Study and Optimization of a Finite Volume Application

Brito, Rui
PG22781
Department of Informatics
University of Minho
ruibrito666@gmail.com

Alves, José
PG22765
Department of Informatics
University of Minho
zealves.080@gmail.com

1 Introduction

High-performance computing has been a fundamental process to science, in order to make viable certain softwares that require massive amounts of calculation. The use of several optimization techniques has help decrease the execution time of different algorithm, providing results and simulations in a viable time for its uses.

This document will describe a study about improving performance of *conv-diff*. This application calculate de heat of a material while spreading though an area. In this study we approach different ways to improve the solution.

This extended abstract presents the aplication domain, explaining its uses and objective. Knowing its domain and application's profile, a foundation is created to develop different optimizations. From here an optimized sequencial version was developed, as well as an OpenMP shared-memory version. The ongoing and future work and a brief conclusion will be presented in the last sections of this extended abstract.

2 Case Domain

The application analyzed for this study is *conv-diff(Convexion-Diffusion)*.This application calculates the heat in the spread of a material, through the course of time. To achieve this end, the surface is represented as a mesh. Each cell of the mesh can be calculate with the velocity of the flux in each of its edges, its edge's normal, and its ϕ. The input and the output of the program is a XML file, that can both be converted to msh format for graphic visualization with gmsh.

Being a finite volumn method each cell only is only dependant of its neighbours in the mesh. This low dependencies between cells, favors parallelization, thanks to the data locality.

3 Profiling

The program consists in two major parts, a cycle that computes the ϕ for each cell, using the functions *makeResidual*, which calls the function *make-Flux*, and an FVLib function, *LUFactorize*, which validates the results of the above cycle. Both the calculated mesh and the real mesh, computed by the LUFactorize, are presented in the output files, together with the error between them.

After analyzing the application, we conclude that the algorithm has a high workload in the *LUFactorize* function,

4 Sequencial Optimization

5 Shared Memory Parallel Optimization(OpenMP)

6 On-going and Future Work

After implementing both the sequencial version and the OpenMP version, a naïve implementation in CUDA was started. This version aims to take advantage of the graphics board performance using its vector units to diminish the compute time. For this a restructuring of the code is necessary removing most of the memory accesses while maintaining an abstraction of the system. After resolving this issue and other minor ones, we expect to achieve a boost in performance.

In future work, it is expected to develop a optimized CUDA version as well as a OpenMPI. A hybrid version using both CPU and GPU could also be a future implementation to have gains in performance. It is also expected to optimize several areas of code, questioning some decisions like using double-precision versus single-precision.

7 Conclusion