P-Threads strikes back

OpenMP is Not as Useful as It Appers

Pedro Bruel
Software Systems Laboratory (LSS)
Institute of Mathematics and Statistics
University of São Paulo
Rua do Matão, 1010
São Paulo, Brazil 05508-090
phrb@ime.usp.br

Paulo Meirelles
FLOSS Competence Center (CCSL)
Institute of Mathematics and Statistics
University of São Paulo
Rua do Matão, 1010
São Paulo, Brazil 05508-090
paulormm@ime.usp.br

Alfredo Goldman
Software Systems Laboratory (LSS)
Institute of Mathematics and Statistics
University of São Paulo
Rua do Matão, 1010
São Paulo, Brazil 05508-090
gold@ime.usp.br

Abstract

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.

CCS Concepts • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Keywords ACM proceedings, LaTeX, text tagging

ACM Reference format:
Pedro Bruel, Paulo Meirelles, and Alfredo Goldman. 2017. P-Threads strikes back. In *Proceedings of 4th International Workshop on Software Engineering for Parallel Systems, Vancouver, Canada, October 2017 (SEPS 2017),* 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

...

In this context, the approach of using Compilation Directives to guide the compiler in the parallelization process has gained popularity. These directives are implemented using the compiler's pre-processing directives and are utilized as annotations that provide tips about the sequential code. Among the tools and extensions that use compilation directives are the `OpenMP` API [4] [3] used for writing programs for multi-core architectures, and the `OpenACC` programming standard [7], used for writing programs for heterogeneous CPU/GPU architectures. The `OpenMP 4.0` specification supports offloading capabilities [8] such as `OpenACC` and `AMD HSA` [1]. These tools aim to make easier the task of designing and implementing efficient and provably correct parallel algorithms. However, the abstraction layers built by the extensions also have the potential to conceal architectural details and to generate new parallel programming challenges.

When comparing `OpenMP` to other available tools for developing multi-threaded applications such as the `pthreads` interface, the API can be considered almost costless. However, when writing more complex applications this may not be true. The design and implementation of such complex applications require more knowledge about platforms and execution models than what can be shown by tutorials, that present high level and trivial examples. These tutorials try to convince the user that programming with directives is simple and easy to do, which is not always true. For instance, to achieve good performance in some cases it is necessary to describe many restrictions on annotations [2] [6].

Krawezik and Cappello [5] evidenced this difficulty by comparing MPI and three OpenMP versions of benchmarks. The most popular use of `OpenMP` directives is at the loop level. The programmer discovers potentially parallel loops and annotates the code with directives starting and closing parallel regions. Good results were obtained only when SPMD was implemented on OpenMP, and achieving good performance required a significant programming effort [5].

In this work we have gathered data from assignments made by graduate students for the *Introduction to Parallel and Distributed Computing* course. Among other exercises, the students were asked to find examples of `OpenMP` code in tutorials and official manuals that failed under special conditions. These conditions could be created by, for example, placing barriers or forcing racing conditions by exposing inaccurate thread management.

The rest of the paper is organized as follows. The following section discusses related work...

2 Background

TO-DO ...

3 Related Work

TO-DO ...

4 Research Design

The motivation to write this paper comes from experiences on teaching. During the classes of Distributed Systems and Parallel Programming, a graduate course at University of São Paulo...

RQ1: *Which API is easier to learn and use for beginning students? ...*

RQ2: *Which API is better to improve the performance for beginning students? ...*

RQ3: *Which API is easier to learn and use for advanced students? ...*

RQ4: *Which API is better to improve the performance for advanced students? ...*

5 Data Analysis

TO-DO ...

6 Lessons Learned

TO-DO ...

7 Conclusions

TO-DO ...

Acknowledgments

The authors would like to thank Students who participated in this exercise ...

References

[1] AMD. 2015. Heterogeneous Computing. (August 2015). http://developer.amd.com/resources/heterogeneous-computing/

[2] E. Ayguade, N. Copty, A. Duran, J. Hoeflinger, Yuan Lin, F. Massaioli, X. Teruel, P. Unnikrishnan, and Guansong Zhang. 2009. The Design of OpenMP Tasks. *Parallel and Distributed Systems, IEEE Transactions on* 20, 3 (March 2009), 404–418. https://doi.org/10.1109/TPDS.2008.105

[3] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. 2007. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation).* The MIT Press.

[4] L. Dagum and R. Menon. 1998. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering* 5, 1 (1998), 46–55. https://doi.org/10.1109/99.660313

[5] Géraud Krawezik and Franck Cappello. 2006. Performance comparison of MPI and OpenMP on shared memory multiprocessors. *Concurrency and Computation: Practice and Experience* 18, 1 (2006), 29–61. https://doi.org/10.1002/cpe.905

[6] Timothy G Mattson. 2003. How good is OpenMP. *Scientific Programming* 11, 2 (2003), 81–93.

[7] OpenACC. 2013. OpenACC Application Programming Interface. Version 2.0. (june 2013). http://www.openacc.org/sites/default/files/OpenACC.2.0a_1.pdf

[8] OpenMP-ARB. 2013. *OpenMP Application Program Interface Version 4.0.* Technical Report. OpenMP Architecture Review Board (ARB). http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf Version 4.0.