

Relativistic Astrophysics on the SiCortex Architecture

Erik Schnetter^{1,2,*} and Steven R. Brandt^{1,†}

¹*Center for Computation & Technology, Louisiana State University, Baton Rouge, USA*

²*Department of Physics & Astronomy, Louisiana State University, Baton Rouge, USA*

(Dated: 2009-03-16)

Numerical simulations of astrophysical systems are often the only way to gain insight into observational data, since observations are limited by the systems' large distances, and experiments are often impossible due to the systems' extreme physical conditions. This is especially true in relativistic astrophysics where the complex features of black holes or neutron stars can not be analyzed in other ways. Such simulations can use formidable amounts of computing resources, and parallel high performance computing systems are the only viable way to perform these simulations.

New parallel systems are projected to provide several Petaflop/s sustained performance for scientific applications, using new hardware architectures and new levels of parallelism to achieve this. Using such systems will likely require the use of higher levels of abstractions such as provided by the Cactus software framework, and/or require a shift to new programming models such as provided by ParalleX.

The SiCortex architecture provides a mixture of traditional features found on today's Linux clusters and novel properties that are also likely to be found on future systems. It can thus serve as stepping stone to help developers research algorithms and tools to ready their applications for Petascale computing.

I. INTRODUCTION

Gamma-Ray Bursts (GRBs) are intense narrowly-beamed flashes of γ -rays of cosmological origin; they are the most energetic events in the modern universe, yet their emission mechanisms remain poorly understood [1]. Understanding these ultra-energetic cosmic explosions and the astrophysical systems and physical processes leading up to them is of key importance for the astrophysical picture of the universe, not only impacting on the theory of stellar evolution, but on astrophysical cosmology as a whole, including, but not limited to the star-formation and chemical-enrichment history of the universe and the co-evolution of black holes and galaxies.

The central engines of GRBs are very likely intimately connected with regions of the universe near a compact object [1, 2], a black hole (BH) or neutron star (NS), where gravity is very strong and where according to Einstein's Theory of General Relativity (GR) the curvature of spacetime is large.

The realm of strong curvature is notori-

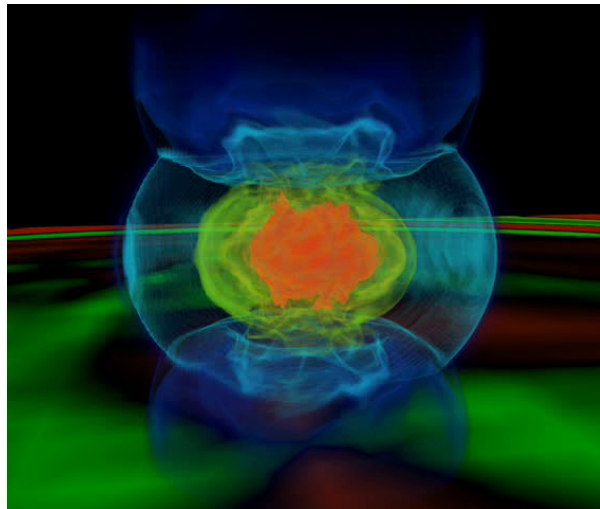


FIG. 1: Rotationally deformed protoneutron star formed in the iron core collapse of an evolved massive star. Shown are a volume rendering of the rest mass density and a 2D rendition of outgoing gravitational waves. Simulation by [3], image by R. Kähler (Zuse-Institute Berlin and KIPAC/SLAC).

ously difficult to investigate with conventional observational astronomy, and some phenomena and intricate dynamical details surrounding the GRB central engine might bear no observable electro-magnetic signature at all and may only

*<http://www.cct.lsu.edu/~eschnett/>

†<http://www.cct.lsu.edu/~sbrandt/>

be visible in neutrinos (if sufficiently close to Earth) or in gravitational waves – ripples of spacetime itself which are predicted by Einstein’s GR. Gravitational waves have not been observed directly to date, but gravitational-wave detectors such as LIGO [4] are in the process of reaching sensitivities sufficiently high to observe interesting astrophysical phenomena.

