GUIDED RESEARCH

Shallow Water Equation simulation with Sparse Grid Combination Technique

Author:
Jeeta Ann Chacko

Examiner:
Prof. Dr. Hans-Joachim Bungartz
Advisors:
Ao Mo-Hellenbrand

April 21, 2017

Fakultät für Informatik
Technische Universität München

Abstract

Scientific applications are affected by the curse of dimensionality. Sparse grids can be used to reduce grid points drastically but they are difficult to parallelize. The sparse grid combination technique can be used to combine lower resolution, anisotropic Cartesian grids to generate a sparse grid instead of directly using sparse grids. The shallow water equation is a hyperbolic PDE that is used to describe the changes in height and horizontal velocities of a fluid. The shallow water equation simulation is implemented using the sparse grid combination technique as a parallelization scheme in this project. The solution is compared with an existing tightly coupled implementation of shallow water equation simulation to evaluate the accuracy.

Index Terms

Shallow Water Equations, Sparse Grids, Sparse Grid Combination Technique, Hyperbolic PDE.

I. Introduction

Scientific applications work on large amounts of data. The usual parallelization technique using domain decomposition based on number of available resources requires a lot of computation since the number of grid points would be enormous. The curse of dimensionality, degree of freedom increases exponentially with the number of dimensions, is a big challenge while solving high dimension PDEs. Sparse grids [1], which have a reduced number of grid points than a full grid, can be used to reduce the number of grid points involved in computations drastically. But directly using sparse grids is complex and the parallelization is difficult.

A different approach to parallelization is the Sparse Grid Combination Technique (SGCT) [2] where multiple lower resolution, anisotropic Cartesian grids (henceforth referred to as component grids) are combined to give a sparse grid which has less grid points than the regular Cartesian grid. The component grids also have less grid points than the regular Cartesian grid. Equations over such component grids can be solved completely independently. Then they can be combined to give a sparse grid solution that would approximate the solution of the equation over a regular Cartesian grid. So, an approximately accurate solution is obtained for a scientific problem with far less computation.

The shallow water equations(SWE) are a set of hyperbolic partial differential equations that are used to describe the behavior of fluids over time based on some initial condition. It defines the height and horizontal velocities of a fluid over time. There is an existing teaching code [3] that implements the SWE over a regular Cartesian grid. This implementation uses classical domain decomposition methods. In this implementation, frequent communication is necessary with the neighboring grids. So every grid requires a knowledge about its neighbors throughout the computation. Therefore this implementation can be described as a tightly coupled parallelization scheme.

In the sparse grid combination technique, the component grids are completely independent. Computations can be carried out on the component grids without any communication with the other component grids. Thus it becomes an embarrassingly parallel program or it can be described as a loosely coupled parallelization scheme. The goal of this guided research was to apply the Sparse Grid Combination Technique on a hyperbolic PDE (Shallow Water Equations) and evaluate the error rate of the solution obtained with comparison to the solution obtained with an existing implementation using regular Cartesian grids. The existing implementation parallelizes based on simple domain decomposition strategy and ghost layer exchanges. Our implementation solves the same shallow water equation using the sparse grid combination technique.

This paper in divided into 5 major sections. Section 2 describes the shallow water equation and the existing implementation of the SWE simulation. Section 3 describes the concept of SGCT. Section 4 explains the implementation details of the sparse grid combination technique. Section 5 explains the general implementation details of this project. Section 6 presents the results and observations made from this implementation. Section 7 gives the conclusion and describes some future work possible for this project.

