International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-2, December 2013

Challenges on Performance Analysis and Enhancement of Multi - Core Architecture, a Solution Parallel Programming Languages

Surendra Kumar Shukla, Vishal Trivedi, Ayush Choukse

Abstract— Performance of computer is major concern in computer architecture. Mores law has gone now, we can not increase the speed of single processor as it has problem of power requirement. So we need to move on multi core processors. Comiler is a main parameter who can give the deatil of parallelism on source code. In this paper we have proposed a scheme where we are utilizing the comiler for detecting and increasing the execution speed of souce code.

Index Terms— **complier , performance , multi-core architecture**

I. INTRODUCTION

Hardware technology advancement is in very high rate[1]. It has created a great challenge to the software developers to make the software in such a way that user can feel that his application is running faster in the high speed hard ware. Hard ware technology is growing specially in microprocessor area, microprocessors are now coming in dual, tri, quad, hexa,octo core chips to even hundred of cores.

It is good news that the processors performance is increasing but flip -side is that, applications are not capable to take the advantage of the processing speed[2]. Sequential programming languages are not capable to exploit the power of multi-core architectures. Applications must be redesign so that processors parallel ability can be utilized. In parallel programming for solving the problem, we are making the program, that program instructions are executed by more than one CPU at a time independently.

Designing and developing parallel programs are manual process. Programmers itself have to identify the parallelism and then he itself have to implement it. But parallel programming have many pitfalls, race conditions, mutual exclusion, dead lock, thread synchronization and load balancing, all these impacts the run time performance, and all these are very hard to fix.

So we can say that processing speed which we can gain from the parallel programs is depends on the total number of cores, also the size of problem and the way it is broken into parts, then how many parts are parallel.

Manuscript published on 30 December 2013.

***** Correspondence Author (s)

Surendra Kumar Shukla*, Department of CSE, DAVV, /SCSIT/ , Indore, India

Vishal TRivedi, Department of Information Technology RGPV, University, Chemeli Devi School of Engineering Indore, India.

Ayush Choukse, Department of CSE, RGPV, University, Chemeli Devi School of Engineering Indore, India

II. REVIEW OF THE WORK

(i) Multi core Processors- A Necessity [3]

One important aspect in increasing the performance of the mulicore architecture the speed up. Speed up can be achieved by increasing the clock speed of the processor. By increasing the more core can also increase the speed up. If we will increase the number of cores then the problems are coming on memory and cache coherence. And communication between cores is also a problem. cache coherence protocol and interconnection network have solved these problems, but some problems are still remain. until programmer will not write the parallel applications, it will not be possible to utilize the multi core architecture for the high performance computing are number of library are available for the programmer to write the parallel programs i.e. OpenMP, openCL, MPI. All these library mainly concentrates on shared memory multi-core architecture. But when a core reads the data from the main memory, he should have to ensure that he is reading the updated data or not.

(ii) Compiler support for work-stealing parallel run time systems[4].

All computers embedded, mainstream, and high-end-are being built using multi core chips, the need for improved productivity in parallel program has taken on a new urgency. Paper specifies that the advent of multi core processors has lead to the emerging of different kinds of programming models to use the parallel processing power available. they have classified the parallel programming models into three categories, these categories are based on their approach to exploit the parallelism. Three classifications are (a) Programming language approach (b) directive based approach (c) library approach. They have specified that programming languages should have to execute the task asynchronously. Here task means independent unit of work. Since task are independent they can be executed independently. But these task may have to coordinate to other task also. This task coordination activity leads to the scheduling problem of the tasks. Solution of this problem is that we need to create the thread for each task, and then these task can coordinate to each other with the help of thread synchronization. But this will be the burden to the operating system, as he need to do the activity of thread synchronization.

Another problem of this approach is that when there are more threads, as compared to the number of processors, some of these threads will remain ideal. As we know that threads takes lot of resources, like memory and may also lead to deadlocks and resource overflow. One another overhead of this solution is to create a thread when task is executed, and destroy the thread when task will be executed completely.

(iii) Performance evaluation of MPI UPC and openMP on multi core architectures[5].

