New Approach to Speedup Dynamic Program Parallelization Analysis

Sudhakar Sah, Symbiosis Institute of Research and Innovation, Lavale, Pune, India & Samsung SDS, Seoul, South Korea

Vinay G. Vaidya, KPIT Technologies, Hinjawadi, Pune, India

ABSTRACT

Development of parallel programming tools has become a mainstream research topic and it has produced many useful tools that reduce the burden of parallel programming. The most important aspect of any such tool is the data dependency analysis, which is very complex and computationally intensive. Fully automatic tools have many demerits such as low efficiency and lack of interaction with the user. In this paper, the authors present their tool called EasyPar having unique features to assist the user at the time of program development as opposed most of the other tools that work on completed programs. Performing parallelization analysis at the time of program development has a potential to exploit more parallelization opportunity by interaction with the programmer. Two most important requirements of such a tool are the accuracy and the performance of the analysis. The authors propose the method that uses database to enable quick and efficient dependency analysis. The proposed method utilizes a database to save program information in structured from and using a query based method for data dependency analysis. The database approach has three benefits. First, it makes the incremental parsing easier and faster; second, query-based approach makes the dependency analysis efficient and third it allows the demand driven program analysis possible. The authors have tested their tool on popular benchmark codes, presented the performance results, and compared it with relevant work.

Keywords: Compiler, Database, Multicore, Parallel Programming, Static Analysis

INTRODUCTION

Program parallelization has become mainstream research topic due to the invention of multicore processors. Multicore processors came to mainstream due to effect described by Moore’s law (Sutter, 2005) according to which the processor speed is doubled after every 1.5 years. The processor technology could not sustain the increase in speed after some limit due to the limitations imposed by power dissipation with increase in clock frequency. Although multicore processors could provide multifold processing speed using multiple cores, but legacy applications could not utilize it as most of the applications were written in serial fashion. Therefore, focus of research shifted to development of tools that could support the development of parallel programs or conversion of serial programs to parallel programs. This

DOI: 10.4018/ijsi.2014100103
requirement leads to the development of three different types of tools. First category of tools supports writing parallel programs manually and using parallelization constructs to support parallel execution like Haskell (Hughes, 1989), Erlang (Official homepage, 2013), Cilk (Frigo, 2007), Go (Go Programming Language Homepage) etc. This technique provides highly efficient parallel programs. However, it is tedious and time consuming to develop parallel programs manually. Hence, second type of tools, which supports automatic parallelization, is more useful. Automatic parallelization tools support source-to-source conversion of a legacy serial code to parallel code. These tools do not require any manual intervention and it is a faster way for developing parallel codes. However, efficiency of these tools is limited by the parallelization offered by legacy programs. Polaris (W. Blume, R. Doallo, R. Eigenmann, J. Grout, J. Hoeflinger, and T. Lawrence, 2002), ParMa(Parallel Programming for Multi-core Architectures), SUIF (S. P. Amarasinghe, J. M. Anderson, M. S. Lam and C. W. Tseng, 1995), Polyhedral parallelization (U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan, 2008), Automatic partitioner (Sarkar, V., 1991), Parallelism discovery (Saturnino Garcia, Donghwan Jeon, Chris Louie, Sravanthi Kota Venkata, and Michael Bedford Taylor, 2010) are some of the famous tools that fall under this category. The limitations of manual and automatic parallelization are taken care of by third type of tool and these tools are called the semiautomatic tools. Semiautomatic tools are capable of automatic dependency analysis but in addition it allows user interaction to increase the efficiency of the code. Parascope (McKinsley, Kathryin S., 1994), Paraassist (Krishnamoorthy, E.T. Hvannberg and M.S., 1998), Prism (Prism), vfAnalyst (vfAnalyst), coarse-grain parallelizer (S. Rul, H. Vandieren-donck, and K. De Boschere, 2008), Alchemist (X. Zhang, A. Navabi, and S. Jagannathan, 2009), dependence profiler (Wu, D. Dasand P., 2010), SD3 (M. Kim, H. Kim, and C.-K. Luk, 2010), Dynamic program analysis (Kim, Minjang, 2011), Kremlin (M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanan, K. Bala, and L. P. Chew, 2007), S2P (Aditi Athavale, Priti Ranadive, M. N. Babu, Prasad Pawar, Sudhakar Sah, Vinay G. Vaidya and Chaitanya Rajguru, 2011) are some of the prominent semi-automatic tools. This paper presents a new semiautomatic tool called EasyPar. EasyPar is a unique semiautomatic parallelization tool because it provides parallelization support at the time of program development unlike most of the parallelization tools, which works on complete program. Analysis of program at the time of development requires two features, first, the offline analysis (storing program information in some file) and second, the demand driven analysis (dependency analysis on select part of code or select parameters). To support these features we have used a database that stores program information in a structured way that can be fetched using database queries. Since, the dependency analysis requires random data fetches about the program, we have used hybrid analysis algorithms that uses database queries combined with the java program. This technique allows us to perform data dependency analysis in faster way. We have compared feature of our tool against other similar tools and we have also presented the performance of our tool against standard benchmark programs.

RELATED WORK

Static analysis of a program for identification of parallelization is an active research area. Every year, we see new tools proposed by researchers or commercial tools released by vendors. Relational databases have already been used for program analysis in the past. However, to the best of our knowledge, this is the first attempt to develop the dependency analysis algorithms and parallelization analysis of a sequential program using database queries. Adam (Dutko, Adam M., 2011) has proposed a tool that converts the program first to a XML and then to a relational database. However, the purpose of this tool is to provide a platform to the programmers so that they can write queries to generate vital
Related Content

Maturing Requirements Engineering Process Maturity Models
www.igi-global.com/chapter/maturing-requirements-engineering-process-maturity/28404?camid=4v1a

Survey of Motivation to Work Among Non-Regular Employees in the Food Services Companies: A Statistical Analysis Considering Length of Employment
www.igi-global.com/article/survey-of-motivation-to-work-among-non-regular-employees-in-the-food-services-companies/213954?camid=4v1a
A Proposed Pragmatic Software Development Process Model
www.igi-global.com/chapter/a-proposed-pragmatic-software-development-process-model/117925?camid=4v1a

Value and Intention Based Information Systems Engineering
www.igi-global.com/chapter/value-intention-based-information-systems/23412?camid=4v1a