II. Shallow Water Equations

The Shallow Water Equations [4] are a set of hyperbolic partial differential equations that can be used to describe the behavior of fluids. It evaluates the change in height (h: water depth) and horizontal velocities (V_x and V_y) over time based on some initial condition. The initial condition would be, for example, the change on the ocean floor caused by an earthquake in case of a tsunami simulation. In the case of tsunami simulation, the uneven ocean floor is also considered (b: elevation of sea floor, bathymetry data) along with the height and velocity. Shallow water equations assume that the effects of vertical flows can be neglected. The changes to height and horizontal velocities are described as a set of PDEs generated based on conservation of mass (1) and conservation of momentum (2) and (3).

$$\frac{\partial h}{\partial t} + \frac{\partial (v_x h)}{\partial x} + \frac{\partial (v_y h)}{\partial y} = 0 \tag{1}$$

$$\frac{\partial (hv_x)}{\partial t} + \frac{\partial (hv_x v_x)}{\partial x} + \frac{\partial (hv_y v_x)}{\partial y} + \frac{1}{2} g \frac{\partial (h^2)}{\partial x} = -gh\frac{\partial b}{\partial x} \tag{2}$$

$$\frac{\partial (hv_y)}{\partial t} + \frac{\partial (hv_x v_y)}{\partial x} + \frac{\partial (hv_y v_y)}{\partial y} + \frac{1}{2} g \frac{\partial (h^2)}{\partial y} = -gh\frac{\partial b}{\partial y} \tag{3}$$

A. Existing Implementation

There is an existing teaching code [4] [3] that solves shallow water equations on a regular Cartesian grid. Each cell of the grid will have the unknowns h, hv_x, hv_y and b. An algorithm is used to calculate the numerical flux on each edge and update the cell values at every time step. This code is parallelized using the classical domain decomposition scheme. An additional layer of cells called Ghost cells exists. Using the ghost values multiple Cartesian grids can be connected to form the entire domain. So, for parallelization, the grids would be distributed among the available processes and each process would work on the given grid. Communication is necessary for exchanging ghost cell values. This implementation is visualized in Fig. 1. This implementation can be described as a tightly-coupled parallelization scheme because of two reasons: The ghost values need to be communicated at every time step and each of the distributed Cartesian grids should have knowledge about its neighbors throughout the computation. This implementation is henceforth referred to as the full grid implementation.

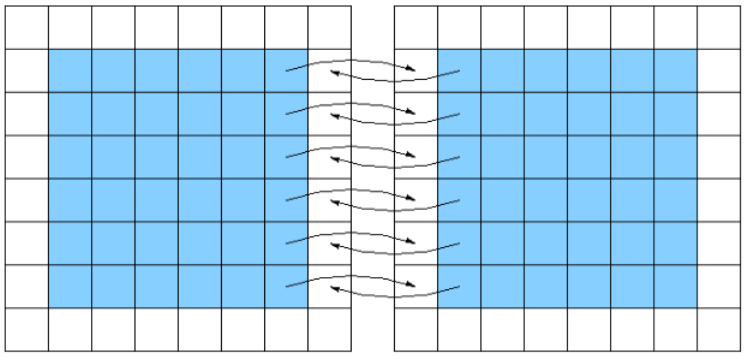

Figure 1. Cartesian Grids and Ghost celll exchange for Shallow Water Equation Simulation Code. Source: [4]

III. Sparse Grids and Sparse Grid Combination Technique

Sparse Grid [1] is a numerical discretization technique. For example, if you consider a 1-D unit line as shown in Fig. 2, it can be split into two by inserting a grid point in the middle. Each half can be further split into halves by adding grid points again in the middles.

Figure 2. Simple discretization example

Each split can be defined as a level. This splitting or discretization can be repeated for different dimensions to generate a sparse grid. The level of discretization for each dimension can be defined as a level vector and is used to describe a sparse grid. [l_x,l_y] is a level vector for a sparse grid of two dimensions which has 2^{l_x} points in x dimension and 2^{l_y} points in y dimension. Sparse grids of different levels are shown in Fig. 3. The left side shows the level vectors and the right side shows the sparse grids for each of those level vectors.

