Memory Requirements of Hadoop, Spark, and MPI Based Big Data Applications on Commodity Server Class Architectures

Hosein Mohammadi Makrani, Houman Homayoun
George Mason University
Fairfax, USA
{hmohamm8, hhmoayou}@gmu.edu

***Abstract*—Emerging big data frameworks requires computational resources and memory subsystems that can naturally scale to manage massive amounts of diverse data. Given the large size and heterogeneity of the data, it is currently unclear whether big data frameworks such as Hadoop, Spark, and MPI will require high performance and large capacity memory to cope with this change and exactly what role main memory subsystems will play; particularly in terms of energy efficiency. The primary purpose of this study is to answer these questions through empirical analysis of different memory configurations available on commodity hardware and to assess the impact of these configurations on the performance and power of these well-established frameworks. Our results reveal that while for Hadoop there is no major demand for high-end DRAM, Spark and MPI iterative tasks (e.g. machine learning) are benefiting from a high-end DRAM; in particular high frequency and large numbers of channels. Among the configurable parameters, our results indicate that increasing the number of DRAM channels reduces DRAM power and improves the energy-efficiency across all three frameworks.**

I. Introduction

Three well-known parallel programming frameworks used by big data community are Hadoop, Spark, and MPI. Hadoop and Spark are two prominent frameworks for big data analytics. Spark has been developed to overcome the limitation of Hadoop on efficiently utilizing main memory. MPI, a de facto industry standard for parallel programming on distributed memory systems, is also used for big data analytics.

While there are literatures on understanding the behavior of big data applications by characterizing them, most of prior works have focused on the CPU parameters such as core counts, core frequency, cache parameters, and network configuration or I/O implication with the assumption of the demand for using the fastest and largest main memory in the commodity hardware [1, 2]. However, none of the previous works have studied the main memory subsystem parameters to characterize big data applications and the underlying frameworks.

The objective of this paper is to evaluate the effect of the memory subsystem on the performance of big data frameworks. To perform the memory subsystem analysis, we have investigated two configurable memory parameters including memory frequency and number of memory channels, to determine how these parameters affect the performance and power consumption of big data applications.

Our evaluation reveals that Hadoop applications do not require a high bandwidth memory subsystem to enhance the performance. Improving memory subsystem parameters beyond 1333 MHz Frequency and a single channel does not enhance Hadoop performance noticeably. On the other hand, Spark and MPI applications can benefit from higher memory frequency and number of channels if the application is iterative such as machine learning algorithms. However, increasing the number of memory channels beyond two channels does not enhance the performance of those applications. This is an indication for lack of efficient memory allocation and management in both hardware (memory controller) as well as software stack.

To the best of our knowledge this is the first work that looks beyond just the memory capacity to understand Hadoop, Spark and MPI based big data applications' memory behavior by analyzing the effect of memory frequency as well as number of memory channels on the performance as well of power consumption.

II. Experimental Setup

In our study, we used Hadoop MapReduce version 2.7.1, Spark version 2.1.0 in conjunction with Scala 2.11, and MPICH2 version 3.2 installed on Linux Ubuntu 14.04. We used BigDataBench [3] and HiBench [4] for the choice of big data benchmarking. We selected a diverse set of applications and frameworks to be representative of big data domain. More details of these workloads are provided in Table 1 and 2.

For running the workloads and monitoring statistics, we used a six-node standalone cluster (Xeon 2650 V2). To have a comprehensive experiment we used different SDRAM memory modules. All modules are provided from the same vendor. We used Intel Performance Counter Monitor tool (PCM) to understand hardware behavior. We collect OS-level performance information with DSTAT tool.

III. Results

In this section, we present a discussion on memory analysis results to help better understanding the memory requirements of big data frameworks.

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

Table 1: Studied workloads

Workload	wordcount	sort	grep	terasort	nweight	bayes	naïve bayes	kmeans	pagerank	aggregation	join	scan
Input size	1.1 T	178.8G	1.1 T	178.8G	17.6G	30.6G	30.6G	112.2G	16.8G	10.8G	10.8G	10.8G
Framework	Hadoop, Spark, MPI	Hadoop, Spark, MPI	Hadoop, Spark, MPI	Hadoop, Spark	Spark	Hadoop, Spark	Hadoop, Spark, MPI	Hadoop, Spark, MPI	Hadoop, Spark	Hadoop	Hadoop	Hadoop
Suite	BigDataBench	BigDataBench	BigDataBench	HiBench	HiBench	HiBench	BigDataBench	BigDataBench	HiBench	HiBench	HiBench	HiBench

Table 2: MPI based Multimedia workloads from BigDataBench

Workload	BasicMPEG	DBN	Speech recognition	Image Segmentation	SIFT	Face Detection
Input size (huge)	24G	MNIST dataset	59G	62G	62G	62G

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

Our system supports four memory channels with the maximum memory frequency of 1866MHz. The maximum and minimum available memory bandwidth are 59.7 GB/S and 10.7 GB/S respectively. We observe in Figure 1 that increasing the number of channels does not have significant effect on the execution time (on average 9%), except for K-means and Nweight in Spark, and for Image segmentation in MPI framework (All of them are iterative tasks). The results show that Spark and MPI based machine learning applications are gaining performance from increased number of channels.

