This paper appears in the 2017 IGSC Invited Papers on Emerging Topics in Sustainable Memories

Understanding the Role of Memory Subsystem on Performance and Energy-Efficiency of Hadoop Applications

Hosein Mohammadi Makrani[1], Shahab Tabatabaei[2], Setareh Rafatirad[1], Houman Homayoun[1]

[1]*George Mason University,* [2] *Rayanmehr Company*

{hmohamm8, srafatir, hhmoayou}@gmu.edu

***Abstract* — The memory subsystem has always been one of the performance bottlenecks in computer systems. Given the large size of data, therefore, the questions of whether Big Data requires big memory and whether main memory subsystem plays an intrinsic role in the performance and energy-efficiency of Big Data are becoming important. In this paper, through a comprehensive real-system experimental analysis of performance, power and resource utilization, we have evaluated main memory characteristic of Hadoop MapReduce, a de facto standard for big data analytics. Through a methodical experimental setup we have analyzed the impact of DRAM capacity, operating frequency, and the number of channels on power and performance to understand the main memory requirements of this important Big Data framework. The characterization results across various Hadoop MapReduce applications from different domains illustrate that Hadoop MapReduce workloads show two distinct behaviors of being either CPU-intensive or Disk-intensive. Our experimental results showed that DRAM frequency as well as number of channels do not play a significant role on the performance of Hadoop workloads. On the other hand, our results indicate that increasing the number of DRAM channels reduces DRAM power and improves the energy-efficiency of Hadoop MapReduce applications.**

Keywords— DRAM characterization; Hadoop MapReduce; performance; power

I. INTRODUCTION

Big Data refers to the data that is massive in volume and variety as well as the velocity and veracity for processing [1]. Hadoop MapReduce [2] has been considered as a dominant framework for Big Data as it supports scalable storage and computing resources for Big Data. Therefore, it is important to understand the behavior of memory subsystem for this class of applications to answer the important question of whether Hadoop applications requires large and high performance memories. Recently, there have been a number of efforts to understand the behavior of Big Data applications by benchmarking and characterizing them on the fastest and largest possible memory subsystem [3][5][12] [14][15][20]. Most of prior studies have focused on the CPU parameters such as number of cores, CPU frequency, and cache size, performing network or disk analysis to understand Big Data application behavior.

The objective of this work is to evaluate the effect of memory subsystem on the performance and power consumption of Hadoop MapReduce applications. In order to perform memory subsystem analysis, we have investigated three important configurable memory parameters including memory capacity, memory frequency, and number of memory channels, to determine how these parameters affect the performance and power consumption of Hadoop MapReduce applications.

Our evaluation shows that Hadoop MapReduce applications do not require a high-end memory subsystem to improve the performance. Increasing memory subsystem parameters beyond 16 GB, 1333 MHz Frequency and a single channel does not enhance Hadoop performance noticeably. In addition, to understand whether our observations on memory subsystem behavior remains valid when changing microarchitecture parameters, we performed further architectural study to understand the impact of increasing core count, cache size and processor operating frequency on memory behavior.

Based on the micro-architectural analysis, this paper makes the following observations: 1) Increasing the number of cores beyond 6 cores/node does not enhance performance as it increases the number of disk accesses, 2) As the cache capacity increases, the accesses to DRAM memory reduces; therefore in future architectures with larger cache capacity, we anticipate that there won't be a major benefit of using high bandwidth DRAM for Hadoop applications 3) Increasing operating frequency of the processor does not improve the performance and energy-efficiency of the system as most Hadoop applications are I/O intensive 4) Emerging DRAM memory technologies such as HMC, HBM, and DDR5 which offers high bandwidth are not going to bring noticeable performance benefits for Hadoop applications.

