HPX Application: HAD Adaptive Mesh Refinement

Matthew Anderson[1,2]
[1]*Department of Physics and Astronomy, Louisiana State University, Baton Rouge, LA 70803-4001*
[2]*Center for Computation and Technology, Louisiana State University, Baton Rouge, LA 70803-4001*

(Dated: February 5, 2010)

Details efforts at creating an adaptive mesh refinement (AMR) toolkit based on HPX.

PACS numbers:

I. INTRODUCTION

There are many important physical scales which need to be adequately resolved when simulating the orbit and merger of compact astrophysical objects like neutrons stars and black holes. These scales include (1) the individual stars, preferably incorporating some of their internal dynamics, (2) the orbital length scale, (3) the gravitational wave zone, and (4) the location of outer boundaries. The computational demands required to resolve these different physical scales are best met using adaptive mesh refinement.

In the past this challenge has been met using the publicly available MPI based computational infrastructure HAD to provide parallel distributed AMR for our codes [1, 2]. As an example, Figure 1 illustrates the resulting mesh structure at a pre-merger stage of a binary neutron star system in our simulations.

Strong scaling results of the MPI based HAD toolkit are illustrated for a particular sample problem in Figure 2. We define speed-up as

$$\text{speedup}(n) = \frac{\text{Run time on one processor}}{\text{Run time on } n \text{ processors}}.$$

The results presented are strong scaling results; the global problem size was kept constant while the number of processors varied. Strong scaling tests are problem dependent and vary according to the size of the global problem selected for investigation. For most problems we investigate, strong scaling using the MPI based HAD toolkit is sufficient only up to about 256 processors.

The HPX implementation of ParalleX provides a way to eliminate the global barriers which impair the scaling of the HAD toolkit. A 1-D implementation of the HAD AMR toolkit has just been completed; a 3-D implementation is coming shortly. In the next section we outline the key concepts and data structures inside the new HPX based HAD for both 1-D and 3-D implementations.

II. HPX BASED HAD

The MPI based HAD toolkit in its original form contains a global barrier every timestep: no computation can proceed once the global barrier has been reached until every point has reached the same timestep in the simulation. This type of global barrier is not unique to HAD: other major finite difference and finite element based AMR toolkits also contain this global barrier. This global barrier causes major problems when running on large numbers of processors because most processors end up waiting for others to reach the global barrier. Improved load balancing can help reduce this problem but will not solve it. ParalleX enables us to remove all global barriers from the simulation pipeline, including the ubiquitous timestep barrier. The HPX based HAD toolkit has been redesigned in order that a node point which is ready to proceed computing the next timestep in the simulation can do so without waiting until the neighboring node point is ready to do the same. The essential concept is illustrated in Figure 4.

Three types of mesh objects are introduced to remove all global barriers. The fundamental data structure is a node point. Each node point is autonomous and may be communicated depending on the granularity desired for a particular problem. Finite difference AMR codes normally pass large blocks of node points to the user defined code; only the boundaries of these blocks are communicated, as illustrated in Figure 3. HPX based HAD currently implements the smallest granularity possible. Eventually a runtime parameter will be introduced so that the user can adjust the optimal granularity for a particular problem on a particular number of processors.

The first type of mesh object in the HPX based HAD toolkit is the unigrid mesh. This is illustrated in Figure 4 for a 1-D space and time simulation where each node point requires information from adjacent neighbors in order to compute a result for the next stage in the simulation. The number of neighbors is adjustable as a runtime parameter in the HPX based HAD implementation.

The second and third type of mesh objects in the HPX based HAD toolkit are tapered meshes. The standard

