한국컴퓨터정보학회 동계학술대회 논문집 제22권 제1호 (2014. 1)

Comparing Energy Efficiency of MPI and MapReduce on ARM based Cluster

Jahanzeb Maqbool[O], Permata Nur Rizki[*], Sangyoon Oh[*]
[O*]Dept. of Computer Engineering Ajou University
{permatarizki, syoh}@ajou.ac.kr, Jahanzeb@ajou.ac.kr[O]

ARM 클러스터에서 에너지 효율 향상을 위한 MPI와 MapReduce 모델 비교

자한제프 마크불[O], 페르마타 눌 리즈키[*], 오상윤[*]
[O*]아주대학교 컴퓨터공학과

● 요 약 ●

The performance of large scale software applications has been automatically increasing for last few decades under the influence of Moore's law – the number of transistors on a microprocessor roughly doubled every eighteen months. However, on-chip transistors limitations and heating issues led to the emergence of multicore processors. The energy efficient ARM based System-on-Chip (SoC) processors are being considered for future high performance computing systems. In this paper, we present a case study of two widely used parallel programming models i.e. MPI and MapReduce on distributed memory cluster of ARM SoC development boards. The case study application, Black-Scholes option pricing equation, was parallelized and evaluated in terms of power consumption and throughput. The results show that the Hadoop implementation has low instantaneous power consumption that of MPI, but MPI outperforms Hadoop implementation by a factor of 1.46 in terms of total power consumption to execution time ratio.

키워드: 고성능컴퓨팅(HPC), ARM, 시스템온칩(SOC), 성능평가(evaluation), MPI, 맵리듀스(MapReduce)

I. Introduction

Recent advancement in science and technology has provided motivation for drastic improvements in high performance computing infrastructure as well as enabling technologies. In 2008, IBM roadrunner was able to achieve the milestone of Petascale computation, since then the experts started to talk about the next milestone of achieving Exascale [5]. The increasing use of cloud computing and Internet services, the energy efficiency of data centres has become more and more critical. The high performance community had already realized that the current trends of building large scale systems with power hungry hardware would probably not lead them to achieve Exascale. Thus, a new trend for "*Green Computing*" [2] started to emerge alongside existing trends.

In recent years, the evolution of GPU computing as general purpose scientific computing has drawn the attention of computer software and hardware industry. The GPUs have evolved over the years to have teraflops of floating point performance [4] with slightly reduced power consumption than of multicore processors. The current industry trends predict that the future computing architectures will be hybrid systems with parallel-core GPUs working with multi-core CPUs. The proof is present in top500 [1] list of supercomputers where among top 5 world's fastest supercomputers some are hybrid – multicore CPUs coupled with GPUs. Due to the emergence of multicore and GPUs for scientific computing, the software programmers could no more rely on processors vendors to improve performance. Instead in order to fully utilize the multiple cores, programmers have started to explicitly program their applications to boost performance. On the other hand,

TABLE I
TopSupercomputersandtheirPowerConsumptionasofJune2012list([1])

Computer	Build	Num Cores	Killowatts
Sequoia	BlueGene/Q	1,572,864	7,890.0
K Computer	SPARC 64	705,024	12,659.9
Mira	BlueGene/Q	786,432	3,945.0

unfortunately, even these hybrid systems could not provide a well adopted solution for energy efficiency and low power computing. Table 1 shows the top 3 supercomputer as of June 2012 and their power consumption in kilowatts. Maximum power consumption of 12.6 Megawatt is a massive amount and results in large electricity bills and reduced system reliability due to excessive heating. Thus, the motivation for a better, energy efficient solution with much reliability and cooling becomes stronger than ever and urges the vendors to come up with an efficient way to tackle these issues.

Considering the fact that ARM based mobile processors provide acceptable computation while consuming significantly low amount of energy and surprisingly low cooling cost gives us enough reason to explore the potential of these processors in more detail and explore the potential of these processors more in detail and leverage the best out of it. This motivates us to explore the feasibility of mobile processors for desktop and data center usage. Our contributions in this paper are given below.

