-BoundReduce 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 iteration, IO-bound in clustering. It is used for machine learning and data mining.

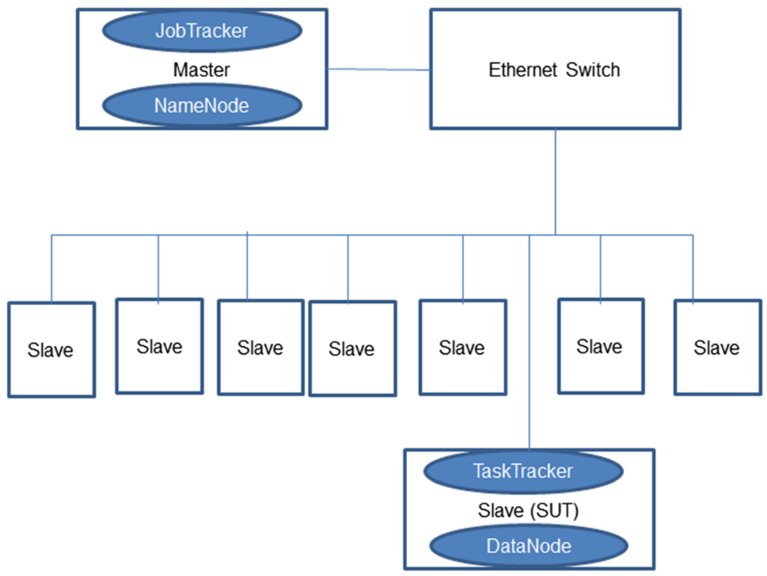

Figure 1 Hadoop master-slave configuration.

DataNode stores and handles read/write operations for HDFS. All Hadoop applications can be categorized as I/O-bound, compute-bound, or in-between. This makes it critical to have configurations with optimal memory size and number of processor sockets, and large numbers of hard drives. Data locality in Hadoop/MapReduce will determine its performance, as Hadoop usually distributes data blocks to multiple nodes based on disk space availability. This is a fair distribution in a homogenous cluster environment. In a heterogeneous computing environment, we have a combination of fast and slow nodes: the faster nodes will complete the processing of data faster than the slower nodes, and the slower nodes will have to transfer part of the data to the faster nodes for processing.

Hadoop characterization and measurements results

In this section, we characterize different benchmarks running on Hadoop framework for performance-per-watt and performance-per-$. The reason why this is important is to understand the benefits for favoring different processor architectures running on backend server in a cloud environment with respect to performance, power, and cost. Moreover, this characterization is used as a baseline for our projection model derived in later section. We specifically look at CPI and power data in characterization. These two variables are essential to derive the performance-per-watt projection model discussed in later section. The metrics we used for performance characterization are latency time, cost, microarchitecture parameters (such as CPI, memory latency and bandwidth) and power. For power, we measured processor power consumption during the entire run for a given workload with the data block size configured at 128 MB. The objective was to calculate performance and performance-per-watt for a specific configuration to establish the baseline needed for the projection. In Table 2, we show measured time for Hadoop framework based benchmarks and speed up ratio between NHM and ATOM processors.

From the measured data, we can conclude that the speedup for NHM ranges from ~3× to ~12× depending on the workload characteristics, the lower the speedup ration

Table 2 Hadoop framework based workloads - NHM vs. ATOM latency and throughput

Workloads	NHM time (sec)	ATOM time (sec)	Speedup Ratio (NHM vs. ATOM)	Throughput (tasks completed/min) NHM	Throughput (tasks completed/min) ATOM	Speedup Ratio(NHM vs. ATOM)
Wordsort	790	2901	**3.67X**	219.5	32.5	**6.75X**
Wordcount	455	5717	**12.5X**	156.1	12.3	**12.69X**
Terasort	3458	28891	**8.35X**	184.2	18.5	**9.95X**
NutchIndexing	218	1834	**8.4X**	-	-	**-**
Kmeans	940	11310	**12.03X**	-	-	**-**

for latency, the better. For throughput, the speedup ranges from ~6x to ~ 12x, and the higher the throughput ration, the better. Next, we measured the power for the same configuration of Hadoop using the same system setup. From the data shown in Table 3, we can conclude that NHM is better than ATOM in terms of performance-per-watt with the ratio ranging from ~0.8x to ~2x for all Hadoop workloads where

$$Performance - per - Watt = \frac{Speed\,up\,Ratio}{Power\,Ratio}, \qquad (1)$$

The performance-per-$ ranges from ~0.3x to ~1x. We would like to see higher performance-per-watt and higher performance-per-$ for a workload since, ideally, the objective is to lower watt and cost for a workload running on a given server. We also did performance, power, and cost assessment for WSM systems. We show that WSM is better than ATOM for CPU-heavy workloads in terms of performance-per-$. In addition, WSM shows further advantage over ATOM in terms of performance-per-watt as shown in Table 4.

Next, we considered microarchitecture parameters that affect the performance as shown in Table 5.

The microarchitecture parameters show that NHM has a lower CPI compared to ATOM for all benchmarks and higher memory bandwidth and lower Last Level Cache (LLC) misses. This shows the clear advantage of NHM over ATOM, which correlates to the conclusion based on performance and power numbers.

Note that Terasort is implemented as a MapReduce sort job with a custom partition. It uses a sort list of *n-1* sampled keys that define the key range for each reduction. Terasort is tested on the ATOM D525 processor and a 1.8 GHz core frequency with a two cores/four threads configuration. We used 10 GB and 100 GB data sizes in compressed and uncompressed modes with combinations of different map and reduction factors.