II. RELATED WORKS

A. Memory characterization

A recent work on Big Data [3] profiles the memory access patterns of Hadoop and noSQL workloads by collecting memory DIMM traces using special hardware. This study does not examine the effects of memory frequency and number of channels on the performance of the system. A more recent work [4] provides a performance model that considers the impact of memory bandwidth and latency for Big Data, high performance, and enterprise workloads. The work in [5] shows how Hadoop workload demands different hardware resources. This work also studies the memory capacity as a parameter that impacts the performance. However, as we will discuss later in this work, their finding is in contrast with ours. In [6]

978-1-5386-3470-7/17/$31.00 ©2017 IEEE

the authors evaluate contemporary multi-channel DDR SDRAM and Rambus DRAM systems in SMT architectures. The work in [11] mainly focuses on page table and virtual memory optimization of Big Data and [12] presents the characterization of cache hierarchy for a Hadoop cluster. These works do not analyze the DRAM memory subsystem. In addition, several studies have focused on memory system characterization of various non Big Data workloads such as SPEC CPU or parallel benchmark suites [7, 8]. Hajkazemi et al. explored the performance of Wide I/O and LPDDR memories [10]. Tran et al. worked on heterogenous memory management [9]. Zhao et al. 3D memory architecture with stacked DRAM [31].

B. Big Data characterization

A recent work introduces a new Big Data benchmark suite for spatio-temporal data and analyzes the redundancy among different Big Data benchmarks such as ICTBench, HiBench and traditional CPU workloads [13]. The work in [14] selects four Big Data workloads from the BigDataBench [21] to study I/O characteristics, such as disk read/write bandwidth, I/O devices utilization, average waiting time of I/O requests, and average size of I/O requests. Another work [15] studies the performance characterization of Hadoop and DataMPI, using Amdahl's second law. This study shows that a DataMPI is more balanced than a Hadoop system. In a more recent work [16] the authors analyzes three SPEC CPU2006 benchmarks (libquantum, h264ref, and hmmer) to determine their potential as Big Data computing workloads. The work in [17] examines the performance characteristics of three high performance graph analytics. One of their findings is that graph workloads fail to fully utilize the platform's memory bandwidth. In a recent work [18], Principle Component Analysis is used to detect the most important characteristics of Big Data workloads from BigDataBench. To understand Spark's architectural and micro-architectural behaviors, a recent work evaluates the benchmark on a 17-node Xeon cluster [19]. Their results show that Spark workloads have different behavior than Hadoop and HPC benchmarks. Again, this study does not consider the effect of memory subsystems on Big Data. The work in [20] performs performance analysis and characterizations for Hadoop K-means iterations. This study also proposes a performance prediction model in order to estimates performance of Hadoop K-means iterations, without considering the memory requirements. Malik et al. characterized Hadoop applications on big-little cores and microservers [28, 29, 30].

III. Experimental Setup

A. Workloads

In our experiments, we used several Hadoop workloads from BigDataBench [21] and HiBench [23] suites including micro kernels, graph analytics, e-commerce, machine learning, web search, and analytical query domains. These workloads are presented in Table 1.

B. Software stack

In this study, we use Hadoop MapReduce (version 2.7) as our software platform installed on Linux Ubuntu (14.2) operating system. Hadoop (Apache) allows for distributed processing of large data sets scaling from one node to thousands of nodes.

C. Hardware platform

To have a comprehensive experiment we used different SDRAM memory modules, shown in Table 2. All modules are from the same vendor. For running the workloads, and monitoring statistics, we used a six-node server with detailed characteristics presented in table 3. While network overhead in general is influencing the performance of studied applications and therefore the characterization results, for big data applications, as shown in a recent work [27], a modern high speed network improves the performance only a small 2% performance. We therefore used a high speed 1 Gbit/s network to avoid making it a performance bottleneck for this study.

D. Methodology

Our experimental methodology is focused on the objective of understanding how Hadoop MapReduce uses the memory subsystem. For this goal we used Intel Performance Counter Monitoring tool (PCM) [22] to understand memory as well as processor behavior. In our experiments, we collect OS-level performance information with DSTAT tool. Some of the metrics that we used for study are memory footprint, memory bandwidth, L2, and Last Level Cache (LLC) hits per instruction (HPI), instruction per cycle (IPC), core C0 state residency, and power consumption. For our experiments, we swept the processors' parameters when using memories with three different frequency setting of 1333 MHz, 1600 MHz, and 1866 MHz. We repeat each experiment for different number of memory channels (1CH, 2CH, and 4CH).

