ISSUES IN IMPLEMENTATION OF PARALLEL PARSING ON MULTI-CORE MACHINES

Amit Barve[1]and Brijendra Kumar Joshi[2]

[1]Asst. Professor,CSE, VIIT Pune,India
[2]Professor,MCTE,Mhow

ABSTRACT

The advent of multi-core architecture has highly influenced the area of high performance computing. Parallel compilation is the area which still needs significant improvement by the use of this architecture. Recent research has shown some improvement in lexical analysis phase. But it is difficult to implement the same technique in parsing phase. This paper highlights some issues related to implementation of parallel parsing on multi-core machines.

KEYWORDS

Syntax Analysis, Parallel Parsing, Multi-core Machines.

1. INTRODUCTION

Compiler is a program that translates a source language into target language. The structure of a compiler is composed of several phases. The first phase is lexical analysis or scanning. This is the only phase which interacts with original source code written by the programmer. It takes stream of characters as input and generates tokens of the form {token name, attribute value} as output. The task that does this is called lexical analyzer or scanner. Lex [1] and Flex [2] are two popular tools for automatically generating lexical analyzers from specifications.

The information about tokens is saved in a special data structure called symbol table. These tokens are then forwarded to the next phase i.e. syntax analysis also known as parsing. Parsing is an important phase in compilers. This phase takes the stream of tokens as input produced by lexical analyzer and converts them into parse trees. A parse tree is a structural representation of grammar being parsed. The tool which performs this task is known as parser. Parser can be automatically generated by YACC [3] and Bison[4] which take grammar specifications as input and produce parsers.

Interaction of the lexical analyzer and the syntax analyzer is depicted in Fig. 1. The details of various phases of a compiler can be found in popular texts [5][6][7][8].

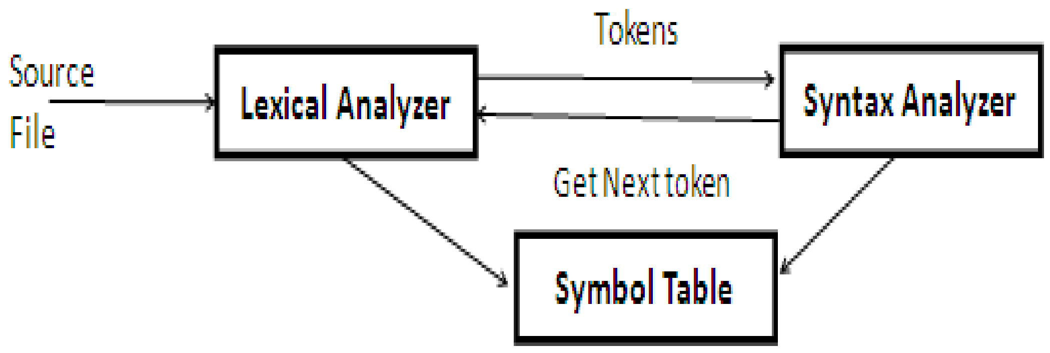

Fig. 1. Interaction of Lexical Analyzer with Parser

2. PARSING TECHNIQUES

The parsing algorithms are primarily classified into two categories, top-down parsing and bottom-up parsing. These refer to the order in which nodes in a parse tree are constructed. In top-down approach the construction of a tree starts from root and proceeds towards the leaves while in bottom up approach construction of a parse tree starts with leaves and proceeds towards the root. Some well known top-down parsing algorithms are recursive decent parsing (also called predictive parsing) and non-recursive decent parsing. Bottom-up parsing includes some algorithms like Simple LR (SLR) parsing, Canonical LR (CLR) parsing, and Look Ahead LR (LALR) parsing.

In LR parsing, parser reads input from left to right and generates a right most derivation in reverse. The name LR(*k*) parser is also used, where *k* refers to the number of unconsumed look ahead input symbols that are used in making parsing decisions. Depending on how the parsing table is created, an LR parser can be called SLR, LALR, or CLR Parser. LALR parsers have more language recognition power than SLR parsers. Canonical LR parsers have more recognition power than LALR parsers. For comparison of these parsers, refer to Table 1.