Table 3 Hadoop framework based workloads: NHM vs. ATOM performance-per-watt and Performance-per-$

Workloads	NHM Average power (W)	ATOM Average power (W)	Power Ratio	Performance-per-watt (NHM vs. ATOM)	NHM Server Cost ($)	ATOM Server Cost ($)	Performance-per-$ (NHM vs. ATOM)
Wordsort	190	44	4.3	**0.85X**	$7300	$650	**0.3X**
Wordcount	260	41	6.3	**1.98X**			**1.05X**
Terasort	215	42	5.11	**1.63X**			**0.7X**
NutchIndexing	220	41	5.36	**1.56X**			**0.7X**
Kmeans	250	42	5.95	**2.02X**			**1.04X**

Table 4 Performance, Price and Power efficiency for WSM vs. ATOM

Workloads	Performance			Price Efficiency				Power Efficiency			
	Job Running Time (sec)		Speedup (WSM vs. ATOM)	Server Cost ($)		Cost Ratio (WSM vs. ATOM)	Performance -per-$ Ratio (WSM-vs. ATOM)	Average Power (W)		Power Ratio (WSM vs. ATOM)	Performance –per-watt Ratio (WSM vs. ATOM)
	WSM	ATOM		WSM	ATOM			WSM	ATOM		
Wordsort	793.0	2948	**3.72X**	7307	628	11.64X	**0.32X**	192.1	44.16	**4.35X**	**0.85X**
WordCount	326.1	5767	**17.69X**				**1.52X**	260.46	41.48	**6.28X**	**2.82X**
Terasort	2837.7	28967	**10.21X**				**0.88X**	215.96	42.3	**5.11X**	**2.00X**
Nutch Indexing	178.7	1819	**10.18X**				**0.87X**	219.88	41.32	**5.32X**	**1.91X**
K-Means	668.3	11357	**16.99X**				**1.46X**	250.05	42.07	**5.94X**	**2.86X**

Typically, a large number of maps are better for performance (excluding power) for two reasons. First, it leads to low map re-execution in case of a failure, and, second, the computation to communication overlap is better. For the reduce phase, typically a 0.95% of the total reduce slot is optimum to reduce task. For the Terasort case in Figure 2, we can conclude that the compressed mode with one map and one reduction combination for 10 GB and 100 GB input size yields the best performance-per-watt.

To determine the time scaling with respect to input size, we used HadoopWordcount benchmark. The actual generated input is a bit different than the requested size. We used the real size (taken by "hadoopfs –dus"). The differences are small (1% –3%). For inputs that are big enough, we can determine a certain processing power of a system that is stable for a wide range of input sizes (10–50 GB) for both ATOM and WSM as shown in Figure 3 and Figure 4.

Hadoop Wordcount does not work very efficiently for small inputs (less than 10 GB for Xeon and 7 GB for ATOM). After 11 GB for Xeon, the execution time increases almost linearly with the input size. This makes the processing rate (Mbytes/second) almost constant. This observation is used to operate Hadoop within certain input sizes for optimized performance as shown in Figure 3 and Figure 4.

In summary for Hadoop characterization, the performance of NHM Server is 3.7× ~ 12.5× compared against the ATOM server, depending on the characteristics of the workloads. NHM server is better than ATOM server for most workloads in terms of performance-per-watt (except Wordsort), while is no better than ATOM in terms of performance-per-$. Microarchitecture metrics also show an advantage on NHM- server over ATOM server.

Memcached overview and characterization

Memcached overview

Memcached is a free open-source, high-performance, distributed-memory object caching system. Its architecture is based on distributed caching layer, which enables the aggregation

Table 5 Microarchitecture parameters for NHM vs. ATOM

Workloads	CPI		Memory BW(MB/sec)		Memory Read Latency(ns)		LLC cache misses/Byte	
	NHM	ATOM	NHM	ATOM	NHM	ATOM	NHM	ATOM
Wordsort	1.51	3.82	3500	1702	54.6	334.2	1.62	3.70
WordCount	1.43	3.04	9470	1192	57.2	275	3.49	5.7
Terasort	1.22	2.79	4710	1139	55.7	296	1.32	2.98

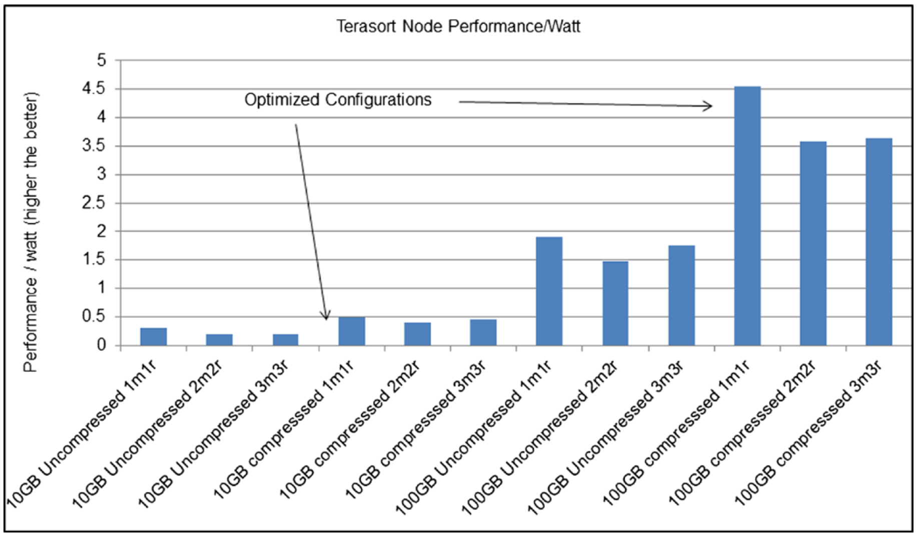

Figure 2 **Performance-per-watt for Terasort at different data sizes (compressed and uncompressed) for different map/reduce combinations.**

of spare memory from multiple nodes. It is typically used to cache database queries, which are very network intensive. It is intended for use in speeding up dynamic web applications by alleviating database load. It is used in large sites such as Facebook, Twitter, and YouTube. It can significantly reduce database load and is suitable for websites with high database loads. Memcached is an in-memory key-object store mechanism for small blocks of data from database, rendering, or API calls. It uses a simple text protocol. It utilizes simple operations such as get, insert, replace, delete, and append. There is one issue for using Memcached; it does not have built-in security features such as authentication to create a fast connection. This issue can be resolved by deploying a firewall and restricting access.

In general, RAM is much faster than disks and can provide higher bandwidths (>100X) with much lower latency (2000x), so cache is used to alleviate the load of slow backend disks, as shown in Figure 5.