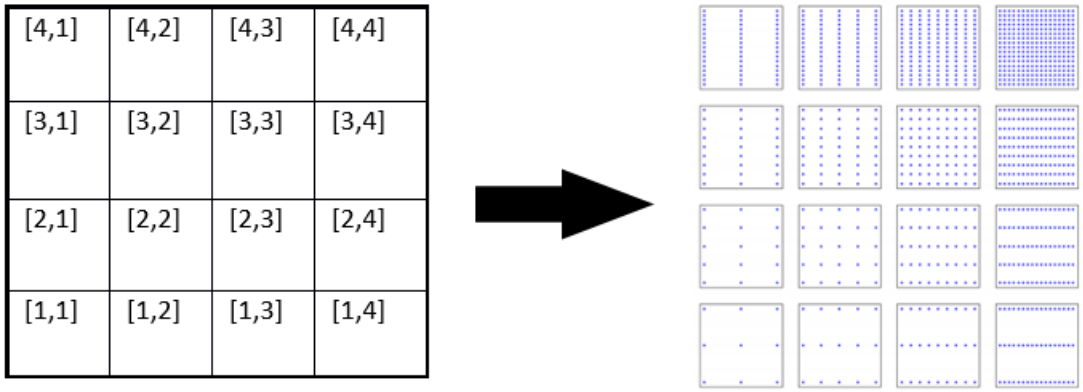

Figure 3. Sparse Grids of different levels

So, a sparse grid does not contain all the grid points in a full grid and thus reduces the curse of dimensionality. The missing points are approximated by interpolation. The advantage of using sparse grids is that the grid points are reduced drastically but a similar accuracy is obtained.

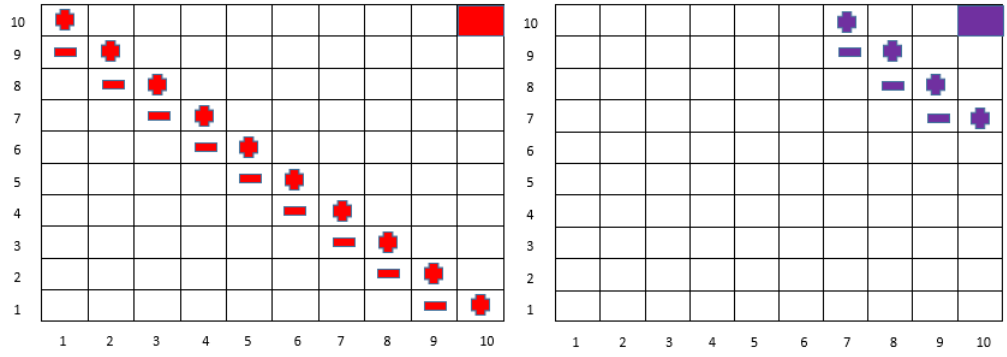

Figure 4. Sparse Grid Combination Technique (left) and Truncated Sparse Grid Combination Technique (right)

Sparse Grid Combination Technique [2] is the process of creating a sparse grid from a set of lower resolution component grids instead of creating it directly. We can define the minimum and maximum levels for choosing

the component grids. For example, to create a sparse grid of resolution 2^{10} x 2^{10} we can select the minimum and maximum level as (1,10) and then the component grids chosen would be as soon in Fig. 4 (left). This is the maximum number of component grids that can combine to give a sparse grid of resolution 2^{10} x 2^{10}. We can also generate the same sparse grid with lesser number of component grids. For example if the minimum and maximum level is (7,10) then the component grids chosen would be as soon in Fig. 4 (right). So when the minimum level is increased the number of component grids decreases but these component grids are finer and is expected to give a better accuracy upon combination. This is known as truncated sparse grid combination technique. In Section VI we compare the results of the normal SGCT and the truncated SGCT.