IV. Result and Discussion

A. Memory analysis

1) Effect of memory channels: The off-chip memory peak bandwidth equation is shown in EQ. (1).

Table 1: Studied workloads

Workload	wordcount	sort	grep	terasort	bayes	naïve bayes	kmeans	pagerank	aggregation	join	scan
Domain	micro kernel	micro kernel	micro kernel	micro kernel	e-commerce	e-commerce	machine learning	websearch	analytical query	analytical query	analytical query
Input type	text	data	text	data	data	data	graph	data	data	data	data
Input size (huge)	1.1 T	178.8G	1.1 T	178.8G	30.6G	30.6G	112.2G	16.8G	10.8G	10.8G	10.8G
Input size (large)	183.6G	29.8G	183.6G	29.8G	5G	5G	18.7G	3.1G	1.8G	1.8G	1.8G
Input size (medium)	30.6G	3G	30.6G	3G	1.6G	1.6G	3.7G	1.3G	1G	1G	1G
Suite	BigDataBench	BigDataBench	BigDataBench	HiBench	HiBench	BigDataBench	HiBench	HiBench	HiBench	HiBench	HiBench

Table 2: Memory modules' part numbers and parameters studied in this work

DDR3	4 GB	8 GB	16 GB	32 GB
1333 MHz	D51264J90S	KVR13R9D8/8	KVR13R9D4/16	---
1600 MHz	D51272K111S8	D1G72K111S	D2G72K111	---
1867 MHz	KVR18R13S8/4	D1G72L131	D2G72L131	KVR18L13Q4/32

$$Bandwidth = Channels \times Frequency \times Width \quad \text{EQ. (1)}$$

Our result shows that increasing the number of channels does not reduce the execution time (at most 4%) of Hadoop applications, presented in Figure 1. Therefore, increasing the memory bandwidth is not a good option to improve the performance.

2) Effect of DRAM frequency: As results in Figure 2 show, similarly we don't observe any major improvement of execution time when increasing memory frequency from 1333 MHz to 1866 MHz. Previous section showed that 4X increase in the bandwidth resulted by increasing the number of channels from 1 to 4 can gain only 4% performance benefit. Therefore, it is clear why a 1.4X increase in bandwidth as a result of frequency increase cannot increase the performance noticeably.

This finding may mislead to use the lowest memory frequency for Hadoop applications. Based on EQ. (3), read latency of DRAM depends on the memory frequency.

$$Read\ latency = 2 \times (CL / Frequency) \quad \text{EQ. (3)}$$

However, for a DDRx technology (DDR4 or 5), this latency is set fixed by the manufacturer with controlling CAS latency (CL). This means two memory modules with different frequency (1333 MHz and 1866 MHz) and different CAS Latency (9 and 13) can have the same read latency of 13.5 ns, but provide different bandwidth per channel (10.66 GB/s and 14.93 GB/s). Hence, as along as reduction of frequency does not change the read latency, it is recommended to reduce DRAM frequency for Hadoop applications. Furthermore, we demonstrated that Hadoop applications do not require a high bandwidth off-chip memory, therefore they do not require a high frequency memory as well. The reason is that Hadoop workloads are not memory intensive. Later in this paper we will discuss the insensitivity of Hadoop applications to main memory parameters.

Table 3: Hardware Platform

Hardware Type	Parameter	Value
Motherboard	Model	Intel S2600CP2
CPU	Model	Intel Xeon E5-2650 v2
	# Core	8
	# Threads	16(disabled)
	Base Frequency	2.6
	Turbo Frequency	3.4
	TDP	95
	L1 Cache	32 * 2 KB
	L2 Cache	256 KB
	L3 Cache	20 MB
	Memory Type Support	DDR3 800/1000/1333/1600/1867
	Maximum Memory Bandwidth	59.7 GB/S
	Max Memory Channels supported	4
Disk (SSD)	Model	HyperX FURY
	Capacity	480 GB
	Speed	500 MB/S
Network Interface Card	Model	ST1000SPEXD4
	Speed	1000 Mbps