Facebook hosts the world's largest Memcached installation by utilizing 800 Memcached servers creating a reservoir of 28 TB of memory enabling a 99% cache hit rate. Memcached packages data in RAM to clients and as data sizes grow, more RAM can be added to servers, as well as more servers can be added to the network. Berezecki, el al. in [12] proposed using high core count, low power consumption systems using TileraTILEPro64 processor

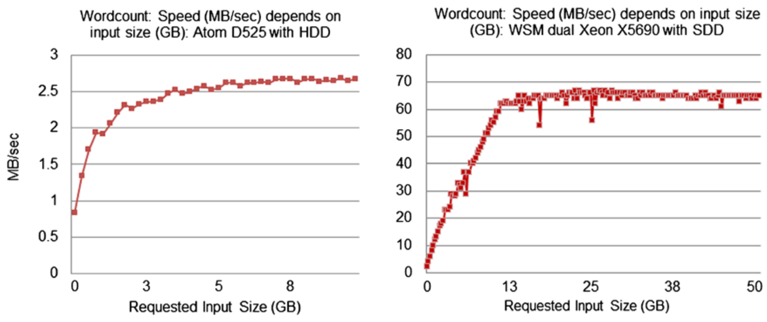

Figure 3 **Wordcount speed with respect to requested input size.**

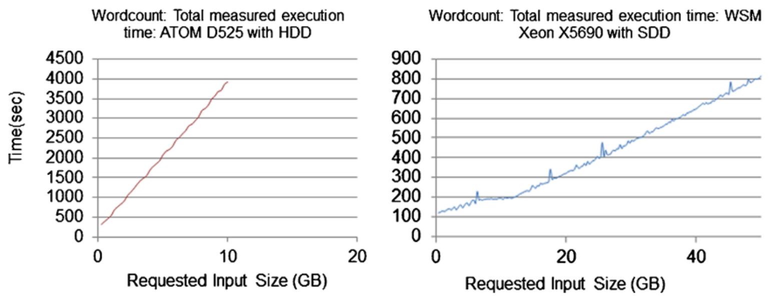

Figure 4 **Wordcount execution time with respect to requested input size.**

architectures for large key-value workloads, as shown in Figure 6. This will cause the CPU to be a bottleneck for wimpy-core-based systems, when the objects are small.

In Memcached clusters, there is no cross-communication among servers, only clients can communicate with the server. Client libraries may consist of PHP/C/JAVA/Python programs, as well as server lists. Clients select consistent hashing to select a unique server per key.

Memcached implements a routing algorithm shown in Figure 7, which consists of the standard modular "hash (key) mod n".

It also has consistent hashing that consists of hash nodes/keys in a continuum (circle), as shown in Figure 8. This provides more flexibility to add/remove a node. The hashing functions that may be used in Memcached are MD5/SHA1/CRC32/FVN. A client will do the routing and sending of requests, and serializing (may compress) of objects that can also provide compression and authentication. The Least Recently Used (LRU) algorithm is used in case storing data size exceeds the cache size, which then requires moving data out of cache.

The two main functions in Memcached are storing and getting data. The "Store" operation is usually transmitted over TCP to ensure the data is copied correctly with no errors, which requires more network bandwidth over large data size, while the "Get" operation can be done over UDP, which requires less network bandwidth but is also less secure.

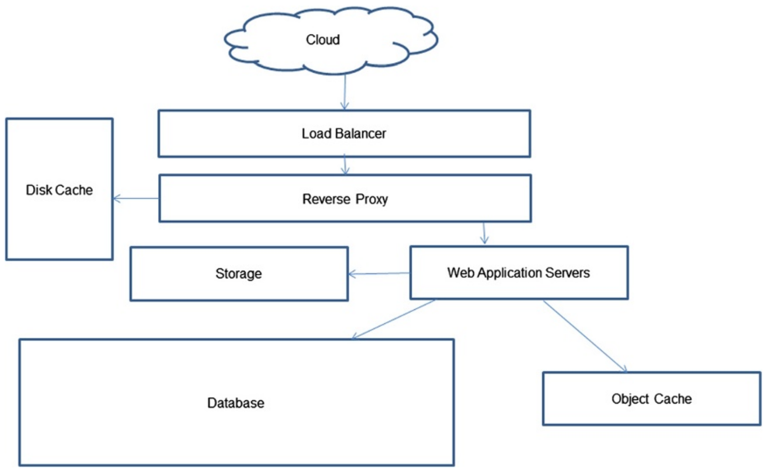

Figure 5 **Memcached configuration block diagram.**

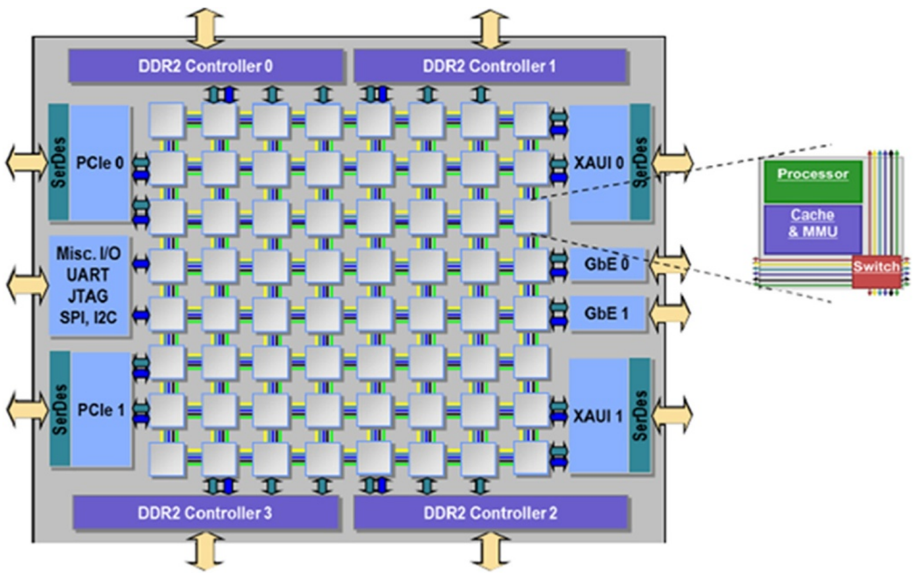

Figure 6 Many-Core, Key-Value Store, Source: Mateusz Berezecki, Facebook, and ILEPro64.