This paper specifies that the current trend to multicore architectures underscores the need of parallelism. This research work evaluated performance against unified parallel c (UPC) and openMP on multicore architectures. Result shows that MPI is best choice for multicore systems, with both hybrid shared and distributed memory, because it takes highest advantage of data locality. Here data locality is a key performance factor.

(iv) Exploiting parallelism of multicore architecture [6].

In this research paper, they have specified that the biggest challenge today is how to exploit the parallelism in the multicore architecture. If threads can be synchronized efficiently, and data exchange between the threads is proper then we can exploit the parallelism. Hardwares are advanced, they have the scope of parallelism, but softwares are not well advanced, they all are not utilizing hard ware properly. Biggest problem of multicore architecture is synchronization. Transitional memory concept is used here to synchronize the multi-threaded program. problem was now that how to synchronize two threads, because both these are at different cores. Solution previously given was the locking, but course grain locks are not scalable, and fine grain locks are hard to design. Transitional memory system have ability to remove the drawback of the locking methods. Transitional memory allows to bundle lot of operations and put all these into thread, bundle is similar to the transaction of database systems. Transactional memory allows to the programmer to define read-modify-write operations. These operations are applied to different memory locations, in multiprocessor environment. For the implementation of this cache coherence protocol is used.
But for the implementation of this scheme is required to add new instructions to the processor an also small auxiliary transactional cache memory . Advantage of this scheme is that it access shared memory very less, because it does not uses the explicit locks.

(v) Intelligent Hotspot prediction for Network-On-chip-Based Multi core systems[7].

hot spot is a router in Network-On-chip-Based Multi core systems, it gets the data when other networked elements will get much data which they can not consume. Hot spot is a problem, physically it is a router. It contains the data which exceeds their buffer size. this paper is focusing on the issues, who are responsible for the creation of the hot-spot in the on-chip multi core systems. if hot spot will be generated it will affect the performance of the network, and indirectly it affects the performance of the multi-core architecture. Prediction of hot-spot in the network is a difficult task, because generation of hot-spot depends on the nature of the application. For the prevention of hot-spots technique described in this paper is based on the artificial neural network(ANN). ANN based technique predicts and avoids the hot spots in the network.
ANN is dynamic method it takes on line-statics data and predicts about the hot-spots. ANN method provides enough time to the multi-core systems, so that they can do some thing so that hot-spots can not be created.
ANN mechanism predicts the location, where the hot-spot can be generated. ANN based predictor can predict about the hot-spot accuracy ranging from 65% to 92%. it uses buffer utilization data for the prediction of the hot-spot. ANN was trained by synthetic traffic model.
Small ANN architecture can predict hot-spot formation with accuracy ranges between 65% and
92%, in future other parameters can also be used I.e link utilization, and topology to improve the
training of ANN. If we need to improve the performance of network-on -chip-based multi-core systems hot spot is a key challenge. It should be overcome by the ANN based methods.

(vi) Understanding off-chip memory contention of parallel programs in multi-core system [8]

Memory contention is an important performance issue in multi-core architecture. Parallel applications suffers because of the off-line memory contention. If problem size is small then less off-chip contention and busty memory traffic, if problem size is big then more off-chip contention and non-busty memory traffic. On the basis of three parameters growth of memory contention, number of active cores, problem size this paper has proposed an analytical model. This analytical model shows, how the memory contention affects, with respect to number of active cores & the problem size, for UMA & NUMA memory access. Memory contention increases exponentially with number of active cores. But if we will use additional memory controller, the memory contention will be reduced. Memory contention increases the number of processor cycles. In this paper they had did the analysis on pentadiagonal solver SP from NPB bench mark, with a large matrix. They have did the analysis with the machine having 24 cores. This paper is inspired by a series of observations derived from experiments on state of the art UMA and NUMA systems using 8,24 and 48 cores previous research has stated that memory traffic is always busty, this paper discovered that the burstiness of memory traffic depends on the problem size.
Program with large size and higher memory requirement leads to large memory contention factors but have non-busty memory traffic. Here they have proposed analytical queuing model for programs with large contention in both UMA & NUMA multiprocessor system.
Analytical model was validated against the two parallel program benchmarks NPB and PARSEC.
Model has high-accuracy and differs from measurements by 5-14% for problems with large contention in the range of problem size and number of cores used in the experiments, NPB, PARSEC, parallel programs.
In a multiprocess system, increasing the number of cores generally also increases memory contention, reason is that, if number of cores will be increased, then it will also increase the load in the interconnection network.

