IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

Overview of Scheduling on Multi-core Processing

Hema K. Reddy[1], Dr. G. R. Bamnote[2]

M.E. Student, PRMIT Badnera, India[1]

Professor, Department of CSE, PRMIT Badnera, India[2]

Abstract: The Multi-core processors are requirement of recent needs. As the computers laptops and mobiles emerged with the need of high end computing as well as battery drainage problem it is much more necessary to work on scheduling. With the evolution of multiprocessors there is increase in battery consumption. Effective utilization of multi-cores of processors will ultimately enhance battery life. ARM's big. LITTLE architecture provides a way to efficiently schedule the tasks on the multiprocessors. This paper explores study on dynamic real time scheduling using Global task scheduling algorithms. This paper illustrates scheduling on current systems and various factors which contribute to the performance up gradation of the system.

Keywords: Multi-core processors, big. LITTLE, Real Time Systems.

I. INTRODUCTION

The era of yesteryears was of single processor, where there was a single core architecture for the computations. As the need aroused the computers were able to work not only for computations but for storage, fun entertainment, audio ,video, and gaming purposes, which in turn lead to demand for high processing speeds and eventually more battery drainage in laptops .

To overcome the issues with Battery drainage the manufacturers concluded that if the processor speed is reduced the battery consumption is enhanced. So there was need for slow but multiple processors, which in turn evolved multicore processor architectures. Theses multiprocessor architectures are mainly to boost up the processing especially for the independent processes so that the output can be combined to form the final solution and get the execution at the distributed level.

Mainly multiprocessors are used in smartphones which executes multiple tasks at a time. Due to these multiprocessors energy is consumed on a large scale and to reduce this energy consumption we need to propose new techniques to reduce battery consumption without compromising with the performance. Previously in most of the mobile phones, a processor giving higher performance or fast processing (Big processors) are not power efficient and processors consuming less power(Little processors)do not provide high performance. Thus our goal cannot be achieved using same type of processors When multiple tasks are running at different cores it is important to know the dependent tasks and independent tasks. Similarly the two tasks are to be grouped in the different categories, and are to be scheduled according the requirement of results.

In Independent processes the processes can be distributed to different cores and are to be executed without much care and botheration, but in case of dependent tasks those are to synchronised so that the results can be obtained at proper sequence as the tasks are interdependent, so their results are. Similarly the processes are to be executed concurrently by taking care of their concurrency, their synchronisations and inter process communication problems.

The problems in scheduling are to considered are based on various factors. As the tasks are running concurrently the Mutual Exclusion Problem, Critical Section Problem, and the deadlocks are to be dealt with. As the tasks under consideration are non real time, if the tasks become real time then deadline becomes an important criteria while deciding the system

The real time system is classified in two types' viz. Hard Real Time system and soft real time system. In Hard Real Time system, if the deadlines are not met then catastrophic failures may occur, but in Soft real time systems failure to some extent may be tolerated, but if average is attained then the soft real time system can be called as successful.

When same processors are replicated in the multi cores the cores are homogeneous, but if there is difference between pair of cores then the combination can be called as heterogeneous cores. Now as in case of mobiles the requirement was for high end gaming softwares, for which high end computing processors are required, much of the time the main functionality viz, calling and texting are useful, which can be executed on low end processors also. To save the power energy which is nothing but battery power is saved when high end computing processors are not employed or idle. To have the best of both worlds, i.e. High end computing Processor and Low end computing processor, ARM has given big.LITTLE platform. To overcome the problem, ARM has launched a technology termed as "big.LITTLE technology" which is a

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

heterogeneous computing architecture consisting of two types of processors.

II. LITERATURE SURVEY

DESPITE the success of current symmetric chip multiprocessors (CMP) to take advantage of thread-level parallelism (TLP), the improvement in performance by TLP will tend to decrease if single-thread performance does not show improvement correspondingly [1]. This viewpoint positive discriminate asymmetric (or hybrid) multicore processors, which merge one or a few big cores with multiple smaller cores. A big core is well suited to execute sequential program sections where all cores—except one—are idle. In current scene, even small rise in sequential performance are amplified by a degree that justifies the employ of techniques that would be deemed inefficient in the absence of multiprocessors [1].

As the thermal and power factors per processor are restricted, increased efficiency of energy and smart handling of wire delays affect directly into higher single-thread performance, especially for the big cores mentioned above. Clustered (or partitioned) micro architectures are a well-known architectural concept that handles the wire delay problem, keeps the important circuit intricacy low, and is particularly power-efficient [2]. Traditionally, load/store queue entries are assigned to memory instructions in program order during the dispatch stage before they go into the out-of-order core. Only after the address was considered and when the instruction enters the memory pipeline, is its queue entry occupied. The entry remains occupied until the instruction commits.[3]

