Real-Time Tsunami Simulation on Multi-node GPU Cluster

Marlon Arce Acuña[1)], Takayuki Aoki[2)]

GSIC, Tokyo Institute of Technology (2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550)

[1)] marlon.arce@sim.gsic.titech.ac.jp, [2)] taoki@gsic.titech.ac.jp

Abstract : With the introduction of GPU, a new revolution has been opened in high performance computing. For accurate early warning for Tsunami disasters, the shallow water equations must be solved in real-time. Even single GPU enables 62-times faster calculation than 1 CPU core and the same domain decomposition is available for multi-GPU computing, where the communications between GPUs are hidden by overlapping with the computation. For the very large-size domain of 4096x8192 mesh with 90m resolution, the GPU Tsunami simulation finishes within 3 minutes in the case of 8 GPUs. The CPU and GPU scalability is compared. Excellent scalability has been achieved on TSUBAME GPU cluster.

1 Introduction

A Tsunami is among the most dangerous natural disasters a country can be exposed to due to its force and the threat to the people living in the coasts. To forecast a Tsunami requires real time calculation to be able to give an early evacuation warning as soon as possible. This requires high performance computing which can be found in the GPU, to reduce the long computation time.

This is why we embark using the new technology provided by nVIDIA: CUDA [1] to increase the acceleration to solve the Shallow Water Equation. Additionally solving it in a highly parallelized structure gives an outstanding improvement in the run time compared with that consumed by previous calculations with regular CPU hardware and single GPU. Pushing the envelope of acceleration Multi-GPU was used and the problem was solved on the newly introduced GPU Cluster at the Tokyo Institute of Technology Super Computer, Tsubame.

2 Background

The Shallow Water Equation [2] can be written in the following form:

$$U_t + F(U)_x + G(U)_y = S(U) \tag{1}$$

with:

$$U = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix} \quad G(U) = \begin{bmatrix} hv \\ hvu \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix} \quad F(U) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix} \tag{2}$$

where h is the water height, u and v are its velocities in the x and y direction respectively, g the gravity and S a source term.

3 Numerical Methods

3.1 Method of Characteristics

Since the Shallow Water Equation is a hyperbolic equation it can be rewritten in its characteristic form and then use its characteristic curves to find its solution. A matrix L is used to diagonalize A, which is found from the eigenvalues λ of A, from its eigenvectors and the Riemann invariants. The point value i can be estimated as two characteristics along C^+ and C^-. If the variables at the upstream departure points of $C^{\pm}$ are denoted by Γ_i^{n+1} and u_i^{n+1}, where $\Gamma \equiv \sqrt{gh}$, then they are given by:

$$\Gamma_i^{n+1} = \frac{1}{2}\left\{\Gamma^+ + \Gamma^- + \frac{1}{2}(u^+ - u^-)\right\} \quad u_i^{n+1} = \frac{1}{2}\left\{u^+ + u^- + 2(\Gamma^+ - \Gamma^-)\right\} \tag{3}$$

To achieve high performance a directional splitting is introduced to keep the calculation of the previously mentioned methods as less intensive as possible.

3.2 Conservative Semi Lagrangian-IDO Method

To solve the Shallow Water Equation the CIP Conservative Semi-Lagrangian IDO Method [3],[4] is used. Two important reasons to work with this method are that it provides a scheme with high accuracy as well as mass and momentum conservation. In this method a point value in the grid and an integrated value between the two adjacent points are considered to find the result.

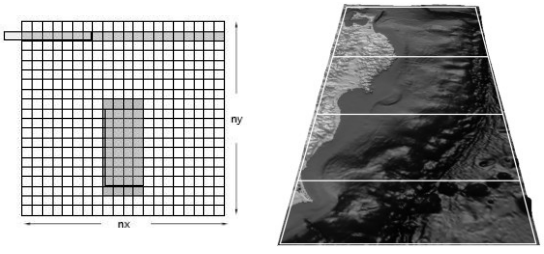
4 GPU Implementation

4.1 Single GPU implementation

Inside the GPU two kernels are run sequentially to solve the directional splitting. First, the equations are calculated in the x-direction: For this purpose the data is loaded from global memory to shared memory in a structured similar to that shown in Figure 1a. A line of length nx and width one from the full domain is break down in four pieces and then two blocks will compute them in two steps. For the y direction the shared memory model changes. In this case a square block instead of a line is used and each block will compute twice its width hence filling all the computational domain. From previous results[5] using a single GPU, an outstanding improvement of 62 times was found when compared the GPU and CPU runtimes.

4.2 Multi-GPU Computing

To solve the Shallow Water Equation in parallel using GPU, the domain is decomposed, distributed along CPUs and transferred to its GPU. MPI is be used to communicate between CPUs, and the CUDA APIs to share data between each GPU and its associated CPU, as shown in Figure 1b.



(a) Shared Memory (b) Multi-GPU

Fig. 1 Shared Memory and Multi-GPU Models

4.3 Transfer and Communication time hiding by overlap

To transfer the data back and forth between CPU and GPU two models were tested. One in which the communications were done synchronously with the calculation and other in which they were asynchronous, as shown in Figure 2.

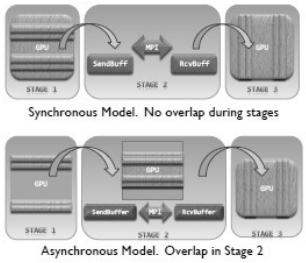


Fig. 2 CPU and GPU communication

In the asynchronous case the domain inner-part calculation is overlapped with the data transfer, therefore when the calculation time is longer than that of the communication an even better performance is achieved by hiding the transfer. Improvements of between 5 to 10% were observed for the asynchronous over the synchronous model.

