ISSN: 0976-2876 (Print)
ISSN: 2250-0138 (Online)

DESIGN OF A PARALLEL MULTI-THREADED PROGRAMMING MODEL FOR MULTICORE ARCHITECTURE WITH RESOURCE SHARING

AJIT GIRI[a1] AND PADMAPRIYA PATIL[b]

[a]M.Tech, Department of Electronics and Communication Engineering, Poojya Doddappa Appa College of Engineering, Gulbarga, Karnataka, India
[b]Professors, Department of Electronics and Communication Engineering, Poojya Doddappa Appa College of Engineering, Gulbarga, Karnataka, India

ABSTRACT

Multi-core architectures have become main stream, and multi-core processors are found in products ranging from small portable cell phones to large computer servers. In parallel, research on real-time systems has mainly focused on traditional single-core processors. Hence, in order for real-time systems to fully leverage on the extra capacity offered by multi-core processor, different design techniques, scheduling approaches, and real-time analysis methods have to be developed. With the arrival of Multi- core Processors (MPs), every processor has now built-in parallel computational power and that can be fully utilized only if the program in execution is written accordingly. Also existing memory system and parallel developments tools do not provide adequate support for general purpose multi-core programming and unable to utilize all available cores efficiently. This study is an attempt to come up with some solutions for the challenges that multi-core processing is currently facing. This project contributes by developing a new multi-thread parallel programming model, "Spmd" (Single program multiple data programming model), for multi-core processor. The SPMD is a serial-like task-oriented parallel programming model which consists of a set of rules for algorithm decomposition and a library of primitives to exploit thread-level parallelism and concurrency on multi-core processors. The programming model works equally well for different classes of problems including basic, complex, regular and irregular problems. Its parallel execution makes it more efficient and less time consuming and its large set of input parameters also provides a wide range of simulation scenarios. Hence in this project the work deals with the synchronization procedure each application is assumed to be allocated on one dedicated core. However, it is further extended the synchronization procedure to be applicable for applications allocated on multiple dedicated cores of a multi-core platform. Likewise, the efficiency calculation of the resource holds times of resources for applications. The resource hold time of a resource for an application is the maximum duration of time that the application may lock the resource.

KEYWORDS: Multi-Core, Concurrent Programming, Parallel Programming, inner product

Multi- core processor (MP) is designed to increase efficiency and performance of the system by increasing multi-tasking, parallelism and throughput. A multi-core processor consists of multiple processing units residing in one physical chip having their own set of execution and architectural recourses which differs with the traditional shared memory parallel architectures (SMP) in both hardware and software designs [F.Nemati, 2012]. The numbers of cores in multi-processor are often limited to four or eight whereas in MP designers are thinking to place hundreds or even thousands of cores in a single chip. The cores in MP are more closely coupled than processors in SMPs. In multi core processor system no cache at any level is shared but in MP, L2 and L3 caches are shared by the multiple cores within a chip. Also, the interconnection scheme used for processor-to-processor and processor-to-memory is static for multi-core processor system and dynamic for MP [M. Behnam, 2007]. Similarly, from software design aspect MP also have different challenges than those of SMPs. These include, program or thread scheduling and better load distribution on the available cores. Next is the level of parallelism, MP favors threading level parallelism whereas multi-core processor works better for process or application level parallelism. Some other differences in software design for MP and SMPs may include, design of threads, algorithm decomposition techniques, programming patterns, operating system support etc [Rajagopalan, B, 2007] [Y .Liu X 2007].

The shift towards multi-core processor (MP) from single core processors systems has resulted in several challenges for hardware and software designers. With changes in technology from micrometre to nanometre, there is a significant increase of the number of cores in a chip. Now, it is computer designer's responsibility to determine a computational structure that can transform the increase in cores into a corresponding increase in computational performance efficiency [K . Lakshmanam, 2009]. This challenge must be

[1]Corresponding author

dealt with on several fronts, like modification in basic architecture design of each processor (core) to increase single or multi-thread performance, change in the architecture of memory system and development of new programming models for multi-core processors.

The SPMD, (Single program multiple data), is a new parallel programming model developed as a part of this project. The development of SPMD is motivated with an understanding that existing parallel developments tools do not provide adequate support for general purpose multi-core programming and unable to utilize all available cores efficiently as they are designed for either specific parallel architecture or certain program structure [F. Nemati, 2010]. The SPMD is developed to equip a common programmer with multi-core programming tool for scientific and general purpose computing. The SPMD provides a set of rules for algorithm decomposition and a library of primitives that exploit parallelism and concurrency on multi-core processors. The programming model is serial-like task-oriented and provides thread level parallelism without the programmer requiring a detailed knowledge of platform details and threading mechanisms. It has also many other unique features that distinguish it with all other existing parallel programming models. It supports both data and functional parallel programming. Additionally, it supports nested parallelism, so one can easily build larger parallel components from smaller parallel components. A program written with SPMD may be executed in serial, parallel and concurrent fashion on available cores. Besides, it also provides processor core interaction feature that enables the programmer to assign any task or a number of tasks to any of the cores or set of cores. Besides, the ability to use SPMD on virtually any processor or any operating system with any C compiler makes it very flexible.

