Elliptic Problem for the shift

- During the 3d evolution, we will "probably" need to solve a non-linear coupled elliptic PDE, with holes cut out of the domain. We have primarily results that suggest that the shift term might need to be chosen such that it satisfies the minimal distortion equation, which is a coupled non-linear elliptic PDE.
- To solve the elliptic problem we want to use a multigrid solver (or Multi Level Adaptive Technique solver).
- DAGH does support multigrid, but it does "nothing special" on the coarse grids. On the coarse grids the communication will outweigh the computation.
- We have implemented multigrid in HPF for "trivial" problems, but problems with holes cut out of the domain will once again provide problems.
- It is unclear how to solve multigrid on the AMR hierarchy. We can solve the full Multi-Level Adaptive Technique (MLAT) equations on the hierarchy. We can also solve each the elliptic equations on each grid independently, using Dirichlet conditions from the parent grid.

Finding the Apparent Horizon

- We have two methods to solve this elliptic problem. One method expands the terms in spherical harmonics, and the other method solves the elliptic problem using finite-differencing with a Newton shooting method with a conjugate gradient method.
- In order for “DAGH” to implement the second method, the algorithm must be completely rewritten, since it uses a “non-parallel” method, and DAGH has *no support* for linear matrix solvers.
- In HPF the second method can make use of available parallel elliptic solvers.
- The first method has been written in DAGH and has been used with one of the black hole evolution codes.

- In HPF, when using 3d mask and WHERE construct, there is no difference for a static and moving black hole. The efficiency of the code (poor load balance) remains constant as the evolution of the hole proceeds.
- Using extrinsic procedures in HPF allows it to reproduce DAGH performance.
- As in case of the static hole, the load balance can be significantly impoved in HPF-2 using generalized block or more general indirect distribution at the cost of additional computation to determine the optimal block size for each processor, and remapping of data (next neighbor communications).

Moving Inner boundary

- During the evolution, the black holes will move on the computational domain. Since the computations around the holes are more expensive, the distribution of the data must be redistributed in order to achieve “better” load balancing.
- In DAGH we are ignoring the load balancing problems around the hole, although DAGH does provide a weight function for grid points, and this feature may be used.
- DAGH does not have any special techniques to handle holes, so on processors that have many boundary points, a large slowdown is expected.
- For non-moving holes, we can place the data so that communication cost will not be large around the holes, i.e. we can move the center of the hole such that one processor contains all of the hole points.

Linear wave problem with matching outer boundary condition for gravational wave extraction.

- The difficulty in this problem for both DAGH and HPF is the following.
- There are two separate evolutions inside the main program. The Cauchy evolution uses a 3D evolution and the characteristic matching problem uses a 1D evolution. The calculation proceeds as follows.
- The ADM code evolves Einsteins equations, and then lattice points from this evolution are interpolated to the points on the 1D lattice.
- Next the 1D evolution evolves this system, and then points from this 1D evolution are interpolated back onto the 3D lattice, to give the outer boundary points.
- The problem is that there are two separate evolutions, and communication must take place between both evolutions.
- DAGH was not built to handle this. (Use MPI “extrinsic”)
- HPF is a general purpose language, so it can handle this. Since the computational cost of the 1d evolution is very small comparing to the 3d evolution, it can be trivially implemented allowing for redundant replicated computations. Extracting the boundary points from the 3d grids necessarily leads to the interprocessor communication (points on the extraction surface must be broadcasted to all processors).

- HPF offers several solutions to this problem. A most natural approach is to introduce 3d mask and employ WHERE statement. The price is additional storage and necessity of scanning 3d volume to identify the boundary points located on the sphere. This solution leads to a similar load imbalance as we have in DAGH.
- The load balance can be improved using general block or indirect distribution introduced in HPF-2. Also, the HPF-2 directive SHADOW allows to define a variable size of the ghost regions so that the very expensive communications needed for computations that involve stencil of size 4 can be easily restricted to the routines that impose the boundary conditions (a feature not available in DAGH).
- The idea of the distributed 1d list introduced in our DAGH implementation is difficult to implement directly in HPF, since it violates the global name space of HPF. However, it can be easily implemented as an extrinsic procedure.
- Other solutions, such as declaring a global 1d array (the list of points) may lead to additional communications, since the block size of the grid functions and the block size of the list owned by a processor (assuming both are distributed blockwise) in general do not match.

