Parallelization of Tsunami Simulation based on CPU, GPU and FPGAs

Fumiya Kono, Naohito Nakasato, Kensaku Hayashi, Alexander Vazhenin, Stanislav Sedukhin (University of Aizu, Japan)
Kohei Nagasu, Kentaro Sano (Tohoku University, Japan), Vasily Titov (NOAA Center for Tsunami Research, USA)

Evaluation and Results

For Code1, when block size is 32x32, we obtained the best performance.

Computing time for 300 steps (Unit: sec.)

<table>
<thead>
<tr>
<th>Device</th>
<th>Code1</th>
<th>Code2</th>
<th>Code3</th>
<th>Code4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIC</td>
<td>MIC</td>
<td>MIC</td>
<td>MIC</td>
<td>MIC</td>
</tr>
<tr>
<td>1</td>
<td>79.8</td>
<td>81.7</td>
<td>96.7</td>
<td>37.8</td>
</tr>
<tr>
<td>4</td>
<td>280</td>
<td>280</td>
<td>280</td>
<td>280</td>
</tr>
<tr>
<td>Toga</td>
<td>122.9</td>
<td>131.7</td>
<td>131.7</td>
<td>131.7</td>
</tr>
<tr>
<td>Toga</td>
<td>371.5</td>
<td>371.5</td>
<td>371.5</td>
<td>371.5</td>
</tr>
</tbody>
</table>

Results of benchmarking on Multi-CPU systems and Many Integrated Core Architecture (OpenMP)

MIC obtained lower performance than multi-CPU systems in any cases due to its memory access speed.

• On multi-CPU systems (Intel Xeon), speedup is proportional to the number of cores.
• On MIC, speedup depends on the implementations.

Code1 did not speedup due to parallelization.
Code2 obtained 20x speedup at maximum and computing times are also shorter.
Matrix transposition improved the cache hit ratio for accessing the 2-D array.
Code3 shows the best speedup (100x) against serial execution.

The number of data copy to the temporary array is minimized in this implementation.

Computation of Code4 is the fastest (about 1 minute) of 4 codes (with 32x32 block).
This is expected greater performance on GPU.

Implementation of MOST on FPGAs

FPGAs (Field-Programmable Gate Arrays) has potential to achieve high-performance and power-efficient computation. We are designing FPGA-based hardware accelerator of MOST which depends on
• temporal parallelism by pipelining to increase the number of operations per cycle
• spatial parallelism by deploying more pipelines in parallel

Iterative stencil computation for stream computing architecture

The data-flow graph of the stencil computation subroutine (for x-direction)

Implementation of the cascaded SPEs with DES-NET board

• Altera Stratix V 5SGX8 FPGA
• Two DDR3 PC12800 SO-RAMs, PCI-Express (PCIe) 3.0 interface
• A peak bandwidth of 25.6 GB/s
• The platform has the three clock domains (50 MHz for PCIe interface, 200 MHz for DDR3 controllers, and 150 MHz for most accelerator)

Computing time at 150MHz for 300 steps with the 256x1x2879 grid: 0.644 second

• 2-stage pipeline
• 154 word inputs
• 5 word outputs

Conslt only of computation nodes and delay nodes.

The stream-processing based hardware designed for MOST

Parallelization of Tsunami Simulation based on CPU, GPU and FPGAs

Results of benchmarking on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Objective of this study and Solution

We accelerate the performance of the tsunami simulation, MOST, to compute the elapsed time of a model faster than real time. (It is useful for tsunami prediction and evacuation.)

We parallelize MOST algorithm written in C++.
We developed three optimized algorithms for parallelization.
We parallelize our optimized codes using OpenMP, OpenACC and OpenCL.
We benchmarked our parallelized codes on
• Intel Xeon multicore CPU
• Intel Xeon Phi (Many Integrated Core Architecture, MIC)
• NVIDIA Tesla K20 (GPU)
• AMD Radeon HD7970 (GPU)

Implementation on FPGAs (as a reference) is also presented.

-•- Results of benchmarking on CPU and GPU (OpenCL)
(Computing time of 32x32 block) is shown which achieved the best performance.

OpenCL code on AMD GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.

Evaluation and Results of MOST on Multi-CPU systems and MIC

CPU: Xeon X5670, X7560, X7550, X5680, X5675, X5680 MIC: Xeon, Xeon Phi OpenMP, OpenACC, OpenCL

Comparison of MOST and GPU

OpenCL code on GPU is the fastest for all implementations.
• In this implementation, 1x1 block is the optimal value for fast computation.
• Larger block size gives performance drop.
• Calculation on AMD GPU can be finished within approximately 3 seconds.