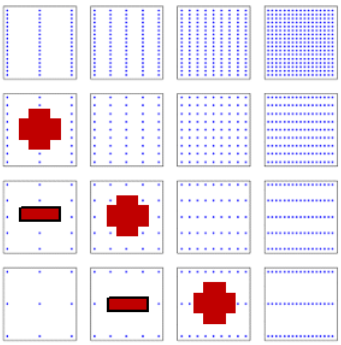

Figure 5. Sparse Grid Combination Technique

If we define a minimum level of 1 and maximum level of 3 then Fig. 5 shows the five component grids necessary to create a sparse grid [5]. First any numerical computations neccesary are solved on the five component grids. Then it is combined through addition or subtraction to generate an approximately accurate solution. The addition/subtraction factor of the component grids are defined as coefficients in the sparse grid combination technique.

Fig. 6 shows an example for the combination technique. First three component grids with level vector [3,1], [2,2] and [1,3] are combined. This results in some duplicate grid points. So, component grids with level vectors [2,1] and [1,2] grids are subtracted from the combination. This results in the generation of a sparse grid.

Figure 6. Sparse Grid Combination Technique

IV. SGCT Implementation

Shallow water equation simulation implements a simple finite volume model [4]. In a finite volume model, the focus is on a fixed volume in space known as a cell. The centroid of this cell is assigned a value which represents the average of the values over the whole volume. Then the changes to this cell centered values due to the neighboring cell center values are observed. Sparse grids are generally defined based on grid points which can be considered as more of a finite difference model where the focus is on a fixed point in space. Due to this difference in models the implementation of the sparse grid combination technique was the major challenge for this project.

If we consider the minimum and maximum levels as 1 and 3 then the component grids generated are shown in Fig. 7. We need to combine these 5 grids to form a sparse grid with resolution 2^3 x 2^3. Because each of the component grids have a different resolution we need to make it uniform before we can do the combination. Since the shallow water equation simulation uses a finite volume model, the values in each cell is assumed to be across the entire volume of the cell. So, a 2^2 x 2^2 grid can be represented as a 2^3 x 2^3 grid by flooding the equivalent

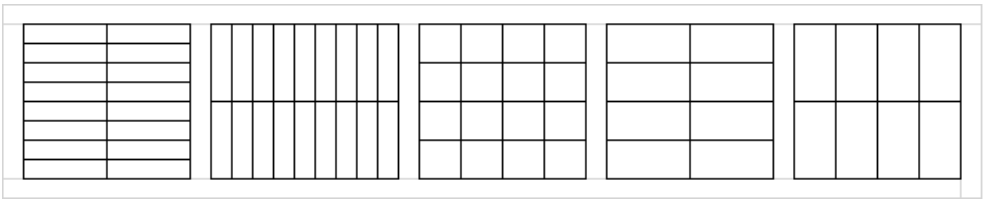

Figure 7. Component grids

volume in the higher resolution grid with values from the lower resolution grid. In this case, one cell of the lower resolution corresponds to four cells (two cells in each dimension) in the higher resolution. Similarly, for representing a 2^1 x 2^3 grid as a 2^3 x 2^3 grid, each cell of lower resolution corresponds to four cells (in one dimension) in the higher resolution. This is illustrated in Fig. 8. The cell values shown in the figure are just random numbers used to clearly describe the implementation.