Table 1. Comparison of parsing techniques

Parsing Technique	**No. of Look Ahead tokens**	**No. of Iterations**	**Grammar recognition Power**	**Grammar used**
SLR	0	Maximum	Least Powerful	Context Free Grammar
CLR	1	Less than SLR	Most powerful Technique	Context Free Grammar
LALR	1	Less than LALR	More powerful than SLR but less than CLR	Context Free Grammar

3. PARALLEL PARSING

Parallel parsing has been attempted by many in the past. The parallel processing was achieved by assigning totally different user jobs to different processors. Zosel[9] focused on recognizing FORTRAN DO-loops that can be collapsed into vector instructions for CDC 7600 machines. Lincoln [10] first proposed the concept of parallel object code for FORTRAN and COBOL job cards in an environment that consisted of IBM 704 uniprocessors and CDC 6500 of ILLIAC IV.

Mickunas and Shell[11] recognized the areas in a compilation process where the parallel processing is inherent. They proposed to divide lexical analysis into scanning and screening. They also developed a parallel parsing technique based on LR parsing. Hickey and Katcoff[12] have analyzed parsing algorithms for upper bound on speedup whereas Cohen and Kolodner[13] have estimated speedup in parallel parsing. Chandwani et al [14] developed a parallel algorithm for CKY-parsing for context free grammars. Khanna et al[15] proposed the partitioning of grammar to make it appropriate for parallel compilation. Object Oriented parsing was proposed by Yonezmva and Oshava[16].

4. MACHINES ARCHITECTURE

Processor is a logic circuitry that responds to and processes the basic instructions that drive a computer.

Single Core Processor is a processor that has only one core (Processor), so it can only start one operation at a time. It can however in some situations start a new operation before the previous one is complete.

Multi-core processor is a processing system composed of two or more independent cores. It can be described as an integrated circuit to which two or more individual processors (called *cores* in this sense) have been attached. Fig. 2 and 3 give a simplified view of single and multi-core machines.

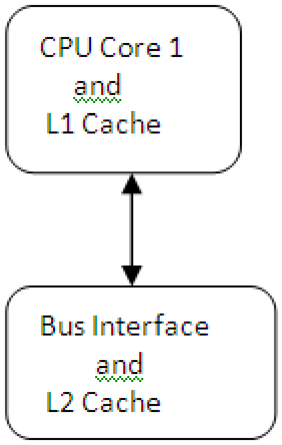

Fig. 2 Single Core Machine

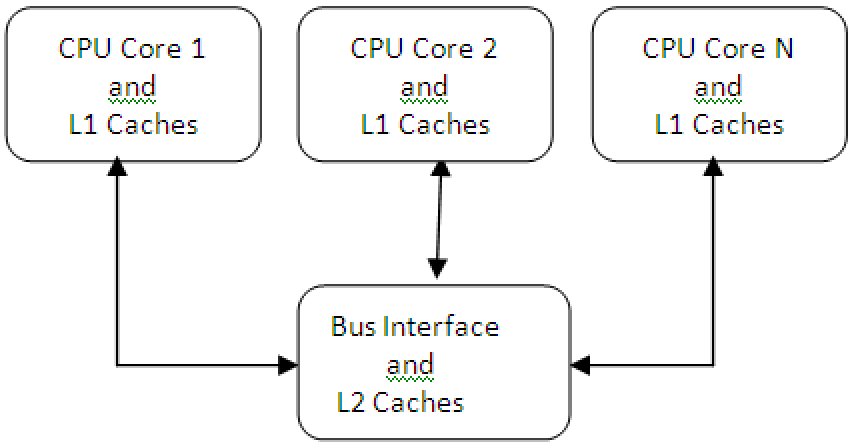

Fig. 3 Multi-Core Machine

Multi-core machines have various advantages like better resource utilization, efficient data sharing (sharing data through memory is more efficient than massage-passing), increased performance etc [17].

The major challenges while designing a multi-core compiler are program optimization, making parallel programming mainstream and development of performance models to support optimization for parallel code. Compiler should be capable of self improvement [18].