3) Effect of memory capacity: Figure 3 presents the impact of memory capacity per node on the average execution time of workloads. The results indicate no significant benefit of using a high capacity DRAM. In order to evaluate the effect of

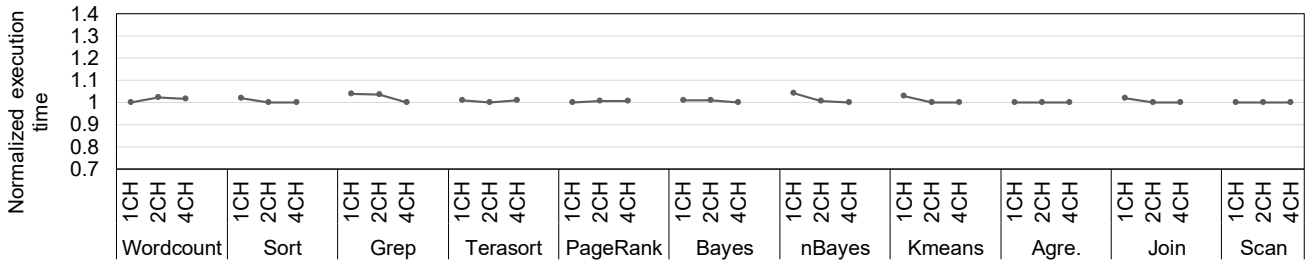

Figure 1: Effect of memory channel on the execution time

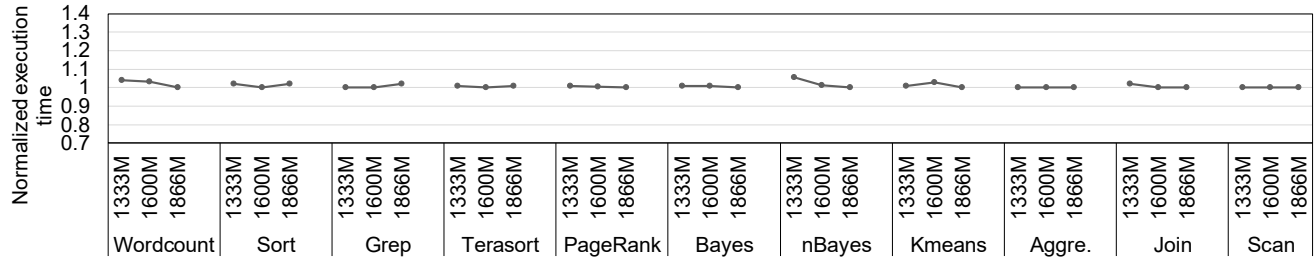

Figure 2: Effect of memory frequency on the execution time

DRAM capacity on Hadoop MapReduce workloads performance, we examined 7 different memory capacities from 4 GB to 64 GB size. Considering the workloads, doubling the capacity (from 4 GB to 8 GB) only reduces the execution time by less than 5%, on average. Beyond 16 GB, DRAM capacity does not affect the performance. This is due to the fact that Hadoop is not an in-Memory system framework and it is more relying on the Disk for operation and data storage.

4) Data size implication on memory capacity usage:

Hadoop uses disk as storage and rely on a cluster of servers to process data in a distributed manner. The ability of MapReduce frameworks is that each map task processes one block of data on HDFS at a time. Hence, this relieves the pressure of large input data on the memory subsystem. Therefore, regardless of input size, the memory subsystem utilization remains almost constant in these frameworks. We have performed experiments with three different input data sizes, namely medium, large, and huge data. Figure 4 shows the memory usage of Big Data frameworks as a function of input data size. Our result shows that the standard deviations of Hadoop memory usage are less than 250 MB and regardless of input data size, the average memory usages and maximum memory usages are close.

5) Data size implication on memory bandwidth usage: Our results presented in Figure 4 reveal that the size of input data does not noticeably change the memory behavior of Hadoop Framework. This is because of the fact that the memory bandwidth usage depends on the cache miss ratio (more on this later in the paper). Cache behavior is more application dependent rather than data-size dependent. Consequently, by increasing the size of input, the cache hit ratio remains almost the same. Therefore, while increasing the input size increases the job completion time, the DRAM bandwidth requirements of applications do not change noticeably.