More cores will try to use the interconnection network at a time. If there is only one core, so 100% bandwidth is available to that core. If two cores are in the network, so probability of first core to get the channel is ½.paper has given the improvements when using the analytical approach,On small program size , AMD NUMA, 48 core processor observed 50% increase of memory contention.
On program with large memory requirement, memory contention increased as 1000% on 24 cores on an inter NUMA system. These tests have been did on SP.C program.
Results shown in this paper are, if problem size is small, less cache miss, then less main memory access, so less traffic and also busty traffic.
If size of the problem is large, more cache miss, more main memory access, so non-busty traffic.
The model proposed in this paper has following limitations- accuracy is decreased for the programs, where less degree of contention is exist. Second limitation is accuracy is decreased for the programs, for low memory requirement.
This model is best for the programs, where the memory requirement is higher. Other alternative solutions can also be there, we can use one special processor who can predict the contention, & perform off-line input output activity for the memory. But as it all depends on the application. So further work can be done on this.

(vii)Task oriented programming: A suitable programming model for multi-core and distributed systems[9].

This paper describes the importance of task-oriented programming languages, to improve the performance of multi-core architectures. Programming languages available, for the multi-core & distributed systems are not efficient. To in crease the performance of the multi-core architectures, we should have to use new programming models., like task-oriented programming.
Older programming languages using multi-threading library p-thread, which is more complex and also low level. Creation of thread involves the system calls, which degrades the performance of the system. Instead of that we should have to focus on the task-oriented programming. OpenMP programming interface is the solution available for the multi-core architecture, but internally openMP is also using the pthread library. openMP must also have to handle the synchronization & concurrency.
MPI has more than 500 functions and hence it is extremely hard for a scientist who needs high performance computing to program his solution. In task oriented programming task and required data both have given to the remote computer and remote computer executes the task and returns it to the needy computer is data scheduling. If data is available far away from the task, it will put higher penalty on the performance of the multi-core architecture.

(viii) Resource management on multi-core system: The Actor approach [10].

Applications running on a system must have to be assigned, small portion of the resource and give them feel that it is running on individual, no one other than that application is running on the system. With this we can know the performance of that application, how much time that application is taking. Same concept can be used in the multi-core architecture, where we can assign the application to those cores where we want to test the performance of application. We can do this with the help of the virtualization. Virtualization is an old concept, where each application has alloted to the virtual environment, that virtual environment is consist with the help of one or more virtual machines. These virtual machines are mapped to the actual machine/physical machine. This concept is use ful in mobile phones, embedded applications where we need to decide, which task we need to give more part of the resource and to whom less resource. After giving the resource to an application we need to check the quality of service with the help of feedback, this feedback will be given by the resource manager. The actors project is useful to portion the application with the help of the resource manager and then distribute these application to different cores. In future this work can be extended for the heterogeneous multi-core architecture.

(ix) Improving Multi-Core Performance Using Mixed-Cell Cache Architecture[11].

In this research work, they have presented that todays multicore architecture related devices are have requirement of less power, if we want that whole device should consume less power, we should have to minimize the power of individual core. Bit lowering the cache voltage causes the reliability issue. This paper states that we need to make two type of cells in the cache one which is required less voltage and one which is required higher voltage, thus it gives the concept of Mixed-cell. This paper shows the improvement of the multicore architecture when storing the modified data in the robust cell(cell required more power), and other data in the less-robust cell.
Mixed-cell cache architectures enable low-voltage operation for a fraction of the cache composed of robust cells. This trade off allows using the entire cache when operations at high voltage, but loses capacity at low voltage. This study improves the performance of multi core architecture by 17% and reducing the power requirement by 50% of L1 cache.

(x) Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors[12]

This research paper have focused on the study of three major performance elements, processor cores, memory hierarchy, interconnection network. If there will be one simulator who can simulate and can show us that how cores are utilized by the threads at run time we can propose a new design scheme. In the same way memory hierarchy is also a important factor, how cache is utilizes by the processor. And the interconnection network, can also be simulated with this simulator. This researchers have given the option of more number of multicore architectures can be proposed with the help of the simulator. Main problem is that the environment if we want to make of multicore architecture is very costly.

