Final Report 2013
Improving the Performance and Walltime of GRMHD Calculations for the Blue Waters Petascale Project

Scott C. Noble, Manuela Campanelli, Joshua Faber, Yosef Zlochower
*Center for Computational Relativity and Gravitation, Rochester Institute of Technology, One Lomb Memorial Drive, Rochester, NY 14623, USA**

Super-massive black hole mergers are believed to happen with significant frequency at the core of most active galaxies. Such events would be very exciting to detect, revealing important information on the birth and growth of their host galaxies, as well as explain how highly relativistic matter behaves in their surrounding accretion disks and in their associated jets. Additionally, it would provide a concrete example of one of general relativity's most spectacular predictions and possibly also allow a test of the validity of general relativity in a truly strong-field regime. Our aim is to provide the field with the first accurate electromagnetic predictions of these circumbinary disk environments using state-of-the-art general relativistic magnetohydrodynamics (GRMHD) simulations and general relativistic radiative transfer calculations of their resultant data. Our Subaward PRAC project focuses on the development of computational frameworks around three codes that will be used for this endeavor: 1) `harm3d`, a relativistic magnetohydrodynamics code for accretion disk calculations in black hole space times; 2) `bothros`, a general relativistic ray-tracing code used to post-process `harm3d` data; 3) `GRHydro`, part of the `Einstein Toolkit` numerical relativity package used to evolve magnetohydrodynamics in dynamical gravitational environments. We report here on our progress in improving the load balance of `harm3d`, `harm3d`'s parallel performance on Blue Waters of the National Center for Supercomputing Applications (NCSA), our progress toward improving the efficient of `bothros` calculations, and our efforts with `GRHydro` development.

I. INTRODUCTION

A major outstanding open question in astrophysics is how supermassive black holes (SMBH) and their host galaxies form and evolve in our universe. SMBHs are observed in the centers of almost all galaxies with a bulge [1] up to very high-redshifts [2], and current models for active galaxies have SMBHs as the central engines [3]. The prevailing theory of galaxy formation asserts that today's massive galaxies were assembled from smaller pieces, as dark-matter halos of progressively greater size merge [4, 5]. In a merging pair of galaxies, dynamical friction from the gas and stars surrounding the SMBHs lead to the formation of tight SMBH binary (see e.g. [6–8]), which may then merge due to subsequent losses of angular momentum (eventually due to gravitational waves). These mergers will strongly affect the gas accreting onto the BHs, leading to electromagnetic (EM) signals from a purely gravitational phenomenon.

To test galactic merger theories, we need to model the EM signals from SMBH mergers. We can then search for these signals to determine if SMBHs merge at the rate and mass-ratio the current theories predict. It is therefore critical to understand the dynamics of accreting magnetized plasma around supermassive black holes during a merger, and the light emitted during these spectacular events, which will be accompanied by an emission of a massive amount of energy in gravitational waves. Intensive, high-cadence astronomical surveys such as Pan-STARRS or the planned LSST make detecting these rare mergers of supermassive black hole binaries (SMBHBs) more likely, yet accurate theoretical predictions for the SMBHB's EM counterpart remain to be done. The goal of this project is to develop accurate theoretical estimates of the EM signature of SMBHBs through GRMHD simulations using the state-of-the-art `harm3d` [9] code, developed by Noble for accretion disk simulations.

These science goals can only be reached with the construction of a highly-sophisticated computational framework. High-performance computation is essential, because tracking the dynamics of these events involves simultaneously solving the equations of radiation transport and magneto-hydrodynamics (MHD) in a general relativistic spacetime, as well as the highly nonlinear Einstein Field Equations.

The primary goals of our project are to: 1) improve the runtime efficiency and load balance of `harm3d` simulations; 2) parallelize `bothros` with OpenMP and add new infrastructure for distributing I/O and data processing effort over several nodes; 3) implement OpenMP throughout `GRHydro` and evaluate its performance on Blue Waters. In Section II, we provide a mathematical description of our problems of interest. We then describe our results in Section III followed by a summary and concluding remarks in Section IV.

*Electronic address: `scn@astro.rit.edu`

II. THEORETICAL BACKGROUND

Our primary science objectives are to perform accurate simulations of the dynamics of magnetized matter in the strong-field regime of gravity. We therefore rely on our ability to solve the nonlinear, coupled PDEs of MHD on curved manifolds (i.e. in the context of the theory of general relativity), and on processing this simulation data with general relativistic radiative transfer. Below, we provide a brief description of the equations involved in these two kinds of calculations.

Before we begin, we note that Greek indices (e.g., $\mu, \nu, \lambda, \kappa$,..) represent space and time components and range over $[0, 3]$, while Roman indices (e.g., i, j, k, l,...) represent spatial components that range over $[1, 3]$. Our physical system resides in a curved spacetime described by the invariant line element

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu \tag{1}$$

where dx^μ are displacements along the space-time extents of our coordinate system, and $g_{\mu\nu}$ is the covariant form of our spacetime metric which describes the gravitational curvature of our system's environment. Our present research goals involve binary black hole scenarios where $g_{\mu\nu}$ is a function of both space and time, necessitating us to advance the metric with the fluid.

