RESEARCH ARTICLE OPEN ACCESS

Features for Parallel Pattern Based Programming System for Multicore

Nilesh Maltare
Asst. Professor IT *(M.B.I.C.T.) New V.V. Nagar Anand, India*

Abstract
This paper provides analysis of features provided by existing Parallel design patterns based Programming System. Objective of this paper is to examine features required to exploit parallelism with ease in Multicore Architectures.
Index Terms—Parallel Computing, Parallel Pattern, Multicore Programming.

I. INTRODUCTION

Recent trends in hardware design towards multicore CPUs with hundreds of cores. It demands for better programs which can exploit multicore. In other words "sequential programs need to be refactored for parallelism"[1]. According to Moore's Law "All software developers have abstracted away the processor assuming that H/W will gets faster and faster" [2]. We need to support new features and larger data sets. It is clear we have to program for performance. There will be emphasis on portability, robustness, malleability and maintainability. After that complexity needs to be managed.

Program can exploit parallelism through various constructs threads,loop level block level parallelism. H/W and Architectural changes offers new way to exploit parallelism. We generally obtain parallelism through task decomposition and mapping of task to processes. Earlier we have observed data movement is bottleneck and we have to pay more cost in communication. With Multicore Architecture we should not afraid to decompose task. Even ultra fast synchronization in Multicore architectures motivate for use of synchronization constructs.

The use of multiprocessor machines has accelerated the importance of parallel programs. The shared multiprocessor machine has become a challenge and the development of the parallel applications is growing slowing. While developing parallel software, programmers must worry about task decomposition, scheduling, communication, and synchronization.

II. CHALLENGES IN EXPLOITING PARALLELISM

Since inception of programming we have followed sequential programming paradigm, Thinking parallel or decomposing problems into parallel tasks is hard for many of us. These all increases complexity and work for programmer. Parallelism is not easy to implement, Parallelism cannot be abstracted.

While Parallel programming programmer is exposed to parallel architectures, which creates parallel program biased to particular architecture. Architecture influences Programming models we can observe:

Message Passing model provides MPI which is suitable for Distributed Shared Memory where processes communicates through messages.

Thread Based Programming models uses threads which uses shared memory approach.

We can notice that there is no unique execution model and programming paradigm to deal wit parallelism. That is the reason programmer needs to work for details of everything. Programmer need to work for lower level API's to exploit parallelism.

Debugging of Parallel program difficult due to:

- Number of possible execution orderings
- Incorrect Synchronization leads to Race condition
- Even Correct program may lead to deadlock

III. PATTERNS AND PARALLELISM

Patterns originated as an architectural concept by Christopher Alexander. Design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.[12] A design pattern is not a finished design that can be transformed directly into source or machine code.[10] Use of Patterns to exploit parallelism is not new. DpnDP, a Design Pattern Based Parallel Programming System was proposed by Siu and Singh, 1997.[5] . Patterns for Parallel Programming demonstrated in book by Matson 2002.[4] CO2P3S (Correct Object Oriented Pattern based Parallel Programming System) is a tool provides abstractions in the PDP.[6]

Pattern–based parallel programming system and ideal characteristics is described in [6] :

- Structuring the Parallelism
- Programming
- Performance
- Portability
- Support

In Structuring Parallelism, we need to provide clean separation between the parallel structure of a program and the application code. It provides easy modification. Code libraries fail to meet this concern. In all code libraries, the structure of the parallelism is embedded directly in the application code via library calls. This structure can only be changed by modifying the code, which may involve significant programming effort. For instance, for message passing libraries such as PVM [21]and MPI[9], the parallel structure is dictated by the communication structure. A program using pthreads or Java threads must explicitly create the threads that will form the parallel structure of the program.

Most parallel programming languages and language extensions also suffer from the same problem by using extra syntax for specifying parallelism in the program code. Some java based languages support data parallelism by including parallel loops (with a forall statement, for example) and extra syntax for parallel array expressions.

Some parallel languages, such as High Performance FORTRAN(HPF) [17] and OpenMP [18] reduce the separation problem by inserting parallel directives as comments that are used only by special compilers. These directives can be added or removed with less effort.