Hence, while astronomical γ -ray and post-GRB observations in lower-frequency bands of the electromagnetic spectrum have helped improving GRB phenomenology, the physical processes creating and powering a GRB remain inaccessible by the means of classical astronomy. Computational modeling is the single, most powerful tool for accessing and understanding GRB physics. Figure 1 presents a stellar collapse calculation carried out by our group.

Modeling GRBs is a true Petascale problem (see [5, 6]) and requires a multi-physics approach that operates on physical length scales from a few meters to millions of kilometers and on time scales from a few tenths of a microsecond to hundreds of seconds. Of central importance in any comprehensive GRB model will be dynamical general relativity in combination with magnetohydrodynamics, radiation transport of neutrinos and photons, as well as neutrino and nuclear microphysics. This multi-physics, hence multi-algorithm requirement puts the GRB modeling problem at the interface of disciplines and requires synergistic action of researchers from multiple fields of astrophysics, applied mathematics, and computer science.

This white paper briefly describes the Cactus software framework that we employ for our simulations (section II), lists our experiences with running on the SiCortex architecture (section III), outlines ParalleX, a new programming model for highly parallel applications (section IV), and concludes with our evaluation of the SiCortex architecture for our research (section V). We assume that our findings are applicable to a much wider range of applications which use similar computational methods for simulating and modeling many kinds of physical problems.

II. THE CACTUS SOFTWARE FRAMEWORK

Cactus [7, 8] is an open-source, modular, and portable programming environment for collaborative HPC computing. It was designed and written specifically to enable scientists and engineers to develop and perform the large-scale simulations needed for modern scientific discovery across a broad range of disciplines. Cactus is used by a wide and growing range of applications, prominently including relativistic astrophysics, but also including quantum gravity, chemical engineering, Lattice Boltzmann Methods, econometrics, computational fluid dynamics, and coastal and climate modeling [9–16]. The influence and success of Cactus in high performance computing was recognized with the IEEE Sidney Fernbach prize, which was awarded to Edward Seidel at Supercomputing 2006.

Development of Cactus and its associated components has been driven from the beginning by user requirements. This has been achieved by developing, supporting, and listening to a large user base. Among these needs have been ease of use, portability, support of large and geographically diverse collaborations, and the ability to handle enormous computing resources, visualization, file I/O, and data management. Cactus must also support the inclusion of legacy code, as well as a range of programming languages.

Some of the key strengths of Cactus have been its portability and high performance, which led to it being chosen by Intel to be one of the first scientific applications deployed on the IA64 platform, and Cactus’s widespread use for benchmarking computational architectures. For example, a Cactus application was recently benchmarked on the IBM BlueGene/P system at ANL and scaled well up to 131,072 processors.

We use Cactus as basis for applications that simulate systems containing black holes, neutron stars, binaries with these objects, and core-collapse supernovae. Cactus and its adaptive mesh refinement infrastructure Carpet [17–19] are also used by many other groups in the numerical relativity community for their research.

Achieving this goal required significant advances in algorithms and hardware. Some parts

of the Intel compiler’s optimization logic targeted the performance of Cactus specifically.

III. CASE STUDY: A GENERAL RELATIVISTIC EVOLUTION CODE ON THE SICORTEX ARCHITECTURE

The SiCortex architecture [20, 21] is a highly parallel distributed-memory supercomputing architecture. While having obvious similarities to the current conventional cluster supercomputing architecture found e.g. in the Cray-XT5 Kraken at NICS [22], the SiCortex architecture differs in several aspects. Of these, we find that two are most prominently visible to the end user, namely

- CPU architecture and
- single CPU performance.

Some other differences will be discussed in section V below.

A. SiCortex Architecture

The SiCortex compute cores use the MIPS architecture [23], which is different from the widely-used Intel architecture [24]. There exist two compiler suites for SiCortex, gcc and PathScale, where PathScale offers the highest performance. Having to use a different compiler than on other systems means having to cope with the limitations and defects of a new compiler. Such limitations and defects are present in all compilers, and a large software package such as Cactus is bound to encounter some of these in each new compiler that is used. Since the SiCortex CPU architecture is fairly new, the PathScale compiler is still seeing rapid development. This is good since existing problems can be quickly addressed, and is bad since this means that there probably quite a few to be addressed.

The other highly visible difference of the SiCortex architecture is its low single-core CPU performance. Many high-end CPU architectures achieve very similar single-core performances (e.g. AMD, IBM POWER, Intel), and users have come to expect to be able to measure the performance of a supercomputer in numbers of cores.