A. GRMHD

Different GRMHD codes use different formulations of the GRMHD equations. For instance, there exist differences between formulations of `harm3d` and `GRHydro`, and these differences are discussed in Appendix A of [10]. We present only the formulation used in `harm3d` here. The equations of motion (EOM) derive from the local conservation of baryon number density, the local conservation of 4-momentum, and the induction equations from Maxwell's equations (please see [9] for more details). They take the form of a set of conservation laws:

$$\partial_t \mathbf{U}(\mathbf{P}) = -\partial_i \mathbf{F}^i(\mathbf{P}) + \mathbf{S}(\mathbf{P}) \tag{2}$$

where $\mathbf{U}$ is a vector of the conserved variables, $\mathbf{F}^i$ are the fluxes, and $\mathbf{S}$ is a vector of source terms.

$$\mathbf{U}(\mathbf{P}) = \sqrt{-g}\left[\rho u^t,\, T^t{}_t + \rho u^t,\, T^t{}_j,\, B^k\right]^T \tag{3}$$

$$\mathbf{F}^i(\mathbf{P}) = \sqrt{-g}\left[\rho u^i,\, T^i{}_t + \rho u^i,\, T^i{}_j,\, \left(b^i u^k - b^k u^i\right)\right]^T \tag{4}$$

$$\mathbf{S}(\mathbf{P}) = \sqrt{-g}\left[0,\, T^\kappa{}_\lambda \Gamma^\lambda{}_{t\kappa} - \mathcal{F}_t,\, T^\kappa{}_\lambda \Gamma^\lambda{}_{j\kappa} - \mathcal{F}_j,\, 0\right]^T \tag{5}$$

where g is the determinant of the metric, $\Gamma^\lambda{}_{\mu\kappa}$ are the metric's Christoffel symbols, $B^i = {}^*F^{it}/\sqrt{4\pi}$ is our magnetic field (proportional to the field measured by observers traveling normal to the spacelike hypersurface), ${}^*F^{\mu\nu}$ is the Maxwell tensor, u^μ is the fluid's 4-velocity, $\mathcal{F}_\mu = \mathcal{L} u_\mu$ is the flux of radiative leakage, $\mathcal{L}$ is the fluid-frame rate of energy radiated away (by photons), $b^\mu = \frac{1}{u^t}\left(\delta^\mu{}_\nu + u^\mu u_\nu\right) B^\nu$ is the magnetic 4-vector or the magnetic field projected into the fluid's co-moving frame, $W = u^t/\sqrt{-g^{tt}}$ is the fluid's Lorentz function, $T_{\mu\nu}$ is the MHD stress-energy tensor defined as

$$T_{\mu\nu} = (\rho h + 2p_m)\, u_\mu u_\nu + (p + p_m)\, g_{\mu\nu} - b_\mu b_\nu \quad , \tag{6}$$

$p_m = b^\mu b_\mu/2$ is the magnetic pressure, ρ is the rest-mass density, $h = 1 + \epsilon + p/\rho$, p is the gas pressure, and ϵ is the specific internal energy. The "primitive" variables are $\mathbf{P} = \left[u, \rho, u^i\right]$. The system is closed with the ideal-gas equation of state: $P = (\Gamma - 1)\,\rho\epsilon$.

In `harm3d`, the EOM are integrated using modern high-resolution shock-capturing finite volume schemes. Volume averages are integrated forward in time. Left and right values of $\mathbf{P}$ are found with shock-capturing piecewise parabolic interpolation. The fastest characteristic wave speed at each cell interface is used in a Lax-Friedrichs-like flux formula. The FluxCT constrained transport scheme is used to maintain the absence of magnetic monopoles [11, 12]. After the conserved variables are advanced in time, the primitive variables are recovered by inverting the set of nonlinear algebraic equations $\mathbf{U} = \mathbf{U}(\mathbf{P})$ via a Newton-Raphson procedure. Physical boundary conditions are employed at the edges of the global domain. The simulation is parallelized by decomposing the global domain into subdomains that are assigned to individual MPI tasks. Ghost zone data is shared between neighboring subdomains so that there is sufficient boundary information available for a MPI task to integrate in time its volume's worth of data locally.

The Christoffel symbols are calculated from derivatives of the metric:

$$\Gamma^\mu{}_{\nu\kappa} = \frac{1}{2} g^{\mu\sigma}\left(\partial_\nu g_{\kappa\sigma} + \partial_\kappa g_{\nu\sigma} - \partial_\sigma g_{\nu\kappa}\right) \quad . \tag{7}$$

For simulations of accretion disks about binary black holes, we use a high-order Post-Newtonian approximation that leads to lengthy expressions for the metric components. Finding closed-form expressions of the Christoffel symbols can only be accomplished using symbolic manipulation software. We avoid this approach as it will likely be far less efficient that our method of choice, which is to calculate Eq.7 via finite difference approximation for both space and time derivatives.

B. General Relativistic Radiative Transfer

In order to predict the radiative emission from a disk simulation, one needs to know how the light travels through the curved spacetime and how its energy spectrum evolves. These two aspects are described, respectively, by the geodesic equations of motion and the radiative transfer equation.

In the theory of general relativity, light travels on curved paths. These paths are determined by the geodesic equations, which can be cast into a set of 8 first-order linear ODEs:

$$n^\mu = \frac{\partial x^\mu}{\partial \lambda} \quad , \tag{8}$$

$$\frac{\partial n^\mu}{\partial \lambda} = -\Gamma^\mu{}_{\nu\kappa} n^\nu n^\kappa \quad , \tag{9}$$

where n^μ is the 4-velocity of a photon (or collection of photons), $x^\mu(\lambda)$ are the coordinates of its worldline (or trajectory), and λ is the affine parameter that parameterizes points along the curve.