III. FACTORS UNDER CONSIDERATION FOR BIG.LITTLE

1. At the high performance end: high compute capability but within the thermal bounds

2. At the low performance end: very low power consumption

ARM big.LITTLE™ technology has been designed to address these requirements. Big.LITTLE technology is a heterogeneous processing architecture which uses two types of processor. ”LITTLE” processors are designed for maximum power efficiency while ”big” processors are designed to provide maximum compute performance. Both types of processor are coherent and share the same instruction set architecture (ISA). Using big.LITTLE technology, each task can be dynamically allocated to a big or LITTLE core depending on the instantaneous performance requirement of that task. Through this combination, big.LITTLE technology gives the answer that is capable of providing the high peak performance demanded by the the current mobile devices, within the thermal limitations of the system, with maximum energy efficiency.[4]

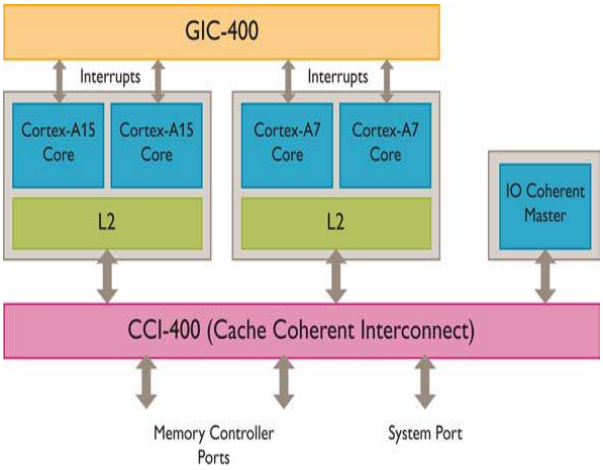

Fig 1: Typical big.LITTLE system

Same architecture but different micro-architectures

The first big.LITTLE processing pair consists of the ARM Cortex®-A15 and Cortex-A7 processors. Since both processors support the same ARMv7-A ISA, the same instructions or program can be run in a consistent manner on both processors. Differences in the internal microarchitecture of the processors allow them to provide the different power and performance characteristics that are fundamental to the big.LITTLE processing concept. Future designs will also utilise the Cortex-A53 and Cortex-A57 processors in a big.LITTLE implementation [4].

In a multi-core system, the processing time of a memory request is highly variable as it depends on the location of the access and the state of DRAM chips and the DRAM controller: There is inter-core dependency as the memory accesses from one core could also be influenced by requests from other cores; the DRAM controller commonly employs scheduling algorithms to re-order requests in order to maximize overall DRAM throughput [5]. All these factors affect the temporal predictability of memory intensive real-time applications due to the high variance of their memory access time. This imposes a big challenge for real-time systems because execution time guarantees of tasks running on a core can be invalidated by workload changes in other cores. Therefore, there is an increasing need for memory bandwidth management solutions that provide quality of service (QoS).[6].

MULTICORE platforms are becoming increasingly popular in modern computing systems since they have a high processing capacity at a comparatively low cost. Shared resources on the multicore chip, such as main memory, are increasingly becoming a point of contention. For example, processing high resolution images on one core, while tracking objects from real-time vision data on another core, may cause a mutual slowdown due to interference for access to the shared memory, which was not a bottleneck when these tasks were run on a single-core system.

While the real-time scheduling problem has been studied for several decades, it has traditionally focused on

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

scheduling CPU computation. One fundamental assumption is that we can estimate Worst-Case Execution Time (WCET) for each task when running alone in the system. However, when considering memory-intensive applications running concurrently on a multicore chip, the measured worst-case execution times can vary significantly, and may change depending on what is running on the other cores on the system. In the worst case, all the cores may simultaneously compete to access the main memory, and the worst-case task execution time can grow linearly with the number of cores in the system [7]

IV. METHODOLOGY

In case of heterogeneous multi-core scheduling, very few studies have focused on global scheduling mainly due to insufficient architectural support for migration in commercial heterogeneous multi-core chips.[8]

The Characteristics and Components of Real Time Operating System

There are many aspects in which Real Time Operating System has similarities with common operating System. To make hardware system available it controls and manages variety of hardware resources. It is responsible for execution of applications and tasks in timely manner.

I. Various Components of RTOS

Following Components are there in most of Real Time Operating Systems: Scheduler – Every Kernel is having scheduler at the core. The responsibility for determining when to execute which task, and it provides corresponding algorithms is performed by scheduler.

Objects-Tasks, semaphores and message queues are the most common RTOS Kernel Objects.

Services- For Real Time embedded system applications to be created the developers gets services from the kernel. In general few services are facilitated like:

Management of Timer

Handling of Interrupts

I/O of Devices

Management of Memory

Various applications are based on embedded systems. The Tasks can be proactive or reactive dependent on the needs like interface, scalability, connectivity etc. Deciding the system for an embedded machine is based on the analysis of the operating System itself and the needs of application.[9]

Characteristics

1. Its real time characteristic- Finish the tasks within the limited time and events are to be responded similarly.

1. High priority task to be executed first is the scheduling objective.

2. The jobs executing on real-time operating system should be definite

3. The data might be extremely sharing in real-time operating System[9]

V. FACTORS THAT AFFECTS REAL-TIME CHARACTERISTICS OF OPERATING SYSTEM

A. Scheduling of Tasks

The real-time operating system should accept scheduling kernel that is pre-emptive, which is based on task priority. The non pre-emptive scheduling mechanism based Operating System, must not have strict real-time characteristic.