MATERIALS AND METHODS

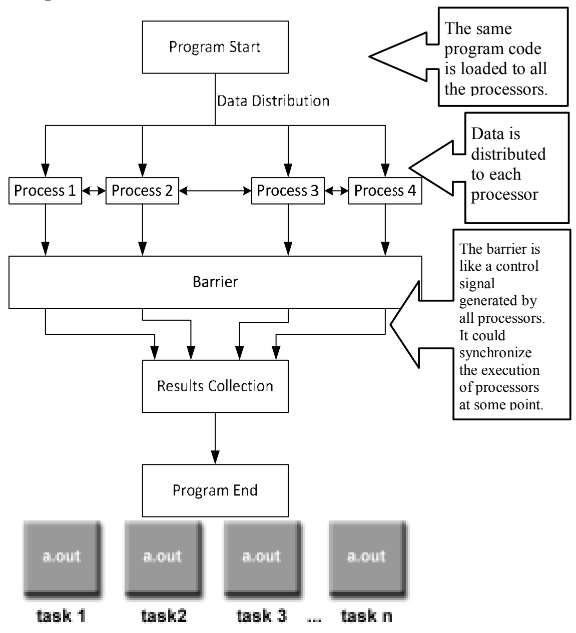
In the past all applications were applied on single core processor with better performance, but also comparison of energy and not suitable scaling. With the recent Era in multi-core helps to increases the scalability and energy management of the processor. This motivates to propose the work in partition which allocates task on multi-core platform in an away that overall amounts of blocking times of tasks are decreased. The parallel task scheduling procedure is built to achieve efficient output and the better resource sharing of the parallel tasks in the multicore processor. The single core processor parallel synchronization procedures have been extended to support inters processor synchronization among tasks. However, under task scheduling methods, the single core processor synchronization procedures cannot be reused without modification.

Instead, parallel synchronization for task and scheduling procedures have to develop to support resource sharing under task scheduling methods, to get desired efficient output from parallel task scheduling and resource sharing with the minimization of execution time and power consumption.

- ➢ Scalability, Portability and Predictability.
- ➢ Minimization of the execution time and energy consumption of task.
- ➢ Increasing the Scalability in multi-core processor.

- Single Program Multiple Data (SPMD)
- SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- SINGLE PROGRAM: All tasks execute their copy of the same program simultaneously. This program can be threads, message passing, data parallel or hybrid.
- MULTIPLE DATA: All tasks may use different data
- SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program - perhaps only a portion of it.
- The SPMD model, using message passing or hybrid programming, is probably the most commonly used parallel programming model for multi-node clusters.SPMD mode is a method of parallel computing, its processors run the same program, but execute different data.

- SPMD could get better computing performance through increasing the number of processors/cores


Program Start
Data Distribution
Process 1
Process 2
Process 3
Process 4
Barrier
Results Collection
Program End
The same program code is loaded to all the processors.
Data is distributed to each processor
The barrier is like a control signal generated by all processors. It could synchronize the execution of processors at some point.
a.out
a.out
a.out
a.out
task 1
task2
task 3
...
task n

Advantages of SPMD

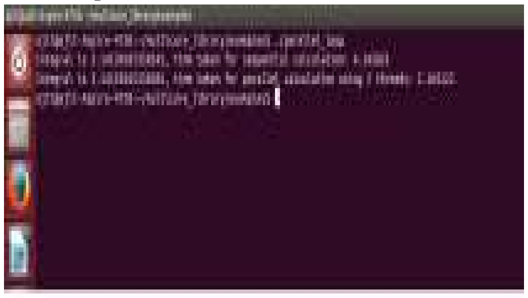
- Locality. Data locality is essential to achieving good performance on large-scale machines, where communication across the network is very expensive.
- Structured Parallelism. The set of threads is fixed throughout computation. It is easier for compilers to reason about SPMD code, resulting in more efficient program analyses than in other models.
- Simple runtime implementation. SPMD belongs to MIMD, it has a local view of execution and parallelism is exposed directly to the user, compilers and runtime systems require less effort to implement than many other MIMD models.