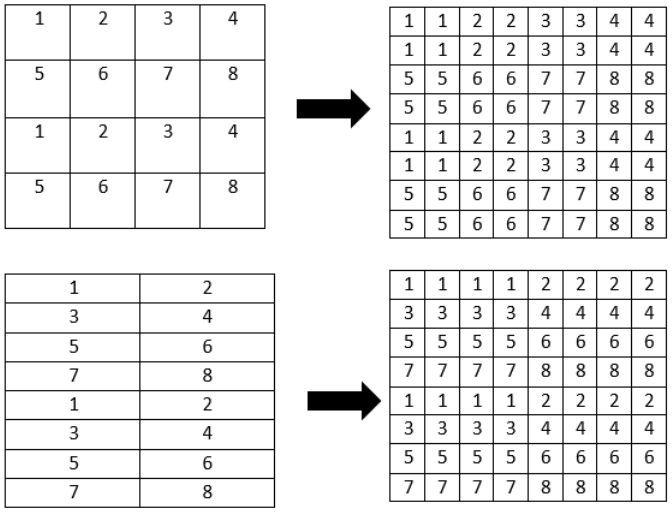

Figure 8. Transformation of component grids

V. Complete Implementation Overview

For implementing the sparse grid combination technique, first the component grids are generated. The program follows a master-worker model where the workers do the numerical computations and the master does the combination of component grids. A sparse grid is generated after combination which can be visually represented. This section explains the different parts of the implementation in detail with the help of examples.

A. Generation of Component Grids

The component grids are generated based on the minimum and maximum levels. For example, if the minimum is 6 and maximum is 9 then the component grids for the level vectors given below will be generated along with their coefficients.The actual resolutions of the component grids are derived from these level vectors by using the formula 2^n where n is the level. So, in this example the grid resolutions would be as given below.

Level Vectors: +[9,6], +[8,7], +[7,8], +[6,9], -[8,6], -[7,7], [6,8]
Resolution: +[512 x 64], +[256 x 128], +[128 x 256], +[64 x 512], -[256 x 64], -[128 x 128], -[64 x 256]

B. Determine the Number of Processes

The program follows the master-worker model and so the minimum number of processes required is two. The maximum number of processes is determined based on the number of component grids generated. As per the earlier example, the number of component grids generated is 7 and so the maximum number of processes would be 8 (1 master and 7 workers). If more processes than this maximum are available, then they are not utilized in the current implementation. Section 6 describes how this can be avoided.

C. Master-worker model

As mentioned before, the program follows a master-worker model. The master does the component grid generation and then decides the number of processes to use. The grid resolutions are send to the workers in a round robin fashion at every time step. The workers calculate the numerical flux for the component grid edges and then updates the values in the grid cells. The numerical flux calculation and cell updating is implemented similar to the existing implementation of the SWE simulation code. The grid cell values are then written into a file. The master is notified after every write in each time step. The master uses the files generated by all the processes in the combination algorithm. It generates a sparse grid after combination. This process is repeated for every time step. This sparse grid solution will be approximately similar to a full grid solution.

D. Combination Algorithm

The combination algorithm combines the component grids and generates a sparse grid. The component grids will have a smaller resolution than the sparse grid. So, this should be taken into account while combining. This was already explained in Section 4. Algorithm 1 shows how this combination technique is implemented.

Algorithm 1 Combination Algorithm

```
1:  function Combination Function
2:      Var evalx= Full_Grid_Resolution_X/Component_Grid_Resolution_X
3:      Var evaly=Full_Grid_Resolution_Y/Component_Grid_Resolution_Y
4:  for i from 1 to Full_Grid_Resolution_X and m from 1 to Component_Grid_Resolution_X do
5:      for j from 1 to Full_Grid_Resolution_Y and k from 1 to Component_Grid_Resolution_Y do
6:          Full_Grid_H[i][j] = Full_Grid_H[i][j] + OR - Component_Grid_H[m][k]
7:          Full_Grid_HU[i][j] = Full_Grid_HU[i][j] + OR - Component_Grid_HU[m][k]
8:          Full_Grid_H[i][j] = Full_Grid_HV[i][j] + OR - Component_Grid_HV[m][k]
9:          if j mod evaly == 0 then
10:             k++
11:     if i mod evalx == 0 then
12:         m++
```

VI. Results and Observations

The SGCT implementation was used to observe the changes in height and horizontal velocities of water during a simple dam break scenario. Results were obtained by using different minimum levels from 3 to 7 and a maximum level of 10 while generating component grids. The grids were combined to form a 2^{10} x 2^{10} sparse grid. These results were compared with the results of the existing full grid implementation for grids of size 2^{10} x 2^{10} and 2^9 x 2^9.

