A MULTI-GPU IMPLEMENTATION OF A 2D SHALLOW WATER EQUATION SOLVER WITH VARIABLE RESOLUTION

Massimiliano Turchetto\(^1\), Renato Vacondio\(^1\) and Alessandro Dal Palú\(^2\)

\(^1\) Department of Engineering and Architecture, University of Parma

\(^2\) Department of Mathematical Physical and Computer Science, University of Parma

Key words: Shallow Water Equations (SWEs), Multi-GPU, CUDA, NVIDIA, MPI

Abstract.
In this paper a Shallow Water Equations (SWEs) solver which can exploit the computational capabilities of multiple Graphics Processing Units (GPUs) systems is presented. This implementation originates from the single-GPU PARFLOOD solver, which has demonstrated to be fast and robust in previous publications and it is equipped with Block Uniform Quad-tree grids to introduce variable resolution. The code has been parallelized using Message Passing Interface (MPI) communication protocol to exploit the computational capabilities of memory distributed High Performance Computing clusters. Weak and strong scalability tests have been performed and they show that an efficiency of more than 90% can be achieved in simulation with up to 10 Tesla P100 GPUs.

1 Introduction

2D Shallow Water Equations has been extensively used to simulate flood propagation. Among other numerical techniques, explicit Finite Volume (FV) schemes have been proven to be particularly robust and accurate. Recently, different GPU-accelerated FV schemes have been proposed in literature to overcome this issue. However, the use of a single GPU is not sufficient for several practical applications which require several million of cells. To overcome this major limitation, we decided to exploit the computational power of modern memory distributed HPC clusters, equipped with several GPUs. The single GPU code PARFLOOD (see [1] for details) has been modified by introducing a spatial decomposition technique: the whole domain has been divided into subdomains, where the computation is performed by means of a different GPU of the multiGPU cluster. The communication between different subdomains is performed by means of the Message Passing Interface (MPI) protocol and point-to-point communications are executed in a non-blocking fashion, in order to overlap the computation of CUDA kernels and data transfer between different MPI processes. In particular, as masking strategy, we anticipate the computation of inner regions of a subdomain that do not rely on communication while data critical for bounds of the subdomain is being received. At the end of each timestep a global blocking MPI communication is needed to compute and distribute the computational timestep, which
Massimiliano Turchetto, Renato Vacondio and Alessandro Dal Palú

Figure 1: (a) Strong and Weak Scalability tests (b) domain decomposition for the strong scalability

is unique for all processes. The time consumed by this MPI communication is the critical point of the present multiGPU implementation.

2 weak and strong scaling tests

The GPU parallelization is performed by mapping each thread to a computational cell, which discretizes the physical domain, and each block to $16 \times 16$ cells. The spatial domain decomposition of the computational mesh has been done in 1D. The computation has been performed using the University of Parma HPC cluster, which has two hybrid nodes, equipped with 5 Tesla P100 GPUs each.

The weak scaling test has been performed by defining each subdomain with $125 \times 32$ blocks (which corresponds to $10^6$ cells) of a Cartesian Grid. Results of the the weak scaling tests are reported in Figure 1-a. For 2 GPUs, the efficiency is equal to 95%, it drops to 92% for 4 GPUs and it remains roughly constant up to 10 GPUs. Our profiling shows the lack of significant overhead in non-blocking MPI communications and their complete masking through the overlapped computation of inner blocks. The degradation of efficiency is caused by the unavoidable synchronization time of the global MPI communication of the computational timestep.

The strong scaling test is performed for a Circular Dam Break test, containing a total of 24000 blocks ($6.1 \times 10^6$ cells) and a Block Uniform Quad-Tree grid is adopted [1]. The domain decomposition is performed in 1D (Figure 1-b) allocating the same number of blocks on each subdomain. For the strong scalability test the efficiency dropdown is around 10-12% when doubling the number of GPUs involved in the simulation (Figure 1-a). Here the efficiency reduction is caused by two factors: (i) global synchronisation for the timestep such as in the weak scaling and, more importantly, (ii) loss of efficiency of CUDA kernels that perform the calculations for blocks on the border of the subdomain.

REFERENCES