

Study and Optimization of a Finite Volume Application

Brito, Rui
PG22781

Department of Informatics
University of Minho
rui Brito666@gmail.com

Alves, José
PG22765

Department of Informatics
University of Minho
zealves.080@gmail.com

1 Introduction

Software optimization has been a fundamental process to science, in order to make viable certain softwares that require massive amounts of calculation. The use of several techniques has helped decrease the execution time of different algorithms providing results and simulations in a viable time for its uses. Thanks to this techniques of optimization, research time is also pushed to a new maximum, decreasing the time between its possible dependencies. This extended abstract presents the application domain, explaining its uses and objective. Knowing its domain and application's profile, a foundation is created to develop different optimizations: optimized sequential version; OpenMP shared memory version. In the last sections of this extended abstract, the ongoing and future work and a brief conclusion will be presented.

2 Case Domain

The application analyzed for this study is *convdiff(Convection-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 calculated with the velocity of the flux in each of its edges, its edge's normal, and its f_i . 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 volume 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 f_i for each cell, using the functions *makeResidual*, which calls the function *makeFlux*, 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 Sequential Optimization

5 Shared Memory Parallel Optimization(OpenMP)

6 On-going and Future Work

After implementing both the sequential version and the OpenMP version, a vanilla 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 an optimized CUDA version as well as an 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