Pre-emptive scheduling supports a good foundation for real-time system. The scheduling systems efficiency can be maximised by making the operating system run with particular real-time scheduling algorithm.

There are few real-time based scheduling algorithms, for example the Liu and Layland Rate-Monotonic (RM) scheduling algorithm and the (EDF) earliest deadline priority algorithm. The static type of scheduling algorithm is RM scheduling algorithm, in this the priority of tasks are considered by the length of the cycle of task, and the shorter cycle of task has a higher priority. Another algorithm which is dynamic priority scheduling algorithms is EDF algorithm that define priority of tasks with respect to their deadlines. Therefore an excellent task scheduling algorithm can improve the performance of operating system's real-time characteristics. However, it also consumes system resources to some extent. Therefore, time complexity of any scheduling algorithm, in turn, has an impact on the real-time characteristic.

B. The context switching time

Processor transferring control from one task which is currently executing to another ready for executing one in Multi Tasking System is Context Switching . There are a lot of conditions that can lead to context switches, such as external interrupt, or freeing of resource which high priority tasks wait for in preemptive scheduling systems. in an operating system the linkages of tasks are achieved by the process control block (PCB) data structure. When context switching occurred, the previous tasks information was saved to the respective PCB or stack PCB given. The new task gets original information from respective PCB. The time switching consumed rely on the processor architecture, as different processors require to preserve and store different number of register .The efficiency of context switch will be affected by Operating system data structures

C. The time of kernel stopping interrupt

The kernel of operating system is required to stop all of the interrupt sometimes. Interrupt are going to crack the sequence of instructions, and may cause loss of data. keeping out interrupt generally delay the response of request and switching of context.. The operations which are non critical can be inserted between the critical areas in order to get better real-time performance of operating system,. We can reduce the prohibition time of interrupt by setting reasonable preemptive points in critical areas[10].

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

VI. CONCLUSION

The Performance of Real Time system greatly depend upon the choice of correct Operating system for particular application In the embedded system the improvement of real-time performance of kernel has great significance. The bond which is complex between the tasks may lead to serious system consumption on internal communication between tasks and corresponding delays.. The separating and scheduling the tasks in the start will also give us suppleness to apply different algorithms for different task queues. Harmonization mechanism between tasks will disallow the real-time performance of system. Therefore which scheduling algorithm to employ a real-time operating system to apply to an actual application system is the answer for all embedded system developer.

REFERENCES

[1] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," IEEE Computer., vol. 41, no. 7, pp. 33–38, Jul. 2008.
[2] V. V. Zyuban and P. M. Kogge, "Inherently lower-power high-performance superscalar architectures," in Proc. Int. Symp. High Perform. Computer. Architecture 2001, pp. 268–285.
[3] Stefan Bieschewski, Joan-Manuel Parcerisa,and Antonio Gonz_alez, Fellow, IEEE "An Energy-Efficient Memory Unit for Clustered Microarchitectures", IEEE Transactions On Computers, Vol. 65, No. 8, August 2016,Pp.2631-2637.
[4] big.LITTLE Technology: The Future of Mobile,White Paper ARM, 2013.
[5] K. Nesbit, N. Aggarwal, J. Laudon, and J. Smith, "Fair queuing memory systems," in Proc. Int. Symp. Microarchit., 2006, pp. 208–222.
[6] Heechul Yun, Gang Yao,et.all.," Memory Bandwidth Management for Efficient Performance Isolation in Multi-Core Platforms", IEEE Transactions On Computers, Vol. 65, No. 2, February 2016 pp. 562-576.
[7] Gang Yao, Rodolfo Pellizzoni,et.all., IEEE "Global Real-Time Memory-Centric Scheduling for Multicore Systems",IEEE TRANSACTIONS ONCOMPUTERS, VOL. 65, NO. 9, SEPTEMBER 2016 ,pp.2739-2751.
[8] Hoon Sung Chwa, et.all. "Towards Energy and Feasibility Optimal Scheduling on big.LITTLE Platform", Proceedings of the 6th Real-Time Scheduling Open Problems Seminar(RTSOPS),July 7, 2015.
[9] P. S. Prasad, Akhilesh Upadhyay," Scheduling Policy and its Performance for the Embedded Real time System", International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 4, April 2013
[10] P. S. Prasad, Akhilesh Upadhyay," Advanced Sheduling Policy and its Performance for the Embedded Real time System", International Journal of Computer Architecture and Mobility (ISSN 2319-9229) Volume 1-Issue 6, April 2013.
[11] P. S. Prasad, Akhilesh Upadhyay,"Interrupt Mechanism for Hybrid Operating System", International Journal of Computer Architecture and Mobility (ISSN 2319-9229) volume 1-Issue 9, July 2013