ICISCA 2015 International Conference on Information, System and Convergence Applications
June 24-27, 2015 in Kuala Lumpur, Malaysia

An Effective Approach for Parallel Processing with Multiple Microcontrollers

Gayeon Kim , Abdul Rahim Mohamed Ariffin, Scott Uk-Jin Lee
Gayeon Kim – Dept. Computer Science and Engineering, Hanyang University, South Korea
ruredi@hanyang.ac.kr
Abdul Rahim Mohamed Ariffin – Dept. Computer Science and Engineering, Hanyang University, South Korea
rahim750413@hanyang.ac.kr
Scott Uk-Jin Lee – Dept. Computer Science and Engineering, Hanyang University, South Korea
scottlee@hanyang.ac.kr

Abstract— **Multithreading is a common technique used to develop a system that processes very large data while producing fast execution and maintaining the efficiency of the program. However, non-determinism aspects of multithreaded program are always ignored due to the low impact on the system. Thus, there are various arguments to be discussed in determining non-determinism as one of the major aspects for developing a multithreaded program. In this paper, we propose a new effective approach for parallel processing with multiple microcontrollers.**

Keywords-component; Multithreading, Non-determinism, Parallel Processing, Microcontroller

INTRODUCTION

Parallel computing is essential for software development. The importance of parallelism is emphasized more than ever due to the rise of multicore hardware and high demand of computation for scientific computing, video and image processing, and big-data analytics. It is also expected to be increased for demand of embedded equipment with the growth of Information and Communication Technology (ICT) industry. Multithreading is one of the mainstream technologies in parallel programming. It is widely used in hardware, operating systems, libraries, and programming languages [1, 2]. However, it remains challenge to implement multithreaded programs. The main reason is that multithreaded programs are non-deterministic. In the sequential programs, the same inputs bring about the same results. In the multithreaded programs, on the other hand, we are not able to predict results when executing the same program with the same inputs. Even if the same multithread program is executed, we have to consider all different interleavings to recognize all possible results [3, 4].

In this paper, we present known solution for non-deterministic solution, deterministic multithreading (DMT) [3, 5, 6] and stable multithreading (Stable MT) [1], and limitation of the solutions. For the overcoming the limitation we propose an effective approach for parallel processing with multiple microcontrollers. Through this approach, non-deterministic problem can be reduced when multiple processing is used instead of multithreading. The performance of the system is also satisfying the concept of distributed computing.

The rest of this paper will be organized in the following manner. Section 2 provides the descriptions of the main problem of multithreading. In section 3, non-determinism as well as deterministic multithreading, stable multithreading along with their related techniques and limitations are described in detail. Then, we propose multiple processing in parallel with connected mircocontrollers in section 4. Finally in section 5, we conclude the paper and present possible future works.

DIFFICULTIES OF MULTIHTREADING

Although developments of concurrent programs are continuously increasing to satisfy the high demand, it is still difficult to implement, test, analyze, and validate them when compared to sequential programs with similar complexity. Multithreading, which has multiple threads running in a process, is the most commonly used type of parallel programs. However, there still are various problems to exploit multithreading in practice.

The major problem of multithreaded programs is non-determinism. It is the behavior of a typical implementation of multithreading where the same output is not guaranteed when the same input is provided. In such situation, it is almost impossible to find bugs with the traditional methods used in sequential programs. In addition, multithreaded programs can cause concurrency errors, such as deadlocks and race conditions due to the non-determinism [7, 8]. In a multithreading environment, sequentially running threads are called interleavings which actually are processes that execute threads in a very short time rather than running then parallel. Hence, with interleavings, threads seem as if it is running in parallel. The sequence of executing threads is determined by various aspects such as priority, request order, and optimization. For instance, optimizations of a compiler may cause a thread to be executed in the order that is not intended by the programmer.