This comparison does not work well on SiCortex, which has been designed to have more numerous and less powerful cores than comparable machines.¹ We suggest that “performance per price”, or even “performance per Watt”, are far better measures when comparing different systems. Unfortunately, due to the complexities of hidden costs, vendor discounts, and total system design, these sorts of evaluations must be deferred to individual bids and cannot be tabulated in general.

The current SiCortex CPU architecture features six cores per node, running at 700 MHz, and delivering 2 Flop/cycle. Comparing this to other “typical” values, found e.g. in the XT5 Kraken at NICS (2.3 GHz, 4 Flop/cycle), leads to the expectation that a SiCortex core should about six times slower than a Kraken core. Our benchmark results (see below) confirm this. Of course, the actual performance of an application depends not only on CPU speed, but also on memory and network performance, so that the true performance of an application is difficult to predict.

The lower single-core performance of the SiCortex architecture means that an application has to use many more cores to achieve the same total performance, which taxes the application’s parallel scalability. Increasing scalability by a factor of six is non-trivial and requires substantial development work.

However, this apparent weakness can also be a strength. By providing a framework which simplifies and stresses scalability, and which makes it easier to benchmark and improve large codes, SiCortex provides a niche service in the laboratory as a near ideal training ground for codes to help them achieve scalability. This is especially relevant since future gains in computing power are expected to come primarily from increased parallelism, forcing many applications

¹ We find that people sometimes fail to appreciate being able to overwhelm a problem with these smaller cores, unconsciously using “performance per core” as a performance metric, and becoming disappointed when they see a 5,000 core SiCortex system delivering a performance comparable to a 1,000 core Intel architecture system.

to become much more scalable if they want to access this future additional computing power.

B. Porting Cactus to SiCortex

The MIPS architecture is somewhat unfamiliar for Cactus developers mostly used to workstations with Intel architectures. The PathScale compilers, the SiCortex MPI implementation, and the Slurm job scheduling software were all different from other systems.

However, libraries and utilities such as LAPACK, HDF5, PAPI, and PETSc were all available by default. These packages can be problematic to install, and are important to Cactus users, so that having these available out-of-the-box was convenient.

The highly portable nature of Cactus meant that there were few surprises while compiling, apart from encountering some PathScale compiler defects:

- The PathScale compiler offers a command-line switch for inter-procedural optimizations. This did not work for us (the compiler seemed to hang), so that we are not using this optimization.
- There was an error in processing certain continuation lines in Fortran 90 code, which has since been corrected in a new compiler release.
- Static initializers in C++ functions calling non-trivial constructors led to segmentation faults. This was also a problem on other architectures, so that we re-wrote the corresponding code to contain explicit if statements.

Such problems are typical when porting to a new architecture or a new compiler.

A new version of Cactus (4.0 beta 16) was released on February 2, 2009, with official support for the SiCortex architecture. We are developing a common set of abstractions that hides differences between various HPC systems and architectures in the *Simulation Factory* [25], and we added to it support for the SiCortex's Slurm job submission mechanism. We are now able to

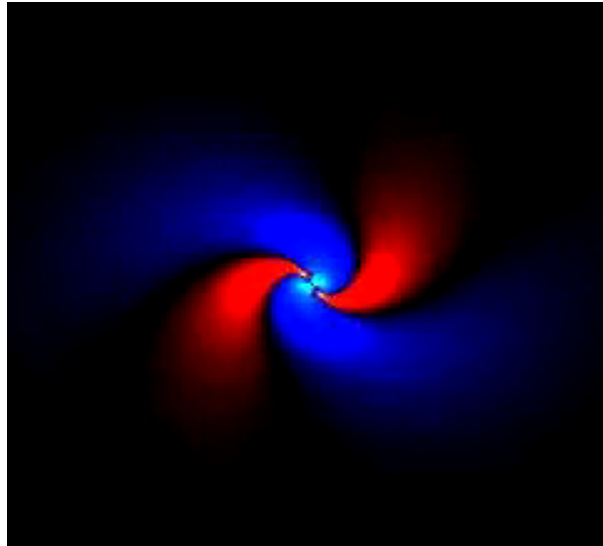


FIG. 2: Gravitational wave indicator Ψ_4 forming a two-armed spiral in the equatorial plane of a binary black hole inspiral. The two black holes are located near the white spots, indicating a region of high curvature. This simulation was performed on a SiCortex SC5832 system.

use SiCortex systems out-of-the-box for our research, in the same way as any other HPC system.