B. Architectural analysis

1) Workload classification: As the main goal of this paper is to study the combined impact of node architecture and workload characteristics, it is important to first classify those workloads. To this goal, we divided workloads into two major groups of CPU intensive and I/O intensive. Our decision criteria for this classification is based on the average Disk bandwidth usage as has been shown in Figure 5. We classify Sort, Grep, and Scan applications to be Disk-intensive while others to be CPU-intensive.

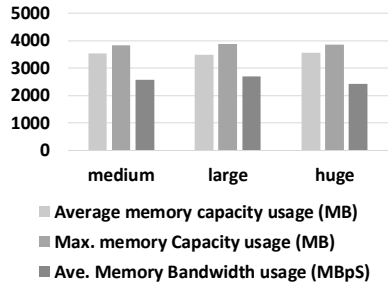

Figure 4: Impact of the size of input data on memory usage

2) Number of cores per node implication: Figure 6 demonstrates the effect of increasing the number of cores on each node on the performance of two groups. For CPU intensive applications and when the core count is less than 6, the performance improvement is close to the ideal case. The interesting trend is that increasing the number of cores does not improve the performance of Hadoop applications noticeably beyond 6 cores. As the increase in the number of cores increases the number of accesses to the disk, the disk becomes the bottleneck of the system. At 8 cores, the CPU utilization is dropped to 15% for I/O intensive applications. It is important to note that our Disk is SSD (Solid State Drive) and the performance benefit could have diminished at even lower number of cores if a HDD drive was used. We also projected the performance of 12 and 16 cores by regression model derived from our experimental results.

4) Cache implication: Xeon processor has a 3-level cache hierarchy. Figure 7 shows Hadoop applications' cache hit rates for level 2 (L2) and last level cache (LLC). This Figure reveals an important characteristic of Hadoop applications. Contrary to the simulation's results in recent work reporting cache hit ratio to be below 10% for Big Data applications [3], our real-system experimental result shows Hadoop applications have a much higher cache hit ratio, which helps reducing the number of accesses to the main memory. On the other hand, the average LLC hit rate for Hadoop workloads are found to be 55% which implies that these applications are cache friendly. The reason of high cache hit ratio is that each parallel task of Hadoop framework processes data in a sequential manner. This behavior increases the chances of a cache hit; therefore, it prevents excessive access to DRAM and eliminates the necessity of using a high bandwidth memory. Consequently, CPUs with larger LLC will show similar characteristics. Therefore, we can conclude that due to cache behavior, Hadoop MapReduce applications are not memory