The Lorentz invariant radiative transfer equation is

$$\frac{\partial I}{\partial \lambda} = j - \alpha I \quad , \tag{10}$$

where I, α, and j are the Lorentz invariant intensity, absorption coefficient, and emissivity, respectively, and I_ν, α_ν, and j_ν are the values measured by some observer that measures the photon to have frequency ν. The relationship between these two sets of quantities is

$$I = \frac{I_\nu}{\nu^3} \quad , \tag{11}$$

$$j = \frac{j_\nu}{\nu^2} \quad , \tag{12}$$

$$\alpha = \alpha_\nu \nu \quad . \tag{13}$$

Here, we have chosen the affine parameterization such that

$$\frac{\partial x^\mu}{\partial \lambda} = u^\mu \Longrightarrow K = -\frac{1}{c\nu} v_\mu u^\mu = 1 \quad , \tag{14}$$

where v^μ is the 4-velocity of a local timelike observer that measures the photon's frequency to be equal to ν. The expression for $d\lambda$ with respect to u^a is:

$$d\lambda = \frac{v_\mu dx^\mu}{v_\nu u^\nu} = \frac{v_\mu dx^\mu}{-\nu c} \quad . \tag{15}$$

For integrations of bolometric luminosity, the absorption term in the radiative transfer equation is sometimes ignored and emissivity is integrated through the entire domain or only up from the photosphere. The photosphere location is surface at some number of optical depths from the camera. We can easily identify the optical depth τ as integrals in the exponents:

$$\tau(\lambda) = K \int_{\lambda_0}^{\lambda} \alpha \, d\hat{\lambda} \quad . \tag{16}$$

It is interesting to note, that the optical depth is actually a Lorentz invariant:

$$d\tau = K\alpha d\lambda = \frac{K\alpha}{K\nu} ds = \frac{\alpha}{\nu} ds = \alpha_\nu \, ds = d\tau_\nu \tag{17}$$

In `bothros`, the geodesic equation is integrated from the simulated camera back in time through the simulation volume. This is because geodesics follow nontrivial paths that are difficult to predict. If one chooses to integrate the geodesics forward in time from the source to the camera, one would need many more geodesics to achieve the equivalent resolution. Once a geodesic is found, simulation data is interpolated along it and the radiative transfer equation is integrated forward in time back towards the camera.

The integration of the radiative transfer equation demands we read in every 3-d snapshot of simulation data from disk and communicate it to those processors performing the ray tracing. The entire procedure involves the following tasks:

- Evaluate the metric at various spacetime points;
- Use finite differences of the metric to calculate the right-hand side of the ODEs;
- Use a variable stepsize ODE integrator (e.g., Burlisch-Stoer) to solve the 8 coupled ODEs of the geodesic equations;
- Interpolate simulation data along the ray;
- Calculate the absorption and emissivity coefficients from the interpolated simulation data;
- Integrate the radiative transfer equation for the observed intensity;
- Output the intensity for each ray.

III. FINDINGS AND ACHIEVEMENTS

A. `harm3d`

As our primary research focus on Blue Waters has shifted to projects involving `harm3d`, we have concentrated our efforts to accomplish its goals first. We will therefore spend the bulk of our discussion on `harm3d` developments.

1. *Load Balancing*

Unlike our previous calculations [13], our next generation of circumbinary accretion disk simulations on Blue Waters place the black holes on the numerical domain. The metric is calculated using different approximations depending on the cell's proximity to the black holes:

- Inner Zone (IZ): the immediate volume about each black hole;
- Near Zone (NZ): the volume encompassing the binary out to a finite radial distance from the binary's center of mass;
- Far Zone (FZ): the volume beyond the NZ to infinity.

The metric in the IZ is that of a distorted single black hole; the metric of the NZ is a high-order Post-Newtonian approximation to a binary spacetime; and the metric of the FZ is that of flat space plus a wave perturbation. They are stitched together in buffer regions using transition functions. In these buffer zones, both neighboring metrics need to be calculated. In terms of computational effort, the IZ metric evaluation is about four times as expensive as that of the NZ. The FZ metric is approximately as costly as the NZ metric. Our profiling measurements indicate that the metric evaluation is—by far—the most expensive part of the calculation, so the metric's computational cost is an accurate indicator of a particular cell's contribution to the entire simulation's effort. The original version of `harm3d` was hard-coded to decompose the global domain uniformly across the MPI tasks. Since the different zones vary in cost by factors of several and these zones are localized in space, then a uniform decomposition scheme would produce a significantly imbalanced load distribution across the MPI tasks. We aim to alleviate this imbalance by considering the distribution of cost over the domain when decomposing the domain, resulting in a nonuniform (in terms of number of cells per MPI task) decomposition.

In order to profile the cost of each zone's metric evaluation, we performed a series of simulations of a single black hole (a "Kerr" black hole) surrounded by a disk of magnetized gas. We compared the performance of this control simulation to two others: the "Kerr+NZ" case where we added a call to the routine that calculates the NZ metric alone, and the "Kerr+NZ+IZ" cases where we added a call to the routine for calculating both the NZ and IZ metrics. These additional calls had no purpose other than to measure the impact on the simulation's runtime. We measured the runtime rate or number of zone-cycles (or cell updates) performed per floating point processor (FPU, or Bulldozer core) per second (aka "zone-cycles/sec/FPU"). The comparisons are illustrated in Figure 1. We find that the "NZ+IZ" calculation can cost approximately a factor of three times that of the NZ-only calculation. Incidentally, we find very little difference in performance between the Cray and PGI compilers.