After porting our Cactus-based general relativity code, we demonstrated a binary black hole coalescence simulation at the Supercomputing '08 conference [26, 27]. This live demonstration featured an inspiralling binary black hole system, using 5,760 cores of a SiCortex machine on the show floor. Figure 2 shows the two-armed spiral of gravitational waves produced in this simulation.

C. Scaling Benchmark Results

The numerical relativity community has defined a number of scaling benchmarks for the Cactus framework [28, 29]. These are intended to measure the suitability of machines for numerical relativity simulations, and also to help improve the efficiency and scalability of the respective simulation codes. The benchmarks evolve a general relativistic spacetime for a certain number of iterations, using nine levels of mesh refinement (AMR) based on the Carpet AMR in-

System	Web	CPU speed [MHz]	Flop/cycle	TPP [MFlop/s]
HLRB II	[30]	1,600	4	6,400
Kraken XT4	[22]	2,300	4	9,200
Queen Bee	[31]	2,330	4	9,320
Ranger	[32]	2,300	4	9,200
SiCortex	[21]	700	2	1,400

TABLE I: Theoretical single-core peak performance (TPP) of the HPC systems used for benchmarking

frastructure [17–19]. The benchmarks are freely available, and we have collected results from a number of current high-performance computing systems, including a SiCortex SC1458 machine with 700 MHz cores in Houston, TX, to which we had access. We employed an updated version of the benchmark with improved parallel scalability that will soon be officially released.

These benchmarks are *weak scaling* benchmarks, so that the problem size increases with the number of cores. This corresponds to our use in numerical relativity simulations, where the typical size of simulations spans orders of magnitude, and our application has to be efficient for a wide range of problem sizes.

Table I lists the systems we used for benchmarking, as well as their theoretical single-core peak floating point performance, which depends on the CPU speed and the number of floating point operations that can be executed simultaneously.

Figure 3 shows the benchmark results. (The per-core benchmark workload for was reduced by a factor of two on the SiCortex system; this has been factored into the result shown.) Following our argument above that “performance per core” is not a good measure, we scale the benchmark results according to the theoretical single-core peak performance of these architectures. This means that this graph cannot be used to compare the systems’ absolute single-core performance, which is intentional since the total performance of these systems depends also on their number of cores, and on our application’s scalability on these systems.

It is clearly visible that Cactus/Carpet scales well to up to more than 12k cores. We also performed benchmarks that use a hybrid par-

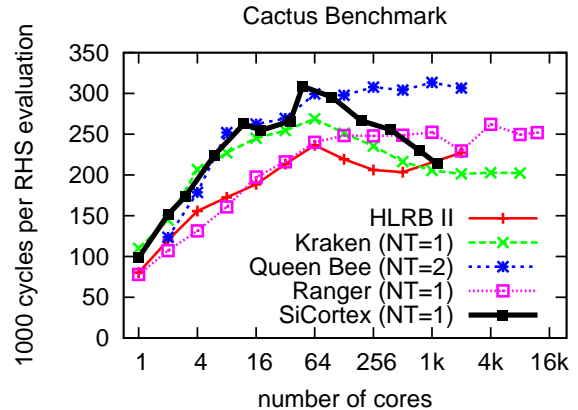


FIG. 3: Weak scaling AMR benchmark results for a set of current, large HPC systems and the SiCortex architecture. Shown is the number of cycles to evaluate the right hand side for one grid point, measured in terms of the theoretical peak performance of the corresponding system. (Note that this does not allow comparing the absolute performance of these systems.) Cactus/Carpet scales well to up to more than 12k cores. On Queen Bee, combining MPI and OpenMP led to increased performance.

allelization model by combining both MPI and OpenMP, and we find that this leads to a (slightly) increased performance on Queen Bee. Whether hybrid parallelization increased performance depends on many details of the application, and we use it in many production simulations since it increases performance there.

