TOWARDS A HIGHLY EFFICIENT AND SCALABLE INFRASTRUCTURE FOR NUMERICAL RELATIVITY CODES

JIAN TAO(1), GABRIELLE ALLEN(1,2), PETER DIENER(1,3), FRANK LÖFFLER(1), ROLAND HAAS(4), IAN HINDER(6), ERIK SCHNETTER(1,3), AND YOSEF ZLOCHOWER(5)

ABSTRACT. The advent of the Petascale era provides a great opportunity as well as a great challenge for computational science and engineering. In order to fully leverage the resources available, scientific applications need to scale to unprecedented numbers of processing cores and adapt to multicore architectures with complex memory and network hierarchies. In the numerical relativity community, the XiRel project has been funded by NSF to help prepare for these upcoming resources. The central goal of XiRel is to develop a highly scalable, efficient computational infrastructure based on the Carpet adaptive mesh refinement library that is fully integrated into the Cactus framework and optimized for numerical relativity applications. This paper presents our work towards building and benchmarking such an infrastructure which will benefit a wide spectrum of scientific applications that are based on Cactus.

1. INTRODUCTION

Petascale computing resources to advance research and education are being designed, developed, and deployed across the planet. These new resources provide a great opportunity for computational science and engineering, but also a number of grand challenges. Application codes, and their underlying numerical algorithms, need to be able to scale to unprecedented numbers of processing cores, adapt to multicore architectures with complex memory and network hierarchies, be tolerant to faults in hardware and software, and provide fast I/O and visualization. This is especially true for numerical relativity, which is aiming at solving Einstein’s equation with the state-of-the-art computational technology. The XiRel project aims to help the numerical relativity community fully leverage computational resources with the development of a highly scalable, efficient, and accurate adaptive mesh refinement (AMR) layer based on the existing Carpet driver. As an example, figure shows the gravitational radiation resulting from the merger of a binary black hole system. Improving the performance of these simulations is one of the aims of the work presented here.

In this paper, we begin with a short introduction to the Cactus-Carpet infrastructure in section 2. In section 3 we construct a suite of relevant numerical relativity benchmarks which provides a basis for measuring and understanding performance gains achieved during the future development of XiRel. We will then present our latest weak scaling benchmark results achieved on Ranger at TACC via the NSF TeraGrid in section 4 and conclude in section 5.

In order to avoid confusion arising from the advent of multi-core architectures, we use the term processing core or simply core instead of processor to refer to a single processing unit.

2. COMPUTATIONAL INFRASTRUCTURE

Cactus is an open source software framework consisting of a central core, the flesh, which connects many software components (thorns) through an extensible interface. Applications based on Cactus are highly portable and scalable. They run on almost all variants of the Unix operating system as well as the Windows platform. Carpet serves as a driver layer of the Cactus framework providing adaptive mesh refinement, multi-patch capability, as well as memory management, parallelization, and efficient I/O. In the Cactus-Carpet computational infrastructure, the simulation domain is discretized using finite differences on

Date: March 19, 2009.
(1) Center for Computation & Technology, Louisiana State University, Baton Rouge, LA, USA.
(2) Department of Computer Science, Louisiana State University, Baton Rouge, LA, USA.
(3) Department of Physics & Astronomy, Louisiana State University, Baton Rouge, LA, USA.
(4) Center for Relativistic Astrophysics, Georgia Institute of Technology, Atlanta, GA, USA.
(5) Center for Computational Relativity and Gravitation, Rochester Institute of Technology, Rochester, NY, USA.
(6) Albert-Einstein-Institute, Potsdam, Germany.

http://www.cct.lsu.edu/XiRel/
block-structured grids, employing a Berger-Oliger block-structured adaptive mesh refinement method [7]. The time integration schemes used include explicit Runge-Kutta methods. The scalability and performance of the Cactus-Carpet computational infrastructure has been greatly enhanced in the latest development of the XiRel project as shown in section 4.

3. XiRel Benchmarks

For measuring and understanding performance gains in the development of the Cactus-Carpet computational infrastructure, we constructed two benchmarks for weak scaling studies, namely Bench_BSSN_Carpet_Unigrid and Bench_BSSN_Carpet_AMR. The benchmarks specify both physical and numerical configurations for solving Einstein’s field equations in vacuum, where the BSSN formalism [2] is used to express the Einstein equations. The main difference between these two benchmarks is the number of refinement levels used (one and nine respectively), which then has implications for the numerical parameters chosen such as number of iterations and number of grid points. Without the complication introduced by the adaptive mesh refinement algorithm, the Bench_BSSN_Carpet_Unigrid benchmark enables us to compare the performance of two numerical relativity codes directly and improve the application codes. On the other side, the Bench_BSSN_Carpet_AMR benchmark helps us to understand and enhance the scalability of the mesh refinement driver Carpet. In this paper, we will only look at the Bench_BSSN_Carpet_AMR benchmark which is relevant to the production runs we carry out on TeraGrid machines.