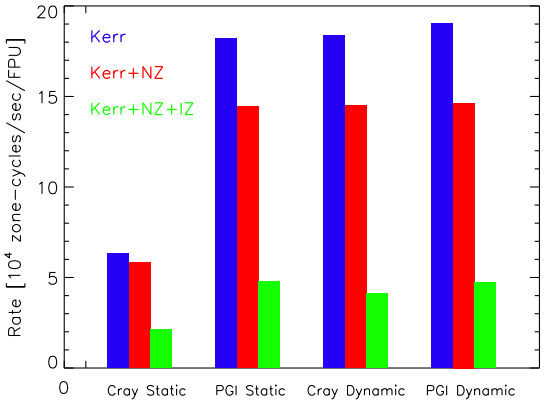

FIG. 1: Runtime efficiency or update rate of `harm3d`'s performance in terms of zone-cycles per second per core when using different compilers (Cray or PGI), code versions ("Dynamic" or "Static"), and metric routines ("Kerr,", "Kerr+NZ," or "Kerr+NZ+IZ"). "Static" and "Dynamic" refer to the different version of `harm3d` that—respectively—involve static and dynamic memory allocation.

A first step of implementing our nonuniform decomposition procedure was to switch from static memory allocation in `harm3d` to dynamic memory allocation. This required surveying the entire source code and rewriting a major portion of it. We anticipated that dynamic memory allocation would have a slight detrimental impact on our runtime efficiency as the compiler no longer has the freedom to optimize with respect to array dimensions. Surprisingly, we find that the dynamic version performs as well as—and sometimes better than—the static version (Fig. 1). The performance results encourage us to use the dynamic version for even the uniform decomposition runs as there are other advantages to using it, e.g., no need to recompile the code when changing the simulation's resolution or the number of MPI tasks. Also, we note that the static version of the code would suffer a segmentation fault when compiled with the default optimization settings of the Cray compiler. For the "Cray Static" run of Fig. 1, we were forced to use a less aggressive set of compiler options, which is why that run performed so much worse that the others. The change to dynamic memory allocation somehow cured this problem; the cause of it still remains elusive to us.

Our load balancing procedure first assumes that the computational cost per cell is known, so that a subdomain's total cost can be easily calculated by taking the sum over all its cells. The algorithm is rather simple:

1. Start with the global domain undivided;
2. Order the subdomains by cost;
3. Bisect the most expensive domain along its longest extent;
4. Assign a new MPI task to one of the new subdomains, while keeping the original MPI task assigned to the other;
5. Determine the neighbors of the two new subdomains;
6. Repeat Steps 2 - 5 until all processors have been assigned or an adequate load distribution has been achieved.

There are several advantages to this algorithm. First, we are likely to have a relatively even load balance with any number of MPI tasks; for instance, uniform decompositions require that the number of subdomains is an integer factor of the number of cells. Second, bisecting a subdomain's longest dimension favors subdomains with smaller ratios of surface area to volume, thereby minimizing the relative amount of inter-domain communication. Third, its simplicity minimizes the risk of errors in its development and allows us to refine the procedure after some time experiencing its performance.

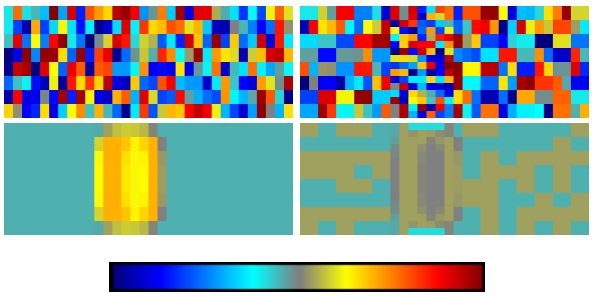

FIG. 2: Top row: illustrations of domain decomposition, with colors used to differentiate adjacent subdomains. Middle row: relative cost offset (R_i) distribution over subdomains. Bottom row: linear color map used in the middle row's plots, ranging over $[-4, 4]$ in terms of R_i. Left column: uniform decomposition. Right column: nonuniform decomposition using our load balancing algorithm. In both decompositions, 256 MPI tasks were used.

Before implementing the full load balancer within `harm3d`, we performed a simulated test of the algorithm in a standalone program. The test constrained the decomposition to only two spatial dimensions. An illustration of its performance is given in Fig. 2. Note that the relative cost offset used to visualize the load imbalance is defined as

$$\tilde{R}_i = \frac{C_i - \bar{C}}{\bar{C}} \tag{18}$$

where C_i is the cost of i^{th} subdomain, and $\bar{C}$ is the average cost per subdomain. We find that the algorithm is rather successful. The nonuniform decomposition results clearly show how smaller domains cover the costlier regions while the larger domains cover the less costly regions, leading to a significantly more balanced distribution over the uniform decomposition. We also note that we are likely to maintain a two-dimensional cost distribution as the costliest IZ regions (about each orbiting black hole) move through the grid several dozens of times over the course of a simulation. Rebalancing the load as the black holes move would likely cost more overhead than what it would save, and would require significantly more development effort. We therefore intend to average the 3-dimensional spatial cost distribution in time, which results in a 2-dimensional cost distribution as the black holes primarily move along one spatial dimension.