Apart from a lower per-core performance by a factor of approximately five, the SiCortex machine shows similar parallel scalability for the Cactus/Carpet benchmark. This factor of five can be entirely explained by the difference in theoretical single-core peak performances. We are therefore happy to report that Cactus/Carpet runs efficiently on the SiCortex architecture.

In detail, the benchmark results show some loss of scalability for small number of cores (up to about 64), and from then on approximately ideal scalability. In some cases (e.g. on Kraken or SiCortex), scalability even improves for very large number of cores. While we have not yet examined this effect in detail, we assume that it is due to the AMR interface conditions in our problem setup. As the number of cores increases, we increase the total problem size to test weak

scaling. Increasing the dimension of the domain by a factor of N increases the number of evolved grid points by N^3 , but increases the number of AMR interface points only by a factor of N^2 . As the problem size increases, the importance of the AMR interface points (and the associated computation and communication overhead) decreases.

IV. BEYOND MESSAGE PASSING: PARALLEX

The ParalleX research effort [33] is addressing the challenges of computing at the Exascale and beyond. This project will take computation beyond the communicating sequential processes model popularized by MPI, combining concepts developed by the computer science community over the last several decades, integrating and extending them to create a new model of computation. These new concepts include active messages [34] (parcels), partitioned global address space memory [35], user space threads, and lightweight concurrency primitives including atomic transactions. Collectively, they will address the challenges of latency hiding, scaling, and productivity.

The HPX runtime is the first implementation of the concepts embodied in ParalleX. It is already being ported to the SiCortex framework where it will leverage and exploit the threading and communication capabilities available there. It will make low level access to the RDMA and OS-bypass networking layer, minimizing latency and maximizing bandwidth.

V. CHALLENGES AND BENEFITS OF THE SICORTEX ARCHITECTURE FOR RELATIVISTIC ASTROPHYSICS

As the time and effort required to develop and debug scientific software has become the bottleneck in many areas of science and engineering, the difficulty of developing high-performance software is recognized as one of the most significant challenges today in the effective use of large scale computers. Traditional, simplified, static applications, developed by single groups

are evolving towards highly complex codes that aim to capture the complexity of nature, including and coupling myriad physical effects, and utilizing adaptive data structures, that require teams of researchers and computer scientists to develop and use. Novel approaches to HPC programming and HPC interaction such as those developed in our *Alpaca* [36, 37], *XiRel* [38], and *ParalleX* [33] projects will be greatly beneficial to any field of science planning to employ Peta-scale computing.

In order to address the problem of simulating Gamma-Ray Bursts, the Cactus framework will need to be continuously ported to newer and faster architectures, and be tested on ever larger scales of parallelism.

As Cactus and Carpet push the frontiers of hybrid parallelism combining MPI and OpenMP, HPX will explore the use of new programming models; in particular, models based on active messages, globally addressable memory, dynamic load balancing and fault tolerance [39]. The SiCortex architecture supports remote direct memory access (RDMA), operating system bypasses for network control, and zero copy transfers [40], which are important enabling technologies for these methodologies.

The fact that the SiCortex architecture has many more cores for the same performance than most current architectures makes it a good candidate for scalability tests. A dedicated SiCortex system can provide quick queue access for rapid development and prototyping. The SiCortex architecture is similar in many ways to the BlueGene/P, providing a large number of low power cores in a collection of SMP nodes. They likewise share advanced networks, capable of zero-copy RDMA, etc. This supports the use of SiCortex as a startup platform, a place to experiment and achieve scalability for codes in preparation for production use on larger machines. At the same time the SiCortex architecture has not been designed to scale up to Petaflop/s performance.

We plan to extend Cactus and Carpet to make use of HPX for communication and load balancing, further increasing scalability and performance across thousands or hundreds of thousands of cores. These modifications will be extensive and non-trivial, but the benefit of hav-

ing a framework for an AMR physics code with vastly increased scalability would be an enormous step forward towards realistic modeling Gamma-Ray Bursts and many other physical phenomena.