3.1. Benchmark Design. We designed the benchmarks to represent important aspects of current and planned production runs for studying black hole physics. In addition to the grid setups shown in section 3.2 we made the following choices for the benchmarks:

1. Cartesian Minkowski spacetime as the initial data,
2. 4th order accurate finite differences,
3. 4th order accurate Runge-Kutta time integrator,
4. 3 timelevels for evolved grid functions,
5. 3 ghostzones for interprocess synchronization,
6. Reflection symmetries at the lower boundaries (octant symmetry),
7. 5th order accurate spatial, 2nd order accurate temporal interpolation at mesh refinement boundaries,
8. 5th order Kreiss-Oliger dissipation [9],
9. A Courant (CFL) factor, dt/dx = 0.5,
10. Commonly used coordinate conditions,
(11) Dirichlet boundary condition with a constant boundary value specified initially,
(12) Grid sizes such that a benchmark run requires approximately 650 MByte per core, allowing it to run
efficiently on systems with 1 GByte per core,
(13) No I/O except outputting the timer report, memory statistics, and timing statistics at the end,
(14) Sufficiently many iterations such that the benchmark runs for about 10 minutes on current hardware.

Note that although we start with a Cartesian Minkowski spacetime, which is a trivial setup as the evolved
fields are all either one or zero, all the terms in the evolution code are activated, and the performance
and scaling metrics for evolving black holes would be the same. Since the Einstein field equations are
invariant under coordinate transformations, various coordinate conditions are implemented in our codes
to fit individual needs. For the benchmarks, we use coordinate conditions commonly used for black hole
simulations.

3.2. Grid Setup. The Bench_BSSN_Carpet_AMR benchmark contains 9 refinement levels with \( 25^3 \times 9 \)
evolved grid points per core, and the system is evolved for 128 fine grid iterations, leading to 64 iteras
tions for the next coarsest level etc. We set up the grid structure in such a way that each level in the grid
hierarchy contains the same number of evolved grid points.

3.3. Constant Grid Hierarchy. To keep things simple, we keep the grid hierarchy constant in time. This
is arguably not very realistic, since the main point of AMR is to adapt the resolution to features of the
solution which need resolving. However, it is difficult to decide on a “typical” regridding behavior since
this depends on the system which is simulated. Consider, for example, a binary black hole simulation where
the refined regions need to track the black holes. In the beginning the black holes move slowly in a large
orbit, later they speed up, plunge, and finally merge. The refined regions move slowly at first, then speed
up with the black holes, then merge (presumably before the black holes merge). During the ringdown of
the final black hole the grid hierarchy remains constant. This wide range of behavior is impossible to capture
in a short benchmark. Another reason not to include regridding in this benchmark is that it is a somewhat
infrequent event, occurring only every several time steps. A design point for the benchmarks is that they
should complete in a short time even before the regridding would take place.

We plan to set up a more realistic, full-physics benchmark in the future which performs an actual binary
black hole simulation, including black hole tracking, horizon finding, and gravitational wave extraction.

3.4. Benchmark Timing Methodology. In this paper, we only consider the evolution time per right hand
side evaluation per grid point for each benchmark. The evolution time is the total time spent evolving the
system. This excludes MPI startup and shutdown, and also excludes the initial grid setup and initial data
calculation, thus it presents the performance of the numerical kernel of the application code. The evolution
time is found by subtracting the initialization time from the total run time. We use the average evolution
time of all the processes for each benchmark run. In order to avoid unexpected hardware problems, we ran
each benchmark setup at least three times and use the minimum time.

4. Enhanced Scalability

We demonstrate the scalability of the Cactus-Carpet infrastructure with two different numerical relativity
codes currently used for modeling the coalescence of astrophysical black holes: (i) CCATIE code [5, 4],
developed by a collaboration between Louisiana State University and the Albert-Einstein-Institute, (ii)
PSU/GaTech code [14], by the group originally at Pennsylvania State University, now at Georgia Institute
of Technology. We compare the scaling of each code on up to 128 cores on Tezpur at LSU in fall 2007 with
the scaling to over 2000 cores on Ranger at TACC in fall 2008 (see figure 3). We see a dramatic improvement
in the parallel scaling between these two periods, thanks to the optimization done in the data structures and
algorithms storing and handling the communication schedule.

5. Conclusion

We have presented our work towards building and benchmarking a highly scalable and efficient computa
tional infrastructure based on the Carpet adaptive mesh refinement library that is fully integrated into
the Cactus framework. The modular design of the Cactus framework hides the detailed implementation of
Carpet from application developers and separates application development from infrastructure develop
tment. The computational infrastructure presented in this paper will benefit a wide spectrum of scientific
applications that are based the Cactus framework.
6. ACKNOWLEDGMENTS

We thank our colleagues in the numerical relativity groups at our institutions and the Cactus team at LSU for their help in setting up and understanding these benchmarks. The CCATIE code was developed by a collaboration between Louisiana State University and the Albert-Einstein-Institute. This work used the Simulation Factory [3] by Erik Schnetter for job management. The XiRel project is supported by NSF awards 0701566 and 0653303. The development of performance measurement tools in Carpet is supported by NSF award 0721915 Alpaca. This work used the computational resource Ranger at TACC via the NSF TeraGrid. It also used the computational resources Tezpur at LSU and Eric and Queen Bee at LONI.

REFERENCES