Memcached characterization and measurements

In this section, we characterize Memcached throughput with respect to power to determine the performance-per-watt. This characterization is used as a baseline for the projection model we derived in next section. For this experiment, we cover the System-Under-Test (SUT) and client step for characterization of Memcached. The SUT components are configured as follows:

SUT:

– 1 ATOM (D525/1P*2c/1.8 GHz/4 GB/1.80 GHz/82574 L–e1000e/i386)

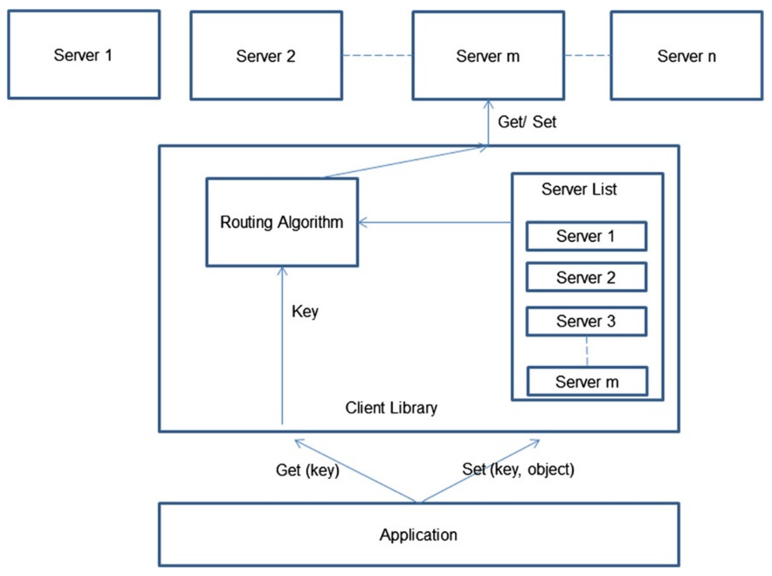

Figure 7 Memcached client-server routing path.

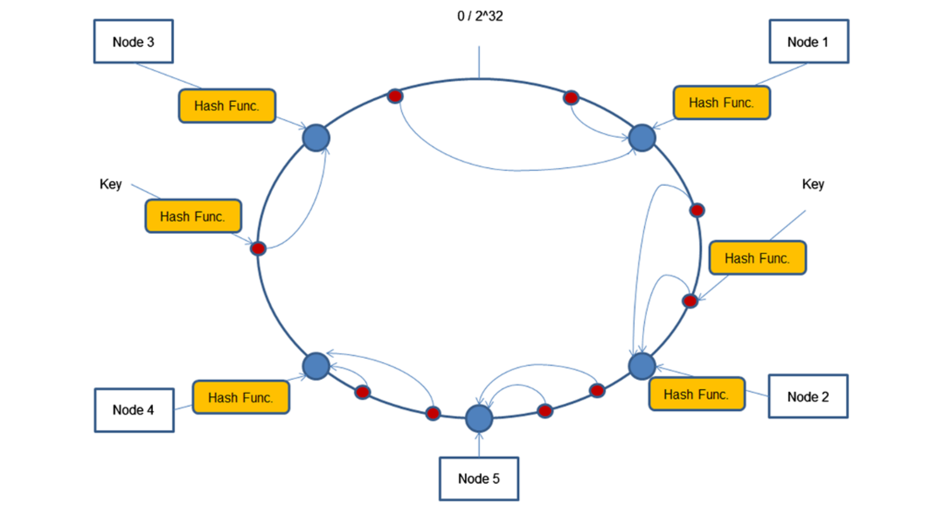

Figure 8 Memached hashing module.

- 1 WSM-EP (L5640 S5500WB/2P*6c/2.3 GHz/24 GB/2.27 GHz/82576-igb/X86-64)
- SMT ON
- OS: RHEL5.4 (updated to 2.6.35.4 for RPS/RFS patch)
- Memcached: 1.3.3 (partition patch)

Client:

- 7 WSM + 4 extra SNB
- Memcached: default 80 client threads
- Binary Protocol + Modular/Default Hashing
- Preload 100 K 64B* objects default
- Pure Get/Multi-Get Operations
- Persistent TCP connections
- Note that the item size used in Facebook is 64B.

1) ATOM -Threads and Partitions

In Figure 9, we show the latency and throughput obtained with various numbers of threads using a 64B Get function. We notice that latency is almost flat beyond four threads while the throughput is also actually flat beyond four threads. We can conclude that the optimum operating point for number of threads for ATOM is four threads. Any increase in number of threads will result in an increase in power consumption with no benefit in performance. Hence, this will lead to a negative performance-per-watt ratio, which is not desirable.

We also show latency and throughput for the various partition numbers. Latency decreases as number of partition increases, as shown in Figure 10. We conclude that beyond four or five partitions, the throughput is flat (~94,500 OPS), while latency shows some variation. Therefore, the optimum operation point is four to five partitions.

2) ATOM– Object Size

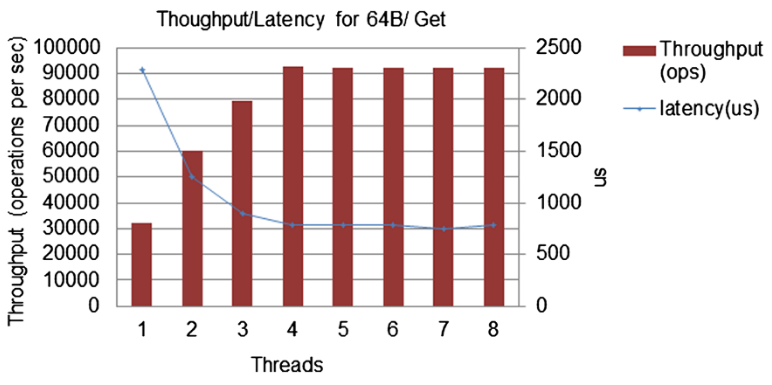

Figure 9 **Throughput and latency vs. number of threads.**

In Figure 11, we show latency for several data sizes in bytes. Latency increases exponentially beyond a data size of 2048B or higher using the Memcached Get function. The smaller the object, the lower the latency, and the lower the bandwidth, the lower the memory capacity requirements. Therefore, lower latency can be achieved by compressing the object size. Operating in an area where latency is increasing exponentially will reduce processor performance significantly.

Memcached can be scaled out simply by adding nodes, but adding nodes is not recommended because it increases the power consumption. Another issue is that each client needs to setup TCP connections to all nodes, which will lead to incast issues for multi-get operations, where latency increases as the number of clients requesting threads increases.