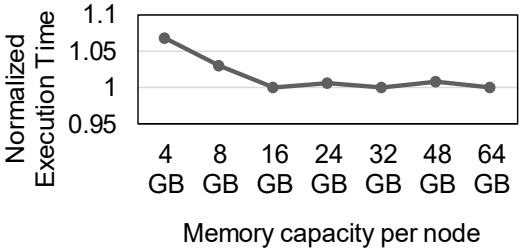

Figure 3: Impact of Memory capacity on the execution time

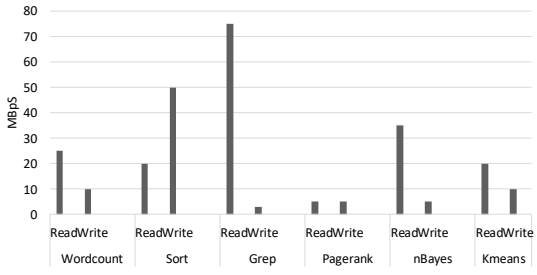

Figure 5. Disk access of Big Data applications

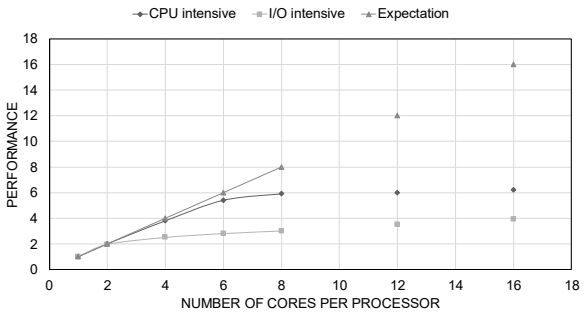

Figure 6: The number of core per processor's effect on the performance

intensive. This explains why increasing the memory frequency and increasing the number of channels cannot improve the performance of Hadoop applications. This finding helps the server designers to avoid over provisioning the memory subsystem for Hadoop MapReduce applications. Therefore, we anticipate that in future architectures with higher cache capacity, the observations made in previous section regarding memory behavior of Hadoop applications remain valid.

C. Power analysis

Figure 8 reports the DRAM power consumption. Our result shows that by increasing the frequency of DRAM from1333 MHz to 1866 MHz, the power consumption increases by almost 15%. This verifies that the major component of DRAM power is the static power. However, increasing the number of channels reduces the power consumption. An important observation is that a memory with four channels consumes 40% less power than a memory with single channel. This is due to the fact that DRAM channels are designed to increase the efficiency of interleaving. As a result, the memory controller can manage accesses more efficiently, which in turn reduces the power consumption. Despite the small/no impact on performance, the number of memory channels significantly affects power consumption.

Figure 9 depicts the average normalized Energy Delay Product (EDP) of Hadoop applications. EQ. (4) indicates how we calculated this metric.

$$EDP = (CPU\ energy + DRAM\ energy) \times Execution\ time \quad \text{EQ. (4)}$$

The results reveal that across all studied applications increasing the number of channels improves EDP, as memory controller power management policy can benefit from such increase and reduce the power consumption accordingly. The

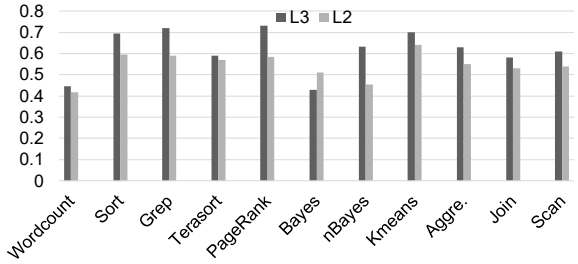

Figure 7: LLC and L2 hit ratio

EDPs results of 1-channel and 2-channel memory are found to be very similar. The trend in this figure shows that increasing the memory frequency increases EDP across all CPU operating frequency points. This implies that a high frequency off-chip memory is not a good choice for EDP optimization for this class of Big Data applications. Furthermore, we observe that high CPU frequency is not an optimal choice for EDP, as it wastes energy (note that most of Hadoop applications are Disk-intensive). A configuration with a single channel running at 1866 MHz memory frequency when CPU is running at 1.2 GHz is shown to have the worst EDP. On the other hand, a configuration with 4-channel and 1333 MHz memory frequency when CPU is running at 1.9 GHz is shown to have the best EDP for Hadoop applications.

1) Guideline for energy efficiency: As a guideline to improve the energy efficiency of Hadoop server clusters, the results suggested using a low frequency DRAM memory with high number of channels which reduces the power consumption of DRAM by 57% without any performance degradation and therefore improves EDP by 16%. Moreover, our study shows that using low capacity DRAM results in significant energy savings for Hadoop applications, as they do not require high capacity memory. Reducing the power consumption of DRAM in a Hadoop cluster, where there are thousands of nodes, is important as our observation shows DRAM consumes 14% of the total server power.

