

Multithreading - An Efficient Technique for Enhancing Application Performance

Mrs.M.Shanthi¹, Dr.A.Anthony Irudhayaraj²

¹ Department of Master of Computer Applications , Velammal Engineering College, Chennai-66,Tamil nadu, India
sha_karnan@yahoo.com

² Department of Information Technology ,Arupadai Veedu Institute of Technology, ,Chennai, ,Tamilnadu, India.
anto_irud@hotmail.com

Abstract— Multithreading has been shown to be powerful approaches for boosting a system performance by taking advantage of parallelism in applications. Producer / Consumer, Web services, Banking and Image compression are good examples of applications that benefit from threading. Performance is increased because multiple threads running on multiple cores can process many requests simultaneously. This paper provides a summary of optimizing a serial designed application to multithreaded applications for Pentium dual core. The exercise uses the Java language as an example. This paper also discusses about the threading tool which is capable of converting a non threaded program into a threaded program to convert it into an efficient multithreaded application. It is observed that multithreading leads to tune the application performance considerably. To realize the advantages of model-driven development in designing parallel applications, a survey is conducted and it is observed that model driven development combined with multithreading can achieve better benefits .

Keywords: Parallelism, Performance , Java, Optimize, Multithreaded

I. INTRODUCTION

A thread is a point of execution within a process. So a multi-threaded process has multiple points of concurrent execution within the process. Multi-threading is useful in any situation where a single thread has to wait for a resource and we can run another thread in the meantime. This includes a thread waiting for an I/O request or database access while another thread continues with CPU work. Multi-threading is also useful if the individual threads can be farmed out to different CPUs (or cores) as they then run truly concurrently. Multithreading involves a new way of programming the parallel execution of tasks and how they interact with each other vs. the traditional line-by-line execution of code . As programmers add more threads to their applications, the complexity of managing those threads jumps exponentially. The situation is analogous to a single developer working on a project vs. a team of developers working on the same project. The team can accomplish more in less time, but the members must communicate among themselves to avoid duplication of effort and prevent any accidental collisions in their work. Regardless of the development environment, multithreading presents many important advantages to developers of Banking and other applications. Developers creating multithreaded applications can use their computer-based systems to the fullest potential as companies continue to standardize on multithreaded operating systems

or acquire increasingly affordable multiprocessor computers to boost performance

In creating a multithreaded algorithm, there are three basic considerations. The first step is to identify the parallelism. This may mean simply decomposing the problem domain of a conventional algorithm into several sections. The second step is to control the access to shared data items. The third step is to optimize the algorithm.

II. RELATED WORK

Blakrishnan and Nandy (2000) showed that multithreaded architectures coupled with SIMD parallelism provide performance improvement in excess of 2% over conventional super scalar architectures. In Pop and Kumar (2006), the results indicate that processors can substantially benefit from multithreading, even in systems with small caches, provided sufficient network bandwidth exists. Increased network contention due to multithreading has a major effect on performance. Their experimental results show that by taking advantage of SMT technology they achieve a 30-70% improvement in throughput over single threaded implementations on in-memory database operations. Costas Kyriacous and Evripidou, (2006) describes the Data-Driven Multithreading is a non blocking multithreading execution model that tolerates inter node latency by scheduling threads for execution based on data availability. Jack L. Lo, Luiz and André Barroso explored the behavior of database workloads on simultaneous multithreaded processors, concentrating in particular on the challenges presented to the memory system. For their study, they collected traces of the Oracle DBMS executing under Digital Unix on DEC Alpha processors, and processed those traces with simulators of wide-issue superscalar and simultaneous multithreaded processors. they believe that these results indicate that Simultaneous Multithreading latency tolerance makes it an excellent architecture for database servers. Hyoseung Kim and Hojung Cha describes in his paper multithreading approach provides a convenient sensor application developing environment with automatic control flow and stack management, it is considered to have a larger data memory requirement and energy consumption than an event-driven model. The experimental results on a common sensor node show that the multithreaded system could be effectively implemented with reasonable overhead.. Jack L. Lo describes Simultaneous Multi-threaded architectures (SMT) allow better utilization of resources, thus a clustered SMT architecture exploits the advantages of SMT

