Design and Development of a Parallel Lexical Analyzer for C Language

Swagat Kumar Jena, Gandhi Institute of Technological Advancement, Department of Computer Science and Engineering, Bhubaneswar, India
Satyabrata Das, VSS University of Technology, Department of Computer Science and Engineering, Burla, India
Satya Prakash Sahoo, VSS University of Technology, Department of Computer Science and Engineering, Burla, India

ABSTRACT

Future of computing is rapidly moving towards massively multi-core architecture because of its power and cost advantages. Almost everywhere Multi-core processors are being used now-a-days and number of cores per chip is also relatively increasing. To exploit full potential offered by multi-core architecture, the system software like compilers should be designed for parallelized execution. In the past, various significant works have been made to change the design of traditional compiler to take advantages of the future multi-core platform. This paper focuses on adapting parallelism in the lexical analysis phase of the compilation process. The main objective of our proposal is to do the lexical analysis i.e., finding the tokens in an input stream in parallel. We use the parallel constructs available in OpenMP to achieve parallelism in the lexical analysis process for multi-core machines. The experimental result of our proposal shows a significant performance improvement in the parallel lexical analysis phase as compared to sequential version in terms of time of execution.

KEYWORDS

Compiler, Lexical Analysis, Multi-Core, OpenMP, Threads, Tokenizer

INTRODUCTION

Usage of multi-core architecture represents the latest trend in modern day computers with their power and cost advantages. In the new era of supercomputer, most of the system’s processors belong to the multi-core processor family. As number of cores continues to grow, a question often confronts the software designers whether software applications running on these platforms can take full advantage of the inherent parallelism offered by these. To fulfill this goal software applications should be designed for parallelized execution and system software like Operating System, Compiler etc. should also support these parallelized platforms. System software should not lead to scalability bottlenecks. If we take the case of compiler design then some excellent work has been done by Mickunas and Shell (Mickunas & Shell, 1987). They proposed a very novel theoretical approach to split the lexical analysis into scanning and screening. The scanning of a text can be done in parallel.

In this paper, we propose a parallel lexical analyzer, which counts the number of lines in the input file and achieve parallelism in lexical analysis phase using the processor affinity concept on

DOI: 10.4018/IJKBO.2018010105
multi-core systems. The result shows a performance improvement in terms of total time of execution in the lexical analysis phase of compiler as compared to sequential lexical analysis used in single processor systems. The rest of the paper is organized as follows. Section 2 presents an introduction to multicore architecture. In Section 3, we briefly discuss about the Lex which is an open source lexical analyzer generator in Linux environment. In Section 4, we discuss the tools for programming and profiling parallel programs. In Section 5, we discuss some significant works in this field. In Section 6, we propose an algorithm for implementing a parallel lexical analyzer. In section 7, we evaluate our proposed algorithm with sample test cases and show the performance enhancement. Finally, we conclude the paper in Section 8.

MULTI-CORE ARCHITECTURE

A Multi-core processor is an integrated circuit die (known as a chip multiprocessor or CMP) with two or more independent actual central processing units called “cores”. Each processor has its own primary cache and pipelined. The multiple cores processors can run multiple instructions at the same time, increasing the overall execution speed for programs which yields to parallel computing. Now-a-days almost all high-performance processors are belonging to multi-core family. While it is generally accepted that we have entered the multi-core era, but still researchers are concerned on scaling and adapting software to multi-core platform.

Some examples of multi-core processors are as follows; Intel Pentium D, Dual Core Opteron, Intel Montecito, Intel Xeon E7-2820 with 8 cores, Sun UltraSPARC IV, IBM Cell, Intel WoodcrestIBM Power4, IBM Power, AMD Quad- and Dual-Core UltraSPARC T1, AMD FX-8150 with 8 cores, Adapteva Epiphany with up to 4096 processors on-chip, SEAforth 40C18, a 40-core processor by IntellaSys etc. This represents the scalability of multi-core architecture.

Figure 1 shows a multi-core architecture with two nodes. One node is based on Non-Uniform Memory Access (NUMA) based design and other is based on Symmetric Multi-Processing (SMP) based design. In NUMA memory access times of cores can be different where as in SMP memory

Figure 1. This figure shows multi-core architecture with two nodes. One node is based on NUMA (Non-Uniform Memory Access) based design and other one is based on SMP (Symmetric Multi-Processing) based design, two nodes can be connected through interconnection network.
Exploring the Role of the Project Manager in Organizational Knowledge Creation
[www.igi-global.com/article/exploring-the-role-of-the-project-manager-in-organizational-knowledge-creation/193193?camid=4v1a](http://www.igi-global.com/article/exploring-the-role-of-the-project-manager-in-organizational-knowledge-creation/193193?camid=4v1a)

Corporate Semantic Webs
[www.igi-global.com/chapter/corporate-semantic-webs/25296?camid=4v1a](http://www.igi-global.com/chapter/corporate-semantic-webs/25296?camid=4v1a)