International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 09 | Dec-2015 www.irjet.net p-ISSN: 2395-0072

A STUDY OF THE OPTIMISTIC MAPREDUCE TECHNIQUES FOR ENERGY MINIMIZATION AND PERFORMANCE ENHANCEMENT FOR MULTICORE CLOUD COMPUTING APPLICATIONS

Priyanka Gupta[1], Kiranbir Kaur[2]
[1]*Department of Computer Engg & Technology, Guru Nanak Dev University, Amritsar, Punjab, India*
[2]*Department of Computer Engg & Technology, Guru Nanak Dev University, Amritsar, Punjab, India*

Abstract— Multi-core architecture is established on a number of processors and has local caches (memories). When all the mandatory actions of a computer are executed on a processor which has more than one core to execute, its processor is known as multi core architecture. Multi-core processing is used to make tasks energy efficient, augment their performance and to make multiple tasks run concurrently in further adequate way as in parallel processing. The overall objective of this paper is to review some well known map-reduce techniques that are used to minimize energy consumption. The review has shown that if fuzzy rules are combined with multi core computing then power consumption of multi core can be further balanced and performance of multi cores can be further increased. Also the use of fuzzy "if then" rules can reduce the potential overheads of Map-reduce technique.

Key Words: Multi-core, energy efficiency, Distributed Task Execution, Map Reduce, Fuzzy rules

1. INTRODUCTION

With the increase in number of chips per core, the demand for adequate programming frameworks which can bring the parallelization of the programs automatically increase and can bring adequate synchronization and communication system also increase [1]. Multi core processing is used to make energy efficient, augment performance and to make multiple tasks run concurrently in further adequate way as in parallel processing, which is a kind of an integrated circuit where two or more than two processors have been connected. The enlargement of many research fields have been derived with the enlargement of multi-core processors. When multi core processors were not seen, there was exponential increment in speed of microprocessors over time so does need of more transistors. As the speed increases, the number of transistors required in processing also increased in a manner that large number of transistors switching at very high frequencies led to very high energy utilization [2]. Multi-core architecture is established on number of processors and has local caches (memories)[3]. When all the mandatory actions of a computer are executed on a processor which has more than one core to execute, its processor is known as multi core architecture.

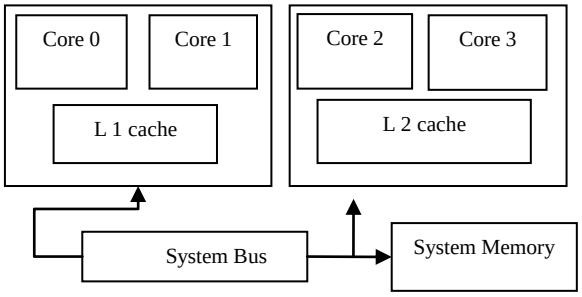

Fig. 1(Multi core Architecture)

In this multi core diagram (Fig.1) dual core processor is shown in which two processors are attached with their local memory, one system bus and one system memory and each processor is divided into two cores. The necessity for warehouse scale data centers who hosts thousands of servers has been initiated for developing web applications like streaming video, social networks and cloud computing . Map Reduce framework is the leading programming schema for handling huge data files in the data centers and other collection of computers. Map Reduce is basically used for handling huge data sets having abundant of nodes. Two functions (Map, Reduce) are specified by the user and the allotment of processes to processors is done by Map Reduce scheduler. One of the major benefits of the Map Reduce schema is that it can be introduced in heterogeneous clutches containing distinct categories of processors. Many computing resources are exhausted for arranging and mapping of programs to the cores and execution of Reduce actions of the Map Reduce scheme is needed, whenever Map Reduce scheme is mapped to either great performance processors or little-energy installed processors directing micro servers [4].

1.1 MAPREDUCE TECHNIQUES:

A. Column-aligning systems: - These systems are highly adequate when an accumulated demands to be figured out over number of rows but only for a remarkably tiny subset of all columns of data as shown in fig. 2. When recent values of a column are equipped for all rows instantaneously, then column- aligning systems are highly adequate. Column data is kind of uniform data, which is capable of giving a few freedoms for storage size expansion [14].