In order to avoid previously mentioned side effect of non-determinism, many programmers apply mutual exclusive locks, semaphores, and monitors. Applying

such techniques involves a very complicated and tedious tasks where there are likely chances of applying these techniques incorrectly. In addition, even a very simple part of a program is difficult to implement in multithreading [4]. Non-deterministic can cause problems like deadlocks or race conditions. Therefore, multithreaded programs have to be implemented very carefully. It is unstable to use common libraries and design patterns when implementing multithreaded programs because they are developed without considering the possibility of non-deterministic problems.

NON-DETERMINISTIC, DETERMINISTIC AND STABLE MULTITHREADING

There are several researchers who have proposed different solutions for non-deterministic problems. Among these proposed solutions, only a few suggests deterministic multithreaded systems to prevent unintended results [3, 5, 6]. Previously, proposed methods devised deterministic multithreading systems to assign each input to a schedule. There are also different approaches suggesting to reduce number of schedules instead of constructing deterministic algorithm [1]. Stable multithreading is based on the idea to reduce possible interleavings by decreasing the total number of schedules. In this section, we provide a comparison between non-deterministic multithreading, deterministic multithreading, and stable multithreading as follows:

L. Non-Deterministic Multithreading

Common multithreaded programs are non-deterministic. Each thread creates interleavings and they are executed at very short period of time with context switch. The sequence of execution for the interleavings changes every time and it also depends on the situation. Normally it is impossible to predict the result of executions. The non-determinism leads to some common issues in parallel processing such as deadlocks or race conditions.

M. Deterministic Multithreading

In deterministic multithreading, threads run the same number of thread interleavings. Consequently, these systems enable multithreaded programs to produce the same results for the same input [3, 5, 6]. There are variety of systems which implement deterministic multithreading. The concepts adopted in these systems are very similar to controlling access to the shared memory where the threads are synchronized at the end of their executions. Hence, the output of the program execution with the same input can always be predicted. However, these systems still have limitations such as large overhead and not providing determinism in particular environments.

N. Stable Multithreading

The main difficulty of multithreading is that multithreaded programs have too many schedules [1]. Even with deterministic multithreading approach, threads can be mapped in a schedule which is prone to produce bugs. Stable multithreading finds unnecessary schedules and excludes them. By reducing the number of possible cases to schedule interleavings, a multithreaded program can obtain better robustness and reliability. However, this system is still immature to be used in application level due to the lack of thorough code analysis and testing [1].

Table 1 describes the features, limitations and goals of deterministic multithreading and stable multithreading. Through the table provided, the differences of each multithreading methods are described.

TABLE I. Features, Limitations, and Goals of Deterministic and Stable Multithreading

Multithreading Methods	Feature	Limitation	Goal
Deterministic Multithreading	Schedule interleavings in deterministic order	Does not work in specific situation or has large overhead	Always get the same output with the same input
Stable Multithreading	Remove unnecessary schedules	Not proper for application yet	Prevent to map buggy schedules

PARALLEL PROCESSES AND MICROCONTROLLERS

Parallel processing provides the characteristics of simultaneous processes producing the same shared inputs is a traditional feature of parallel computing. In order to produce a fast and accurate output while maintaining the performance of the parallelism have always been a major concern to developers.

O. Parallel Computing

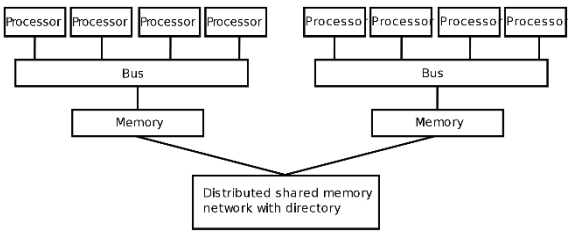

Figure 1 shows the overview of parallel computing architecture

Parallel computing is a form of computation in which calculations are carried out simultaneously [9]. The inputs for the computations are derived from the distributed shared memory where the buses are authorized to send the inputs towards multiple processors as shown in Figure 1. Parallelism has been traditionally used to develop high-performance computing. For such purposes, various forms of parallel computing such as bit-level, instruction-level, data, and task parallelisms have been developed [10]. However, in recent years, parallel computing has been used for different purposes such as concurrent processes using multi-core processor and parallel processes in microcontrollers.

