Enabling Science at the Petascale: From Binary Systems and Stellar Core Collapse to Gamma-Ray Bursts

Jian Tao
On behalf of Ernazar Abdikamalov, Marek Blazewicz, Steven R. Brandt, Peter Diener, Roland Haas, Ian Hinder, David M. Koppelman, Frank Löffler, Philipp Moesta, Christian Ott, Christian Reisswig, Sherwood Richers, Erik Schnetter

Center for Computation & Technology
Louisiana State University
jtao@cct.lsu.edu

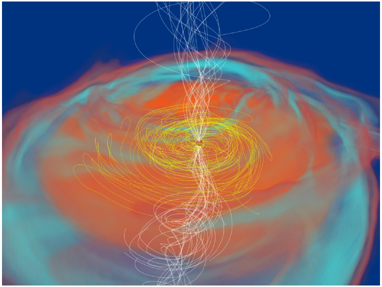
Blue Waters Symposium

May 21th, 2013 Urbana, IL

1. Scientific Questions
2. Background & Motivations
3. The Chemora Project
4. CFD Example
5. Conclusions & Future Plans
6. Acknowledgments

Binary neutron star mergers as an engine for short GRBs

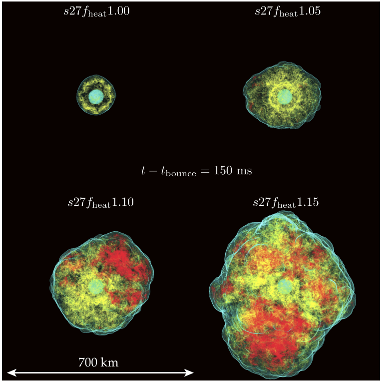
- Luminosity $L \simeq 10^{48} \mathrm{erg\,s^{-1}}$
- Burst duration < 2s
- Black hole as a result of the merger surrounded by accretion disk generates a jet
- r-process nucleosynthesis in the disk and the jet
- Ideal multi-messenger source in neutrinos, gravitational waves, and X-rays


[Etienne et al. 2013]

LSU Center for Computation & Technology

Core collapse supernovae

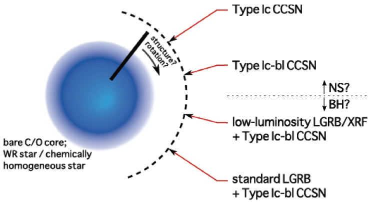
- Death of massive stars; explosion energies up to $10^{53}\mathrm{erg\,s^{-1}}$
- 99 % of that energy released in neutrinos
- Explosive nucleosynthesis enriches the interstellar medium with heavy elements
- Typical transient astronomy sources, lightcurves powered by radioactive decay of ejected material



[Ott et al. 2012]

Supernova + long gamma ray burst connection

A fraction of type Ic-bl supernovae are observed in combination with a long gamma ray burst/x-ray flash:



What is the engine behind this process and how does it depend on the progenitor parameters? Proto-magnetar or accretion powered collapsar?

Stellar collapse / neutron star merger simulations

Stellar collapse:

- Jet propagation speed, instabilities and asymmetries in the jet geometry
- Tracer particles to track the explosive nucleosynthesis along the outflows of stellar material
- Composition and mass of the ejected material and asymmetries in the ejecta

Neutron star mergers:

- Modeling of merger, accretion phase, black hole and jet formation
- r-process nucleosynthesis in ejected material

Modeling:

- (Ideal) MHD: fluid and magnetic field dynamics
- GR: space-time dynamics, neutron star radius
- Realistic nuclear tabulated EOS: Nuclear interactions
- Neutrino physics and transport: Neutrino interactions (heating, cooling, etc)
- Computational infrastructure: Adaptive mesh-refinement (AMR)
- Multi-D modeling (turbulence, convection, MRI) in massively parallel environments needed!

We use our open-source code *GRHydro* which is part of the *Einstein Toolkit (www.einsteintoolkit.org)*!