Black holes (Inner Boundary Condition)

- For uni-grid calculations with black holes cut out of the domain, apparent horizon boundary conditions allow for special forward and backward differencing to determine the derivatives of the grid functions at the "boundary" lattice points.
- These points are not uniformly distributed across the domain because the are generated from placing a spherical boundary onto Cartesian coordinates.
- We have a DAGH version of this code which allows for black holes in the computational domain. The spherical boundary surface is mapped onto 1d list of boundary points, and each processor maintains a local part of it. Creating the distributed list requires no interprocessor communication, and subsequent computations are performed only by processor that own the boundary points (load imbalance).
- DAGH has a weighting function for lattice points, which weights the cost of the computation per lattice point; but it does not have a systematic way of weighting the communication cost which is much more expensive around the holes. (We use a fourth order operator to evaluate the derivatives around the hole, and these are forward and backwards differenced. Thus, the stencil size around the hole can be as large as 4.)

Linear waves

- Linear waves in the BBH GC code have been run in both DAGH and HPF. Both implementations allow for very good parallel speedup, since all calculations are regular.

Timings of HPF run with pghpf (version 3.02) using `(*,*,Block)` distribution for small problems, and `(*,Block,Block)` for larger problems(n=129)

SP2	33	65	129	T3d	33	65
1	170	-	-		208	-
2	104	541			113	
4	65	303			63	
8	55	220			44	220
16	38	113	607		38	138
36		75	286			
64			222			

- Although we have no current numbers for the parallel speedup of DAGH, we have seen almost linear speedup on this “simple” problem.

Implementation

- The PDE unigrid solver can be implemented in a data parallel paradigm, and a HPF implementation works very efficiently on systems such as SP-2 or T3D. Using some of the HPF-2 features will allow to achieve a decent performance of the code even with the black hole boundary conditions, and wave extraction.
- Parallel implementation of AMR in HPF is difficult. HPF-1 lacks adequate language features, such as distributed components of derived types (to maintain grid hierarchy), and some of other features such as pointers to distributed data has not been implemented in the current generation of the HPF compilers.
- HPF-2 makes implementation of AMR easier. In particular, support for pointers and derived types allows for efficient dynamical memory management, and generalized block distribution and elements of task parallelism (SUBSET, TASK_REGION, INDIRECT) make it possible to control the load balance communication cost better. Nevertheless distributing the grid hierarchy is very challenging, and a custom data distribution not supported by HPF may be necessary (c.f. DAGH).
- HPF-2 INDIRECT may not be as efficient as DAGH since it uses a spacetime filling curve distribution.
- Implementation using “raw” message passing library, such as MPI is tedious.
- Anticipating these problems the Alliance decided to use the DAGH system (Distributed Adaptive Grid Hierarchy) developed by Browne and Parashar.

- It is particularly inconvenient when advanced data structures are used in Fortran. For example, the Black Hole code, using Fortran 90 derived types, aggregates grid functions into 3-, 6-, 9- or 18-component tensor with specific symmetries built in. Here, the DAGH distributed grid hierarchy is flattened, send to Fortran as a set of arrays, and the tensor structure is build. This operation must be repeated each time when an interpocessor communication is needed.
- Fortunately, we have not found any noticeable performance penalty of doing this, and the burden of generating the calling sequences is removed by using custom developed `perl` scripts for this purpose.
- DAGH is not a complete language, such as Fortran or C++ (and it has never been meant to) and therefore it has some limitations of expressiveness. On several occasions we found missing advanced features, such as support for interoperability of grid hierarchies of different dimensionality (3d interior evolution and 2d boundaries). These can be custom developed using a mechanism resembling HPF extrinsic procedure mechanism, though.