architectures and hides much of the inter cluster communication latencies incurred due to clustering. This is achieved by executing instructions from a different thread when a thread stalls waiting for inter-cluster communication. Xiaofeng Guo experimented with network processors employ a multithreaded, chip-multiprocessing architecture to effectively hide memory latency and deliver high performance for packet processing applications. Stefan Link and et al (2009) they focused their research work on key performance for human interaction. According to Johan there are five types of MDD they are programming - the manual work, Code generation - automation MDA-abstraction,DSL-specializing and MDE - from problem to solution Salvador and et al (2007) described that MDD is an emerging paradigm for software construction that uses models to specify programs, and model transformations to synthesize executables.

III. EXPERIMENTS AND OBSERVATIONS

A. Banking Applications

Program is written in Java with and without threads for banking application. Program with thread takes less time than the program with out thread. In this application we consider only two operations deposit and withdrawal with twenty(20) records. Study shows that searching time varies based on the position of the record and the value of the amount deposited or withdrawn(size of the figure 1000, 100000). Since the process involves user interaction with the system , timing difference plays a major role in the analysis . Thus the analysis is repeated ten times with the same input value and the average is considered as the final value. The following table shows the analysis for the deposited value of Rs.1000. The program is run in Pentium processor with HT supported machine and database used MS access and JAVA swing

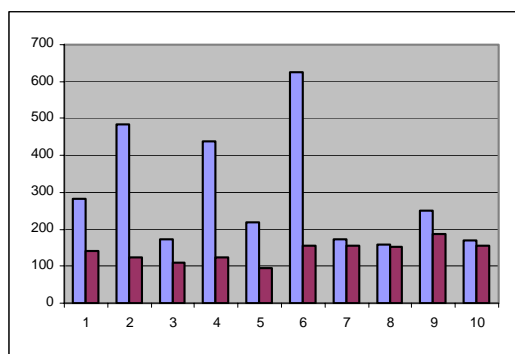


Figure 1. Banking Application

B. Client Server applications

The evolution of the multi-threaded processor design is the trend for the next generation desktop processors. To add a record from the client side, a program is written and executed in Java with and without thread. The results are tabulated .If we compare the average value of two, program with threading takes the advantage (117ms) over without threading(157ms) . So it is evident that threading in client-server application is faster always irrespective of number of records.

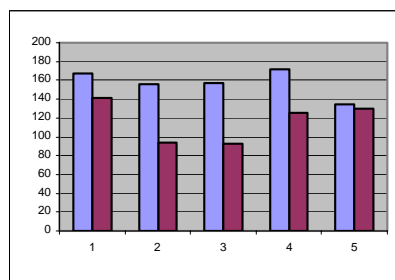


Figure 2 : Threading in Client Server applications

C.Thread Operation Models

Dispatcher-worker thread model

In this model, [Cameron Hughes et al 2004] the process consists of a single dispatcher thread and multiple worker threads. The dispatcher thread accepts requests from clients after examining the request , dispatches the request to one of the free workers threads for processing of the request. Each worker thread threads works on a different client request therefore multiple client requests can be processed in parallel.

Peer to Peer model

All threads behaves as equal in the sense that there is no dispatcher-worker relationship for processing clients' requests. Each thread gets and processes clients' request on their own. this model is often used for implementing specialized threads with in a process. therefore multiple types of requests can be simultaneously handled by the process.

Pipeline model

This model is useful for applications based on producer consumer model in which the output generated by one part of the application is used by another part of the application. In this model , the threads of a process are organized as a pipeline so threads the output data generated by the first thread is used for processing by the second thread, the output of the second thread is used for processing the third and so on . the out put of the last thread in the final output of the process to which the thread belong.

Three models were implemented in java and the observation is graphically represented. This representation concludes that peer to peer model seems to be the best.

