Parallel and Distributed Autotuning for High-Performance Computing

Pedro Bruel

Text submitted
to the
Institute de Mathematics and Statistics
of the
University of São Paulo
for the
Qualifying examination
for the
Doctoral Degree in
Computer Science

Advisor: Professor Alfredo Goldman Vel Lejbman

During the development of this work, the author was supported by CNPq №XXXX, Hewlett-Packard Enterprise and CAPES

São Paulo, March 7, 2017

Parallel and Distributed Autotuning for High-Performance Computing

This is the original version
submitted for evaluation
by Pedro Bruel

Abstract

BRUEL, P. **Parallel and Distributed Autotuning for High-Performance Computing**. 2017. xxx f. Qualifying Examination (PhD) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2017.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós- Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação.

Keywords: keyword1, keyword2, keyword3.

Contents

List of Symbols

ω	Frequência angular
ψ	Função de análise *wavelet*
Ψ	Transformada de Fourier de ψ

List of Abbreviations

CFT	Transformada contínua de Fourier (*Continuous Fourier Transform*)
DFT	Transformada discreta de Fourier (*Discrete Fourier Transform*)
EIIP	Potencial de interação elétron-íon (*Electron-Ion Interaction Potentials*)
STFT	Tranformada de Fourier de tempo reduzido (*Short-Time Fourier Transform*)

List of Figures

List of Tables

Chapter 1

Introduction

A: Bilmes *et al.* (1997)

1.1 Objectives and Expected Contributions

1.2 Text Structure

Chapter 2

Autonomous Search

2.1 Algorithm Selection and Autonomous Solvers

2.2 Off-line Configuration

2.2.1 Evolutionary Computing

2.2.2 Stochastic Local Search

2.2.3 Machine Learning

2.3 On-line Control

2.3.1 Adaptive Parameter Configuration

2.3.2 Credit Assignment

2.3.3 Reinforcement Learning

Chapter 3

Autotuning

3.1 OpenTuner

3.1.1 Context

3.1.2 Software Architecture

3.2 Search Techniques

3.2.1 Numerical Methods

3.2.2 Evolutionary Computation

3.2.3 Stochastic Local Search

3.2.4 Machine Learning

3.3 Benchmarks

3.3.1 Solvers of NP-Complete Problems

3.3.2 Algorithm Selection and Configuration

3.3.3 Compiler Configuration

3.3.4 Measurement Time

Chapter 4

Case Studies

Chapter 5

Parallel and Distributed Autotuning in the Julia Language

Chapter 6

Summary and Discussion

6.1 Objective and Future Work

6.2 Schedule

Bibliography

Bilmes *et al.***(1997)** Jeff Bilmes, Krste Asanovic, Chee-Whye Chin e Jim Demmel. Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology. Em *Proceedings of International Conference on Supercomputing, Vienna, Austria*. Citado na pág. 1

Index