The only parallel programming language that eliminate this problem is P3L, a pattern based programming language. A P3L program consists of a set of named code fragments and a separate pattern description that indicates how the fragments and patterns are composed into a larger program. It is observed that parallel structure can be separated from the application code through indirection.

In explicit message–passing systems such as DPnDP [5], Tracs, and Parsec, all messages are sent through channels via ports. The ports do not contain any reference to the process that will actually receive the data, and so decouple the two communicating processes. Thus, a process may be freely interchanged with another that exchanges the same data.

Parallel Architectural Skeletons (PAS) [8] take a slightly different approach. They create an additional process that serves as the fixed entry point to the processes that make up the pattern. This fixed entry point allows a pattern to be replaced with another in a seamless manner.

A system should allow patterns to be composed hierarchically, refining the computation within a given pattern using another pattern. In general, a single parallel structure cannot be used to effectively parallelize all parts of a large computation. Since a user can generally place communication calls anywhere in application code, message–passing and thread libraries meet this characteristic. Without hierarchical resolution, the user must draw one large graph for the structure of the complete program. Programs built using skeletons or frameworks can experience composition problems. Root cause of these problems is that frameworks and skeletons were created with the assumption that only one will be used in a given application. There should be no rules regarding how patterns can be composed. While code libraries meet this concern because of their generality. Among the pattern–based programming systems (DPnDP, Enterprise,FrameWorks, Parsec, Tracs, and PAS), only FrameWorks, the oldest system, does not exhibit independence of patterns. Specific combinations of structures could not be properly supported.

Pattern based Programming System can provide many feature to extract parallelism available but it is important to focus on Ease of Programming also. Ease of Programming and Code Efficiency are contradictory objective. Use of pattern based approach can simplify hard parallel code.

It should be possible to achieve the best possible performance for a program but it is dependent on selection of the parallel patterns. General–purpose parallel libraries can provide the best performance. Programmer can optimize program by reducing communication and synchronization costs.

Applications should be able to ported on different architectures. The performance of a program may suffer on different architecture, but application should continue to run. Unlike Shared memory machines, message–passing systems continue to communicate using expensive network messages rather than taking advantage of cheaper memory–based communication mechanisms.

IV. FEATURES IN PARALLEL PATTERN BASED PROGRAMMING SYSTEM FOR MULTICORE

Although there are number of Parallel Programming Platforms tools and techniques available ,still we need Parallel Programming System for multicore provides:

- Ease of Programming : Programmer need not to learn difficult syntax, lower level API's to exploit parallelism. It makes parallel programs also difficult to debug and understand. Abstraction will be key to achieve same. Abstraction can eliminate problem faced in debugging Parallel Program.

- Language Support : It should support common languages to leverage skills of programmers. Parallelism can be extracted by large group if System meet same. Message passing library does not impose any changes to programming language. Programmer needs to use extra communication calls. Most of the programmers are comfortable with common languages like C,C++, Java that is the reason most application can take benefit of Parallelism.
- Support for Improving Performance : System should help programmer to achieve the best possible performance for a program. Programmers can select parallel patterns to reduce communication and synchronization cost. Dynamic threading and other techniques should be available with convenient model of use.
- Flexibility : System simplicities changes in parallel program, because in parallel programming small changes introduces bugs. Flexibility and ease of programming are complementary goal in building such system. If System provides abstraction at desired level readability, malleability, and maintainability can be achieved.
- Portability : Program should not be biased with particular architecture. One code written for shared address space will work for distributed memory architectures also and vice versa.

V. Conclusion

We have explored different Parallel Platforms for programming and also observed parallel constructs available. Pattern based approach to exploit parallelism is evaluated for achieving ease of programming, correctness and performance. In general programming languages, if parallelism is available with ease we can benefit large number of applications.

VI. Acknowledgment

I wish to acknowledge my Ph.D. Advisor and mentor Dr. V. N. Kamat for valuable guidance and support.

References

[1] Vandierendonck, H. , Mens, T. "Averting the Next Software Crisis." Computer Volume:44, Issue: 4 Digital Object Identifier: 10.1109/MC.2011.99, Publication Year: 2011 , Page(s): 88- 90.