Parallel Execution in DAGH

- DAGH systems automatically decomposes all grids into “grid components” and distribute them onto available processors. The algorithm used guarantees the optimal load balance and alignment of grid components on different refinement levels defined on the same computational subdomain. Rebalancing of the load after adaptive regridding requires only next neighbor communications.
- Each processor is assigned a set of grid components so that the computations are performed in a data parallel fashion. Interprocessor communication are facilitated by the prefetched ghost regions and a library of DAGH communication functions implemented in C++ on top of MPI (reductions, moving subregions defined by bounding boxes, etc.).
- DAGH is implemented in C++, and the user routines, including the PDE solver can be written in Fortran (particularly important when porting an existing code into DAGH). Fortran routines take grid components (a flat set of arrays) as arguments. The price of it is that the DAGH communication library cannot be called from Fortran routines and all computations must be restricted to local data augmented by the prefetched ghost regions. All other interpocessor communications must be performed by the driving C++ routine.
- As a consequence, the Fortran code must be partitioned into “local access only” fragments (node programs where DAGH generates ghost regions/boundary values) to be called from a driving C++ routines. Splitting the code in such a way may be tedious and error prone.

DAGH System

- DAGH system provides a set of high level abstractions that can be used to implement an existing sequential unigrid PDE solver in the AMR regime on multiprocessor systems.
- User defines a cartesian computational domain (“base grid”), and specifies parameters for creating a grid hierarchy. The list of parameters includes maximum levels of refinement, refinement ratio, switch between cell-centered and vertex-centered grids, and more.
- User provides routines to update the grid and boundary conditions, and routines that provide data for testing regridding criteria (typically truncation error estimate).
- DAGH provides a library of routines for intergrid data movement (prolongation and restriction), and the user is allowed to replace it by his/her own that better matches the problem in hand.
- For parallel execution, the user specifies the size of ghost regions.

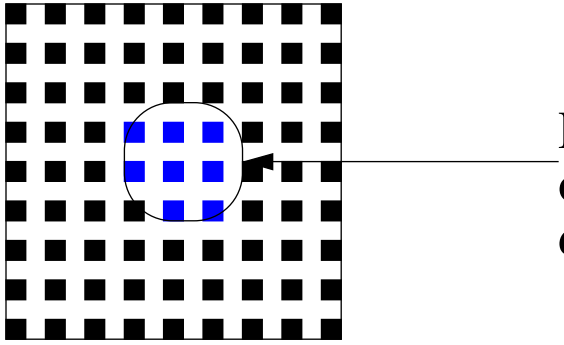
Difficulties of the BBH Problem

Difficulties for H(HPF), H2(HPF-2) and D(DAGH) are given from 1-3, where 1=easy, 2=medium and 3 =difficult

Difficulties inherent to the BBH problem	H	H2	D	Comments
Linear wave problem. No “inner boundaries”	1	1	1	lapse=1, shift=0, no holes.
Linear wave problem with matching outer boundary condition for gravitational wave extraction.	2	1	3	The outer boundary requires a 1d evolution after the 3d evolution.
Adaptive Mesh Refinements (AMR)	3	2	1	DAGH was designed for this purpose.
Multigrid/MLAT + AMR	3	2	1-3	Only a certain class of problems can be solved with DAGH.
Spherical “inner boundary” in Cartesian coordinates.	2	2	2	DAGH: large stencils on the inner boundary - expensive communications. HPF: use 3D “sparse” array or extrinsics.
Moving “inner boundary”.	3	1	1	Irregular boundary makes it difficult to implement in HPF.
Finding Apparent Horizon.	1	1	1	Expand spherical harmonics.
Causal Differencing with a “real” shift.	2	2	1	Requires additional storage in HPF.

Apparent Horizon boundary conditions.

- To achieve long-term stability we must cut out the singularity from the computational domain.
- Previous black-hole codes produced long term instabilities due to problems associated with grid points being "sucked" inside the singularity.
- Cosmic Censorship allows us to cut out the points inside the even horizon.
- These points are casually disconnected from our universe and do not effect our the other points from our universe.
- Since we can not find the even horizon for general black holes, we find the "Apparent" horizon, which is defined from the equation of trapped surfaces. It is the innermost trapped surface.
- The apparent horizon is defined by the solution of an elliptic equation.