(xi) Cache-Based Memory Copy Hardware Accelerator for Multicore Systems[13]

for improving the performance of the multicore architecture, this paper has focused on the one important issue that is data movement between main memory and cache memory. If we can increase the speed of data movement speed this is called Accelerator scheme.

In this paper they have used the theory called open-queuing theory so that the can study how performance can be improved with the help of the accelerator concept. accelerator provides speedups from 2.96 to
4.61 for the receiver-side of the TCP/IP stack, reduces the number of instructions from 26 percent to 44 percent and achieves a higher
cache hit rate Utilizing the analytical analysis, this accelerator reduces in the average number of cycles executed per instruction up to 50 percent for one of the CPUs in the multicore system
Limitation of this scheme was that it can improve the performance only in case we are doing copy operation between the main memory and the cache memory.
But this scheme is not applied when real time updation is done in main memory and the cache memory. Because now a days our multi core architectures are used in real time applications. So in future research can be done on accelerator for the updation in cache and memory at real time or run time.

(xii)Performance prediction and improvement techniques for parallel programs in multi processors[14]

Magnus Broberg has developed some techniques to analyze the sequential program, and then visualize it through the tool he has developed. Visualization process shows, if same program will be executed in the parallel machine, what hurdles, performance limitation will arise.
Magnus Broberg has tested their tool with the help of the uniprocessor. Main motivation was that, a developer creates the programs in sequential machine. He has tried to develop the parallel program in sequential machine.

III. PROPOSED SOLUTION

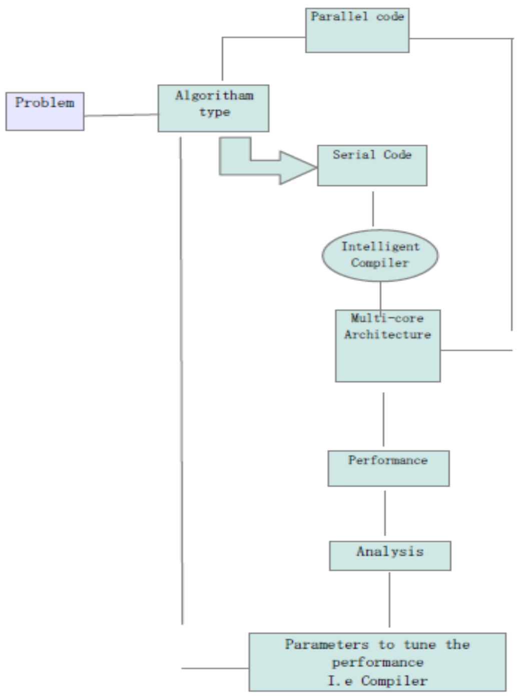

Figure 5.1 Proposed methodology used in the Performance improvement.

In the proposed research work, we will take a problem then select the type of solution is available for the problem. If type of solution is sequential, then it will produce the serial source code. For finding the scope of parallelism, we need to use the Intelligent compiler. Compiler will try to detect the instructions in the source code who are independent. Then compiler will make another file based on the serial source code, that new file is having the scope of parallelism, and multi core architecture can utilize that parallelism for the performance improvement. After the execution of that partial parallel code speedup will be note down. Then we will analyze the speedup. During the analysis, we will focus whether our source code have used all the cores or not. If some cores was ideal, we will try to find out the reason of that why cores was ideal. Reason may be lack of parallelism, inter thread communication problem, data availability. After this analysis we will make the changes in the source code with the help of the intelligent compiler. Changed code will be again given to the multi core machine. This process will be repeated till we will not get the improved speed up.
If the problem solution is made with the help of parallel programming languages, then the code will be directly given to the multi core machine. Speed up will be noted down. Then the analysis will be done for the speed up. A list of performance barrier factors will be checked, I.e interconnection network delay, cache-coherence. After the analysis we will try to find out who is problematic area in the multi core architecture whose can be corrected for the performance. A visual scheme can be used to check for this. We need to use a simulator who can detect the problem and can suggest for the improvement.