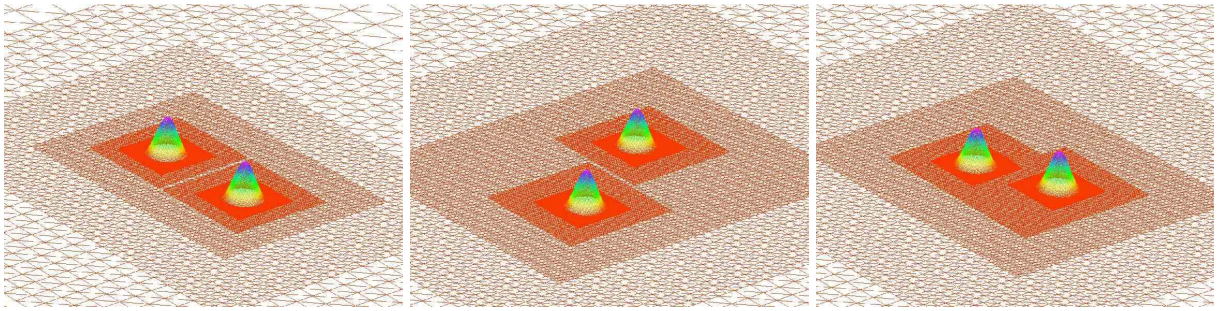


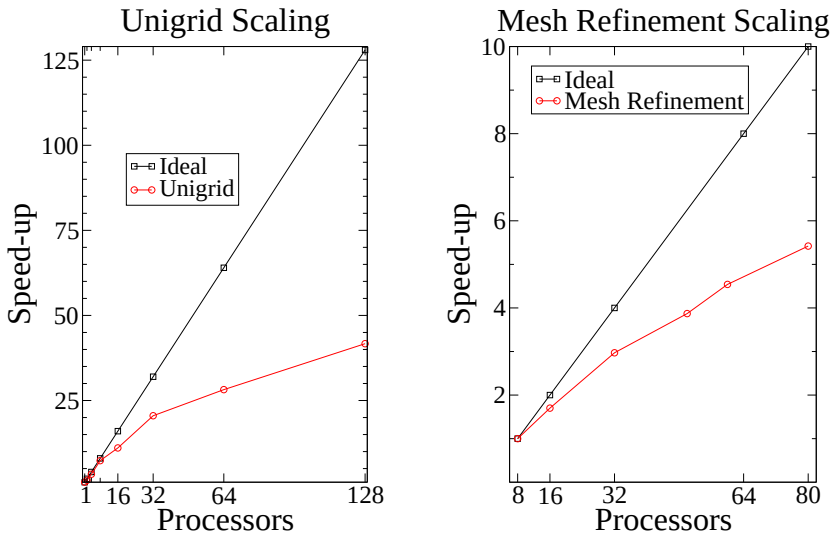
FIG. 1: The AMR mesh structure at three different times for the pre-merge stage of the simulation with a highest resolution of 32 points across each star. This simulation was performed using the MPI based HAD toolkit. The simulation had seven different resolution levels depending on the relative error measured locally in the computational domain. Five of those resolution levels are visible here. Simulations were performed on 128 processors.

Berger-Oliger AMR [3] calls for interpolation in time at the boundaries of finer meshes. The tapered-grid boundary method [4] offers an improvement by eliminating the need for interpolation in time while significantly reducing spurious reflections at interface boundaries. In the tapered-grid boundary method, the boundary points of an AMR region are discarded each update cycle resulting in a tapered spacetime mesh that respects the domain of dependence for the problem. The tapered meshes in HAD incorporate the tapered grid boundary method; an illustration of the two tapered meshes needed for second order spatial differencing with AMR is found in Figures 5 and 6.

Using the meshes illustrated in Figures 4, 5, 6 we have created a parallel 1-D AMR application code evolving the advection equation without any global barriers with arbitrary levels of refinement. The data flow of the HPX based HAD AMR code is illustrated in Figure 7 for a single level 1-D AMR simulation. Snapshots of a multiple refinement level simulation at two instances in time are shown in Figure 8. Figure 9 follows the evolution of a multiple refinement level simulation in *physical* time to better illustrate the impact of removing global barriers from a simulation.