5. IMPLEMENTATION ISSUES IN PARALLEL PARSING

The efforts cited in reference [11]-[16] to develop parallel parsing algorithms are of theoretical significance only. Their practical implementations have not been seen so far in real programming languages for multi-core machines because of issues discussed next.

a) **Division of code and Synchronization:** Barve and Joshi[19][20][21] developed some algorithms for doing parallel lexical analysis on multi-core machines. Their approach is to divide the source code into number of blocks and perform lexical analysis on individual blocks. Their approach was good for parallel lexical analysis. If we use the same approach for syntax analysis, building of a common symbol table is an issue as multiple instances of syntax analyzer would be in action. These syntax analyzers would generate individual symbol tables corresponding to the source code at their disposal.

b) **Processor Issues:** In the past, the researchers assumed that if n processors are available then task is divided into several parts and is assigned to any of the available processors that do the job independently. In multi-core machines this task can be done by the use of processor affinity concept [22][23]. To obtain higher degree of precision in time consumption, it is required that the underlying operating environment be attached to a single processor relieving remaining processors for exclusive use by the parallel parsing algorithm. Binding entire operating system to a single processor is not straightforward.

c) **Threading:** Threading is an essential feature of multi-core machines which enables us to achieve parallelism. Run time libraries like PTherad[24], Thread Building Blocks(TBB)[25] and OpenMP[26] are used for this purpose. Threading is also responsible for performance degradation. Some time more threading takes more times as compared to serial counterpart of the target program. So, it is essential that threading be used only when it is required and which results in increased performance.

d) **Task Distribution:** Task distribution is also an important factor which affects performance. The distribution of tasks may be done in such way that no processor will be free after finishing its task. Rajan et al have evaluated the performance of such distribution on High Performance Computing (HPC) clusters [27][28][29].

e) **Context Switching:** System has to pay the cost when context switching is done specially in multi-core systems. Chuanpeng Li. Et al have shown the results of experimentally quantifying the indirect cost of context switching using a synthetic workload. They have also measured the impact of program data size and access stride on context switch cost [30].

6. CONCLUSION

In this paper various issues in implementation of parallel parsing algorithms on multi-core machines were discussed. It is imperative to pay attention to synchronization among threads for shared resources. This point has been addressed numerous times since the decades. The problem becomes more serious as the number of core per machines and clock speed of processors increase. Still a good amount of dedicated efforts is required to explore inherent property of parallel processing present in multi-core machines targeting parsing.

REFERENCES

[1] M. E. Lesk, E. Schmidt; “Lex- A Lexical Analyzer Generator”; Computing Science Technical Report No. 39, Bell Laboratories, Murray Hills, New Jersey, 1975.

[2] http://flex.sourceforge.net/

[3] S. C. Johnson; “YACC: Yet Another Compiler Compiler”; Computing Science Technical Report no 32, Bell Laboratories, Murray Hills, New Jersey, 1975.

[4] www.gnu.org/s/bison. (Last accessed on 05-Aug-2014)

[5] Alfred V. Aho, Ravi Sethi, Jeffrey D.Ullman; “Principles of Compiler Design”; Addison Wesley Publication Company, USA, 1985.

[6] Alfred V. Aho, Ravi Sethi, Jeffrey D.Ullman; “Compilers: Principles, Techniques and Tools”;Addison Wesley Publication Company, USA, 1986.

[7] Jean Paul Tremblay,Paul G. Sorenson;”The Theory and Practice of Compiler Writing”;McGraw-Hill Book Company USA 1985

[8] David Gries; “Compiler Construction for digital Computers”; John Wiley & Sons Inc. USA, 1971.

[9] M. Zosel; “A Parallel Approach to Compilation”; Conf. REc. ACM Sysposium on Principles of Programming Languages, Boston, MA, pp. 59-70, October 1973.

[10] N. Lincoln; “Parallel Compiling Techniques for Compilers”; ACM Sigplan Notices, 10(1970), pp. 18-31, 1970.

[11] M. D. Mickunas, R. M. Schell; “Parallel Compilation in a Multiprocessor Environment”; Proceedings of the annual conference of the ACM, Washington, D.C., USA, pp. 241–246, 1978.

[12] Timothy Hickey, Joel Katcoff; “Upper Bounds for Speedup in Parallel Parsing”; Journal of the ACM (JACM), Vol. 29, No. 2, pp. 408 – 428, 1982.

[13] J. Cohen, Stuart Kolodner; “Estimating the Speed up in Parallel Parsing”; IEEE Transactions on Software Engineering, January 1985.