Fig. 9 shows the visualization of water height and data ranges obtained using the SGCT implementation at time 15 seconds (last time step) and the corresponding full grid implementation. From this we can observe that the

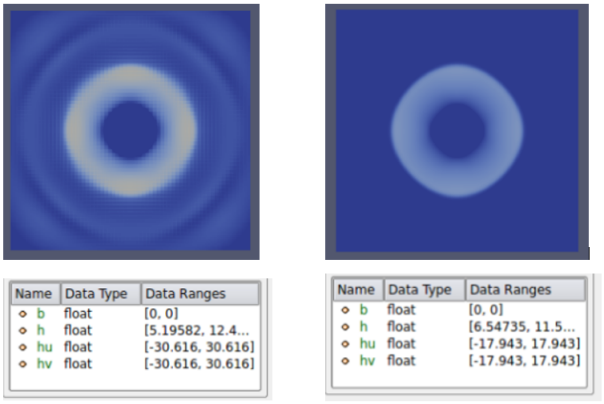

Figure 9. Visualization of water height in SGCT implementation with minimum and maximum levels (7,10) (left) and full grid implementation (right) at the last time step and the corresponding data ranges

data range of water height is from 5 to 12 for the SGCT implementation while it is from 6 to 11 for full grid implementation. This is a small difference when compared to the difference in horizontal velocities between the two implementations. The horizontal velocities range from -30 to +30 for the SGCT implementation while it is -17 to +17 for the full grid implementation. A possible explanation for this is that the SGCT implementation uses a combination of very coarse grids. So, in every cell of the grid the water can travel further than when compared to the cells of a very fine grid (full grid implementation). Hence the data range of horizontal velocities for the SGCT implementation are much higher than the full grid implementation. This also explains the difference in the visualization of water height. The higher velocities cause the higher water height to be dispersed further away from the center (where the dam break occurs) in the SGCT implementation than in the full grid implementation.

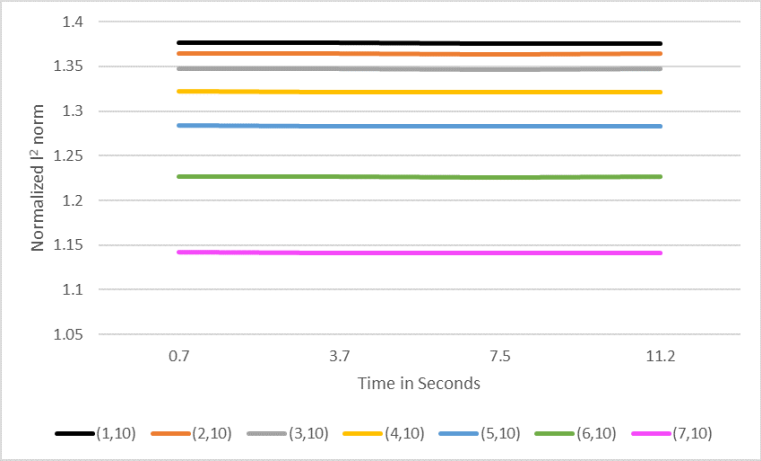

Figure 10. Normalized l^2 norm of water height with full grid mplementation and SGCT implementation. The water height is calculated for different minimum levels (1 to 7) and a maximum level of 10 for the SGCT implementation. This is compared with a full grid implementation with grid resolution 2^{10} x 2^{10}

The values of water height at different time for the SGCT implementation and full grid implementation were compared. Both implementation simulates the scenario over a duration of 15 seconds with computations done at every time step (approximately every 0.05 seconds). The normalized Euclidean distance (l^2 norm) was calculated to compare the water heights of both implementations. This is represented in Fig. 10. The X-axis shows the time step and the Y-axis shows the l^2 norm. The water height is calculated for different minimum levels (1 to 7) and a maximum level of 10 for the SGCT implementation. This is compared with a full grid implementation with grid resolution 2^{10} x 2^{10} (Fig. 10).

