Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/108873, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Comparative Study of Multithreading APIs for Software ofICT Equipment

Isma Farah Siddiqui, Asad Abbas, Abdul Rahim Mohamed Ariffin and Scott Uk-Jin Lee*

Department of Computer Science and Engineering, Hanyang University, South Korea; isma2012@hanyang.ac.kr, asadabbas@hanyang.ac.kr, rahim750413@hanyang.ac.kr, scottlee@hanyang.ac.kr

Abstract

In the infrastructure of Information and Communication Technology (ICT), the design and architecture of software deployed over equipment plays a vital role in system's concurrent efficiency. The high performance computing nexus with fine-grain parallel processing environment calls for optimized and effective multithreading strategies for ICT's software implementations. This paper briefly identifies different state-of-art multithreading APIs used in multi-core coherent processors. The paper investigates various APIs based on main categorization of operating system level and lightweight thread level. Operating system level multithreading relies on operating system kernel architecture asnumerous threads or tasks achieve concurrency provided by the underlying hardware, thus considered effective for high core programming. On the other hand, various application level lightweight thread models are being offers with lighter mechanism for high parallelism and massive concurrency. Lightweight models are optimized to combine low-latency thread and task scheduling with optimized functionality fordata-movement. This comparative study aims to demonstrate major working models and principal differences among well-established libraries in each category. This paper projects an early investigation for the identification of most efficient thread library for ICT equipment.

Keywords: ICT Software, Parallelism, Multithreading Models

1. Introduction

Software applications require the parallel scalability of hardware and software for high performance processing. Multiprocessor architecture enhance parallel processing of multi-threads. In multi-threaded applications, a noticeable problem is data race. Mutex-lockapproach reduces the data race problem by giving the permission with synchronous access onshared resources. Due the mutex-lock approach deadlock may occur when one tread holds the mutex-lock for one process and at the same time other processes are continuously waiting for that process. In multithreading approach, the main challenge is to access the shared resources without data race and deadlock conditions which is resolved viably through different implementations of multithreading libraries.

Multi-threading in embedded software applications are on two levels 1): Operating System (OS) level and 2): user level (light weight).OS level multi-threading and resource scheduling is implemented by fault tolerance and patching integrating applications. Responsibility of operating system is to handle the hardware errors and ensure error prone user level application support[1]. However, user level thread is initiated by application level that share storage and control registers with more than one user level threads. More than one registers can be shared and visible to first user-level, second user-level or more with given instruction from OS-level threads for synchronization processing[2].

Embedded software applications based on ICT equipment are increasingly used in business and governance because of high speed and less resource consumptions at core level of multi-threaded system[3]. Most of organizations pay more attention on ICT hardware rather than software. However, ICT software have strong impact on the usage of resource and energy of hardware. Developing the efficient multi-threaded software

*Author for correspondence

indirectly will consume less equipment resources[4]. ICT based operating systems classified as[5]:

- Multi-users: More than one users are allowed to run the programs simultaneously on operating system. In big organizations hundred or even thousands of users are allowed to run the same program on the same operating system.
- Multi-processing: Permits to run a program on multiple CPUs.
- Multi-tasking: Operating system that allow to operate multiple programs simultaneously.
- Multi-threading: Multiple parts of single program run same time.

ICT operating systems are configured to operate and start-up in different ways. However, configurations are dependent of end user and business requirements. In this paper we have studied current multithreading APIs that are providing multithread support for solving known problems of parallel processing environments. The categorization of these APIs is based on their application at operating system or user level. The study compares and highlights major differences among these APIs. The purpose of this study is to collect and compare available state-of-art multithreading APIs for their adaptation in ICT infrastructure particularly software.

The rest of the paper is organized as follows: Section 2 gives related works of multithreading and their optimization issues in ICT industry. Section 3 shows categorization of several available multithreading APIs along with highlighted differences. Finally, section 4 gives conclusion and future works.

2. Related Work

Multi-threading is important in embedded software to get predictability of execution time, real-time work deadlines and less consumption of energy resources. In current industry, time and energy is more important to get speedy execution in embedded software. To improve the efficiency of multi-threading, embedded hardware processors XMOS XS1-L series has been used for ICT which is programmed on C language. Author presented energy model of multi-threading for these processor to reduce the energy consumption, increase the execution speed and deterministic system[6].

Reduced energy consumption in Internet of Things (IoT) and ICT equipment by using multi-threading approach in embedded software got interest in current research. Multi-threaded embedded software ultimately increases the efficiency for both hardware and software. Hardware become more efficient to reduce the power such as mobile phones must be energy efficient. This only can be achieved with reduced energy consumption consumed by software code during execution time. Multi-threading is efficient method to reduce the consumption of resources and ultimately effects overall energy consumption[7].