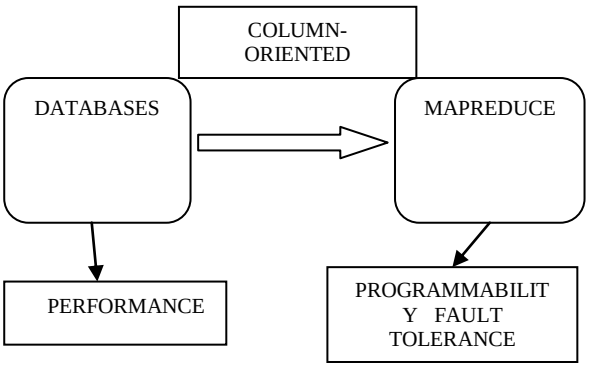

Fig. 2(Column Aligning Systems)

B. Collating :- When a set of components and operations of a single component are available, It is recommended to preserve each component which has the similar value of operation into separate file or implement some other processing that desire all those components to be computed as a organization. Its example is construction of inverted indexes. Then for all those components, Mapper figures out a operation and provides value of the operation in the form of a key and value is component itself. The task of reducer is to collect all those components which are arranged by operation value and then compute them or save them. In inverted indexes, components are terms (words) and operation is a document ID where the term was developed. It is used in Inverted Indexes.

C. Refining, Disintegration and Approval

There is a collection of files(records) and it is necessary to gather those files which satisfy few condition or change the representation of each file(not depend upon other files) into another form. This phase consists of text disintegration and value elicitation, transformation then Mapper deals with files individually and generates confirmed elements or their reconstructed variants. Applications are Data Querying, Log resolution, Data approval

D. Distributed Task Execution

A giant problem is split into several sub problems and outputs of all sub problems are collected to get an end result then problem narration is divided into a set of terms and these terms act as input material for Mappers. Each Mapper yields a term process it accordingly and give output. The task of reducer is to combine all generated sub problems into the end result [15].

The sharing resource in an environment of multi-core systems is common bus of the System-on-Chip (SoC) which is mutually used by multiple cores of the master. An arbiter is an electronic device which allows processes to share resources. To provide an authorization to use the shared resources adequately then an arbiter shows an essential role. During any cycle, It assures that at a time, at least one master can approach to the bus by observing that different requests expressed by various masters. It fragments the number of requests and take decisions which master can approach the shared bus. To appoint processes to be carried out by the process is the main goal of an arbitration process or in such an aspect that it meets the targets such as energetic process or utilization, bandwidth optimization and low latency [5].

1.2 FUZZY RULES

A new technique is used which is based on fuzzy rules i.e. fuzzy based map reduce. A fuzzy rule is basically a conditional statement in the pattern: IF v is P THEN w is Q. where v and w are linguistic variables; P and Q are linguistic values preserved by fuzzy sets on the universe of discourse V and W, proportionately. All the choices we make are all based on computer like if-then statements. In this human knowledge is represented by if-then rules. eg, If you are going then i will also go. Rules specify relation between events and associate ideas. Fuzzy machines, which always try to take decisions like man, work in similar manner. However, fuzzy sets replace the outcome and the means of choosing that outcome and fuzzy rules replace the rules. Fuzzy rules also work using a list of if-then utterances. For eg, if V then P, if W then Q, where P and Q are all sets of V and W.

2. RELATED WORK

Phoenix Map Reduce framework has used to naturally parallelize and schedule programs, which is a multi-core programming scheme. A unique method established on scratchpad memory architecture, is used to quicken Map Reduce functions by indexing and dealing with the key/value pairs. The suggested scratchpad memory strategy can be mapped onto programmable logic or

multi-core processors chips as a coprocessor to quicken Map Reduce functions (Kachris et al., 2015).

In many function territories energy-adaptive job arrangement is a crucial argument, like energy protection for mobile appliances and application of green computing data centres. Present day processors are based on Dynamic voltage and frequency scaling (DVFS) on a per-core based, i.e., the CPU can regulate the voltage or frequency of all cores. As a consequence, a core in a processor may have distinct measuring skills and power utilization. To preserve energy in multi-core platforms, job arrangement algorithms are suggested that take advantage of per-core DVFS and accomplish a equity among performance and energy utilization (Lin et al., 2015).