To confirm the latency observations we have seen in cloud clusters, we set up Memcached on one WSM machine and ran memslap (a traffic generator) from a different WSM system over a dedicated network. On the server side, Memcached defaults to four threads, while on the client side, we varied the number of requesting threads and the data size.

Figure 12 shows that the execution time (latency) increases exponentially with the problem size, particularly with lager data sizes as the number of requesting threads

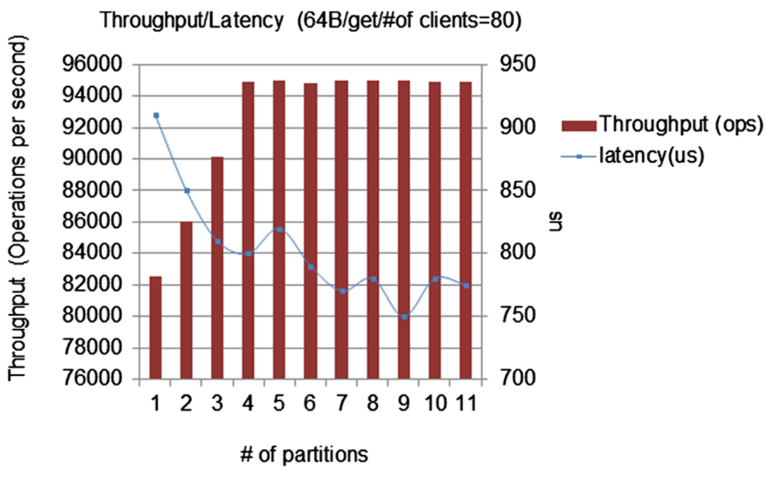

Figure 10 **Throughput and latency vs. number of partitions.**

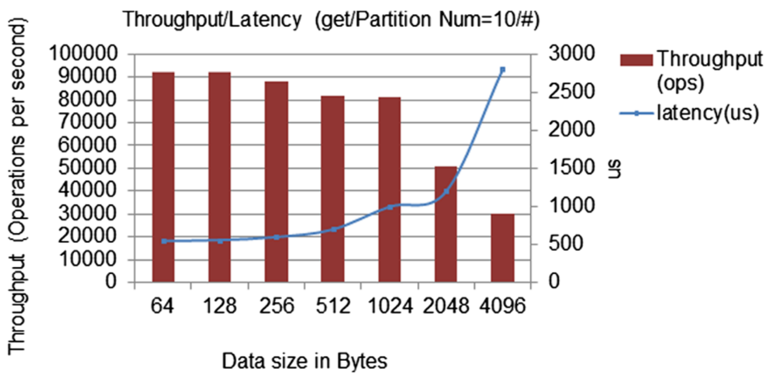

Figure 11 Throughput and latency vs. data object size.

increases beyond 32. This shows an optimized threshold to operate Memcached within below 32 threads to avoid exponential increase in execution time.

Next, we study the performance from the client side by increasing the number of threads while transmitting the same amount of data (1 M requests). The results show that there is no benefit to running more than 16 threads on the client side when generating traffic.

3) Memcached data results comparison table

From the measured performance parameters for power and microarchitecture, as shown in Table 6, WSM has lower CPI compared to ATOM and the performance per core is higher. This shows the clear advantage of WSM over ATOM for Memcached.

There is a significant difference in throughput ~14.79× between the two systems, for almost the same latency. There are several factors contributing to making the performance for WSM better than for ATOM D525. The first factor is that WSM has 24 threads (2 sockets * 6 cores * 2 threads/core), while ATOM has four threads. The second factor is that CPU utilization for ATOM is higher than for the WSM processor. The difference in core frequency and memory size is also a contributing factor. However, the increased performance for WSM comes at the expense of power consumption;

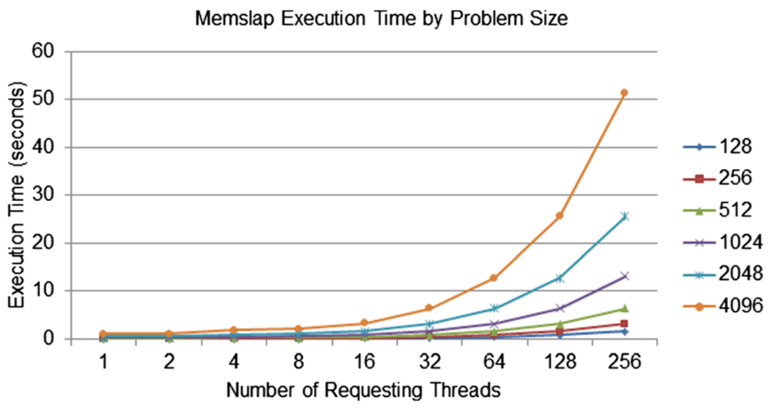

Figure 12 Memslap execution time by problem size with respect to requesting threads over direct network connection.

Table 6 Memcached performance, power, and microarchitecture parameters

	ATOM D525	**WSM-EP**
CPU	**1 Socket *2 Cores**	**2 Sockets *6 Cores**
	1.80 GHz	**2.27 GHz**
CPU Utilization	97.40%	84.80%
Memory	4 GB	24 GB
Throughput	95,336	1,410,475
Latency (us)	985	959
Per Node	1	14.79x
Per Socket	1	7.40x
Per Core	1	2.47x
Power (watt)	35.4	207.6
CPI	6.04	3.05

the power for ATOM is ~5.9× lower than the power consumption for WSM, and the price difference (performance-per-$) is an advantage for ATOM.

Recommendations to reduce performance bottlenecks

From the characterization results for Memcached, we identified three different kinds of bottlenecks. The CPU is the first bottleneck for wimpy core-based servers [12] when the objects are small. The second bottleneck is the network bandwidth if the object size is large enough. The third bottleneck is the cache user-level lock contention. This can be resolved or minimized by partitioning the hash table in a way in which each partition uses its own cache lock. Running multiple instances in single node can be another way to partition the big hash table. From the measured power data, we can also conclude that the WSM system is more power efficient than the ATOM D525 and provides higher power proportionality in a wimpy-core based server.