Acknowledgments

We thank A. Purkayastha for sharing his expertise on the SiCortex architecture, and we also

thank our colleagues at the CCT and in particular the members of the Cactus group for their insights and comments while porting the Cactus framework. The scaling studies on the NSF TeraGrid machines were funded by the NSF SDCI grant no. 0721915 *Alpaca* [36, 41] and the NSF PIF grant no. 0701566 *XiRel* [28, 29]. They were also supported by TeraGrid allocation TG-MCA02N014, and by allocations on LONI [42] and at the LRZ [43].

-
- [1] P. Mészáros, *Gamma-ray bursts*, Rep. Prog. Phys. **69**, 2259 (2006), astro-ph/0605208.
- [2] S. E. Woosley and J. S. Bloom, *The supernova gamma-ray burst connection*, Annual Rev. Astron. Astrophys. **44**, 507 (2006), astro-ph/0609142.
- [3] C. D. Ott, H. Dimmelmeier, A. Marek, H.-T. Janka, I. Hawke, B. Zink, and E. Schnetter, *3d collapse of rotating stellar iron cores in general relativity including deleptonization and a nuclear equation of state*, Phys. Rev. Lett **98**, 261101 (2007), astro-ph/0609819.
- [4] LIGO: Laser Interferometer Gravitational Wave Observatory, URL <http://www.ligo.caltech.edu/>.
- [5] C. D. Ott, E. Schnetter, G. Allen, E. Seidel, J. Tao, and B. Zink, *A case study for petascale applications in astrophysics: Simulating Gamma-Ray Bursts*, in *Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities* (ACM, Baton Rouge, Louisiana, 2008), no. 18 in ACM International Conference Proceeding Series, URL <http://doi.acm.org/10.1145/1341811.1341831>.
- [6] E. Schnetter, C. D. Ott, G. Allen, P. Diener, T. Goodale, T. Radke, E. Seidel, and J. Shalf, *Cactus Framework: Black holes to gamma ray bursts*, in *Petascale Computing: Algorithms and Applications*, edited by D. A. Bader (Chapman & Hall/CRC Computational Science Series, 2007), chap. 24, arXiv:0707.1607 [cs.DC].
- [7] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, *The Cactus framework and toolkit: Design and applications*, in *Vector and Parallel Processing – VECPAR’2002, 5th International Conference, Lecture Notes in Computer Science* (Springer, Berlin, 2003).
- [8] Cactus Computational Toolkit home page, URL <http://www.cactuscode.org/>.
- [9] S.-H. Ko, K. W. Cho, Y. D. Song, Y. G. Kim, J. su Na, and C. Kim, *Lecture Notes in Computer Science: Advances in Grid Computing - EGC 2005: European Grid Conference, Amsterdam, The Netherlands, February 14-16, 2005, Revised Selected Papers* (Springer, 2005), chap. Development of Cactus Driver for CFD Analyses in the Grid Computing Environment, pp. 771–777.
- [10] F. Dijkstra and A. van der Steen, *Integration of Two Ocean Models*, in *Special Issue of Concurrency and Computation, Practice & Experience* (Wiley, 2005), vol. 18, pp. 193–202.
- [11] B. Talbot, S. Zhou, and G. Higgins, *Review of the Cactus framework: Software engineering support of the third round of scientific grand challenge investigations, task 4 report - earth system modeling framework survey*, URL http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20020069014_2002111115.pdf.
- [12] ASC Website, *Astrophysics Simulation Collaboratory (ASC) home page*, URL <http://www.ascportal.org>.
- [13] K. Camarda, Y. He, and K. A. Bishop, *A parallel chemical reactor simulation using Cactus*, in *Proceedings of Linux Clusters: The HPC Revolution, NCSA* (2001), URL <http://www.cactuscode.org/Articles/Camarda01.doc>.
- [14] J. G. Kim and H. W. Park, *Advanced simulation technique for modeling multiphase fluid flow in porous media*, in *Computational Science and Its Applications - Iccsa 2004, LNCS 2004*, by A. Lagana et. al. (2004), pp. 1–9.
- [15] D. Rideout and S. Zohren, *Evidence for an entropy bound from fundamentally discrete gravity*, Class. Quantum Grav. (2006), gr-qc/0606065.