The problem of increment in power dissipation with clock frequency multi core architecture has been introduced. For single core systems introduction of deeper pipelines, speculative threading etc. were not ready to carry much increment in performance as related to their correlated power overhead. Performance scaling with multiple cores has always been an objection for multi core architectures. With less extent of code in parallel having large number of cores for an operation might just grant higher power dissipation rather than bringing some performance advantage. Here a fuzzy logic based design space exploration procedure has been discussed that aims to upgrade a multi core architecture according to the workload prerequisite in order to obtain optimum equity between energy and throughput of the system (Quadri et al., 2015).

In today's microprocessors power management is one of the most demanding issues in the design. The aim of power management is to increase performance within a given power cost. The main goal was to audit and to consider the ongoing power management approaches (Attia et al., 2015).

In sequential computing algorithms textbooks and portable software are resources that facilitate software systems to be recorded that are comfortably convenient across changing hardware platforms. Multi-core architecture is still missing is these resources, where a programmer searching for big performance has no equal freedom to frame on the highbrow attempts of others. In order to locate this issue, here given a bridging model whose major objective is to occupy the general resource criteria of multi-core architectures. Portable algorithms would consist of adequate designs for all acceptable grouping of the general resource parameters and input sizes, and would create the base for implementation or compilation for specific machines (Valiant, 2010).

For future extreme-scale computing system parallelization of task is treated to be a big issue. The big issue to deal with is the accomplishment of high CPU core usage through increment in task parallelism by keeping moderate bus bandwidth allocation. In order to tackle the aforesaid problems, a novel arbitration technique, known as Parallel Adaptive Arbitration (PAA) has been suggested for the masters designed according to the traffic nature of the data flow. The given arbitration approach is a capable case in favour of fair bandwidth optimization and high CPU utilization (Akhtar et al., 2015).

Synthetic benchmarks are one of the ordinarily used approaches to speedup previous architectural research and performance estimations of modern hardware architectures. A novel automatic thread-level synthetic benchmark generation framework has been discussed in (Sen et al., 2015). The emerged thread-level synthetic benchmarks are human-readable, portable and fast. This is determined that multi-threaded synthetic benchmarks can be accomplished for actual-life PARSEC and Rodinia benchmarks. The captured conclusion represent that synthetic benchmarks not only conserve thread-level micro-architecture dependent and independent attributes but also parallel programming arrangement, which are great-quality results to issues occurring at short intervals in parallel programming

Latest approach that let billions of transistors on a chip, that is microprocessor design is becoming popular in multi core world. Similarly, the resource providers will also enhance their machines to the multi core architecture, which allow customers to use their computing elements and storage. Also, compute Grids as dominating resource sharing platforms, are facing a big objection from Cloud computing. Cloud computing is appealing more users to accompany its dimension with an accessible interface, large-level services, and on-demand arrangement of compute resources. Future cloud environments will be generally based on multi core technologies. A review over multi core technologies in the ambience of Cloud computing has been given (Wang et al., 2010