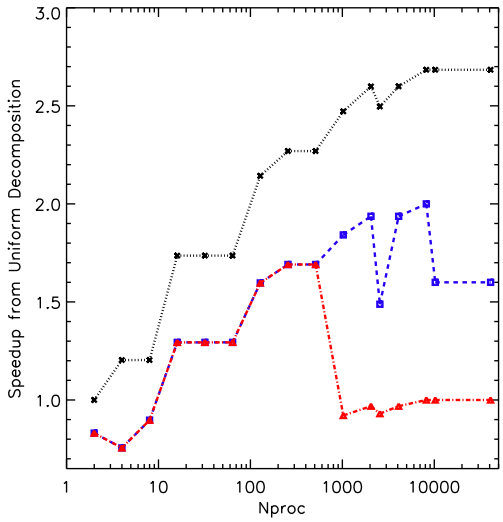

FIG. 3: Ratio of the runtime rates between a given decomposition method and uniform decomposition versus the number of processors (or subdomains) for a variety of decomposition models: perfect load balance (black exes), 2-d nonuniform decomposition (red triangles), and 3-d nonuniform decomposition (blue squares). The global number of cells used in this simulated scaling experiment was kept fixed as the number of cores/subdomains were increased. Saturation of the 2-d nonuniform method occurs at 1000 cores.

Even though we intend to constrain the cost distribution to two spatial dimensions, we will still allow for nonuniformly decomposing the the subdomains in all three spatial dimensions. The additional degree of freedom to bisect the subdomains is necessary to scale our code to more than 10^3 cores with a balanced load. The advantage of 3-d nonuniform decomposition is seen in Figure 3. Here we plot the speedup in update rate over uniform decomposition method for three different cases: 1) perfect load balance where the global load is exactly evenly split amongst the cores (exes); 2) nonuniform decomposition in 2-dimensions with the third dimension uniformly decomposed (triangles); 3) nonuniform decomposition in all three dimensions (squares). The speedup is measured as the ratio of the rates of the slowest subdomain for a particular decomposition to the rate of the slowest subdomain from a 3-d uniform decomposition. We find that the 2-d nonuniform decomposition saturates at a particular FPU count and becomes no better than uniform decomposition. The saturation is caused by the discrete limit in grouping an integer number of cells; we set the minimum extent a subdomain's dimension can have to be 12 cells. If this limit were any smaller, a subdomain of this minimum size would be communicating nearly its entire volume as ghost zone data to neighboring subdomains—leading to a prohibitive communication load. The minimum extent constraint limits the 2-d nonuniform decomposition algorithm from adequately load balancing. That is, once the 2-d nonuniform method saturates, the most expensive subdomain will have this minimum extent in the 2-dimensions and so cannot be subdivided further to improve the load balance. With 3-d nonuniformity, we can subdivide its third dimension nonuniformly and continue to balance the load. We note that the perfect limit is likely unreasonable as we are constrained to discretely partition the load and maintain block-shaped subdomains. Even though the 3-d nonuniform decomposition is not at the ideal limit, it can reach a factor of two speedup over the uniform scheme. This remarkable speedup factor is even seen with our use of a cost imbalance that is less severe than what the latest profile suggests we will see in practice (Fig. 1). Our results have encouraged us to begin implementing the load balancer within `harm3d`.

2. *Performance on Blue Waters*

As we developed the load balancing infrastructure, we performed a variety of scaling experiments on Blue Waters. Our results present a baseline from which to improve, and they inform us how we may best tailor our code's execution on Blue Waters. We performed a weak scaling test, wherein the per core load was held constant while the global problem size increased with core count; and a strong scaling experiment, wherein the per core load decreased with increasing core count while the global problem size was held constant. The calculation involves a circumbinary magnetized disk using the NZ metric, using the same initial conditions of [13]. The small problem size used in the strong scaling experiment is the same resolution as that of the production runs reported in [13], while the large problem size series uses 8 times the number of cells or twice the number of cells per dimension. Each run lasted for an hour to minimize the influence a run's initialization has on its total runtime; this initialization procedure never exceeds 50 seconds and is most often about 30 seconds long.

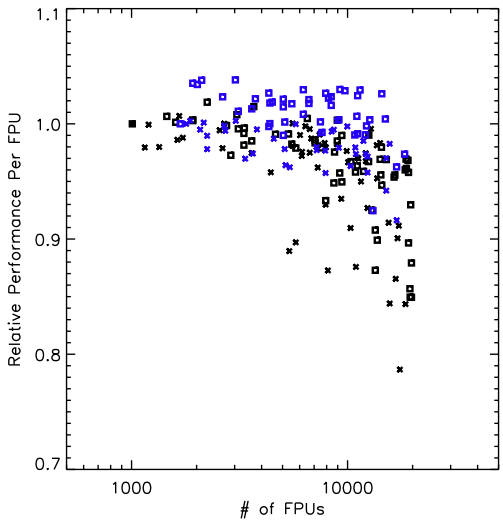

FIG. 4: Ratio of update rates between a given case and the smallest core count instance versus total core count for two different per subdomain cell extents and two different compilers: $16 \times 16 \times 16$ (black), $20 \times 16 \times 25$ (blue), PGI compiler (squares), Cray compiler (exes).