- [16] S. Major, D. Rideout, and S. Surya, *Spatial hypersurfaces in causal set cosmology*, *Class. Quantum Grav.* **23**, 4743 (2006), gr-qc/0506133.
- [17] E. Schnetter, S. H. Hawley, and I. Hawke, *Evolutions in 3D numerical relativity using fixed mesh refinement*, *Class. Quantum Grav.* **21**, 1465 (2004), gr-qc/0310042.
- [18] E. Schnetter, P. Diener, N. Dorband, and M. Tiglio, *A multi-block infrastructure for three-dimensional time-dependent numerical relativity*, *Class. Quantum Grav.* **23**, S553 (2006), gr-qc/0602104, URL <http://stacks.iop.org/CQG/23/S553>.
- [19] Mesh refinement with Carpet, URL <http://www.carpetcode.org/>.
- [20] M. Reilly, L. Stewart, J. Leonard, and D. Gingold, *Sicortex technical summary* (2006).
- [21] SiCortex, URL <http://www.sicortex.com/>.
- [22] Kraken, supercomputer at NICS, URL <http://www.nics.tennessee.edu/computing-resources/kraken>.
- [23] MIPS architecture, URL http://en.wikipedia.org/wiki/MIPS_architecture.
- [24] Intel architecture, URL <http://en.wikipedia.org/wiki/X86>.
- [25] Simulation Factory, URL <http://www.cct.lsu.edu/~eschnett/SimFactory/>.
- [26] S. Brandt and E. Schnetter, *From black holes to gamma-ray bursts: Cactus on SiCortex* (2008), talk given at SiCortex and LSU booths at Supercomputing Conference in Austin, TX.
- [27] E. Schnetter, *XiRel: Cyberinfrastructure for numerical relativity* (2008), talk given at GA Tech booth at Supercomputing Conference in Austin, TX.
- [28] J. Tao, G. Allen, I. Hinder, E. Schnetter, and Y. Zlochower, *XiRel: Standard benchmarks for numerical relativity codes using Cactus and Carpet*, Tech. Rep. CCT-TR-2008-5, Louisiana State University (2008), URL <http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-5>.
- [29] XiRel: Next Generation Infrastructure for Numerical Relativity, URL <http://www.cct.lsu.edu/xirel/>.
- [30] HLRB II, supercomputer at the LRZ, URL <http://www.lrz-muenchen.de/services/compute/hlrb/>.
- [31] Queen Bee, supercomputer at LONI, URL <http://www.loni.org/systems/system.php?system=QueenBee>.
- [32] Ranger, supercomputer at TACC, URL <http://www.tacc.utexas.edu/services/userguides/ranger/>.
- [33] ParalleX (PX) parallel execution model, URL <http://parallex.cct.lsu.edu/>.
- [34] T. Eicken, D. Culler, S. Goldstein, and K. Schauser, *Active messages: a mechanism for integrated communication and computation*, in *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on* (1992), pp. 256–266.
- [35] W. Carlson, J. Draper, D. Culler, K. Yelick, E. Brooks, and K. Warren, *Introduction to UPC and language specification* (Center for Computing Sciences, Institute for Defense Analyses, 1999).
- [36] E. Schnetter, G. Allen, T. Goodale, and M. Tyagi, *Alpaca: Cactus tools for application level performance and correctness analysis*, Tech. Rep. CCT-TR-2008-2, Louisiana State University (2008), URL <http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-2>.
- [37] Alpaca, Alpaca: Cactus tools for Application-Level Profiling and Correctness Analysis, URL <http://www.cct.lsu.edu/~eschnett/Alpaca/>.
- [38] XiRel, XiRel: Cyberinfrastructure for Numerical Relativity, URL <http://www.cct.lsu.edu/xirel/>.
- [39] L. Kale and S. Krishnan, *CHARM++: A portable concurrent object oriented system based on C++*, in *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications* (ACM New York, NY, USA, 1993), pp. 91–108.
- [40] L. Stewart, D. Gingold, J. Leonard, and P. Watkins, *RDMA in the SiCortex Cluster Systems*, *Lecture Notes in Computer Science* **4757**, 260 (2007).
- [41] Alpaca: Tools for Application-Level Profiling and Correctness Analysis, URL <http://www.cct.lsu.edu/~eschnett/Alpaca/>.
- [42] LONI: Louisiana Optical Network Initiative, URL <http://www.loni.org/>.
- [43] LRZ: Leibniz Supercomputing Centre, URL <http://www.lrz-muenchen.de/>.