[14] M. Chandwani, M. Puranik , N.S. Chaudhari, “On CKY- Parsing of Context Free Grammars in Parallel”; Proceedings of the IEEE Region 10 Conference, Tencon 92, Melbourne Australia, pp. 141-145, 1992.

[15] Sanjay Khanna, ArifGhafoor, AmritGoel; “A Parallel Compilation Technique Based on Grammar Partitioning”; Proceedings of ACM annual conference on Cooperation, Washington, D.C., USA, pp. 385 – 391, 1990.

[16] Akinori Yonezmva, Ichiro Ohsawa; “Object-Oriented Parallel Parsing for Context-Free Grammars”; Proceedings of the 12th conference on Computational linguistics – Vol. 2, Budapest, Hungry, pp. 773–778, 1988.

[17] Valeriy Shipunov, Andrey Gavryushenko, Eugene Kuznetsov,” Comparative Analysis of Debugging Tools in Parallel Programming for Multi-core Processors” CADSM’2007, February 20-24, 2007, Polyana, UKRAINE IEEE.

[18] Mary Hall, David Padua and Keshav Pingali,”Compiler Research:The Next 50 Years”, Communication of the ACM Feb 2009,Vol. 2.

[19] Amit Barve and Dr. Brijendra Kumar Joshi;”A Parallel Lexical Analyzer for Multi-core Machine”; Proceeding of CONSEG-2012,CSI 6th International confernece on software engineering; pp 319-323;5-7 September 2012 Indore,India.

[20] Amit Barve and Brijendrakumar Joshi, "Parallel lexical analysis on multi-core machines using divide and conquer," NUiCONE- 2012 Nirma University International Conference on Engineering , pp.1,5, 6-8 Dec. 2012. Ahmedabad, India.

[21] Amit Barve and Brijendrakumar Joshi; “Parallel lexical analysis of multiple files on multi-core machines”; International Journal of Computer Applications; Vol. 96, No.8, June 2014.

[22] http://www.linuxjournal.com/article/6799?page=0,1.

[23] http://www.cyberciti.biz/tips/setting-processor-affinity-certain-task-or-process.html

[24] David R. Butenhof, “Programming with POSIX Threads”, Addison-Wesley Longman Publishing Co., USA 1997.

[25] http://openmp.org/wp

[26] http://www.threadingbuildingblocks.org.

[27] Rajan, A; Joshi, B.K.; Rawat, A; Jha, R.; Bhachavat, K., "Analysis of process distribution in HPC cluster using HPL," 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), 2012, pp.85,88, 6-8 Dec. 2012 Solan India.

[28] Rajan A., Joshi B.K., Rawat A., Gupta S.”Analyitical Study of HPCC Performance Using HPL”;International Journal of Computer Science and its Applications, Vol. 2, no. 1, p. 47-49, Apr. 2012.

[29] Rajan A., Joshi Brijendra Kumar, Rawat A.”Critical Analysis of HPL Performance under Different Process Distribution Patterns”.CSI 6th International Conference on Software Engineering (CONSEG-2012), DAVV, Indore, Sep., 5-7, 2012

[30] Chuanpeng Li, Chen Ding, Kai Shen;”Quantifying the cost of context switch”,ExpCS’07’ Proceeding of the 2007 workshop on Experimental computer science; article 2; ACM New York USA;2007.

Authors

Mr. Amit Barve is an Assistant Professor in Computer Engineering at Vishwakarma Institute of Information Technology, Pune (M.H.) India. He has completed BE in Computer Science and Engineering from MIT Ujjain; M.Tech. in Computer Engineering from VJTI Mumbai. His research interests are parallel processing, HPC, and compiler design.

Dr. Brijendra Kumar Joshi is a Professor in Electronics & Telecommunication and Computer Engineering at Military College of Telecommunication Engineering, Mhow (M.P.), India. He has obtained BE in Electronics and Telecommunication Engineering from Govt. Engg. College Jabalpur; ME in Computer Science and Engineering from IISc, Banglore, and Ph.D. in Electronics and Telecommunication Engineering from Rani Durgavati University, Jabalpur, and M.Tech in Digital Communication from MANIT, Bhopal. His research interests are programming languages, compiler design, digital communications, mobile ad hoc and wireless sensor networks, software engineering and formal methods.