The weak scaling tests use two different per core subdomain dimensions: a symmetric one with $16 \times 16 \times 16$ cells, and an asymmetric one with $20 \times 16 \times 25$ cells (Figure 4). Note that the subdomain extents are listed in the order of outermost loop to innermost loop (i.e. the last dimension is the "fastest" running dimension). This test primarily measures the performance of the MPI communication routines and how efficiently they are at sharing data between processors as the total core count increases, while keeping most everything else held fixed. We note that there is a subtlety with the weak scaling test in that even though two points with the same core count have the same number of cells, the global number of cells per dimension may be different. This difference affects the physics of the simulation (e.g., the height of

the disk may be better resolved in one run while the other's disk's radial extent is better resolved) which then could affect the runs' runtime rates (e.g., a more rapidly changing region will have a harder time recovering its primitive variables). It can also affect the mapping between location in the problem space and the cluster node on which it lives. Because of asymmetries in Blue Waters' interconnect, this difference in task distribution may alter the runtime performance. We find that the symmetric case performs worse in general than the asymmetric configuration. One reason could be that each core has more work to do in the asymmetric case (the number of cells per core is larger in this case), so these runs have less communicating to do compared to their intracore computation. Another reason is that the asymmetric case has a larger extent in the fastest loop dimension and so can vectorize more operations. And, it appears as though the asymmetric runs exhibit less scatter for the same core count, implying that there is less of an effect from the global resolution and/or task distribution over the interconnect. The increase in runtime performance at large core counts indicates that more attention needs to be paid to `harm3d`'s parallel performance for very large runs.

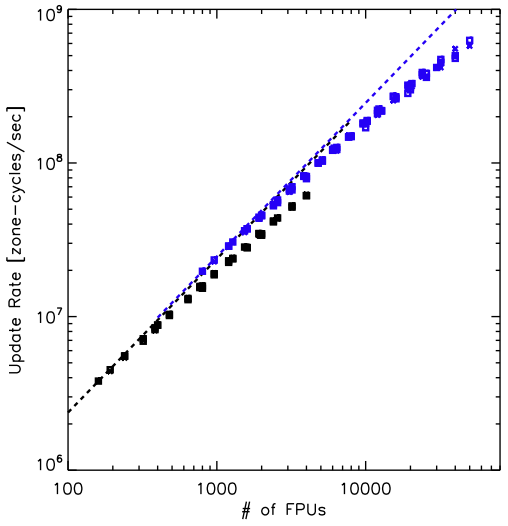

FIG. 5: Update rates versus core count for two different problem sizes and two different compilers: $300 \times 160 \times 400$ (black), $600 \times 320 \times 800$ (blue), PGI compiler (squares), Cray compiler (exes).

The strong scaling experiments demonstrate how effective the code and the cluster act in concert to divide a problem of fixed size (Figures 5 - 6). In the test, we have often chosen multiple ways of decomposing the domain across the given number of cores, in order to see if the way we decompose the domain

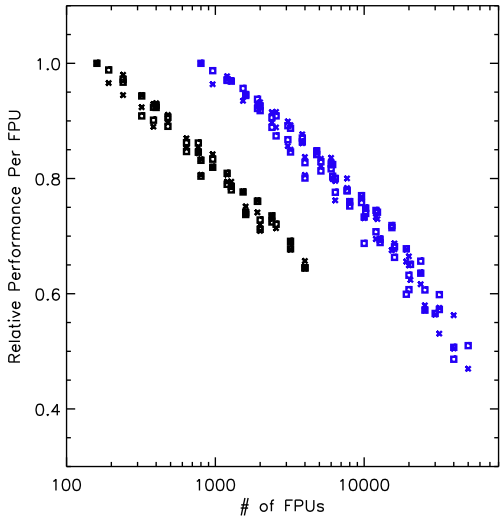

FIG. 6: Ratio of update rates between a given case and the smallest core count instance for the strong scaling test of Fig. 5.

has an affect. We find that both problem sizes follow similar trends which indicates that the differences in the physical problem are not significant, consistent with the weak scaling results. From the vertical scatter in the plots, we find that different domain decompositions lead to a variation of runtime performance of a few percent, suggesting that it is not a significant contributing factor. For each problem size, we find that it takes an order of magnitude increase in core count to yield a 75% performance penalty. This level of scaling efficiency is surprisingly good as no special accommodation was made to optimally distribute the runs over the cluster's interconnect; using Cray's advance task placement tools may significantly improve this performance. For example, it could be that the tasks were distributed remotely across the interconnect's 3-d torus topology. Some other PRAC teams report better scaling performance using similar codes, indicating to us that additional investigation could produce improved performance.

B. `bothros`

Even though post-processing `harm3d` simulation data with `bothros` costs only a small fraction of the particular simulation's computational cost, the ray-tracing procedure—as it currently stands—will not scale to much larger problem sizes than we currently use. The primary bottleneck is reading data from the Lustre filesystem. Previous simulations typically use 10 TB of storage, with future runs projected at sev-

eral times this. Since `bothros` is currently a serial code, ray tracing involves multiple redundant reads of the data from disk with as different instances must read the data independently. The primary goal of our parallelization of `bothros` is to dramatically reduce the number of times a simulation time slice (i.e. 3-d volume of grid function values) is read.