Disk IO performance and power evaluation

Several cloud workloads using Hadoop framework are IO-bound, which means that Disk IO performance becomes a bottleneck for achieving higher performance. Memcached is memory-bound workload, so disk IO performance-per-watt characterization may not be applicable for Memcached.

For example, in Hadoop, some data operations may not all fit in main memory, so disk IO operations are needed to complete the operation for specific servers with a small RAM. For such workloads, the disk latency is an important factor that affects performance.

In this section, we evaluate disk IO performance and power watts on the ATOM D525 and WSM EP X5660 systems used in our experiments. For this evaluation, we used a disk traffic generator tool to drive IO load with different IO parameters and collect IO performance. We used a performance-profiling tool to look at CPU utilization at different IO parameters. A power meter (Yokogawa) is used to collect power at idle and different load cases.

Disk IO evaluation methodology

In this section, we discuss the method used to evaluate disk IO on both processors. A few parameters will affect disk I/O behavior, and only the following combinations are selected in our experiment. The Read/Write ratio, where read-only is 100/0 and Write only is 0/100 for both the sequential and random pattern behaviors. The block size (KB) is 4 KB for random behavior and 32 KB for sequential behavior. The values for Queue Depth (QD) parameters for variable load used are 1, 2, 4, 8, 16, and 32. The performance indicator used for random pattern is IOPs and for sequential pattern is IOBW(KB/sec).

The IO performance and latency are well matched on ATOM D525 and WSM X5660 platforms with 3% better performance and 3% lower latency at QD = 1 in random write pattern. Latencies are proportional to QD for both patterns. Also, IO performance and latency are well-matched on ATOM D525 and Xeon X5660 platform, with a 9% worse performance and 2% higher latency at QD = 1 in sequential write pattern. Latencies are proportional to QD for both patterns.

Disk IO power efficiency

In this section, we show measured power and performance data for different pattern behaviors as shown in Table 7 and Table 8.

At peak performance (QD = 32), IOPS is the unified performance indicator for both random and sequential patterns. ATOM D525 shows much better performance-per-watts than WSM X5660 for all patterns. Sequential patterns show better performance-per-watt than random patterns on both platforms. In summary, similar disk behavior was noted on I/O performance of ATOM D525 and WSM-EP X5660, but much better performance-per-watt was seen for ATOM D525 than for WSM-EP X5660.

Performance-per-watt estimation model

In this section, we present an analytical model to project performance and performance-per-watt for Hadoop and Memcached. The baseline used for performance and power model derived in this section is based on characterization data discussed in previous sections. The model we published in [2,3] is based on Amdahl's law, which is implemented in a regression form to project performance for different workloads. In this paper, we use this model for CPI scaling with respect to change in core frequency for Hadoop and Memcached workloads; we also introduce the power factor in the projection model. First, we identify the microarchitecture variable affecting performance. The total execution time is a function of data block size, Cycles per Instructions (CPI), Path Length (PL), core frequency, processor efficiency, and number of cores:

$$\begin{aligned} Total\,Execution\,time = {} & (Data\,Size \times CPI \times PL \\ & \times Efficiency)/Core\,Frequency/of\ cores \end{aligned} \tag{2}$$

where

Table 7 Disk IO performance-per-watt for ATOM D525 Disk

ATOM D525	IOPS	Latency(ms)	CPU %	Watts	Performance-per-watt
Random Read	404	79.01	1.18	31.79	12.71
Random Write	452	70.70	1.46	31.56	14.32
Sequential Read	3686	8.88	10.79	33.25	110.88
Sequential Write	3686	8.85	16.04	34.75	106.09

Table 8 Disk IO performance-per-watt for WSM Disk

WSM X5660	IOPS	Latency(ms)	CPU %	Watts	Performance-per-watt
Random Read	400	79.97	0.17	155.51	2.57
Random Write	450	71.09	0.19	157.81	2.85
Sequential Read	3636	9.00	0.72	185.72	19.58
Sequential Write	3638	8.98	0.99	188.24	19.32

$$PL = a \times \ln(of\ cores) + b. \tag{3}$$

PL increases with the number of processes because of increased inter-process communication. Our measurement suggests that the number of instructions increases logarithmically with the number of processes, assuming PL is independent of the platform. The *a* and *b* variables for the path length are determined by curve fitting the total number of instructions retired relative to the number of cores scaling. The *a* and *b* variables are constant and change for different benchmarks. We used the Amdahl's law regression method published in [2] to analyze the CPI scaling with respect to higher core frequencies [13,14]. The projection model requires at least two measured data points to establish a measured baseline. This baseline is measured on a processor of similar architecture for the one to which we are projecting. For example, if our measured baseline is for the ATOM with two performance data points at two different core frequencies, we can use this baseline into the model to project performance for the same ATOM architecture but at higher core frequencies. In case we have to project for a different processor architecture family, a new measured baseline is required. We also derive the maximum performance a processor can achieve as core frequency increases to higher values.

CPI is one of the critical parameters affecting processor performance; we used the model to project for CPI scaling with respect to core frequency. We took two measured data points at two different frequencies for Memcached using ATOM and WSM systems. The regression model generates estimated CPI values at different core frequencies, as shown in Figure 13. The CPI for WSM is lower compared to CPI for ATOM, which is expected given the higher frequency and core count for the WSM processor.

Next, we transform the non-linear curves in Figure 13 to linear equations. For ATOM, we derive the equation y = -250x + 0.304 and for WSM y = -402x + *0.505*. At x = 0, we

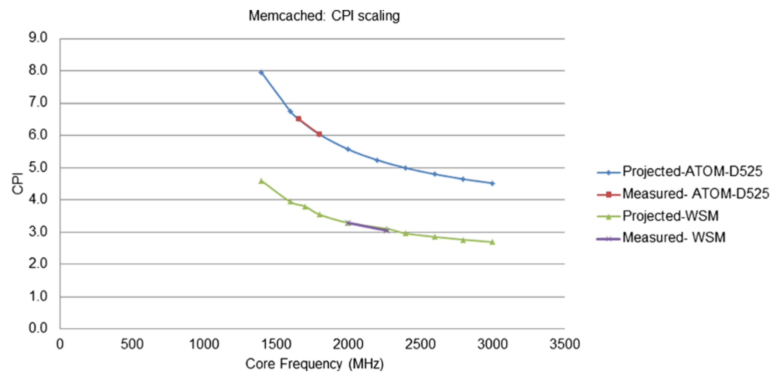

Figure 13 CPI scaling with respect to higher core frequencies.

get y = 0.304, taking the inverse $1/y$ =3.28. This means that the CPI will not reach the lower bound of 3.28 as frequency increases to much higher values. Similarly for WSM processors, we get $1/y$ = 1.98. The same method can be applied to project for total execution time using execution time as a baseline instead of CPI. We also use the model to project CPU performance for Hadoop framework based benchmarks as shown in Figure 14.

