ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

Framework for Dynamic Scheduling and Balancing in Multicore Asymmetric Thread Processing

J Akshya*, B Srinivasan

School of Computing, SASTRA University, Thanjavur, Tamilnadu, India

E-mail: akshya0503@gmail.com
E-mail: srinivasan@core.sastra.edu

Abstract: Today's modern technologies have leads to many innovations in all the fields. One such area is Multicore Architectures where several cores are used to improve the efficiency and performance of a processor. Methods\Statistical Analysis the Symmetric processors have identical cores. But, in the case of asymmetric processors, it is different. They do not have identical cores. Findings: Various studies have shown that using an Asymmetric Multicore processor produces better results than the Symmetric Multicore Processors. This is due to the various hardware designs present in the AMPs. The existing schedulers for an AMP are used to schedule the jobs in a sequential way. Application Improvements: The proposed Asymmetric scheduler is used to schedule various jobs in parallel. Experimental results are shown using Proteus simulator where 25 percentage of the total time has been reduced by the proposed scheduler.

Keywords: Multicore processors; Asymmetric architecture; Proteus simulator; Scheduling; Dynamic; Profiling; Load balancing.

I. INTRODUCTION

A processor is a logical circuit that responds to the basic instructions provided by the users to the computer. A processor might contain one or many cores within them generally known as multicore architecture. A multicore architecture is in turn divided into symmetric and asymmetric multicore architecture systems. When all cores in the system function independently then they are known to be symmetric else asymmetric multicore processors. Symmetric systems contain a single chip with multiple cores, where the entire core functions independently. Whereas, in the asymmetric system, it contains a single chip with multiple cores, where all the cores might be of various designs. The cores in the AMPs are not identical and they may be used for various works.

1.1 Fundamentals of Processor

A processor in a system is used to process all the tasks that are given to it and perform all the operations that are given to them. The processor consists of a chipset which is the collection of a set of chips. A chip is used to help the processor to interact with the physical memory and all the rest of the components in the processor.

1.2 Multicore Architectures

Overview: Nowadays, with the increase the modern technologies many new inventions have arisen. One such invention is the multicore architectures. So what could be the main difference between an ordinary single core and a multicore architecture? The detailed description is as follows.

1.3 Symmetric Multicore Architecture

In symmetric multicore architecture more than one core is present. But here all the cores are identical. That is they have the same hardware designs such as the area occupied, the cache miss, the amount of memory needed and so on. They do not differ from one another in any aspect. For example, Intel quad core has four cores present within itself and all the four cores are identical to each other. They don't differ from each other in any aspect.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

1.4 Asymmetric Multicore Architecture

In an asymmetric multicore architecture more than one core are present. But not as like in a symmetric multicore system where the cores are identical here the cores may have different characteristic designs. The cores may not be identical and they may have various hardware designs. The best example of asymmetric multicore architecture is IBM Cell processor where it has eight cores. Here, seven cores are identical and they are used for the input-output operations. Whereas the remaining one core is used for vector operations.

II. PROBLEM STATEMENT

Earlier experts have designed a system which could only support the underlying homogeneous architectures. Homogeneous architectures are the system where all the cores that are present are identical that is, they have the same hardware designs. Then in the later stage, the experts designed a new system where it could also support the heterogeneous systems, that is, where all the cores may not be similar. They were not designed to support parallel execution. This increased the time and the power consumption to a larger extent.

2.1 Problem

When the system doesn't support the parallel execution then there is a chance of higher energy consumption. The amount of memory requests issued by processor cores increases memory starvation. This drawback on the number of memory accesses decreases the performance of the modern multicore system. The system totally fails in the case of heterogeneous a processor which has to be executed in parallel.

2.2 Solution

In order to reduce the time and power consumption to a greater extent we need to design such a system that it could be run or executed in parallel. When parallel execution takes place, it reduces the system power consumption to a greater extent. When sequential applications are executed the amount of memory requests that are given by the processor cores increases memory starvation. Scheduling is done independently irrespective of the clock speed of the heterogeneous processors. The model framework is very dynamic and fast. It also schedules and balances the loads on the fly.

2.3 Solution-Benefits

The main advantage or the benefits of the solution is it has very low computational overhead. It consumes less energy when compared with the existing systems. Memory consumption is not uniform based on the requirement. It speeds up the threads very dynamically and parallel execution is done. Job or task completion is fast and accurate.

2.4 Objectives of the Study

The main objective of the proposed system is to develop an effective scheduler and a balancer for multithreaded operations that utilizes the memory - RAM efficiently The system should also cause lesser computational overheads when using heterogeneous architecture based processors. This is done by using the SIMIC library files which show the execution of the jobs in parallel. This is done in order to reduce the power consumption and reduce the computational overhead in the system.