The size of the problem is based on how many light rays we integrate. We integrate a series of level sets or synchronous planes of photons from different vantage points at the same radius. Each plane of photons is composed of $N_x \times N_y$ pixels or points of origin for photons. There are N_t snapshots per vantage point. There are $N_\theta \times N_\phi$ vantage points. Each ray-tracing run therefore requires $N_{\rm pixels} = N_t N_x N_y N_\theta N_\phi$. Typical numbers are $N_x = N_y = 300$, $N_\phi = 2$, $N_\theta = 8$, $N_t = 2000$, or $N_{\rm pixels} \simeq 3 \times 10^9$ photons. In the original version of the code, each snapshot is decomposed into frames or sets of photons so that the computation can fit into memory. In our new scheme, each distribution of effort will be some segment in the space of $N_x \times N_y \times N_t \times N_\theta \times N_\phi$. We let (i, j, n, l, m) index a photon over $N_x \times N_y \times N_t \times N_\theta \times N_\phi$ space, respectively, where decomposition of effort is performed slowest to fastest going from the leftmost to the rightmost dimensions or indices. This way the photons local in memory will also more likely be closer in spacetime as well, which will minimize the amount of simulation data needed by the local MPI task. Note that each photon represents a geodesic path along which simulation data must be interpolated, and along which the radiative transfer equation is integrated back to the camera to take a particular time's snapshot.

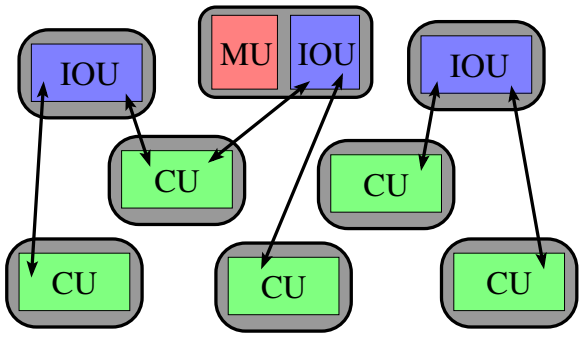

FIG. 7: Strategy 1: Diagram of how different units or roles are assigned to threads or MPI tasks, how they communicate information, and how they are distributed over nodes. Grey boxes indicate nodes, and the colored rectangles represent the units (threads or MPI tasks) assigned to that node. In this strategy, only one unit is assigned per node and a role's responsibilities are distributed on the node via OpenMP threads. Units communicate with each other using simple non-threaded MPI calls (black arrows).

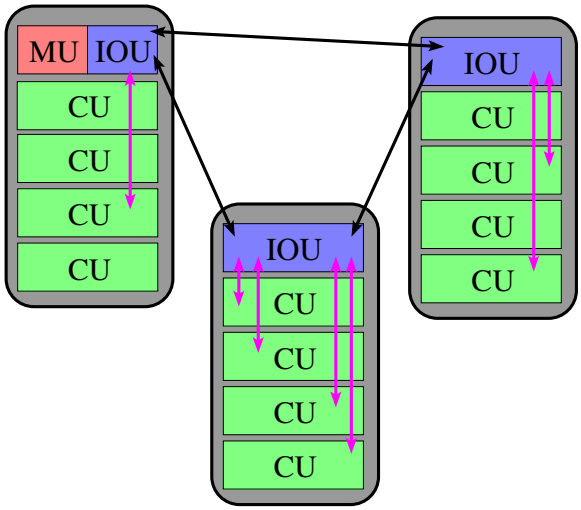

FIG. 8: Strategy 2: Diagram of how different units or roles are assigned to threads or MPI tasks, how they communicate information, and how they are distributed over nodes. Grey boxes indicate nodes, and the colored rectangles represent the units (threads or MPI tasks) on that node. In this strategy, units are assigned to threads on each node. Internode communication (between IOUs) is performed via MPI calls from a specified thread (black arrows). A node's CU threads retrieve data from other nodes through exchanges with that node's IOU thread (magenta arrows). This way, there is only one thread that performs MPI calls.

We have developed two strategies for parallelizing a `bothros` run, illustrated by diagrams in Figures 7 - 8. The strategies define an MPI task or OpenMP thread to have one or more of three possible roles:

- Master Unit (MU): responsible for evaluating the problem size, assigning units their roles, informing the IOUs what data is available, writing the results to disk, and ending the run;
- I/O Unit (IOU): responsible for reading data from disk, serving it to CUs, and cycling through the available time slices on disk while coordinating with other IOUs to prevent redundant reads;
- Compute Unit (CU): responsible for asking and receiving data, integrating the geodesic and radiative transfer equations, and return the results of the ray tracing to the MU;

Each unit or role will be assigned a subset of the computational tasks. A processor may have more than one unit's worth of duties (e.g., the MU may also be an IOU). The MU, once performing managerial duties at the beginning of a run, will be assigned IOU tasks. CUs will likely always perform CU duties and not share in other roles.

We have tested the threading procedure for Strategy 2 (Figure 8) without the MPI communication step implemented and with only one node. Specifically, we parallelized the computation step with the with disk read step via OpenMP directives, so that other threads may process the current one while the next time slice's data is read in. What remains to be written is the infrastructure code to handle the MU duties and IOU communication routines. Our plan is to finish this development while our `harm3d` production simulations run on Blue Waters so that it will be ready once the data is available.