P. Microcontrollers

Microcontrollers have become the backbone of many appliances. They are widely used in embedded systems

such as robots, cars, peripherals and other appliances. New development of microcontrollers occurs at very fast pace where even multi-core microcontrollers become available in recent years [11]. This has created a more visible availability in solving multithreading non-determinism problem. Solving non-determinism constraint in multithreading is very important to provide a multithreaded system or application with better robustness, maintainability, and testability.

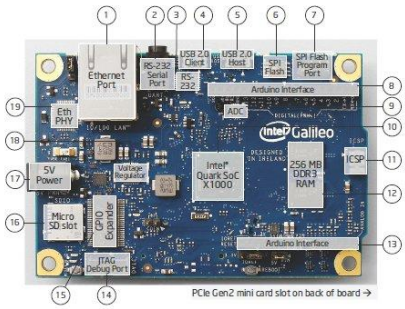

Figure 2. Intel Galileo Microcontroller Unit Chip

Q. Parallel Processing with Multiple Microcontrollers Unit (MCU) System

Microcontrollers in embedded system controls external hardware operations and also provide cost efficiency in terms of having small number of program tasks stored in permanent memory with lowest possible cost. Thus, multiple controllers running concurrent processes provide a solution to non-deterministic aspects in multithreading by enhancing the performance of each operations run by MCUs. Conceptually, multithreading is equivalent to a context switch at the operating system level. The difference is that a multithreaded CPU can do a thread switch in one CPU cycle whereas a normal context switch requires hundreds or thousands of CPU cycles. This is achieved by replicating the state hardware (such as the register file and program counter) for each active thread. A further enhancement is simultaneous multithreading which allows superscalar CPUs to execute instructions from different programs/threads simultaneously in the same cycle [12]. However, similar to other multithreading techniques, the non-deterministic aspect of multithreading is still being ignored in developing a multithreaded program. Thus, in this paper, we propose a new solution to handle non-deterministic aspects of multithreading by applying parallelism method such as parallel computing in microcontroller system unit. In terms of performance, microcontroller works much faster than a single computer handling multithreading processes because memory in a microcontroller is much smaller in size. Hence, the tasks operated by the microcontroller will produces the output faster in term of time. Recently, the prices of microcontrollers such as Arduino, Intel Galileo (Figure 2) and others have reduced dramatically. By just having this factors, programmers with less knowledge on multithreading will have the encouragement and aspiration to do simultaneous or parallel operation similar to multithreading through applying parallelism method with microcontroller system units. Previously, the proposed approach were controversial due to the high cost of microcontrollers. However, a new approach for programming or developing a concurrent and parallel applications can now be introduced due to the decrease of price for microcontrollers. According to the 2015 McClean Report [13], the estimation of all types of microcontrollers (MCU) with 8-bit, 16-bit, and 32-bit designs used in new systems being attached to the Internet of Things in 2019 is expected to be about 1.4 billion. This is a dramatic increase for the demands of MCUs when compared to 306 million in 2014. It proves that microcontrollers are currently on high demands for the most current system and technology especially in the field of embedded system where multiple microcontrollers are required.