2.5 System Architecture

All the systems have their own architectural designs. The architectural design gives us the entire concept in a diagrammatic way. It is used to better understand the concept which takes part in the framework. The architecture diagram is split into multiple parts. The detailed architecture diagram is given in Figure 1. Each part has their own jobs that need to be done during the execution time. All the jobs that are given here are processed in parallel and not in a sequential manner. This reduces the time consumption to a greater extent.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

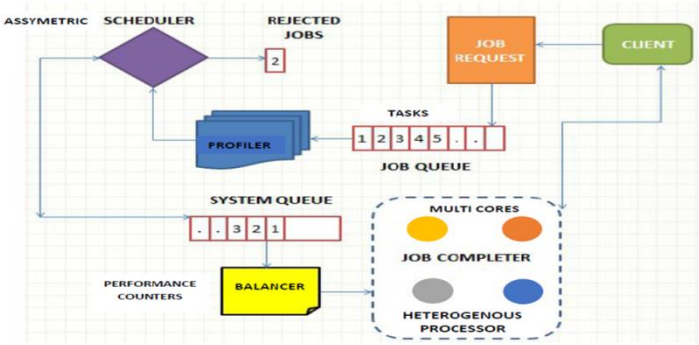

Figure 1: Architectural design.

III. METHODOLOGIES AND APPROACHES

Each framework has its own methodologies and approaches that are carried out by them self. They stock with the methods that are given and produce the output to the end user. Same as like in other frameworks in the proposed framework also some methods are used in order to get the desired out. The system follows some algorithms as referred in order get the best output [1].

3.1 Algorithms

3.1.1 Profiler: mechanism of profiler:

- Create a new thread to receive and deal with the profiling data
- Repeat profiling
- Send profiling report to data dealer of the model
- Terminate Repeat loop

3.1.2 Data dealer: receives and deals with the profiling data:

Input: Provide with profiling data

- Repeat till profiling mechanism terminate
- Receive data and filter them from received profiling data
- If loading is unbalanced or changes in behaviour of processes or important core are idle
- Compute AMP Speedup Factor
- Sort the processes in AMP-list
- Trigger schedule by sending signal
- Terminate If condition
- Terminate Repeat loop

3.1.3 Schedule: schedule asymmetric processor:

- Compute number of powerful core based on load balancing policy
- Retrieve from AMP-list the processes to be scheduled on powerful cores
- If Retrieved processes is not equal to current processes on powerful cores
- Migrate process
- Terminate If condition
- If present Resource Contention Degradation is very high
- The processes with heavy Contention Degradation Factor is scattered
- Migrate processes

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

- Terminate If condition

3.2 Policies

There are various policies that are used in the framework. But the major two policies that are present in the system are as follows.

3.2.1 Asymmetric-aware schedule policy:

There are two types of processors in multicore systems. First, where few high-speed cores are present and another one where a larger amount of slower cores will be present. The first discussed core is called as the faster core and the second as the slower core. Both cores have many differences. Some of them are the complexity of execution, performance of execution and so on. Now when a process wants to be scheduled it should be decided on which core the process should be executed. It may either get executed on the faster core or else might be on the slower cores [2].

3.2.2 NUMA-aware schedule policy:

As in increase with the high level of access to the memory and data starvation of the process, the usage of NUMA architecture is constantly increasing. The other major problem that is present in the existing systems is the remote memory access. In the existing systems, the NUMA is very non-uniform which reduces the performance of the system and the time consumption is high. But in the proposed system the NUMA is uniform so that it consumes less amount of time when compared to the previous systems. The idea of inter-communication transfer keeps on increasing between the various threads in the processor [3-7]. Figure 2 shows the design of NUMA - aware scheduler.

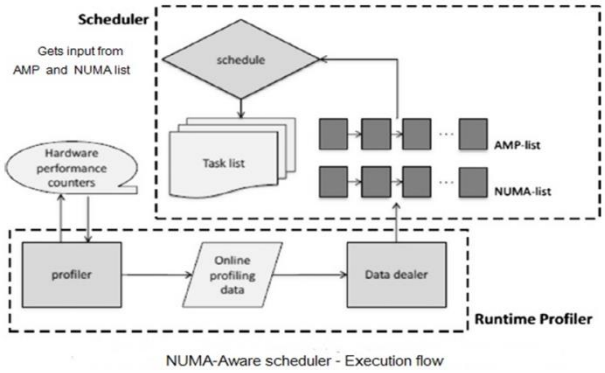

Figure 2: Numa - aware scheduler.

IV. OUTPUTS

The results of the design are shown with the help of two simulators. One is Proteus simulator and the other one used is the Simics simulator. Initially, the Proteus simulator is used to show the framework in a graphical representation. The entire core design is shown here. Then the Simics simulator is used to show how the jobs are scheduled and how it makes use of the benchmark programs that are given to it. The Simic simulator gives the main design of the framework and the same sequence happens in the entire framework [8-10].