Using the same method as we did for Memcached, we transferred the curves to linear line equations, setting x = 0 and taking the inverse of y to determine the lower bound for CPI for each of the Hadoop workloads. The scaling for CPI at higher core frequencies is used to determine the change in total execution time Eq(2) given that all other variables remain fixed as the core frequency changes. We verified the model using the Hadoop Wordcount workload. The CPI projection values for different core frequencies were derived using the Amdahl's law regression method. For path length, the a and b values derived for word count workload were -31500 and 70702 respectively and the data size used was 128 MB. We projected the total execution time for ATOM D510, ATOM D525, NHM processors, as shown in Table 9 and Table 10.

The error deviation between projected and measured times is <10% for all three processors. In summary, the performance model consists of two sections. The first section is the Amdahl's law regression method [2] used to analyze the sensitivity curve for CPI at different core frequencies for a given processor architecture. The limitation for the Amdahl's law regression method is that each processor architecture family (i.e. ATOM or Xeon) needs a different measured baseline to be able to project for different core frequencies.

Once we obtain the CPI for a given frequency on a given processor architecture, we use that CPI in Eq(2) in which we have also to include the number of cores and the data size to predict the execution time. Each of the workloads requires its own path length equation. For example, in Hadoop, the Wordcount Path Length equation cannot be used for Wordsort, as each has its own Path Length equation. However, the same Path Length equation is common for different processor architectures because the Path Length is derived for a specific workload, not processor architecture. As indicated in Table 9, we used the same Path Length for Wordcount for both ATOM and NHM

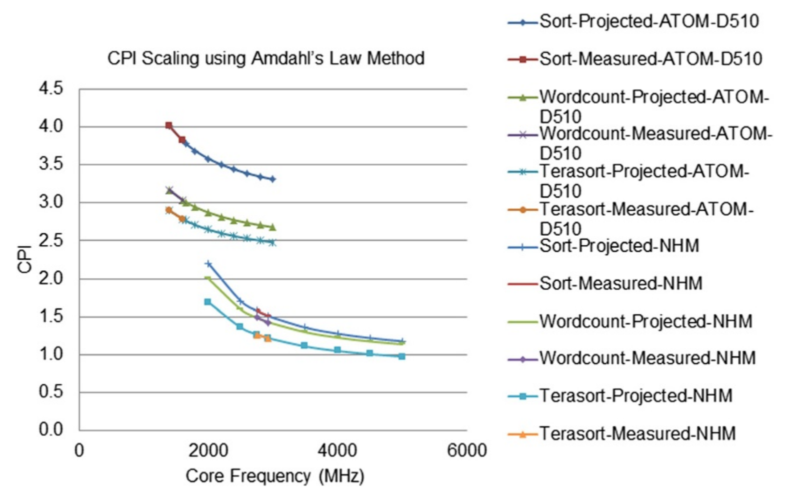

Figure 14 Hadoop framework based benchmarks on ATOM and NHM systems.

Table 9 Wordcount Time (sec): Projected vs. Measured

	ATOM-D510 @ 1.66 GHz	ATOM-D525@ 1.88 GHz	NHM@ 2.93 GHz
Measured	5717	4150	455
Projected	**5652**	**4491**	**431**
% Error	1.15%	7.61%	5.55%

which have different architectures, and the projection results presented $<10\%$ error deviation.

Power projection model

Energy consumption for data center is becoming a large component of operating costs. MapReduce manages large components of these datacenters, and Memcached became a requirement for cloud data storage and retrieval. Therefore, it is important to understand the power consumption for a given cluster configuration. It is also important to estimate processor power consumption relative to performance especially for future processors. This is where the performance-per-watt term becomes critical as described in [3]. The performance-per-watt is defined by the rate of computation delivered by the processor under test for every watt of power consumed. The energy dissipated in a processor is related to its supply voltage. The power consumption relation for CMOS devices is given by:

$$Power = k \times C \times V^2 \times Frequency, \tag{4}$$

Where k is an application specific constant, C is the total switching capacitance of the processor, V is the input voltage, which changes with frequency, and f is the core frequency. The dynamic power equation is a function of input voltage V and core frequency f. Note that f and V are directly proportional. The power projection equation is derived of the frequency fraction multiplied by the power difference:

$$\text{Pr}ojected\,Power = P_0 + (P_1 - P_0)\frac{f_1}{f}. \tag{5}$$

Where P_1 is the measured execution power at frequency f_1 and P_0 is the non-scale power. We can write P_0 in terms of a second measurement P_2 at f_2:

$$P_0 = \frac{P_2 f_2 - P_1 f_1}{f_2 - f_1}, \tag{6}$$

It is not desirable for a workload to operate in a negative performance per watt slope. The optimum solution is to operate at an increasing performance-per-watt slope. Positive slope for performance-per-watt means that the workload is gaining performance as the core frequency increases compared to the amount of power consumed associated with the increase in core frequency. A negative slope means that the system is consuming more power at a

Table 10 Wordsort time (sec): Projected vs. Measured

	ATOM-D510 @ 1.66 GHz	ATOM-D525@ 1.88 GHz	NHM@ 2.93 GHz
Measured	2901	2340	790
Projected	**2947**	**2446**	**793**
% Error	1.59%	4.36%	0.38%

higher rate than the performance gained as core frequency increases. For the Memcached performance-per-watt analysis, the baseline is established by taking the throughput to power ratio for two different frequency points, so the performance unit for Memcached is throughput, not time. We then used the regression model to project for throughput/power at higher core frequencies as shown in Figure 15, where the power-measured baseline is the total AC power for the system under test, which is expected to scale with core frequency. The throughput-per-watt for WSM is higher compared to ATOM processors. This is because the power ratio of WSM to ATOM is 5.9× and the performance ratio is 14.7×. As the core frequency increases up to 3000 MHz, the throughput/power ratio increases for both processors at a non-proportional rate, as shown in Figure 15.