We can observe from Fig. 10 that the accuracy increases when the minimum level increases. This is because if the minimum level is low the number of component grids generated will be more but there will be a lot of very coarse grids. This will decrease the accuracy. For example for a minium and maximum level (3,10) there are 15 component grids but it includes very coarse grids like 2^3 x 2^{10} and 2^{10} x 2^3. So when the component grids are combined and compared with a very fine grid like 2^{10} x 2^{10} the error will be high. A minimum and maximum level of (7,10) generates only 7 component grids but they are more fine grids (i.e 2^7 x 2^{10} , 2^8 x 2^9 etc.) when compared to a minimum and maximum level of (3,10).

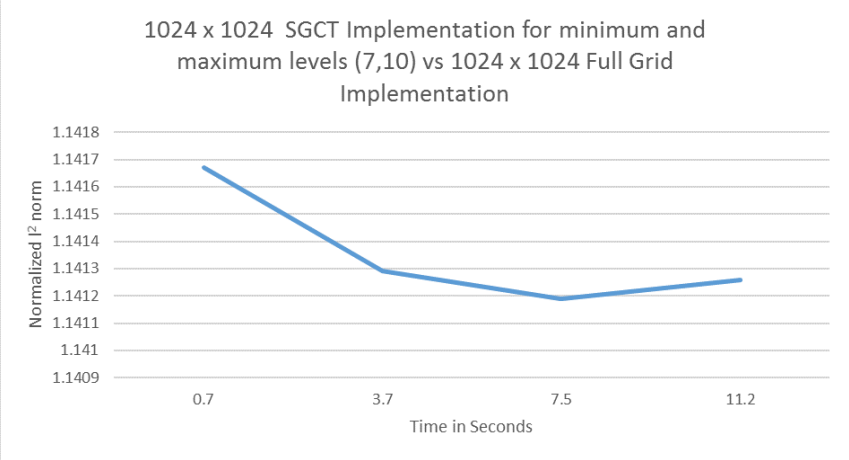

Figure 11. Normalized l^2 norm of water height with full grid mplementation and SGCT implementation. A full grid implementation with grid resolution 2^{10} x 2^{10} was compared with the SGCT implementation for sparse grid resolution 2^{10} x 2^{10} with minimum and maximum levels (7,10)

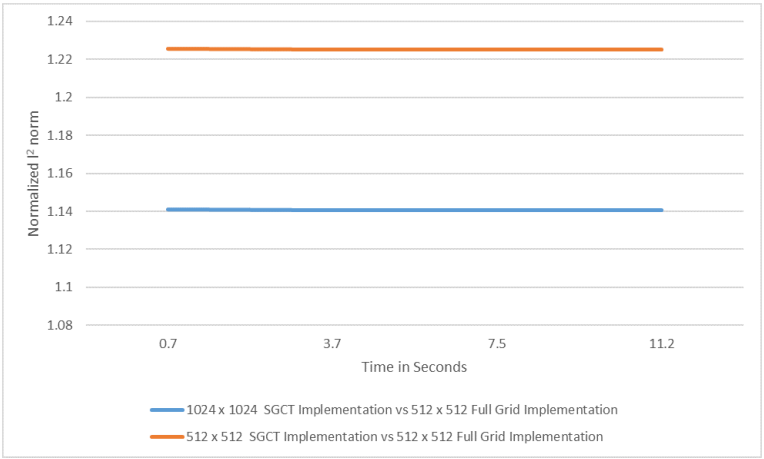

Figure 12. Normalized l^2 norm of water height with full grid mplementation and SGCT implementation. A full grid implementation with grid resolution 2^9 x 2^9 was compared with the SGCT implementation for sparse grid resolution 2^9 x 2^9. Also, a full grid implementation with grid resolution 2^9 x 2^9 was compared with the SGCT implementation for sparse grid resolution 2^{10} x 2^{10} .