3. COMPARISON ANALYSIS

REF NO.	YEAR	TECHNIQUE	BENEFITS	LIMITATIONS	ENERGY EFFICIENCY	LOAD BALANCING
[1]	2015	Map reduce scratchpad memory	a) Increase performance, b) reduce time c) less energy consumption	a)Complexity of large functions is more	Yes	Yes
[2]	2015	DVFS, Thread motion, Variable size cores	a) Low energy consumption b) balanced power management	a)Dependents jobs are not considered. a)Fuzzy rules are ignored. b) Overheads are ignored.	Yes	No
[3]	2010	Portable algos for matrix multiplication, FFT	a) It makes easy to change software on hardware platforms	a)Complex in nature b) Fuzzy rules are ignored c)Overheads are ignored	No	Yes
[4]	2014	Map reduce accelerator	a) Performance improvement b) less execution time	a)To accelerate several cloud computing apps	No	No
[5]	2015	Parallel adaptive arbitration(PAA)	a) Fair bandwidth optimization b) high parallelism c) minimum fluctuation reduces system latency	a)Dependents jobs are not considered b) Overheads are ignored	No	Yes
[6]	2013	GPU enhanced multicore machine	a) Solved large combinatorial optimization problems b) reduces thread divergence c) Reduces CPU communication with GPU	a)On combining multicore & GPU, found that RLL-GB & B is not beneficial	Yes	Yes
[7]	2015	Fuzzy logic based throughput and design space MPSoCs	a) Achieve optimum throughput b) Optimum energy consumption	a)Effect of overhead is not considered	Yes	No
[8]	2015	Thread level synthetic benchmark	a) Fast b) Portable c) human readable d) provide high quality parallel programming	a)Dependents jobs are not considered b) Fuzzy rules are ignored	No	Yes
[9]	2010	Multicore for cloud computing	a) Easy migration b) Parallelization c) Latencies between clusters is very low d) data transfer rate of multicore system is high	a)Fuzzy rules are not considered	Yes	Yes
[10]	2015	Patch based directional wavelet(PBDW)	a) Improve image reconstruction b) take less time to give output image	a)Time consuming method b) large computations	No	No
[11]	2015	Task scheduling with per core DVFS	a) a balance between performance and energy consumption	a)Fuzzy rules are not considered	Yes	Yes
[12]	2012	Time Predictable scratchpad	a) Performance maximise b) energy efficient	a)Fuzzy rules are ignored	Yes	No

		memory(SPM)				
[13]	2011	Hybrid approach(combination of OpenMP & MPI	a) Load balancing improved	a)Performance of OpenMP is limited b) interaction between OpenMP & MPI threads is not good	No	Yes
[14]	2012	Intro of high-performance energy efficient multicore embedded computing	a) Performance increased	a)Power consumption is ignored	No	Yes
[15]	2015	Multicore parallelization of RSMF(Robust structured multifrontal factorization)	a) Effective to achieve parallelization	a)Effect of dependent jobs is not considered	No	Yes
[16]	2008	Multicore for Chemo metrics calculation	a) Less execution time b) high speed	a)Large number of computing cores	No	Yes
[17]	2014	Multi BSP(Bulk Synchronous Parallel) model for multicore	a) Parallelization is achieved b) high performance	a)Effect of dependent jobs is missed	No	No
[18]	2012	Multicore for Finite-difference time domain method(FDTD)	a) Powerful for large scale simulations b) Parallelization	a)Effect of overhead is missed	No	No

4. GAPS IN LITERATURE

The subsequent section contains following limitations:-

1. The use of fuzzy membership has not been explored much while mapping the jobs on multi core environment.
2. The effect of dependent jobs is also not considered in existing literature. A job is said to be dependent if and only if it can be executed after another job i.e. it may require output of another jobs as input.
3. Effect of potential overheads has also not been considered in earlier techniques. Overheads i.e. extra communication delay may occur while reducing the problem in Map-reduce form.

5. CONCLUSION AND FUTURE WORK

The review has clearly shown that still much improvement is required in existing techniques. The comparison among the existing technique has shown that no technique is effective in nature. The use of fuzzy membership is not much explained in existing literature while mapping jobs on multi-core environment. The effect of dependent jobs is also not considered in existing literature. Effects of overheads are also ignored. This work has focused on multi-core computing by using fuzzy membership functions. The overall objective is to schedule given set of tasks in more efficient manner than earlier technique. In near future we will design and implement a fuzzy logic based membership function for mapping the data among multi-core.

6. REFERENCES

[1] Kachris, Christoforos, Georgios Ch Sirakoulis, and Dimitrios Soudris. "A MapReduce scratchpad memory for multi-core cloud computing applications." Microprocessors and Microsystems 39, no. 8 (2015): 599-608.

[2] Attia, Khaled M., Mostafa A. El-Hosseini, and Hesham A. Ali. "Dynamic power management techniques in multi-core architectures: A survey study." Ain Shams Engineering Journal (2015).

[3] Valiant, Leslie G. "A bridging model for multi-core computing." In Algorithms-ESA 2008, pp. 13-28. Springer Berlin Heidelberg, 2008.

[4] Kachris, Christoforos, Georgios Ch Sirakoulis, and Dimitrios Soudris. "A Reconfigurable MapReduce accelerator for multi-core all-programmable SoCs." In System-on-Chip (SoC), 2014 International Symposium on, pp. 1-6. IEEE, 2014.