Non-determinism of threads and processes are still ongoing problem and also not easy to handle where large number of threads are waiting for shared common resources. To remove data race and deadlock from concurrent system Pthread and Dthread has been used. Free deadlock system and accurate output from each thread is possible by synchronized execution of threads. In a comparative study between Pthread and Dthread on same concurrent system to remove data race and deadlock, authors in [9] have found Dthread as more efficient and effective to get synchronized execution of threads and free deadlock system[8].

3. State-of-Art Multithreading APIs

Based on execution environment, the multi-threading approach is categorized ini) OS (kernel) level multi-threading and, ii) user level (light weight) multi-threading. Kernel structure, scheduler of processes and programing model are the characteristics of OS. Key points of OS multi-threading are process scheduling and efficient programing model[9]. Moreover, kernel structure manages the threads i.e. resource allocations, access to shared resources. On the other hand, end-user application manages the user level multithreading, which is developed according to application requirements. Management of user level thread scheduling and resource allocations are comparatively easy because additional resources of kernel are not required for user level threads and also no need to utilize all OS level threads[10].

3.1 OS Level Multi-Threading

OS level multi-threading is managed by two types of POSIX threads Pthread and Dthread.

3.1.1 Pthread

The Problems of data race and deadlock in parallel computing system can be overcome by POSIX thread API

library of C/C++ that consist on Pthreads. Pthread control the non-deterministic parallel thread execution in sense of scheduling and shared resource allocation. Concurrency problem may happen with Pthread by using IEEE POSIX 1003 standard programming interface which specified to achieve the portability of threaded system [9]. However, Pthread is used to achieve optimum performance on multiprocessor architecture and also do need intermediate copy of memory because within a single process threads share common address space[11].

3.1.2 Dthread

Dthread is extended POSIX compliant library for replacement of Pthreads in C/C++ applications for more deterministic multi-threaded (DTM) systems. Determinism is guaranteed by using Dthread even data race occurrence in multi-threaded program. OS events always generate the output in same sequence as input is given by using Dthreads. Various limitation is present in Dthread implementation such as: external determinism, unsupported programs, issues with memory consumption and consistency. However, the numerous advantages are by far better than Pthread has been investigated by many researchers. Dthread is portable and ensures internal determinism with no data race and deadlock. The internal design of Dthread also bypass false sharing which is another notorious problem of multithreading over multi processors[12].

Table 1 briefly highlights the differences between Pthread and Dthread libraries and clearly indicate that the Dthreads is more efficient, deterministic and safe than Pthreads for many applications.

3.2 User Level (Lightweight) Multi-threading

User level threads are created, synchronized and executed with transparency of operating system. Run time libraries are linked with software application and manage user level threads therefore, kernel level thread change is avoided. User level threads can be modified by the code without any changes in operating system kernel. These threads have limited access of shared resources such as processes within the specific application[13]. These libraries are above OS level threads but are executed with OD participation, using own programming models.

3.2.1 Qthread

Lightweight multi-threading is supported by Qthread that maximizes architecture portability and primitives of synchronization. API of Qthread consists on basic three components: lightweight command set, threaded loops and commands for awareness of resources limitation[14]. Qthread offer mechanism for high number of user-level thread handling with high synchronization using mutex.

3.2.2 MassiveThread

Multi parallel processing need tasking layer that fulfills performance scales, synchronization and communication of inter-node runtime system. MassiveThread is lightweight thread library and also compatible with Pthreads. MassiveThreads trigger user level I/O calls instead of OS level threads[15]. When a new user level thread is created, MassiveThread execute it immediately and thread that is already executed is moved into ready queue. MassiveThread use the hardware resources as worker concept such as CPU or core, environmental variables define the number of workers i.e. resources consumption[15].

3.2.3 Argobots

Argobots is another lightweight threads library that allows developers to create their own programing models with massive concurrency.Argobots library provides the complete access of supported resources to the processes i.e. OS level resources. The architecture consists of an execution model and a memory model, supporting two levels of parallelism using *execution streams* and *work units*. The working units of Argobotsare lightweight execution units such asuser-level threads or tasklets. The benefits of Argobots for high performance computing includes its high support with lowest-level constructs in hardware and OS. These includes efficient notification mechanisms,

Table 1. Comparison of Pthreads and Dthreads

APIs	Guarantee Determinism	Recompilation	False sharing	Eliminate false sharing	Fault Protection	Mechanism
Pthreads	×	✓	✓	×	×	Threads
Dthreads	✓	×	×	✓	✓	Processes

Table 2. Comparison of lightweight thread libraries

User-Level Threads	Unit of works	Levels	Thread support	Queue Based	Group Control	Tasklet
Qthreads	1	3	✓	Private	✓	×
MassiveThreads	1	2	✓	Private	✓	×
Argobots	2	2	✓	Public & Private	✓	✓

data movement engines, memory mapping, and efficient strategies for data placement[16].

