Shallow Water Equations - Optimization Strategy

Gaurav Kukreja, Evangelos Drossos

January 6, 2014

Abstract

Shallow Water Equations (SWE) are typically used for tsunami or dam-break simulations. The numerical discretization techniques and resulting computational challenges can be applied to other hyperbolic PDE's as well. Examples are Earthquake simulations, weather and climate simulations or aerodynamics. The task of the project is to optimize SWE code.

Our objective for the project is to optimize the code by using Compiler intrinsics to improve vectorization in the code. Further, we will parallelize implementation using OpenMP and MPI. We will also try to improve Cache Performance by using Cache Blocking and compare results by using different strategies for the same. Detail description of our approach is discussed in this report.

1 Compiler Intrinsics to Optimize Solvers

The code under *src/solvers* contain the implementation of the stencil operations. This code can be optimized by using Compiler Intrinsics for optimizing vectorization. Multiple approaches for solvers are implemented. We will focus on optimizing the F-wave solver. All functions are inline. The Intel Compiler is capable of vectorizing the code automatically. However, the performance can be improved by using Compiler Intrinsics.

If time permits, we will try to optimize the Augmented Riemann Solvers as well.

2 Cache Blocking

We will implement various Cache Blocking Optimizations using existing SWE_Blocks and implementing blocking within each SWE_Block. We will to compare our implementations and produce a report based on our understanding of the bottleneck issues.

If time permits, we will try to extend the analysis using Scalasca to better identify the bottlenecks.

3 Parallelization using OpenMP and MPI

We will try to find the best strategy to parallelize our implementation using hybrid implementation with OpenMP and MPI.

4 Performance Analysis using Scalasca

If time permits, we will try to manually instrument the SWE code. We will present our understanding about the bottleneck issues and demonstrate how our implementation resolves these issues.

5 Distribution of Tasks

We will start our implementation with tasks of vectorization and hybrid parallelization with OpenMP and MPI. Gaurav will focus on parallelization while Evangelos will work in parallel on improving vectorization using Compiler Intrinsics.

We will then collaborate on identifying bottlenecks with Cache, and solve them by implementing Cache Blocking strategies. We hope to be able to complete these tasks by 18 January 2013.

We will work on fixing bugs if any, and producing results to demonstrate the merits of our implementation and identify future scope.

Eventually, we will try to instrument the code using Scalasca and extend our analysis to support our understanding of the problem and demonstrate how our implementation mitigates the problems.