We present two different parallel implementation of Black-Scholes option pricing formula using MPI [6] and Hadoop [7]. We evaluate our implementation on our 16-core ARM SoC based cluster named 'weiser'. We analyse the performance in terms of execution time and energy efficiency. The rest of the paper is organized as follow:

Section II describes the Black-Scholes application formula. Section III shows the details of our testbed software and hardware configuration. Section IV describes the implementation details and algorithm flow. Section V discusses the experiment results and details. Section VI concludes the work with possible future directions.

II. Black-Scholes

Black-Scholes [10] is a mathematical model of a financial market. Black-Scholes formula calculates the prices for European options by using Black-Scholes partial differential equation. The input dataset varies from 1 (*test*) to 10 million (*native*) options [8]-[9]. We used *native* dataset which

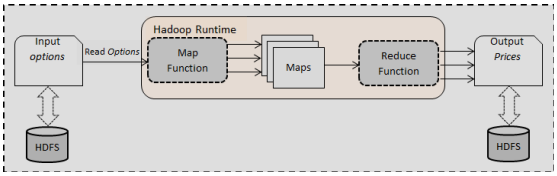

Fig. 1 Black-Scholes Hadoop implementation flow.

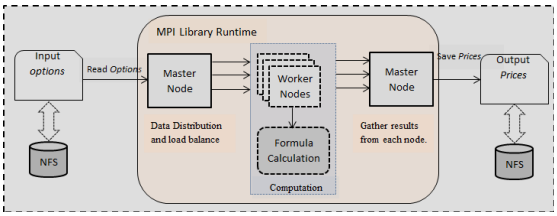

Fig. 2 Black-Scholes MPI Implementation flow.

consists of 10 million options. The partial differential equation of Black-Scholes model is given in (1).

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \qquad (1)$$

Where S is the price of the stock, r is the annualized risk-free interest rate continuously compounded, σ is the velocity of the stock's returns.

III. Hardware and Software Setup

This section describes the software and hardware configuration of our testbed cluster weiser.

A. *Software Setup*

We have Ubuntu Linaro 12.04 installed on our *weiser* cluster. We use mpich3.0 as our message passing library. NSFv4 is also setup for shared access to the input data and executable files. We used Apache Hadoop as our MapReduce framework due to its widely available support.

B. Hardware Setup

The *weiser* cluster consists of 4 nodes of ODROID-X SoC development board. Each board is 4-way SMP with 2MB of shared L2 cache and 16GB SD card as storage. The input data and the program executable are placed in NSF drive which is a 2TB external storage mounted on a directory server.

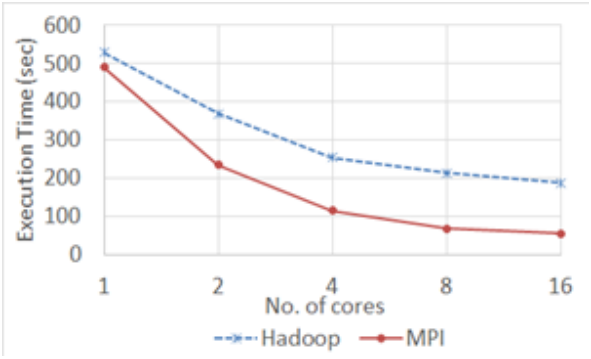

Fig. 3 Black-Scholes execution time graph on MPI and Hadoop.

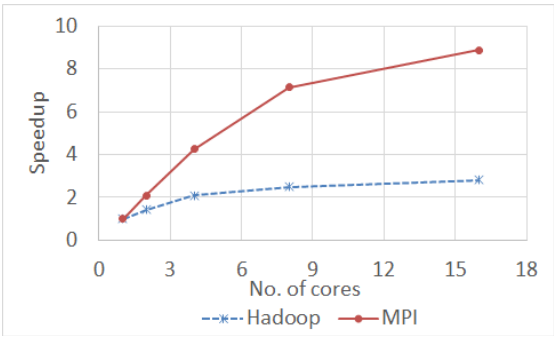

Fig. 4 Black-Scholes Speedup graph on MPI and Hadoop.