Table 2 demonstrates main impacting differences among three discussed user level thread libraries. The use of these thread libraries are dependent on application multithreading requirements. These light-weight approaches seem better than POSIX threads for fine-grain task parallelism or highly nested par-allel structures.Most of these implementations are ableto provide lightweight execution model with low-latency thread and task scheduling along with meeting optimized data-movement.

4. Conclusion

The paper highlights early investigation about various multithreading libraries and categorized these based on interaction level within multi-core systems architecture. The OS level multithreading comprised of POSIX Pthread and its replacement in C/C++ application by Dthread is discussed. Dthread is found to be much efficient and fully deterministic approach while compromising with exter-nal-event determinism and high memory consumption. User level (lightweight) thread models are found to be more efficient with low-latency threads and task sched-uling. These approaches are found to be more feasible for fine-grained parallel codes and nested task parallel structures.

In the future, we plan to perform comparative verification for more properties required in an efficient multithreading approach. We will investigate other APIs and approaches to obtain most appropriate multithreading approach for ICT equipment.

5. Acknowledgement

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

6. References

1. Döbel B, Härtig H, Engel M. Operating system support for redundant multithreading. Proceedings of the tenth ACM International Conference on Embedded Software; 2012Oct. p. 83–92.
2. Grochowski E, Wang H, Shen JP, Wang PH, Collins JD, Held J, Kundu P, Leviathan R, Ngai T-F. Method and system to provide user-level multithreading. U.S. Patent Application 15/088,043; 2016 Mar 31.
3. Kundu P, Das D, Banerjee A. Plug-in instrumentation: A futuristic ICT approach for teaching the runtime behav-iour of software. UGC Sponsored Seminar on ICT in Higher Education: Opportunities and Challenges in the 21st Century Proceedings, SPS Education India Pvt. Ltd., Kolkata; 2012 Mar. p. 24–7.
4. Mahmoud SS, Ahmad I. A green model for sustainable software engineering. International Journal of Software Engineering and its Applications. 2013; 7(4):55–74.
5. Yull S, Lawson J. AQA AS GCE applied ICT single award (Paperback); 2005.
6. Kerrison S, Eder K. Modeling and visualizing net-worked multi-core embedded software energy consumption[Internet]. 2015. Available from: https://arxiv.org/abs/1509.02830.
7. Kerrison SP. Energy modelling of multi-threaded, multi-core software for embedded systems. Ph.D diss., University of Bristol; 2015.
8. Fei Y, Zhu H, Wu X, Fang H. Comparative modeling and verification of Pthreads and Dthreads. 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE);2016 Jan. p. 132–40.
9. Baccelli E, Hahm O, Gunes M, Wahlisch M, Schmidt TC. RIOT OS: Towards an OS for the Internet of Things. 2013 IEEE Conference onComputer Communications Workshops (INFOCOM WKSHPS); 2013 Apr. p. 79–80.
10. Danjean V, Namyst R, Russell RD. Linux kernel activations to support multithreading. 18th IASTED International Conference on Applied Informatics (AI 2000);2000.
11. Barney B. POSIX threads programming. National Laboratory. [Internet]. 2009. Available from:https://com-puting.llnl.gov/tutorials/Pthreads/.

12. Liu T, Curtsinger C, Berger ED. Dthreads: Efficient deterministic multithreading. Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles; 2011 Oct.p. 327–36.
13. Anderson TE, Bershad BN, Lazowska ED, Levy HM. Scheduler activations: Effective kernel support for the user-level management of parallelism. ACM Transactions on Computer Systems. 1992;10(1):53–79.
14. Wheeler KB, Murphy RC, Thain D. Qthreads: An API for programming with millions of lightweight threads. IEEE International Symposium on Parallel and Distributed Processing. IPDPS 2008; 2008 Apr. p. 1–8.
15. Nakashima J, Taura K. MassiveThreads: A thread library for high productivity languages. Concurrent Objects and Beyond, Springer Berlin Heidelberg; 2014.p. 222–38.
16. Seo S, Amer A, Balaji P, Beckman P, Bordage C, Bosilca G, Brooks A, Castelló A. Argobots: A Lightweight Low-level threading/tasking framework [Internet]. 2015. Available from: https://collab.cels.anl.gov/display/ARGOBOTSS/.
17. Castelló A, Pena, AJ, Seo S, Mayo R, Balaji P, Quintana-Orti ES. A review of lightweight thread approaches for high performance computing. IEEE Cluster 2016; 2016.