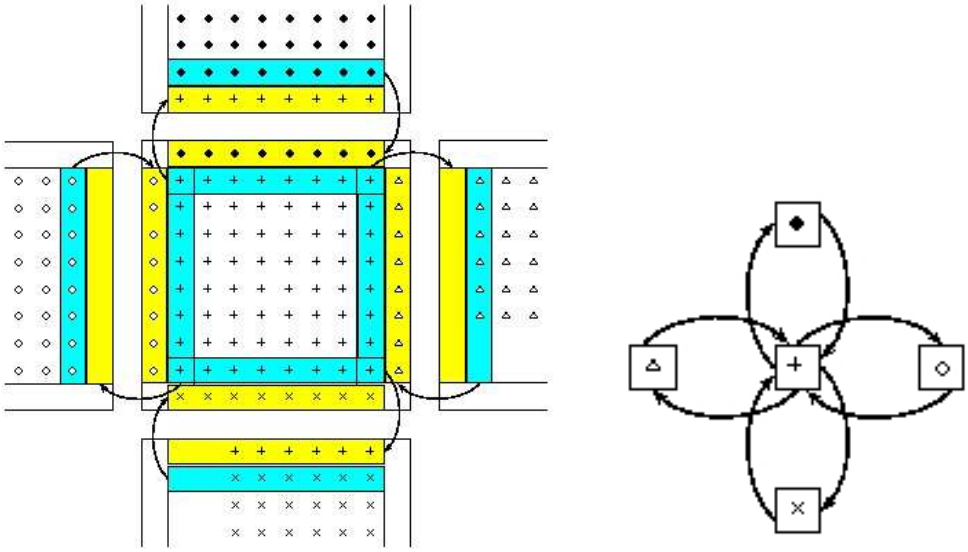
Acknowledgments: We would like to thank S. Liebling and L. Lehner for stimulating discussions.

[1] http://www.had.liu.edu/.
[2] S. L. Liebling, Phys. Rev. **D66**, 041703 (2002).
[3] M. J. Berger and J. Oliger, J. Comp. Phys. **53**, 484 (1984).
[4] L. Lehner, S. L. Liebling, and O. Reula, Class. Quant. Grav. **23**, S421 (2006).



(a) **(b)**

FIG. 2: This figure shows strong scaling results for single grid and mesh refinement using the MPI based HAD AMR toolkit. A spherical blast wave initial dataset wase run for a fixed problem size as the number of processors is varied. The left frame shows the unigrid strong scaling results, and the right frame includes mesh refinement. For the unigrid scaling, the data were evolved for 80 iterations, and the global grid size was 121^3. For this problem size, the communication overhead begins to overshadow the local process computation on ≥ 64 processors. For the mesh refinement scaling, thirty iterations were performed on a coarse grid of size 81^3 and a single level of refinement. Since the test problem would not fit in memory on a single processor, speed-up was measured using 8 processors as the base value. All tests were performed on an Intel Pentium IV 3.0 GHz cluster with Myrinet.



(a) **(b)**

FIG. 3: Two different approaches to structured mesh based communication: (a) 2-D representation of a typical communication pattern for a finite difference based AMR code. Large blocks of memory are passed to the user for computation where only the boundaries of the blocks are communicated among processors. In this figure, the yellow regions (frequently referred to as ghostzones) are communicated regions originating from blue zones on a distributed memory block as indicated by the arrows. The current HPX based HAD approach is seen in (b). Here each point is communicated giving the simulation the smallest granularity possible. While the amount of communication required in paradigm (b) is substantially more than in (a), those communication costs are reduced by the locality management inherent in HPX. Paradigm (b) is advantageous for eliminating global computation barriers. HPX based HAD will eventually be capable of both paradigm (a) and (b) controlled as a runtime parameter so the user can adjust the optimal granularity for a particular problem on a particular number of processors.

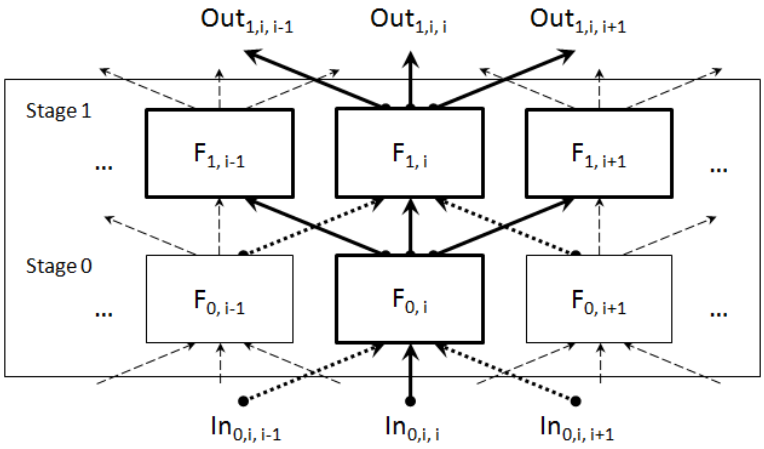


FIG. 4: The unigrid mesh structure for a 1-D space and time simulation where each node point requires information from both adjacent neighbors in order to compute a result for the next stage in the simulation. When the input data (indicated by "**In**") required for operation $F_{0,i}$ to compute a result for Stage 0 is available, the computation proceeds. The $F_{0,i}$ operation can proceed without waiting for the adjacent operations $F_{0,i-1}$ and $F_{0,i+1}$ to be ready.