In summary, the performance-per-watt model presented in this paper is based on performance prediction method derived and power prediction method as described earlier. The method shows the importance of how processor behavior will be at higher core frequencies by taking the ratio of projected performance relative to projected power. This enables the analysis of performance-per-watt for core frequencies we cannot measure or for different processors of similar architectures (i.e., higher core count, core frequency) that are not available in the market yet for measurement.

Conclusion

We presented a detailed performance and power analysis and characterization for Hadoop and Memcached workloads that led to identifying several bottlenecks that can be avoided to improve performance. The performance, cost and power analysis were implemented on different processor architectures such as WSM, NHM, and ATOM processors running on a backend server cluster. We identified several bottlenecks for performance and power in which optimum operating points are identified. In addition, we provided a comparison to show performance-per-$ between different processor architectures. Both performance-per-watt and performance-per-$ need to be minimized for an optimum solution in a cloud cluster. Furthermore, we proposed a projection analytical model to project performances and performance per watt with error deviation <10% between projected and measured data. More importantly, the projection model is flexible as it can be applied by establishing a CPI

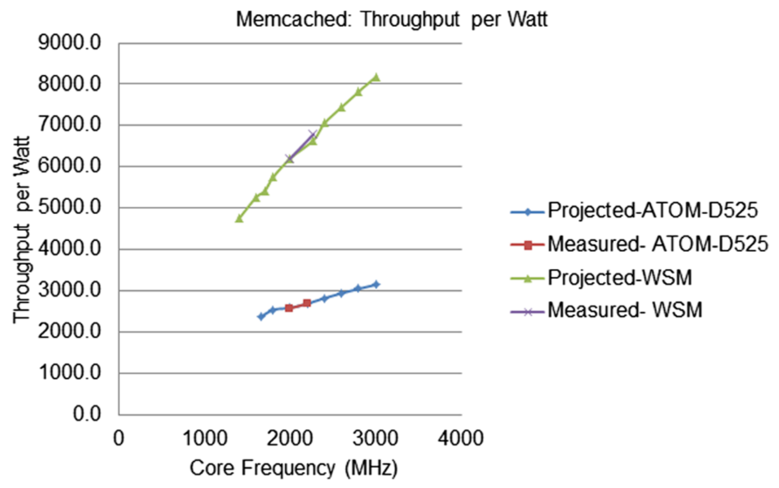

Figure 15 Throughput-per-watt for Memcached.

measured baseline for a given processor architecture and project from that CPI baseline to a different core frequency of the same processor architecture. The method does not require traces or simulations; it does require a code to implement.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The research presented in this paper is the result of teamwork. All authors read and approved the final manuscript.

Authors' Information
Joseph A. Issa received his B.E in computer engineering from Georgia Institute of Technology in 1996. He obtained his master's degree in computer engineering at San Jose State University in 2000. Currently he is a PhD candidate computer engineering major at Santa Clara University. His research interests are in areas of performance and power prediction, analysis and characterization.
Silvia M. Figueira received the B.S. and M.S. degrees in Computer Science from the Federal University of Rio de Janeiro (UFRJ), Brazil, and the Ph.D. degree in Computer Science from the University of California, San Diego. Currently, she is an Associate Professor of Computer Engineering at Santa Clara University. Her research interests are in the areas of performance and energy consumption modeling and prediction.

Received: 5 April 2012 Accepted: 7 June 2012
Published: 12 July 2012

References
1. Apache Software Foundation: *Official apache hadoop website*. 2011. http://hadoop.apache.org.
2. Issa J, Figueira S: *Graphics Performance Analysis Using Amdahl's Law: IEEE/SCS SPECTS*. Ottawa, Canada: International Symposium on Performance Evaluation of Computer and Telecommunication System; 2010.
3. Issa J, Figueira S: *Performance and power-consumption analysis of mobile internet devices*. Orlando, Florida: IEEE IPCC–International Performance Computing and Communications Conference; 2011.
4. Vianna E: *Modeling performance of the hadoop online prototype*. Espirito Santo: International Symposium on Computer Architecture; Vitoria; 2011.
5. Xie J: *Improving Map Reduce performance through data placement in heterogenous Hadoop clusters*. Atlanta, Georgia, USA: IEEE International Symposium on Parallel & Distributed Processing; 2010.
6. Dejun J, Chi GPC: *EC2 performance analysis for resource provisioning of service-oriented applications*. Stockholm, Sweden: International Conference on Service-Oriented Computing; 2009.
7. Ibrahim S, Jin H, Lu L, Qi L, Wu S, Shi X: *Evaluating MapReduce on virtual machines: The hadoop case*. Beijing, China: International Conference on Cloud Computing; 2009.
8. Stewart R: *Performance and Programmability of High Level Data Parallel Processing Languages*: 2010. http://www.macs.hw.ac.uk/~rs46/files/publications/MapReduce-Languages/Complete_Results_Chapter.pdf.old.
9. Leverich J, Kozyrakis C: *On the energy (in)efficient of Hadoop clusters, ACM SIGOPS Operating systems Review*. New York:; 2010:61–65.
10. Wiktor T, *et al*: *Performance analysis of hadoop for query processing*. Biopolis: IEEE,International Conference on Advanced Information Networking and Applications; 2011.
11. Jiang DR, Ooi BB, Shi L, Wu S: The performance of mapreduce: an in-depth study. *Proceedings of the Very Large Database Endowment* 2010, **3**(1-2):472–483.
12. Berezecki M, Frachtenberg E, Paleczny M, Steele K: *Many-Core Key-Value Store*. Orlando, Florida: International Green Computing Conference and Workshops (IGCC); 2011.
13. Hennessy JL, Patterson DA: *Computer architecture: A quantitative approach*. 4th edition. Morgan Kaufmann: Elsevier; 2007.
14. Hennessy JL, Patterson DA: *Computer organization & design: The hardware/software interface*. 4th edition. Morgan Kaufmann: Elsevier; 2009.

doi:10.1186/2192-113X-1-10
Cite this article as: Issa and Figueira: Hadoop and memcached: Performance and power characterization and analysis. *Journal of Cloud Computing: Advances, Systems and Applications* 2012 **1**:10.