IV. Implementation

The two versions of the Black-Scholes application were implemented using MPI and Hadoop. We describe our implementation in the following sub-sections.

A. MPI Implementation

We used mpich as our message passing library. The input datasets are taken from PARSEC [8] benchmark. The input file consists of a list of *options,* each row of the file consist of the required variables to input the Black-Scholes formula to calculate the option *price*. Fig. 2 describes the flow of the MPI implementation. We read the file into and array of option objects. This large array contains one option at each index. The *master* processor is responsible for reading the input from the file and storing it in optionObject array. Once the file is read in the memory, master processor then divides the data into *N/p* chunks and distributes it among the *slave* processors. Each *slave* processor, upon receiving the data into its local sub-array, computes the option pricing formula and store the result into optionPrice array. Once the computation is done, each slave processor sends its optionPrice array to the master processor. Finally, *master* processor stores data back to the output file and computation is over. All the necessary communication were handled manually and blocking communication primitives were used.

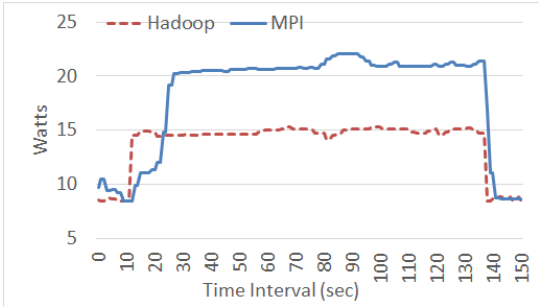

Fig. 5 Power consumption of MPI and Hadoop between a certain time interval running on 16 nodes.

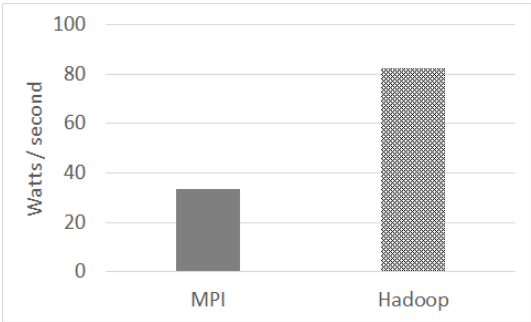

Fig. 6 Total power consumed to total time to finish ratio of Black-Scholes MPI and Hadoop implementation. (Lower the better). Running on 16 nodes.

B. Hadoop Implementation

In Hadoop implementation, we also use *master-slave* paradigm for computation. The input is read using "*TextInputFormat*" class and stored as key value pairs of the following form.

<key, value>:: Keys(0); values: (individual options vars)

Fig. 1 shows the execution flow of the Hadoop implementation. In Hadoop all the necessary communication is and data movement is handled by the library runtime. We define the *Mapper* function to map the input data on the slave processors and *Reducers* to gather the final results. The Black-Scholes is an embarrassingly parallel application which is best suited for MapReduce programming model. The scheduling of jobs is also handles by *JobTracker* and *TaskTracker* provided by MapReduce runtime.

V. Results and discussion

The results of the experiments performed on 16-core ARM based cluster are depicted here. Fig. 3 shows the execution time of running MPI and Hadoop implementation of

Black-Scholes. As we see that MPI

performs better and as the number of cores varies, the performance is much better. The Hadoop runtime is built upon Java, thus it automatically inherits the JVM related extra overheads which cause execution time slowdown in most of the Java based applications. Similarly, in the strong scaling speedup test shown in Fig. 4, we observe that MPI scales better and show reasonable speedup, while Hadoop implementation suffers from scalability issues. The reason for this behaviour is that MPI workloads put the processor to 100% usage and exploits the multicore capability of underlying nodes and Hadoop does not.