2) Emerging memory technology consideration: Based on our real-system experimental results on DDR3 memory technology we anticipate that using emerging memory technologies such as DDR5, High Bandwidth memory (HBM) and Hybrid Memory Cube (HMC) is not going to bring noticeable performance benefit in Hadoop MapReduce applications. DDR4 only provides more bandwidth per channel while increases read latency by up to 2-3% as compared to DDR3. HMC is an expensive technology which provides 5X~10X more bandwidth that is far beyond Hadoop

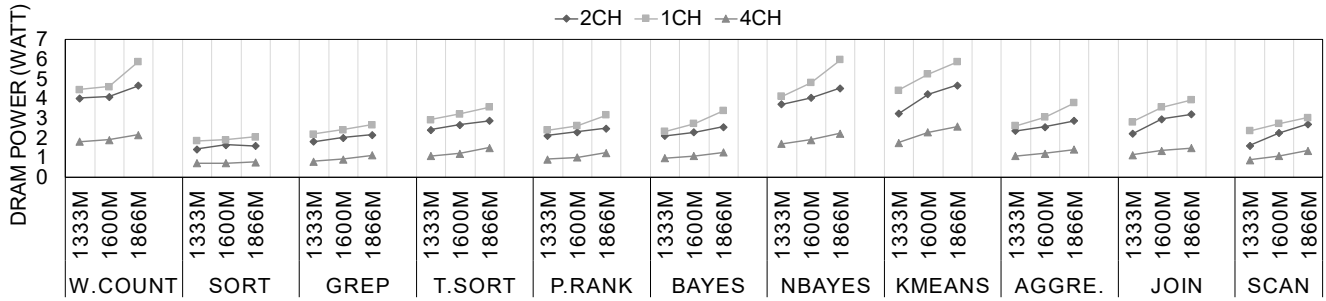

Figure 8: DRAM power consumption

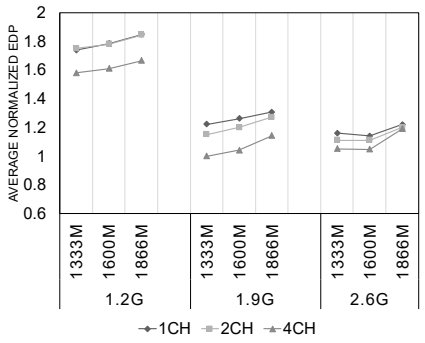

Figure 9: Average normalized energy delay Product

applications' requirements. Additionally, our results indicate that using embedded memory can be an energy efficient solution for Hadoop applications without degradation in the performance. DDR3L and LPDDR3 both can provide the required memory bandwidth and capacity for Hadoop applications with the same read latency and consuming significantly less energy.

V. CONCLUSION

This paper answers the important questions of whether Hadoop MapReduce applications running on an non-virtualized platform require high capacity-bandwidth DRAM memory and what the role of memory for energy-efficient processing of this important class of applications is. Characterizing memory behavior of Hadoop applications is the key to answer these important questions as it helps guiding scheduling decision in cloud scale architectures as well as it helps making decisions in designing server cluster for MapReduce computing. While latest work have performed a limited study on memory characterization of Hadoop applications, this work performs a comprehensive analysis of memory requirements of Hadoop applications through a methodical real-system experimental evaluation setup to provide new insights. Based on the results presented in this paper and contrary to reports in latest work, we observed that Hadoop MapReduce applications aren't memory intensive. This shows that Hadoop applications do not require high frequency, high capacity, and large number of channels memory for higher performance. Based on microarchitectural analysis we anticipate that in future architectures with higher number of cores, larger cache capacity and higher operating frequency, similar observations will remain valid indicating no urgent need for a high-end DRAM for Hadoop MapReduce Computing.

REFERENCES

[1] Bertino et al., "Big Data-Opportunities and Challenges," in IEEE 37th Annual Computer Software and Applications Conf., pp. 479-480. 2013.

[2] The Apache Software Foundation, "What is Apache Hadoop?" Available at: https://hadoop.apache.org/

[3] M. Dimitrov et al., "Memory system characterization of Big Data workloads," in IEEE Conf. on Big Data, pp. 15-22, October 2013.

[4] R. Clapp et al., "Quantifying the Performance Impact of Memory Latency and Bandwidth for Big Data Workloads," in IEEE Symp. on Workload Characterization (*IISWC*) , pp. 213-224, October 2015.

[5] I. Alzuru et al., "Hadoop Characterization," in *Trustcom/BigDataSE/ISPA* 2015, Vol. 2, pp. 96-103, August 2015.