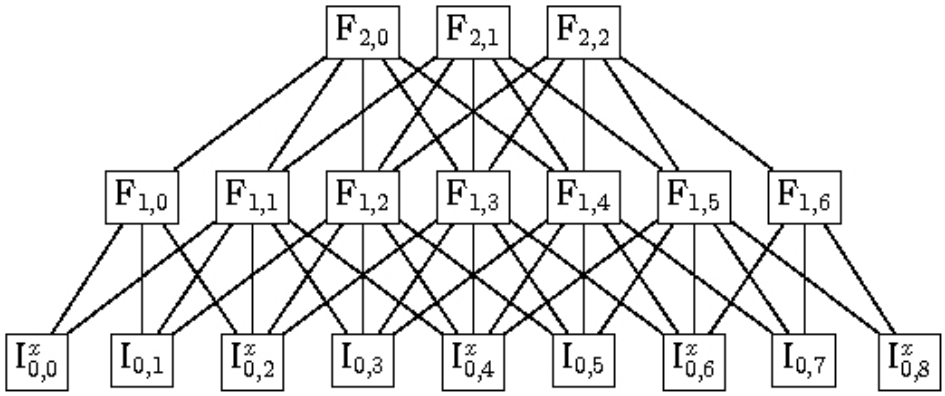


FIG. 5: The left tapered mesh structure. One of two meshes used for computing finer meshes. See also Figures 6 and 7.

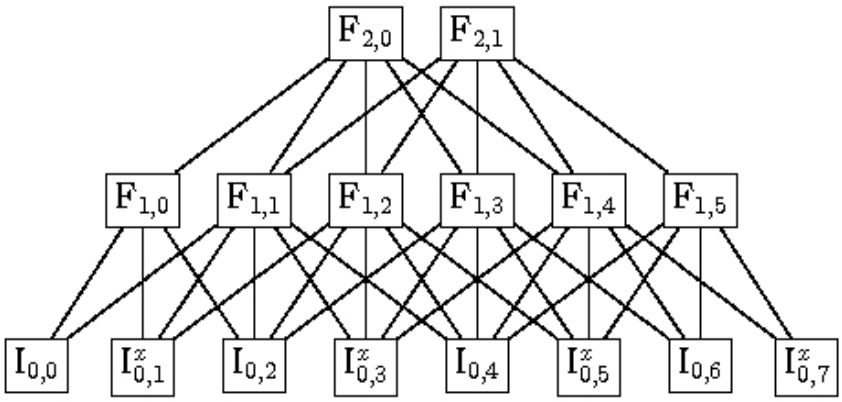


FIG. 6: The right tapered mesh structure. One of two meshes used for computing finer meshes. See also Figures 5 and 7.