In the energy efficiency comparison in Fig. 5, we ran the experiment for a certain time interval, say 150 seconds, and observed the power consumption of MPI and Hadoop. As we see that the power consumption of MPI during this interval is higher than that of Hadoop because of the high CPU utilization of MPI over Hadoop. This seems like Hadoop has lower energy consumption as compared to MPI, which is good. But, when we run another experiment in which we considered the ratio of total power consumed during a run to the total time taken to finish the job, we found that higher finish time of Hadoop offsets its low power advantage. Because the slow performance of Hadoop causes longer execution time, hence consumes more total power during the execution. Fig. 6 shows the results of this experiment. As we see that MPI also performed significantly well in total energy to time ratio comparison.

VI. Related Work

The usage of ARM based processors for desktop computing is an emerging trend. Recent efforts are going on in the performance evaluation for HPC workloads. *Tibidabo* [11] was the first attempt to build a high performance cluster based on Tegra2 SoC chips. They demonstrated the basic prototypes and initial performance benchmarks in their recent papers [12]-[13].

Edson *et al*. performed the comparison of execution time, power consumption between two different categories of SoC boards [14]. They concluded that different scenarios i.e. time critical or energy efficient, need different kind of SoC.

However, our approach differs from existing approaches in a sense that we use two widely used parallel programming models i.e. MPI and Hadoop, which are considered *de-facto* standard for parallel programming in the domains of distributed memory cluster and cloud computing respectively. We present our finding by parallelizing Black-Scholes, a widely used financial model, and demonstrate our findings on our own build SoC cluster.

Conclusion

In this paper, we presented two different parallel implementation of Black-Scholes option pricing model and evaluated the performance on low powered ARM SoC cluster running MPI and Hadoop. Our analysis shows that MPI implementation shows better performance due to high utilization of available processors, while Hadoop, due to inherent JVM overhead and extra features like fault tolerance, and data redundancy, tend to perform slower. The instantaneous power consumption of Hadoop implementation is lower, however, the total power consumption in reaching the solution i.e. total power to total time ratio is higher than MPI. We conclude that the ARM based low energy processors can be a good candidate for Exascale but research is needed to come up with efficient runtime libraries, particularly optimized for such systems. A large gap lies ahead in terms of supported software and available APIs for these processors.

Acknowledgement

This research was jointly supported by the MKE of Korea, under the ITRC support program supervised by the NIPA (NIPA-2013-(H0301-13-2003)). A part of this research is also funded by BK21 Plus Program of Korea National Research Foundation. The graduate students were supported by a grant from the NIPA (National IT Industry Promotion Agency) in 2013. (Global IT Talents Program).

References

[1] Top500 List. [Online]. Available: http://top500.org

[2] Green500 List. [Online]. Available: http://green500.org

[3] Sharma, Sushant, Chung-Hsing Hsu, and Wu-chun Feng. "Making a case for a Green500 list." Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International. IEEE, 2006.

[4] Collange, Sylvain, David Defour, and Arnaud Tisserand. "Power consumption of gpus from a software perspective." Computational Science–ICCS 2009. Springer Berlin Heidelberg, 2009. 914-923.

[5] Bergman, Keren, et al. "Exascale computing study: Technology challenges in achieving exascale systems." Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep (2008).

[6] MPI Homepage. [Online]. Available: http://www.mcs.anl.gov/research/projects/mpi/

[7] Hadoop Homepage: [Online] Available: http://hadoop.apache.org/

[8] PARSEC Homepage: [Online] Available: http://parsec.cs.princeton.edu/

[9] Bienia, Christian, and Kai Li. Benchmarking modern multiprocessors. Princeton University, 2011.

[10] Black-Scholes Info: [Online]: http://www.investopedia.com/terms/b/blackscholes.asp

[11] Rajovic, Nikola, et al. "Tibidabo: Making the case for an arm based hpc system." Barcelona Supercomputer Center, Tech. Rep (2012).

[12] Rajovic, Nikola, et al. "The Low Power Architecture Approach Towards Exascale Computing." Journal of Computational Science (2013).

[13] Rajovic, Nikola, et al. "Supercomputing with commodity CPUs: are mobile SoCs ready for HPC?." Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2013.

[14] Padoin, Edson L., et al. "Evaluating Performance and Energy on ARM-based Clusters for High Performance Computing." Parallel Processing Workshops (ICPPW), 2012 41st International Conference on. IEEE, 2012.