[6] Z. Zhu, and Z. Zhang, "A performance comparison of DRAM memory system optimizations for SMT processors," in the 11th *HPCA*, pp. 213-224, February 2005.

[7] Barroso et al., "Memory system characterization of commercial workloads," ACM SIGARCH Computer Architecture News 26, no. 3, pp. 3-14, 1998.

[8] Jaleel, and Aamer, "Memory characterization of workloads using instrumentation-driven simulation–a pin-based memory characterization of the SPEC CPU2000 and SPEC CPU2006 benchmark suites," Intel Corporation, VSSAD, 2007.

[9] Tran, Le-Nguyen, et al. "Heterogeneous memory management for 3D-DRAM and external DRAM with QoS." *In ASP-DAC,* 2013.

[10] M. Hajkazemi et al., "Wide I/O or LPDDR? Exploration and analysis of performance, power and temperature trade-offs of emerging DRAM technologies in embedded MPSoCs," In ICCD, pp. 62-69, 2015.

[11] Basu et al., "Efficient virtual memory for big memory servers," ACM SIGARCH Computer Architecture News, vol. 41, no. 3, pp. 237-248, 2013.

[12] Jia et al., "Characterizing data analysis workloads in data centers," in IEEE *IISWC*, pp. 66-76, 2013.

[13] W. Xiong et al., "A characterization of Big Data benchmarks," in IEEE Conf. on Big Data, pp. 118-125, October 2013.

[14] F. Pan et al., "I/O characterization of Big Data workloads in data centers," in *BPOE*, pp. 85-97, 2014.

[15] F. Liang et al., "Performance characterization of hadoop and data mpi based on amdahl's second law," in *NAS*, pp. 207-215, August 2014.

[16] K. Hurt, and E. John, "Analysis of Memory Sensitive SPEC CPU2006 Integer Benchmarks for Big Data Benchmarking," in Proc. *PABS*, pp. 11-16, February 2015

[17] S. Beame et al., "Locality exists in graph processing: Workload characterization on an Ivy Bridge server," in the IEEE *IISWC*, pp. 56-65, October 2015.

[18] Z. Jia et al., "Characterizing and subsetting Big Data workloads," in IEEE *IISWC*, pp. 191-201, October 2014.

[19] T. Jiang et al., "Understanding the behavior of in-memory computing workloads," in the IEEE *IISWC*, pp. 22-30, 2014.

[20] Issa, J. "Performance characterization and analysis for Hadoop K-means iteration" Journal of Cloud Computing, 5(1), 1, 2015.

[21] Lei et al. "Bigdatabench: A Big Data benchmark suite from internet services," in IEEE 20th *HPCA*, pp. 488-499, 2014.

[22] Available at: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[23] S. Huang et al., "The hibench benchmark suite: Characterization of the mapreducebased data analysis," in IEEE *ICDE*, pp. 41–51, 2010.

[24] M. Malik et al., "Characterizing Hadoop applications on microservers for performance and energy efficiency optimizations," in *ISPASS*, pp. 153-154, 2016.

[25] Ch. Bienia et al., "The PARSEC benchmark suite: characterization and architectural implications," in Proc. *PACT*, pp. 72-81, 2008.

[26] Ferdman, Michael, et al. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." *ACM SIGPLAN Notices*. Vol. 47. No. 4. ACM, 2012.

[27] Ousterhout, Kay, et al. "Making Sense of Performance in Data Analytics Frameworks." *NSDI*. Vol. 15. 2015.

[28] M. Malik et al., "Big data on low power cores: Are low power embedded processors a good fit for the big data workloads?," *In ICCD*, pp. 379-382, 2015.

[29] M. Malik et al., "System and architecture level characterization of big data applications on big and little core server architectures," *In IEEE Big Data,* pp. 85-94, 2015.

[30] M. Malik et al., "Big vs little core for energy-efficient Hadoop computing," *In DATE*, pp. 1480-1485, 2017.

[31] Zhao et al., "Temperature aware thread migration in 3D architecture with stacked DRAM." In ISQED, 2013.