[5] Akhtar, M. Nishat, Junita Mohamad-Saleh, and Othman Sidek. "Design and simulation of a parallel adaptive arbiter for maximum CPU utilization using

multi-core processors." Computers & Electrical Engineering 47 (2015): 51-68.

[6] Chakroun, Imen, Nordine Melab, Mohand Mezmaz, and Daniel Tuyttens. "Combining multi-core and GPU computing for solving combinatorial optimization problems." Journal of Parallel and Distributed Computing 73, no. 12 (2013): 1563-1577.

[7] Qadri, Muhammad Yasir, Nadia N. Qadri, and Klaus D. McDonald-Maier. "Fuzzy logic based energy and throughput aware design space exploration for MPSoCs." Microprocessors and Microsystems (2015)

[8] Sen, Alper, and Etem Deniz. "Thread-level synthetic benchmarks for multicore systems." Microprocessors and Microsystems 39, no. 7 (2015): 471-479.

[9] Wang, Lizhe, Jie Tao, Gregor von Laszewski, and Holger Marten. "Multicores in cloud computing: research challenges for applications." Journal of computers 5, no. 6 (2010): 958-964 .

[10] Li, Qiyue, Xiaobo Qu, Yunsong Liu, Di Guo, Zongying Lai, Jing Ye, and Zhong Chen. "Accelerating patch-based directional wavelets with multicore parallel computing in compressed sensing MRI." Magnetic resonance imaging 33, no. 5 (2015): 649-658 .

[11] Lin, Ching-Chi, You-Cheng Syu, Chao-Jui Chang, Jan-Jan Wu, Pangfeng Liu, Po-Wen Cheng, and Wei-Te Hsu. "Energy-efficient Task Scheduling for Multi-core Platforms with per-core DVFS." Journal of Parallel and Distributed Computing 86 (2015): 71-81 .

[12] Liu, Yu, and Wei Zhang. "Exploiting multi-level scratchpad memories for time-predictable multicore computing." In Computer Design (ICCD), 2012 IEEE 30th International Conference on, pp. 61-66. IEEE, 2012 .

[13] Jin, Haoqiang, Dennis Jespersen, Piyush Mehrotra, Rupak Biswas, Lei Huang, and Barbara Chapman. "High performance computing using MPI and OpenMP on multi-core parallel systems." Parallel Computing 37, no. 9 (2011): 562-575.

[14] Floratou, Avrilia, Jignesh M. Patel, Eugene J. Shekita, and Sandeep Tata. "Column-oriented storage techniques for MapReduce." Proceedings of the VLDB Endowment 4, no. 7 (2011): 419-429.

[15] highlyscalable.wordpress.com."mapreduce patterns":Google,February 2012.

[16] Zhang, Zhi-Min, Yi-Zeng Liang, and Qing-Song Xu. "Multi-core computing: A novel accelerating method for chemometrics calculation." Chemometrics and Intelligent Laboratory Systems 96, no. 1 (2009): 94-97.

[17] Zuo, Xian-yu, Ze-yao Mo, Tong-xiang Gu, Xiao-wen Xu, and Ai-qing Zhang. "Multi-core parallel robust structured multifrontal factorization method for large discretized PDEs." Journal of Computational and Applied Mathematics 296 (2016): 36-46.

[18] Francés, Jorge, Sergio Bleda, Cristian Neipp, Augusto Márquez, Inmaculada Pascual, and Augusto Beléndez. "Performance analysis of the FDTD method applied to holographic volume gratings: Multi-core CPU versus GPU computing." Computer Physics Communications 184, no. 3 (2013): 469-479.

[19] Fathoni, M.F.; Sridadi, B. "Multicore computation of tactical integration system in the Maritime Patrol Aircraft using Intel Threading Building Block", in Advanced Computer Science and Information Systems (ICACSIS), 2014 International Conference on , vol., no., pp.1-6, 18-19 Oct. 2014 doi: 10.1109/ICACSIS.2014.7065821

[20] Munir, Arslan, Sanjay Ranka, and Ann Gordon-Ross. "High-performance energy-efficient multicore embedded computing." Parallel and Distributed Systems, IEEE Transactions on 23, no. 4 (2012): 684-700.