Generic Modeling for Pthreads with Cilk

Abdul Rahim Mohamed Ariffin, Scott Uk-Jin Lee*
Department of Computer Science and Engineering, Hanyang University, South Korea
rahim750413@hanyang.ac.kr, scottlee@hanyang.ac.kr*

Abstract—**Implementation of pthreads in embedded and other hardware-related systems are common, especially when dealing with systems that require parallel process executions and reduced memory consumption. However, implementation of pthreads are still unable to solve issues regarding arbitrary output due to the occurrence of deadlock and data races. In this paper, we study existing benchmarks in order to overcome the common issues which exists in pthreads. We investigate existing parallel programming framework and suggest which frameworks are best to be implemented with pthreads. We then give suggestions on new benchmarks for pthreads in embedded systems.**

Keywords-pthreads, parallel programming, embedded systems, threads

I. INTRODUCTION

Pthreads or POSIX threads programming are one of the most essential programming methods to compute multiple processes while accessing shared memory in a system. Although, it allows multiple threads to execute asynchronously in a period of time, the major concerns in pthreads programming are the existence of data race and deadlock issues within threads executions. Hence, a number of researches on issues that exploit the properties of pthreads and introduces different programming methods, such as dthreads that is a deterministic executions of threads. Pthreads are most commonly used in embedded systems and general purpose systems, due to pthreads being a lightweight-based programming which supports parallel executions and multiple accesses in shared memory. Therefore, there exist a number of studies on the benchmarks of pthreads in embedded systems and hardware-related systems. Existing benchmarks are unable to support unexpected deadlock and data race occurrences, which makes other parallel programming framework, such as OpenMP and dthreads, a more preferred solution in avoiding these issues than pthreads. In this paper, we will evaluate the properties of pthreads and study existing benchmarks on the performance of pthreads in embedded systems. Then, we provide new benchmarks for pthreads in order to improve its performance and support for unexpected issue occurrences. We will also discuss related studies for pthreads and our motivation for proposing new benchmark for pthreads in embedded system.

This paper is organized as follows: In Section 2, we provide a brief introduction of pthreads. Section 3, discusses the general approaches with existing benchmarks and we suggest new benchmarking criteria. In Section 4 we discuss the existing analysis study on pthreads in comparison with other parallel programming frameworks and methods. We conclude this paper by discussing future work to find pthreads' correctness properties for performing verifications in Section 5.

II. APPLICATIONS ON PTHREADS

Pthreads and many other parallel programming frameworks such as MPI, OpenMP and dthreads support multiple executions of processes for accessing shared memory. Through these criteria, many embedded systems implement these frameworks. Parallel programming, such as pthreads, are lightweight and also support its own interface library, which allows complex embedded systems that require simple programming for faster performance and less shared memory space consumption. Pthreads is a parallel execution model which allows a program to control multiple overlapping threads in a period of time [1]. Pthreads application program inference (API) control the creation and flows of threads in parallel system, also for thread synchronization.

OpenMP is a shared-memory multiprocessing API for easy development of shared memory in parallel programs [2]. It is a suitable framework for embedded systems due to its ease of use with a low-level complexity to code. In addition, OpenMP provides a set of compiler directives, library routines and environment variables. With these properties, OpenMP is able to easily create, synchronize and manage threads in a shared memory embedded system.

Message passing interface (MPI) is an open source library used in most programming such as C and C++ functions [3]. It acts as a collection of processes that can exchange messages between channels. MPI provides message passing model, especially for parallel or distributed systems. MPI supports the sending and receiving operations between threads and also allows dynamic creation of processes through one of its version model called MPI-1 and MPI-2.

Dthreads is a programming model that provides deterministic execution of processes in multithreaded systems [4]. Dthreads determines the interleaving of each threads, after selecting prioritized threads and calculating the expected execution time. It provides

different addresses for the threads and provides token for threads to access shared memory.

III. New Benchmarks

A. General Approach

Existing benchmark for pthreads currently are still unable to fully utilize the properties of pthreads. There exist a number of performance evaluation study for pthreads which only focuses on the energy consumption, workload balancing and speedup on computing complex algorithms. Therefore, to overcome these issues we use Cilk which is a general-purpose programming languages designed for multithreaded parallel computing [5]. Cilk is an extension of constructs based on C and C++ programming languages to express parallel programming, such as loops. Cilk support data and task parallelism, allowing it to execute the three main components in pthreads API; mutex variables, condition variables and control management.

```
pthread_mutex_lock (&mutexsum);
dotstr.sum += mysum;
pthread_mutex_unlock (&mutexsum);

pthread_exit((void*) 0);
```

Figure 1. Pthread's mutex lock and unlock command

Mutex variables, which is a locking mechanism, is used to prevent simultaneous access to multiple threads in a shared memory, i.e. executing the same data at the same time.

```
if (count == COUNT_LIMIT) {
  pthread_cond_signal(&count_threshold_cv);
  printf("inc_count(): thread %ld, count = %d
         my_id, count);
  }
```

Figure 2. Pthread's condition variables for signaling threads

Condition variables is a mechanism to select which thread is more critical than the other threads that is executed in parallel manner. Condition variables also allow for synchronization of threads through the actual value of data. Without condition variables, it is very resource consuming to allow threads to exit the execution process, since threads are always being prompted at every phase in the execution process. By using Cilk as benchmark, it does not only increase the performance, but increases/decreases energy consumption as well. Other than that, execution period (speedup) are measured and other possible pthreads attributes can be used for testing. Other attributes including mutex locking, condition variable checking can also be measured through Cilk.