2) Memory frequency implication: Figure 2 shows the effect of memory frequency on the execution time. Note that increasing the frequency from 1333 MHz to 1866 MHz translates to almost 40% increase in the bandwidth. Previous section showed that 400% (4X) increase in the bandwidth resulted by increasing the number of channels from 1 to 4 can gain only 9% performance benefit. Like memory channel, only iterative tasks on Spark and MPI take advantage from high frequency memory.

3) Power analysis: Figure 3 reports the DRAM power consumption. The first observation is that by increasing the frequency of DRAM by about 40% (1333 MHz to 1866 MHz), the power increases by almost 15%. It shows the static power is the major component of DRAM power. However, the DRAM power consumption is reduced when we increase the number of channels. An interesting observation is that a memory with 4 channels consumes 42% less power than a memory with 1 channel. Because with more channels, the memory controller can manage accesses more efficiently.

IV. Conclusion

This work performs a comprehensive analysis of memory requirements of big data applications through an experimental evaluation setup. We study diverse range of applications from microkernels, graph analytics, machine learning, E-commerce, social networks, search engines, and multimedia in Hadoop, Spark, and MPI. This gives us several insights into understanding the memory role for these important frameworks. We observe that most of studied big data applications in MapReduce based frameworks such as Hadoop do not require a high-end memory. On the other hand MPI applications, as well as iterative tasks in Spark (e.g. machine learning) benefit from a high-end memory. Moreover, we show that increasing the number of memory channels reduces memory power noticeably across all studied applications and improves the energy-efficiency.

References

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

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

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

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

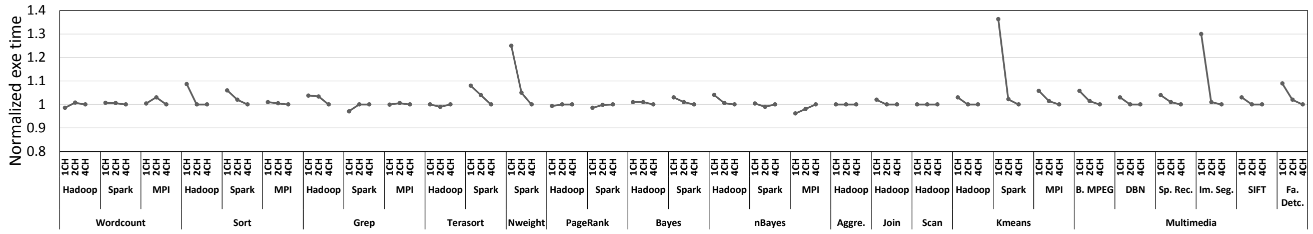

Figure 1: Effect of memory channel on the execution time (Normalized to 4CH)

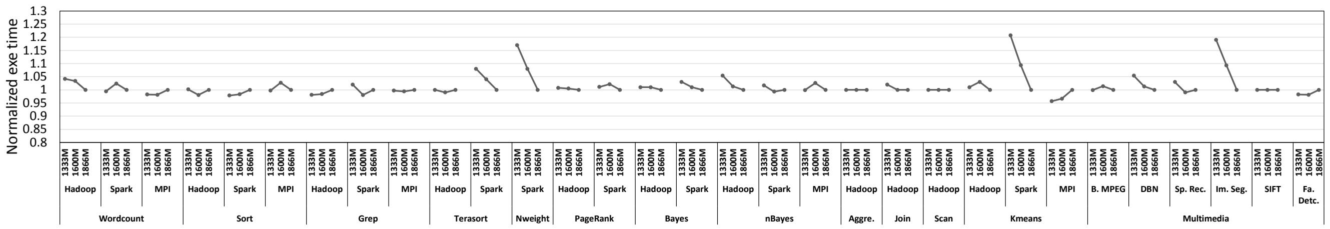

Figure 2: Effect of memory frequency on the execution time (Normalized to 1866M)

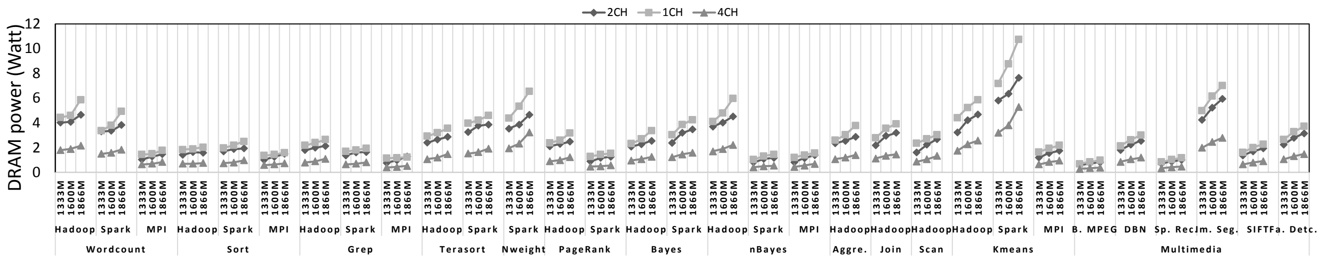

Figure 3: DRAM power consumption