Advantages

- 2nd-order convergent and Courant stable for any shift.
- Courant stability condition independent of shift.
- Natural way to repopulate grid points flowing out of the Apparent Horizon.

Disadvantages

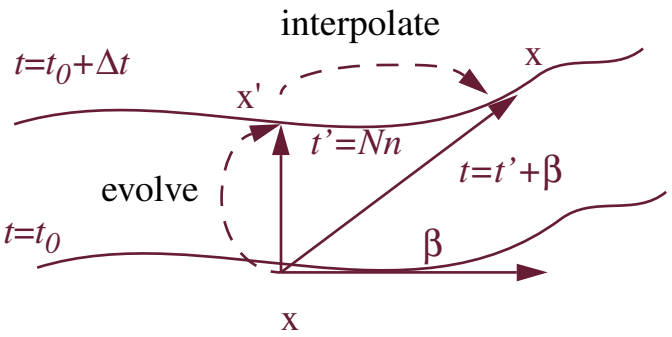
- Innermost grid points computed by extrapolation, not interpolation.
- Long-term stability is questionable.

Causal Differencing

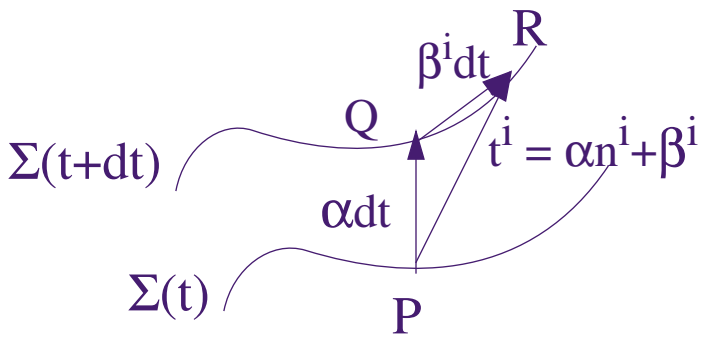
- Causal Differencing handles the $\beta^k \partial_k$ terms in the evolution equations (the advective terms).
- β^k moves the world lines of particles in the coordinates and changes the coordinate basis vectors, i.e. the shift "shifts" the coordinates.

Our implementation is as follows

1. Evaluate the RHS ${}^{(g)}\Re_{ij}^{(n)}$, ${}^{(K)}\Re_{ij}^{(n)}$ at time level *n*.
2. Compute the shifted coordinates at levels *n-1* and *n*.
3. **Interpolate** the RHS's to the new shifted coordinates at the *n* time level.
4. **Interpolate** g_{ij} and K_{ij} at the shifted coordinates on the *n-1* time level.
5. Carry out a leapfrog update.



- α determines the lapse in proper time, and β^i describes the shift in the spatial coordinates, and t^i are the tangents to a coordinate observer who in moving through the spacetime moves in a timelike direction by αdt, and in a spacelike direction by β^i.



- The extrinsic curvature is a spacelike object which determines the contraction and shear of the normals to $\Sigma(t)$.
- The evolution equations we solve are

$$\partial_t g_{ij} - \beta^k \partial_k g_{ij} = -2\alpha K_{ij} + g_{lj}\partial_i \beta^l + g_{il}\partial_j \beta^l$$

$$\partial_t K_{ij} - \beta^k \partial_k K_{ij} = -\partial_i \partial_j \alpha + \Gamma^k_{ij}\partial_k \alpha + \alpha(R_{ij} + K K_{ij} - 2K^l_i K_{lj}) + K_{lj}\partial_i \beta^l + K_{il}\partial_j \beta^l$$

- We have the ability to use either a 3-level non-staggered leapfrog scheme or a 2-level semi-iterative Crank Nicholson scheme to evolve the equations forward in time.
- In order to evolve the equations in a stable manner, we make use of causal differencing (explained on the next foil).