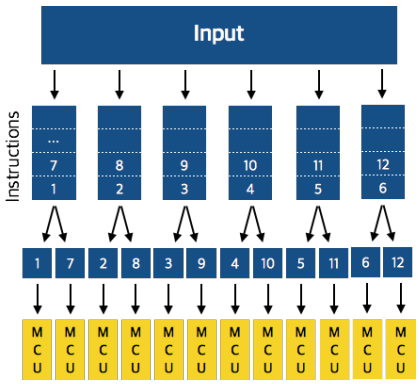

Figure 3. MCUs executes single task per one thread

Through our approach, we provide an approach to solve non-deterministic issues in multithreading by applying multiple controllers. Although there has been similar research in recent years, the proposed solutions from related research are still unable to provide direct solution to non-determinism. Figure 3 illustrates that the input from the distributed shared memory are divided and processed by multiple MCUs to produce the output where each MCU will executes a single thread. The goal is to provide better performance with multiple controllers instead of traditional multithreading methods. Through parallel processing with multiple controllers, the system is able to prevent deadlock occurrence by specifically running a single thread in one microcontroller. As discussed earlier, each microcontroller will run a single thread and produces output in a short amount of time since microcontroller have a small size memory spaces.

R. Limitations and Discussion

Although the proposed approach provides sufficient solution, there still are some unresolved issues that may occur in the future and become a concern for the application of MCUs in parallel computing system. One of the issues is the maintainability caused by using large number of MCUs for parallel processing. Maintaining each of the MCUs will be very tedious. Another issues is rather physical where programmers require to purchase multiple MCUs and stack it up on top of their work

station. When a large number of MCUs is required, it can lead to the waste of physical spaces as well as the effort required to configure each MCUs. However, this problem will be resolved as the size and price of a microcontroller are reducing and performance of microcontrollers are increasing continuously.

CONCLUSION

In this research, we have designed conceptual approach for replacing multithread with multiple processing by connected multiple microcontroller. We are planning to improvise this approach in future research by deriving the experiments for better performance through implementation and quantitative analysis. We will also compare our approach with multithreading in the economical perspective through the experiments. Surely, providing reliable, low-priced, and easy to implement multithread programming methodology is the main objective. However, there is no known solution which satisfies all three conditions. Parallel processing with interconnected multiple microcontrollers not only satisfies the conditions for taking parallel processes instead of multithread, but also provides sufficient performance by adapting the concept of distributed computing. Moreover, it costs much less than using multithreaded program. Therefore, we believe the proposed approach can be a reasonable alternative until the proper development of competent solution for multithreading covering non-deterministic problem.

ACKNOWLEDGMENT

This work was supported by the ICT R&D program of MSIP/IITP.[12221-14-1005, Software Platform for ICT Equipment].

REFERENCES

Junfeng Yang, Heming Cui, Jingyue Wu, Yang Tang, Gang Hu, "Determinism is not enough : making parallel programs reliable with stable multithreading", Columbia University (2013)

Heming Cui, Jingyue Wu, Chia-che Tsai, Junfeng Yang, "Stable Deterministic Multithreading through Schedule Memoization", Computer Science Department, Columbia University, 2010

Tongping Liu, Charlie Curtsinger, Emery D. Berger, "DThreads : Efficient and Deterministic Multithreading", SOSP '11, October

Edward A. Lee, "The problem with threads", Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report, January 10th, 2006

Nissim francez, C. A. R. Hoare , et. al, "Semantic of Nondeterminism, Concurrency, and Communication", Journal of Computer and System Science (1979)

Emery D. Berger, Ting Yang, Tongping Liu, Gene Novark, "Grace : safe multithreaded programing for C/C++", OOPSLA 2009

Marek Olszewski, Jason Ansel, Saman Amarasinghe, "Kendo : efficient deterministic multithreading in software", ASPLOS 2009

Robert H.B. Netzer, Barton P. Miller, "What is race conditions? : Some issues and formalizations" , ACM 1992

http://en.wikipedia.org/wiki/Parallel_computing (visited 04/2015)

W. Pornsoongsong, P. Chongstitvatana, "A Parallel Compiler for Multi-core Microcontrollers", IEEE 2012

Derek G. Murray, Steven Hand, "Non-deterministic parallelism considered useful", University of Cambridge Computer Laboratory, 2011

http://en.wikipedia.org/wiki/Microarchitecture (visited 052015)

Ic Insights, "Microcontroller Sales Regain Momentum After Slump", February 2015.