4.1 Proteus Simulator

The model of the framework before and after the execution is shown in the images that are presented in the upcoming section. The simulation is shown in the VSM analyzer and the simulation log also depicts the form in which the simulation takes place inside the processor. The Proteus simulator is the simulator that provides the design how the core is actually designed in the system and what are the major components of the system. The hardware that is present

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

in the computer is depicted in the Proteus simulator. The wiring's and the execution flow of the data is shown in the simulator with the help of the simulation.

4.1.1 Simulation - before

The panel shows the toolbar where all the operation commands are present such as a file, view tools and so on. The left panel shows the view of the microcontroller and other devices that can be used in the framework. The panel below the frame shows the buttons which trigger the simulation in the system. The design shows the 8051 microcontrollers that are used in the system. All the controllers are wired with each other and there is a digital analysis that shows the digital representation of the data that flows through the microcontroller. The controllers are grounded and the voltage supply is given to the microcontroller.

4.1.2 Simulation - after

The simulation is very useful for knowing how actually the process occurs and what are the graphical representation of the process that is prepossessed in the simulator. The VSM analyzer depicts various parameters such as memory used, the cache used, bandwidth, network and all other parameters that can be measured by the simulator. The program is written in C language and is loaded into the simulator. According to the coding the simulator runs.

4.2 Simics Simulator

The complete design of the Simics simulator is given in Figure 3. The Simics simulator is used to give the job to the framework. The user is asked to give any job to the simulator and see the results. Here the model is divided into two modules.

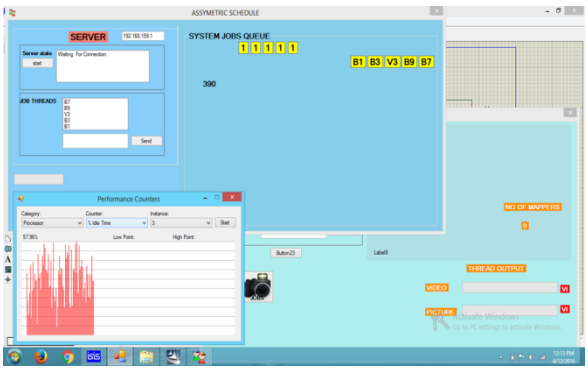

Figure 3: Asymmetric scheduler.

4.2.1 Module 1 - asymmetric scheduler:

The asymmetric scheduler is the module tin which the jobs are scheduled. In this framework after connecting to the server and after when the jobs are loaded from the Thread job giver module we can see how the scheduling takes place in the framework. Each and every job is scheduled one after the other and the usage of memory is seen in the framework. The framework is built using the Java JDK. The number of jobs that are scheduled is also shown in the framework.

4.2.2 Module 2 - thread job giver:

The second module is the job giver. As mentioned it is used to give the job to the framework. The job can be given by connecting the client and the server together. For that, the IP address is required. When both the IP addresses match then the framework can be loaded with the job. The thread output shows how the thread's output varies when different jobs are given and the number of mappers gives the number of how many jobs are given to the scheduler.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 6, Issue 4, April 2018

V. CONCLUSION

A dynamic asymmetric framework was created which was able to schedule various jobs given to it in a parallel way. Threads were created for scheduling the jobs with lesser computational overhead. The results obtained were better than the results obtained from the existing systems by at least 12 percentages. The framework involved some simulations with the use of Proteus and Simic's simulators. The workloads that were given to scheduler were not same. Heterogeneous workloads were given t the scheduler. The executions that were done on the framework were dynamic. All the loads that were given to the framework were balanced properly and then scheduled by consuming lesser energy.

VI. REFERENCES

1. D Jiun-Hung, C Ya-Ting, et al. An efficient and comprehensive scheduler on Asymmetric Multicore Architecture systems. Journal of System Architecture 2013; 60:305-314.
2. K Rakesh Kumar, FI Keith, et al. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. Annual International Symposium on Computer Architecture 2004.
3. L Tong, B Dan, et al. Efficient operating system scheduling for performance-asymmetric multi-core architectures. IEEE Conference on Supercomputing 2007.
4. S Juan Carlos, P Manuel, et al. A comprehensive scheduler for asymmetric multicore processors. Conference on Computer Systems 2010: 139-152.
5. LA Barroso, K Gharachorloo, et al. A scalable architecture based on single-chip multiprocessing. IEEE International Symposium on Computer Architecture 2000.
6. O Beaumont, A Legrand, et al. The master-slave paradigm with heterogeneous processors. IEEE International Conference on Cluster Computing 2001; 14:897-908.
7. J Burns and JL Gaudiot, SMT layout overhead and scalability. IEEE Transactions on Parallel and Distributed Systems 2002; 13:142-155.
8. RH Denard, The design of ion-implanted MOSFETs with very small physical dimensions. IEEE Journal of Solid-state Circuits 1974; 98:256-268.
9. R Espasa, F Ardanaz, et al. Tarantula: A vector extension to the alpha architecture. IEEE International Symposium on Computer Architecture 2002.
10. R Figueiredo, J Fortes. The impact of heterogeneity on DSM performance. In Sixth International Symposium on High-Performance Computer Architecture. 2000.