C. `GRHydro`

The `GRHydro` code contained within the Einstein Toolkit is a publicly available set of routines that can evolve the equations of general relativistic hydrodynamics (GRHD) or magnetohydrodynamics (GRMHD) [14]. It builds upon other functionality contained within the Toolkit, including routines that provide grid-based structures, parallelization infrastructure, data I/O and analysis routines, as well as scientific modules that provide the underlying evolution system for the spacetime metric itself (while the evolution set for the latter is similar in some ways to the set describing hydrodynamical evolution, the smooth nature of the fields allows for simple and easily implemented finite-differencing techniques that are not applicable to non-smooth fluid configurations). In the past year, the Einstein Toolkit consortium released a new version of the code that includes many new routines for simulating MHD for numerical relativity applications [10]. Some of our PRAC team members were integral to the development of this code. As it was written, much attention was paid to the incorporation of threading (OpenMP) throughout the new `GRHydro` modules.

IV. SUMMARY AND CONCLUSION

In this report, we have presented a description of the many significant accomplishments made toward our Subaward PRAC project. Our `harm3d` results demonstrate satisfactory progress toward performing efficient calculations of circumbinary accretion disks around supermassive black holes. Our nonuniform decomposition algorithm was demonstrated to function as expected with significant improvement in load balance for approximate models of the cost distribution. Current work is underway to incorporate the load balancer in our new dynamic memory allocation version of `harm3d`. Further, strong and weak scaling experiments were performed to demonstrate the parallel efficiency of `harm3d`. Its performance is adequate, but additional avenues (e.g., advanced placement tools, profiling of MPI calls) need to be explored that may result in improved performance. In addition, we have described our parallelization strategies for `bothros`, and are underway at implementing these ideas.

There are a variety of future directions that can be explored. First, once all the new pieces are implemented in `harm3d`, an extensive set of scaling and profiling experiments will be run on Blue Waters using production-level simulation configuration to evaluate `harm3d`'s ultimate performance. Additional adjustments to MPI calls or the load balancer will likely be needed once these experiments have been performed. Utilizing Cray vast pool of tools will be critical in this phase. Second, regarding the load balancing procedure, it would be convenient to have infrastructure in place for measuring a subdomain's computational cost (e.g., with timers embedded throughout the code). With such a scheme, a simulation could measure its performance and redecompose its domain based on this data automatically. Further, such a scheme could potentially adapt for a heterogeneous computing environment, placing less load on slower nodes. Cell scale cost data may be hard to measure reliably with timers, so we will need to explore ways in which we can use subdomain-scale measurements to inform us about the cell scale performance. Third, even though preliminary tests imply threading certain routines in `harm3d` does not provide substantial improvement in performance, further experiments should be made to be sure. Possibly, gains will only be dramatic for the largest runs at scales not explored in our initial tests.

V. ACKNOWLEDGMENTS

We thank Jing Li of the NCSA for her tireless efforts to help us better use Blue Waters, solve problems we encountered along the way, and perfect our PRAC reports. We also thank the various technical staff at NCSA and Cray that improved our experience on Blue Waters along the way and have closed our mass of tickets. The authors received NSF support from NSF PRAC subaward 2077-01077-26 (Prime: OCI-0725070) and PRAC OCI-0832606. S.C.N., M.C, and Y.Z. also acknoweledge support from AST-1028087 for related scientific research with `harm3d`. M.C. J.F and Y.Z. also acknoweledge PHY-0903782, PHY-1212426 and PHY-0969855 for work in relation with the `Einstein Toolkit` and GRMHD.

[1] K. Gultekin, D. O. Richstone, K. Gebhardt, T. R. Lauer, S. Tremaine, et al., Astrophys. J. **698**, 198 (2009), 0903.4897.

[2] D. J. Mortlock, S. J. Warren, B. P. Venemans, M. Patel, P. C. Hewett, et al., Nature **474**, 616 (2011), 1106.6088.

[3] J. H. Krolik, *Active galactic nuclei : from the central black hole to the galactic environment* (Active galactic nuclei : from the central black hole to the galactic environment /Julian H. Krolik. Princeton, N. J. : Princeton University Press, c1999., 1999).

[4] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. White, Astrophys. J. **292**, 371 (1985).

[5] J. M. Bardeen, J. Bond, N. Kaiser, and A. Szalay, Astrophys. J. **304**, 15 (1986).

[6] M. Dotti, M. Colpi, and F. Haardt, Mon. Not. Roy. Astron. Soc. **367**, 103 (2006), astro-ph/0509813.

[7] M. Dotti, M. Colpi, and F. Haardt, in *Relativistic Astrophysics Legacy and Cosmology - Einstein's Legacy*, edited by B. Aschenbach, V. Burwitz, G. Hasinger, and B. Leibundgut (2007), p. 125.

[8] M. Colpi, S. Callegari, M. Dotti, S. Kazantzidis, and L. Mayer, AIP Conf. Proc. **924**, 705 (2007), 0706.1851.

[9] S. C. Noble, J. H. Krolik, and J. F. Hawley, Astrophys. J. **692**, 411 (2009), 0808.3140.

[10] P. Moesta, B. C. Mundim, J. A. Faber, R. Haas, S. C. Noble, T. Bode, F. Loeffler, C. D. Ott, C. Reisswig, and E. Schnetter, ArXiv e-prints (2013), 1304.5544.

[11] G. Tóth, Journal of Computational Physics **161**, 605 (2000).

[12] C. F. Gammie, J. C. McKinney, and G. Toth, Astrophys. J. **589**, 444 (2003), astro-ph/0301509.

[13] S. C. Noble, B. C. Mundim, H. Nakano, J. H. Krolik, M. Campanelli, Y. Zlochower, and N. Yunes, Astrophys. J. **755**, 51 (2012), 1204.1073.

[14] Einstein Toolkit home page: `http://einsteintoolkit.org`.