Fig. 11 shows the normalized l^2 norm of water height with full grid mplementation and SGCT implementation. A full grid implementation with grid resolution 2^{10} x 2^{10} was compared with the SGCT implementation for sparse grid resolution 2^{10} x 2^{10} with minimum and maximum levels (7,10). We can observe that the accuracy is low at the initial time steps and then it increases over time. This is because, at the first time step the dam break has just occurred and there is a high water height and high velocity at the center. Coarse grids would disperse water further from the center than fine grids. But as time increases the heights and velocities decrease and both coarse and fine

grids would produce a more similar simulation.

A full grid implementation with grid resolution 2^9 x 2^9 was compared with the SGCT implementation for sparse grid resolution 2^9 x 2^9. Also, a full grid implementation with grid resolution 2^9 x 2^9 was compared with the SGCT implementation for sparse grid resolution 2^{10} x 2^{10}. The result is shown in Fig. 12. We can observe that by increasing the resolution of the sparse grid we get better accuracy.

From these observations, we can conclude that to get a good accuracy when compared to a full grid implementation we need to generate higher resolution sparse grids using the SGCT implementation. We should also take care that the component grids do not include very coarse grids. So optimum minimum and maximum levels to generate the component grids should be determined.

VII. Conclusion and Future Work

In this project, we implemented the sparse grid combination technique to solve hyperbolic partial differential equations. The shallow water equations were used for this purpose. SWE described the height and horizontal velocities of a fluid over time. Lower resolution anisotropic grids (component grids) were generated and distributed among the available processes. The SWE equations were solved on them in parallel. The combination algorithm created a sparse grid solution that was approximately similar to the full grid solution. After analyzing the results we observed that to get a good accuracy when compared to a regular Cartesian grid implementation we need to generate higher resolution sparse grids using the SGCT implementation. Also very coarse grids should not be used for the combination technique because it decreases the accuracy.

There are many future extensions possible for this project. Currently each of the component grids are solved on different processes. This can be further parallelized by performing a domain decomposition on the component grids and distributing it across different processes. This will lead to more communication for the exchange of ghost layers and so the performance changes need to be well evaluated. Another future work possibility is to make the program more resource elastic. This can be implemented by using the concept of elastic MPI [6].

References

[1] H. joachim Bungartz, M. Griebel, H. j. Bungartz, and M. Griebel, "Doi: 10.1017/s0962492904000182 printed in the united kingdom sparse grids."

[2] M. Griebel, M. Schneider, and C. Zenger, "A combination technique for the solution of sparse grid problems," 1992.

[3] A. Breuer and M. Bader, "Teaching parallel programming models on a shallow-water code," in *Proceedings of the 2012 11th International Symposium on Parallel and Distributed Computing*, ser. ISPDC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 301–308. [Online]. Available: http://dx.doi.org/10.1109/ISPDC.2012.48

[4] "Swe - a simple shallow water code," https://www5.in.tum.de/SWE/doxy/, 2016, [Online].

[5] "Tolerating hard and soft faults with the sparse grid combination technique," http://scc.acad.bg/ncsa/articles/library/Library2016_Supercomputers-at-Work/ISC2016_Frankfurt/Tolerating-Hard-and-Soft-Faults.pdf, 2016, [Online].

[6] I. Comprés, A. Mo-Hellenbrand, M. Gerndt, and H.-J. Bungartz, "Infrastructure and api extensions for elastic execution of mpi applications," in *Proceedings of the 23rd European MPI Users' Group Meeting*, ser. EuroMPI 2016, 2016.

[7] J. Garcke, "Sparse grids in a nutshell," in *Sparse grids and applications*. Springer, 2013, pp. 57–80.