IV. EXPECTED OUTCOME OF THE PROPOSED WORK

(a) We will Find the concurrency in a program, by the intelligent compiler.
(b) We will find the efficient task scheduling strategy who will help in exploiting the power of multi core architecture.
(c). We will solve the data locality problem, with the help of efficient cache allocation stretgy.
(d) We will come with higher performance in the multi core architecture in terms of speed up.
(e) We will use the capability of parallel programming languages for the performance improvement, we will tell what features are more important for the performance improvement.
(g) We will find out which API is efficient in process communication MPI, openMP.
(h) We ill suggest the synchronization constructs and protocols that enable programmers write program free from deadlock and race conditions.

V. CONCLUSION

Multicore architecture performance improvement have big requirement. There are so many performance parameters available for performance improvement, we have discussed here only comipler. In future simulator can be implemented for this concept. Manual parallelism identification is a very tedious job. Automated tools are the requirement for the identification of parallelism.

REFERENCES

[1] http://software.intel.com/en-us/blogs/2008/12/31/top-10-challenges-in-parallel-computing

[2] Advanced Computer Architecture, parallelism, scalability, programmability, Kai hwang 2nd ed. McGraw-Hill, 01-Feb-2003

[3] Bryan Schauer “Multi core Processors – A Necessity” Pro Quest Discovery Guides, September 2008

[4] Raghavan Raman, “Compiler Support for Work-Stealing Parallel Runtime Systems”, Ph.D. dissertation, Dept. Computer Science, Rice University, Houston, Texas , May 2009

[5]. Damian A. Mallon, et. al, “Performance Evaluation of MPI, UPC and OpenMP on Multicore Architectures” in under Project TIN2007-67537-C03-02, spain.

[6]. Dmitri Perelman, “Exploiting Parallelism of Multi-Core Architectures”, Ph.D dissertation, Department of Electrical Engineering, Israel Institute of technology , Haifa Israel September 2012.

[7]. Kakoullie, E. et. al, “Intelligent Hotspot Prediction for Network-on-Chip-Based Multicore Systems” in computer aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol.61 no.3 , pp. 418 – 431, 2012.

[8]. Tudor, B.M. And Young Meng Teo, “Understanding Off-Chip Memory Contention of Parallel Programs in Multicore Systems”, IEEE Conference in Parallel Processing (ICCP), Taipei , Singapore, 2011, pp. 602 - 611

[9]. Shahrivari, S & Sharifi, M, “Task-Oriented Programming: A Suitable Programming Model for Multicore and Distributed Systems”, IEEE Conference in Parallel and Distributed Computing (ISPDC), Cluj Napoca, Iran, 2011, pp. 139 - 144 .

[10]. Bini, E et. Al, Resource Management on Multicore Systems: The ACTORS Approach, published in IEEE Conference in Micro, 2011, pp. 72-81.

[11]. Khan, Samira M,et. Al “Improving Multi-Core Performance Using Mixed-Cell Cache Architecture”, in High Performance Computer Architecture (HPCA2013), IEEE 19th International Symposium, Shenzhen, China, 2013, pp. 119 - 130

[12]. Ubal, R et.al “Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors”, in Computer Architecture and High Performance Computing,19th International Symposium, Rio Grande do Sul, 2007 pp. 62-68

[13]. Durate, F and Wong, S “Cache-Based Memory Copy Hardware Accelerator for Multicore Systems”, Published in Computers, IEEE Transactions, (Volume:59 , issue:11), 2010, pp. 1494-1507.

[14] Magnus Broberg, **“**Performance prediction and improvement techniques for parallel programs in multi processors”, Ph.D dissertation, Department of software Engg. & computer sc., Blekinge institute of technology Sweden, 2002

Surendra Kumar Shukla has completed his UG degree from SGSITS Indore in computer Engineering and completed his Masters of Engineering from IET DAVV Indore in Computer Engineering Branch and presently pursuing PhD from DAVV Indore ,He published many research papers in various national and international conferences , he is doing research work in the area of multi core architectures , he won the gold medal from IET DAVV for his studies in the year 2011

Vishal Trivedi Completed his BE degree from RGPV Bhopal in Information Technology Branch, after that he completed his Masters of Engineering from DAVV Indore in Information Security Branch, he is a member of CSI , and published research papers in international journals

Ayush Choukse has completed Be from Chameli Devi Group of Institutions RGPV University Bhopal in year 2013 .