5 Results

To evaluate our scheme and the acceleration using GPU, two test bathymetries were introduced with an initial wave resembling the properties of a Tsunami (Fig. 3 and 1b). The first bathymetry used was solved for four grid sizes: 512x512, 1024x1024, 2048x2048 and 4096x4096. The second bathymetry is real data for Japan`s Tohoku region, adapted from NASA's SRTM Mission[6] and NOAA's ETOPO[7] with a 90m resolution and grid size 4096x8192, covering a vast 370x735km area. First a single machine was chosen to run the simulation and then on Tsubame's GPU Cluster using multi-node for high performance.

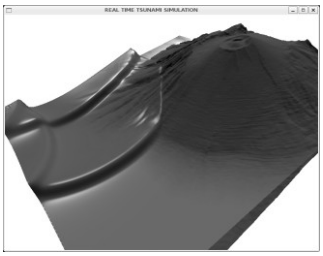


Fig. 3 Test Bathymetry with ongoing Tsunami Simulation

The specifications of the single machine in which the computation was done are: CPU: Intel Core i7 CPU920@2.67GHz; 8GB Ram memory; GPU: GTX295 x 4. For Tsubame Super Computer: Sun Fire X4600 server (8AMD DualCore); 32GB Memory; GPU: Tesla S1070 (2GPU/node) and Infiniband Network Voltaire ISR 9288x8.

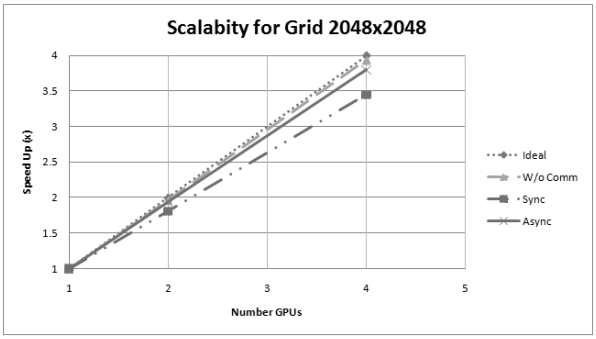


Fig. 4 Scalability for 2048x2048 Grid, single machine

The results for the GPU scalability are shown in Figure 4 for the 2048x2048 grid, this shows the runtime rates between the multi GPU and the single GPU, when 2 and 4 GPUs were used, the dot line represents the ideal case. The bigger the grid size the better the results because the computing time increased giving chance to hide the communication in the asynchronous model.

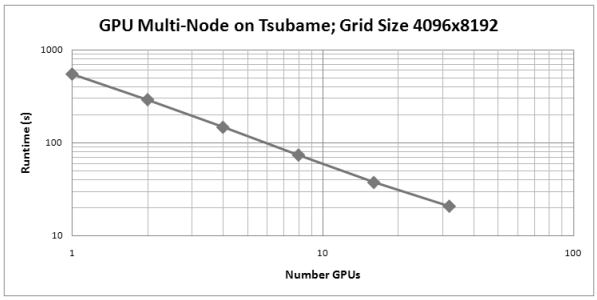


Fig. 5 Runtimes for Japan Bathymetry on Tsubame

In the case of the topography for Japan Tohoku region, the calculation was done on Tsubame using multi-node taking advantage of its high speed network, each node had 2 GPUs. The results can be seen in Figure 5 where the run time per number of GPUs is shown. Figure 6 shows a comparison of the CPU and GPU scalability using the single CPU as based time. GPUs presented a high scalability with efficiency around 90 and 98% moreover even though the CPU scalability is not low the number of devices required to match the GPU performance is overwhelming e.g. around 1024 CPUs are required to match 16 GPUs performance.

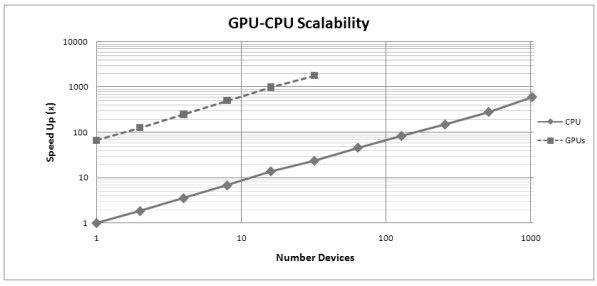


Fig. 6 GPU-CPU Scalability (1 CPU Core based)

6 Conclusion

Test bathymetries were used to study the acceleration for a Tsunami simulation provided by GPUs. Based on the outstanding speed up performance results on a single machine and on Tsubame and the significant less number of GPU devices required for high performance we see the parallel GPU as a promising and revolutionizing technology to develop a warning tool for a High Performance Real-Time Tsunami Simulation

References

[1] CUDA Zone http://www.nvidia.com

[2] E.F. Toro. Shock-capturing methods for free-surface shallow flows. John Wisley&Sons Ltd 2001, London

[3] T. Aoki, Comp. Phys. Comm., Vol.102, No.1-3, 132-146 (1997)

[4] Y. Imai, T. Aoki and K. Takizawa, J. Comp. Phys., Vol. 227, Issue 4, 2263-2285 (2008)

[5] GPU driven acceleration for solving the Shallow Water Equation. Marlon Arce Acuña, Takayuki Aoki. JSMES 21st Conference, November 2008

[6] NASA SRTM www2.jpl.nasa.gov/srtm/index.html

[7] NOAA www.ngdc.noaa.gov/mgg/global/global.html