[2] Kozyrakis, C.E. , Patterson, D.A. "A new direction for computer architecture research" Computer Volume:31, Issue: 11 Digital Object Identifier: 10.1109/2.730733, Publication Year:1998 , Page(s): 24 - 32

[3] Ananth Grama,George Kar. "Introduction to Parallel Computing" Addison-Wesley, 2003, ISBN13: 9780201648652.

[4] Timothy G. Mattson,Beverly A. Sanders, Berna L. Massingill. "Patterns for Parallel Programming", 2005 ISBN-10: 0321228111 ISBN-13: 9780321228116,Addison-Wesley Professional .

[5] D. Goswami, A. Singh, and B. Priess. "Architectural skeletons: The reusable building-blocks for parallel applications". In Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applciations (PDPTA'99), pages 1250–1256, 1999..

[6] S. MacDonald, D. Szafron, J. Schaeffer, and S. Bromling. "From patterns to frameworks to parallel programs". Journal of Parallel and Distributed Computing, 2001. .

[7] J.L. Ortega,"Arjona Design Patterns for Communication Components", Proceedings of the 12th European Conference on Pattern Languages of Programming and Computing (EuroPLoP2007), Kloster Irsee, Germany, 2007

[8] Xin Liu , Jingyu Zhou , Daqiang Zhang , Yao Shen , Minyi Guo, "A Parallel Skeleton Library for Embedded Multicores Parallel Processing". Workshops (ICPPW), 2010 39th International Conference on DOI: 10.1109/ICPPW.2010.21, Publication Year: 2010 , Page(s): 65- 73

[9] Kamal, H. , Wagner, A.. FG-MPI: Fine-grain MPI for multicore and clusters, Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium, Digital Object Identifier: 10.1109/IPDPSW.2010.5470773.

[10] G. Booch. "Back to the future". IEEE Software, 2008.

[11] F. Buschmann, K. Henney, and D. C. Schmidt. "Pattern-oriented Software Architecture": On Pattern and Pattern Languages, volume 5. John Wiley and Sons, 2007.

[12] Gamma Eric. "Agile, open source, distributed, and on-time - inside the Eclipse development process", International Conference on Software Engineering 2005, Digital Object Identifier: 1010.1109/ICSE.2005.1553528.

[13] Ampatzoglou Apostolos, Kritikos Apostolos ,Kakarontzas, George and Stamelos, Ioannis. "An empirical investigation on the reusability of design patterns and software packages". Journal of System Software

2011, Digital Object Identifier: 10.1016/j.jss.2011.06.047.

[14] Autili, Marco and Di Benedetto, Paolo and Inverardi, Paola. ”A hybrid approach for resource-based comparison of adaptable Java applications”, Elsevier Journal of Science & Comp. Programming 2013 volume 78, pages :987-1009, Digital Object Identifier:10.1016/j.scico.2012.01.005

[15] DeHon, A. and Adams, J. and deLorimier, M. and Kapre, N. and Matsuda, Y. and Naeimi, H. and Vanier, M. and Wrighton, M.. “Design patterns for reconfigurable computing”. Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on 2004, pages :13-23, Digital Object Identifier:10.1109/FCCM.2004.29.

[16] Jain, Dolly and Yang, Helen J. ”OO Design Patterns, Design Structure, and Program Changes”: An Industrial Case Study. Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01) 2001, pages: 580 Digital Object Identifier:10.1109/ICSM.2001.972775.

[17] Pazat,J.L.. Tools for high performance FORTRAN: A survey. In The Data Parallel Programming Model, volume 1132 of Lecture Notes in Computer Science, pages 134–158. Springer–Verlag, 1996.

[18] OpenMP 4.0 Specifications, http:// openmp.org/wp/openmp-specifications/

[19] JavaDocs,http://docs.oracle.com/javase/7/docs/api/java/util/concurrent

[20] J.L. Ortega, “Arjona Patterns for Parallel Software Design”, John Wiley & Sons, 2010.

[21] Geist, A. , Beguelin, A. , Dongarra, J. , Jiang, W. , Manchek, R., Sundam, V.. “PVM:A Users' Guide and Tutorial for Network Parallel Computing”, Page(s): 11-17,1994