ADM Equations

3+1 Splitting of Einstein's equations

- We decompose the four-dimensional covariant version of Einstein's equations so that it is possible to pose them as a Cauchy evolution problem.

$$G_{00} = 8\pi T_{00}$$
$$G_{0i} = 8\pi T_{0i}$$
$$G_{ij} = 8\pi T_{ij}$$

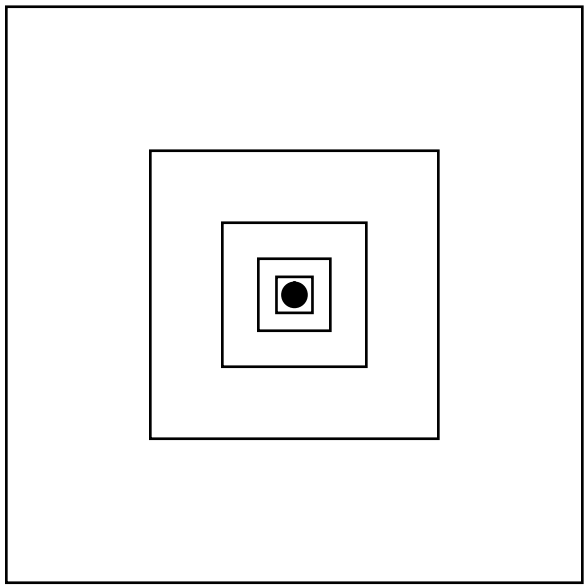
- A relationship between the components of the metric of spacetime and those of the spatial 3-metric must now be defined.
- We can look at the Arnowitt-Desser-Misner (ADM) form of the metric to see this

$$ds^2 = -\alpha^2 dt^2 + \gamma_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt)$$

where the functions α and β^i are called the lapse and shift functions, and γ_{ij} is the 3-metric.

Computational Infrastructure

- To get accurate waveform extraction, for stable evolution codes, our memory requirement is (unigrid codes): $8(100)((1/32)^{-1}(2)(32))^3 = 6871$ Gigabytes, where (8=bytes/word, 100=total number of 3d grid functions, 1/32 = mesh spacing, +,- 32=outer boundary placement relative to the center of the domain(0,0,0).
- If we use Adaptive Mesh Refinement, our memory requirements can be greatly reduced. $8(100)(5)(128)^3 = 8$ Gigabytes, where we can use 5 grids with 128^3 points per grid, as shown on the figure to the left.



- The grids will dynamically adapt to the equations, using a truncation error estimate.
- `DAGH` (Distributive Adaptive Grid Hierarchy) is used to handle the memory management of adaptive meshes on parallel computers.
- `DAGH` must also handle the solution of elliptic PDE's, as well as hyperbolic PDE's.
- We use finite difference techniques for the spatial and temporal derivatives.
- HPF can also be used for problems which do not require adaptive meshes.

Goals of the Alliance

- Develop a problem solving environment for Einstein's equations including a dynamical adaptive multilevel parallel infrastructure.
- Provide controllable convergent algorithms to compute gravitational waveforms which arise from black hole encounters, and which are relevant to astrophysical events and may be used to predict signals which for detection by ground and space based detectors (LIGO).
- To provide representative examples of computational waveforms.

Products

1. A Parallel Adaptive Computational Infrastructure (DAGH), HPF.
2. Initial Value Solvers.
3. Outer Boundary Solvers.
4. 3D Evolution Codes.
5. Coordinate Conditions.

HPF & DAGH usage in the Binary Black Hole Grand Challenge

T. Haupt, S. Klasky, G. Fox

NPAC, Syracuse University

haupt@npac.syr.edu, scott@npac.syr.edu, gcf@npac.syr.edu

- Goals of the alliance
- Computational infrastructure
- ADM Equations
- Causal Differencing
- Apparent Horizon boundary conditions
- Difficulties of the BBH problem in DAGH and HPF
- Implementation

Comment: The difficulties of the efficient use of the BBH GC code make the community run this code on SMP machines (favoring the **Cray vector machines (T90!)**)