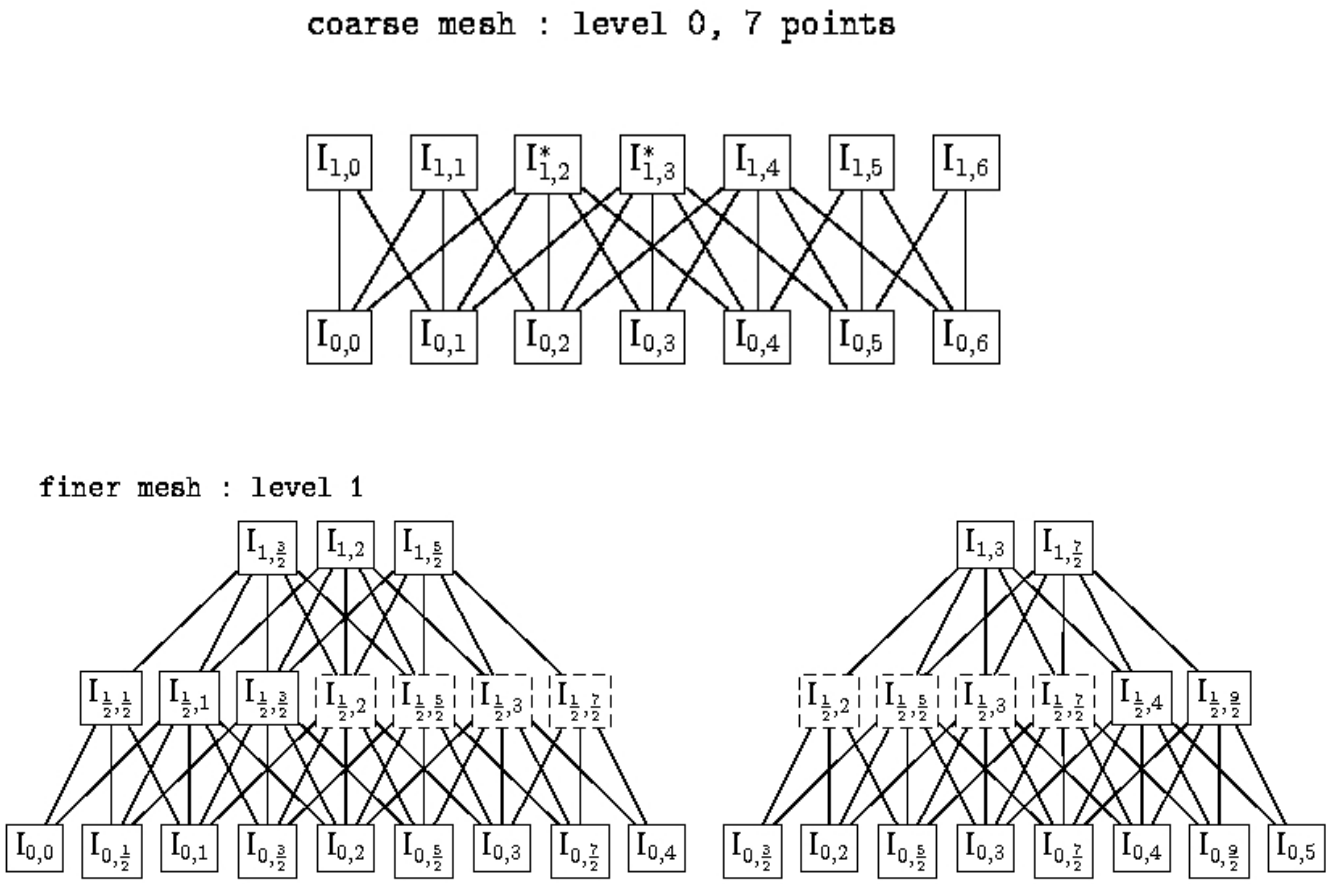
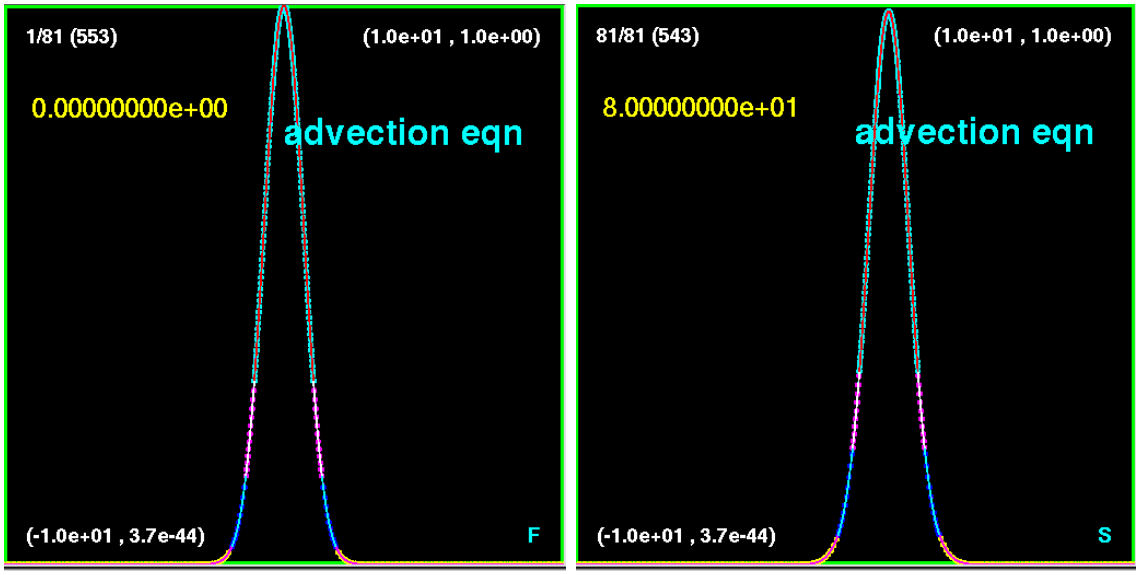