Disadvantages of SPMD

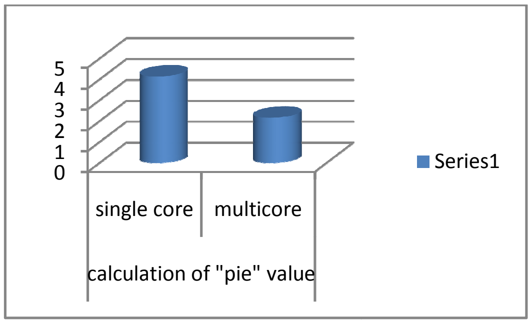
- SPMD is a flat model, which makes it difficult to write hierarchical code, such as divide-and-conquer algorithms, as well as programs optimized for hierarchical machines.
- The second disadvantage may be that it seems hard to get the desired speedup using SPMD.

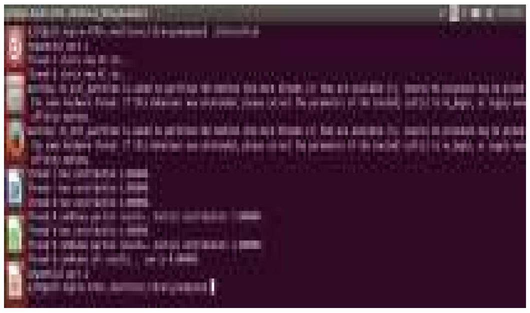
RESULTS AND RESULT ANALYSIS

1. Obtained result of the program calculating the value of "pi" using both the single and multi-core processor



```
[illegible]
```





```
[illegible]
```

The results from this project work show that the Multicore programming model provides a simpler, effective and scalable

Way to perform any applications on multi-core processors. With the concurrent function of multi-core programming model, the programmer can execute a simple and standard applications algorithm

Concurrently on multi-core processors in much less time than that of standard parallel Open-MP. This multi-core programming model will be further worked out for the introduction of some more parallel and concurrent functions and Synchronizing tools.

Case study: Inner product

Testing the applicability of the created **mclib** interface to scientific computing problems requires a practical use case scenario. Its solution has to be sufficiently computationally intensive to require parallelization with the need of synchronization at multiple points. This chapter describes such a use case, giving a step-by-step derivation of the computational Inner Product Computation.

Algorithm for Inner Product Computation

A simple example of a **mclib** algorithm is the following computation of the Inner product α of two vectors

x = (x0, . . . , xn−1)T and **y** = (y0, . . . , yn−1)T,

$$\alpha = \sum_{i=0}^{n-1} x_i y_i$$

To run the ip program on at most three processors:

```
$ mcrun -npes 3 ip
```

How many processors do you want to use?

```
2
```

Please enter n:

```
100
```

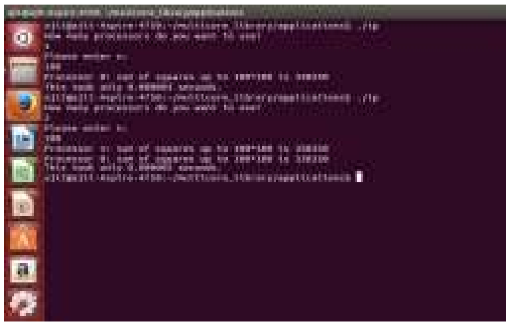
Processor 0: sum of squares up to 100*100 is 338350

This took only 0.000005 seconds.

Processor 1: sum of squares up to 100*100 is 338350

The program or the user can decide to use less than three processors; here two processors are actually used.

Note that the output of the different processors is multiplexed and hence can be garbled.



```
[illegible]
```

ACKNOWLEDGEMENT

The authors are thankful to the Principal, Poojya Doddappa Appa College Of engineering, Gulbarga, Karnataka, India for providing necessary facilities.

REFERENCES

F.Nemati and T. Nolte, "Resource sharing among prioritized real-time applications on multiprocessors". Technical report April, 2012.

M. Behnam, I. Shin, T. Nolte, and M. Nolin. SIRAP, "A synchronization protocol for hierarchical resource sharing in real-time open systems". In Proceedings of 7th ACM & IEEE International conference on Embedded software (EMSOFT'07), pages 279–288, 2007.

Rajagopalan, B .T .Lewis, and T .A. Anderson, "Thread scheduling for multicore plarform" 2007.

Y .Liu X . Zhang , H. Li, and D.Qian, "Allocating tasks in multi-core 2007.

K . Lakshmanam, D.de Niz , and R Raj Kumar, "Co-ordinated task scheduling, allocation and synchronization on multiprocessor" 2009.

F. Nemati, T. Nolte, and M. Behnam, "Partitioning real-time systems on multiprocessors with shared resources", In Proceedings of 14th International Conference on Principles of Distributed Systems (OPODIS'10), pages 253–269, 2010.