B. Modelling with Cilk

There are 3 basic components of Cilk code, which are cilk, spawn and sync. Each component described as cilk; a function for running processes in parallel, spawn; procedure representation of parent and child procedure which is able to perform parallel execution in a program, while sync; a control mechanism which lock the program until all the procedure has been met and satisfied. This method is similar to pthreads condition variables which require some conditions to be met before executing the processes. Figure 3 shows code snippet in Cilk for parallel execution of a Fibonacci program.

```
cilk int fib (int n) {
    if (n<2) return (n);
    else {
        int x,y;
        x = spawn fib(n-1);
        y = spawn fib(n-2);
        sync;
        return (x+y);
    }
}
```

Figure 3. Cilk code snippet in Fibonacci program

To overcome the issues which still exists in pthreads, verifications are required to be conducted on them. SPIN [6], which is a model checking tool for multithreaded applications may be the best option avalaible to verify properties of pthreads. Figure 4 presents an example of one of the code snippet that are available in SPIN using PROMELA language to verify a mutex lock process.

```
void
lock(pthread_t Pid)
{
busywait:
    x = Pid;
    if (y != 0 && !pthread_equal(Pid, y))
        goto busywait;

    z = Pid;
    if (!pthread_equal(x, Pid))
        goto busywait;

    y = Pid;
    if (!pthread_equal(z, Pid))
        goto busywait;
}
```

Figure 4. Mutex lock verification in SPIN

IV. Discussions

In [7], pthreads outperforms OpenMP and MPI in parallelism control and workload distribution for embedded and general purpose systems. However, pthreads requires additional programming effort, since pthreads programming are more direct and hardcoded. The authors claim that implementing pthreads for embedded systems would be the best suite compared to other parallel programming frameworks. When performing complex algorithm in parallel programming, pthreads provides the best speedup performance outcome when compared to OpenMP and Microsoft Parallel Patterns APIs [8]. However, the speedup performance of the increased matrix dimensions' result in pthreads and OpenMP having similar percentages and similar execution time.

[9] proposed the modeling of pthreads and dthreads in order to perform detection on data race and deadlock occurrences. Authors claim that pthreads unable to eliminate undesirable deadlock and data races from occurring in a parallel system. While dthreads easily eliminate the existing issues in pthreads such as data race occurences, due to dthreads execution behavior is deterministic. However, implementing dthreads in embedded system will not be suitable, since its behavior of threads executions are deterministic.

There exist previous studies on benchmark for pthreads such as in [10]. However, it is still not enough to support pthreads which are commonly implemented in embedded systems. Therefore, we provide an evaluation study to use a new benchmark for pthreads using Cilk so that pthreads performance in [11] and [12] can be improved when compared with existing parallel programming framework and models.

V. CONCLUSION

This paper presented current issues in pthreads implemented in embedded systems and provides suggestion for new possible benchmarks for pthreads. We provide a suggestion for the new benchmarks for pthreads so that it will be able to outperform existing parallel or distributed programming frameworks. Cilk is able to provide lightweight and ease of use in developing parallel programs and also support better synchronization and load balancing in accessing shared memory. As our future work, we will expand our studies on pthreads' correctness properties for performing verifications. We will apply SPIN/Promela for exploring possible errors since SPIN provides good quality of counterexamples.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Nichols, Bradford, Dick Buttlar, and Jacqueline Farrell. Pthreads programming: A POSIX standard for better multiprocessing. "O'Reilly Media, Inc.".

[2] Dagum, Leonardo, Rameshm Enon, "OpenMP: an industry standard API for shared-memory programming," Computational Science & Engineering, IEEE, 1998 vol. 5, no. 1, pp. 46-55.

[3] A. Stamatakis, M. Ott, "Exploiting fine-grained parallelism in the phylogenetic likelihood function with MPI, Pthreads, and OpenMP: a performance study," In Pattern Recognition in Bioinformatics,2016, pp. 424-435.

[4] Liu, Tongping, Charlie Curtsinger, Emery D. Berger, "Dthreads: efficient deterministic multithreading," In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011, pp. 327-336.

[5] https://en.wikipedia.org/wiki/Cilk (last visited: 5/22/2016)

[6] Zaks A, Joshi R, "Verifying multi-threaded c programs with SPIN," Proceedings of 15th SPIN, USA, 2008, pp. 325–42.

[7] A. F. Lorenzon, M. C. Cera, A. C. S. Beck, "Performance and Energy Evaluation of Different Multi-Threading Interfaces in Embedded and General Purpose Systems, " Journal of Signal Processing Systems, 2015, vol. 80, no. 3, pp. 295-307.

[8] Sharma, Mukul, Pradeep Soni, "Comparative Study of Parallel Programming Models to Compute Complex Algorithm," International Journal of Computer Applications, 2014, vol. 96, no. 19, pp. 9-12.

[9] Y. Fei, H. Zhu, X. Wu, H. Fang, "Comparative Modeling and Verification of Pthreads and Dthreads," IEEE 17th International Symposium on High Assurance Systems Engineering, 2016, pp. 132-140.

[10] B. R. de Supinski, J. May, "Benchmarking Pthreads Performance," Proc. International Conference on Parallel and Distributed Processing Techniques, Las Vegas, Nevada, United States, 1999.

[11] Kindlmann, Gordon, Charisee Chiw, Nicholas Seltzer, Lamont Samuels, John Reppy, "Diderot: a Domain-Specific Language for Portable Parallel Scientific Visualization and Image Analysis," IEEE Transactions on Visualization and Computer Graphics, 2016, vol. 22, no. 1, pp. 867-876.

[12] S. J. Kang, S. Y. Lee, K. M. Lee, "Performance comparison of OpenMP, MPI, and mapreduce in practical problems," Advances in Multimedia, 2015, vol. 2015, pp. 7.