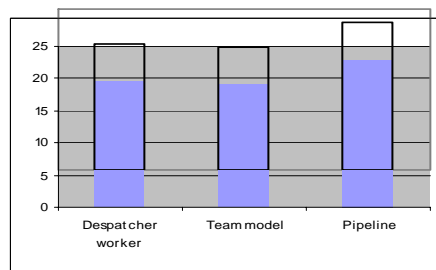


Figure 3 Threading models

IV PROPOSED WORK

Model-driven approach

The main aim of this paper is to propose and implement model driven methodology with multithreading support to increase the performance of any real time applications.

Increasingly systems have the ability to undertake decisions and execute actions without reference to people in either the choice of decision or the course of action. Online booking system is an ideal feature for any client who would like to allow customers or clients the ability to book a resource/course/training event online. The booking systems can be date based where a calendar is shown in order to help choose an event date. A form can then collect all of the delegate information and store the information for company to process. Each online booking system is designed individually for each client and their specific purpose or application. They are designed to attract the highest conversion on visitors to the site to number of online bookings

We first defined a design methodology and a development flow for online application system using object oriented technology. We then included a UML profile for modeling software applications. The new code sign flow is supported by an environment, which allows system modeling at higher abstraction levels (from a functional executable level to a register transfer level) and supports automatic code-generation/back-annotation from/to UML models.

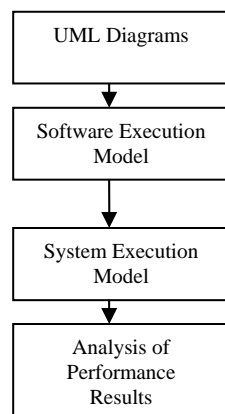


Figure 4 Proposed diagram for online applications

Step0: Analyze the requirement specification in detail and collect the data.

Step2 : Determine the number of use cases
And the actors

Step3 : Determine the relationship between the
use cases

Step3: Draw Sequence diagrams to identify parallel
and sequential activities

Step4: Construct the software execution model
and evaluate Performance parameters

V.CONCLUSIONS

Multithreading has emerged as one of the most promising and exciting techniques used to increase the performance of applications. Many software applications deliver appreciable performance gains on the Pentium

processor. We present a performance-based methodology, a method to analyze, specify and model for the systems, particularly to be employed for computing applications. MDD with multithreading approach is proposed. The use-case approach is chosen because it has been widely accepted as a tool for requirements analysis. Using online reservation system as a case study, we found that our method is viable enough to be applied for other real life cases.

REFERENCES

- [1] Balakrishnan, S. and S.K. Nandy, 2000. Performance evaluation of multithreaded architectures for media processing applications. In: Proc. IEEE. Int. Symp. Circ. Syst. ISCAS, 1: 531-534.
- [2] Intel Corporation Threading Methodology, 2003 Principles and Practices version 2.0.
- [3] Pop. R and S.Kumar, 2006 on performance improvement of concurrent applications using SMT processors.
- [4] John, L.H. and D.A. Patterson, 2003. Computer Architecture A Quantitative Approach 3rd Edition.
- [5] Guitart, F.J., 2005. Performance Improvement of Multithreaded Java Applications Execution on Multiprocessor Systems Performance
- [6] Stefan Link, Philip Hoyer, Tilmann Kopp, 2009 Sebastian A Model-Driven Development Approach Focusing Human Interaction
- [7] Johan den hann 5 types of Model Driven Software Development 2009
- [8] Gregor Engels, Model-Driven Development for End Users, too!?
- [9] Sabina Jeschke¹, and Helmut Vieritz¹ Accessibility and Model-Based Web Application Development for e-Learning-Environments.
- [10] Javier Muñoz, Estefania Serral, Carlos Cetina and Vicente Pelechano Applying a Model-Driven Method to the Development of a Pervasive Meeting Room
- [11] MJ. Escalona and G. Aragon Model-Driven approach for webrequirements. May/June-2008.
- [12] Marco Brambilla, Stefano Ceri, Piero Fraternali Process Modeling in Web Applications
- [13] James Gan, Model-driven Development Tool for Parallel Application 2007.
- [14] Soumya Gupta Comparative Study of Threading Models Software and Solutions Group, Intel Corporation 2007.
- [15] Allan McNaughton Multi-Threading in a Java* Environment 2007.