FIG. 7: AMR data flow example for a simulation with 1 level of refinement and a stencilsize of three so that the adjacent neighbors of each point are required for a computation. The numbers in each frame indicate the timestep and the coordinate location of the node. The simulation begins at timestep 0 on the coarse mesh (level 0) where data points $I_{0,0\to 6}$ are provided as initial data. The initial data is communicated as indicated. Data points at timestep 1, $I_{1,0\to 6}$, are then computed as the data becomes available and the refinement criteria is applied to the result. In this example, the two points labelled $I_{1,2}$ and $I_{1,3}$ are flagged for refinement as indicated by the *. At this point, the computation of $I_{1,2}$ and $I_{1,3}$ is repeated, but this time using the finer meshes illustrated. Data points $I_{0,\frac{1}{2}}$, $I_{0,\frac{3}{2}}$, $I_{0,\frac{5}{2}}$, $I_{0,\frac{7}{2}}$, and $I_{0,\frac{9}{2}}$ are then found from the initial data (or from a previous finer mesh cycle if available; interpolation from neighbors if not), and the computation of $I_{1,2}$ and $I_{1,3}$ is repeated at twice the resolution. Neighboring points $I_{1,\frac{3}{2}}$, $I_{1,\frac{5}{2}}$, and $I_{1,\frac{7}{2}}$ are also computed for use in future AMR cycles. This process is entirely non-blocking and allows for each point to refine autonomously without knowing if its neighbor is refining or not. The cost of this, however, is that some computations are duplicated. Duplicate computations are indicated by dashed frame boxes.



(a) (b)

FIG. 8: The plots in (a) and (b) show the AMR mesh structure of a wave with 3 levels of refinement traveling to the right at two instances in time: $t = 0$ and $t = 80$. The refinement criteria was set to be the amplitude of the wave in this case for demonstration purposes. The coarsest mesh is colored orange; the finer meshes (each finer than the last by factor of two) are colored blue, purple, and light blue-red. The refined regions track the wave as it moves to the right. This example was run on two operating system threads.

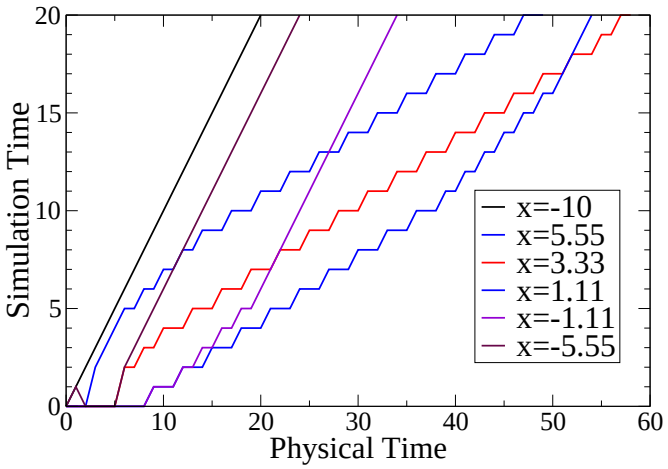


FIG. 9: Results from evolving the 1-D advection equation using two levels of refinement for a simulation time of $t = 20$. This figure compares the simulation time versus physical time (i.e. real time) for six different x coordinate positions. Notice that some points in the simulation long before others do (i.e. the slope is much higher for some points than for others). This is a consequence of removing global barriers from the computation. If a global barrier were introduced, then all x coordinate points would like on the same line and the code performance could be no better than the lowest slope line in this figure.