Einstein Toolkit

The Einstein Toolkit is

- an open software
  scientific tool kit for relativistic astrophysics.
- to provide the core computational tools that can
  - enable new science,
  - broaden our community,
  - facilitate interdisciplinary research,
  - take advantage of emerging petascale computers and advanced cyberinfrastructure.
- available at `http://einsteintoolkit.org/`

Challenges for Computational Scientists

In addition to addressing the **performance** and **scalability** issues, the developers for next generation HPC applications will also need a **sustainable development strategy** to enhance overall **programming productivity**.

LSU Center for Computation & Technology

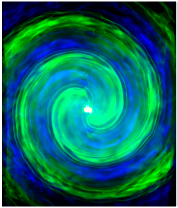
Opportunities for Multidisciplinary Research

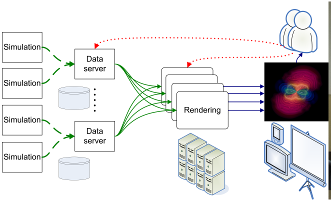
Whenever there is a challenge/difficulty, there is an opportunity. A lot of **multidisciplinary research and analytical work** is involved in

- designing and implementing the right algorithm (applied mathematics, computer science) for the right set of equations (all sciences) on right computing systems (computer science, electrical engineering);
- finding and categorizing the programming patterns (computer science);
- designing and implementing scientific applications (computational sciences);
- and much more...

Our Attempts to Address These Challenges

The prize-winning work at the **SCALE 2009 Challenge** at CCGrid09 is one of our attempts to demonstrate our **multidisciplinary and collaborative research efforts** and **framework-based solutions** to these challenges.

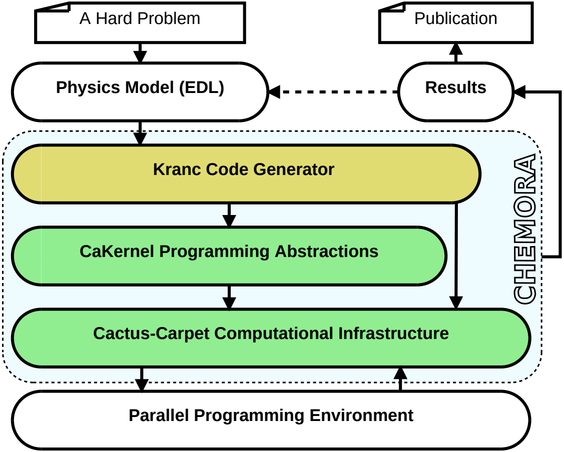




Center for Computation & Technology

Chemora for Heterogeneous Systems

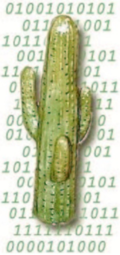
To carry our success to next generation petascale/exascale heterogeneous systems, based on the **Cactus Computational Framework**, we start the **Chemora** project.



Cactus Computational Framework

Cactus is

- a computational framework for developing portable, modular applications solving partial differential equations.
- focusing, although not exclusively, on high-performance simulation codes.
- designed to allow domain experts in one field to develop modules that are transparent to experts in other fields.
- supporting adaptive mesh refinement via the Carpet library.
- available at `http://cactuscode.org/`

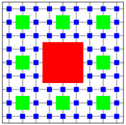


LSU | Center for Computation & Technology

Carpet Adaptive Mesh Refinement Library

Carpet is

- a driver layer of Cactus providing
  adaptive mesh refinement, multi-patch capability, as well as parallelization and efficient I/O.
- written primarily in C++.
- led by Erik Schnetter at LSU and Perimeter Institute.
- available at `http://carpetcode.org/`



Accelerator Framework

The naive "copy to - compute - copy back" approach in GPU programming very often fails to give optimal performance. As a module (or thorn) in Cactus, we developed an accelerator framework.

- We track which data (and what part of the data) are read and written by a particular routine, and where this routine executes (host or GPU).
- Data are copied only when necessary, and then only those portions that are needed.
- Data are not only accessed for computations; inter-process synchronization and I/O also access data, and are typically executed on the host.

CaKernel Programming Abstractions

CaKernel is

- a kernel abstraction;
- a parallel programming framework suitable for solving some types of PDEs;
- a collection of Cactus modules/thorns;
- able to automatically generate and execute CUDA, OpenCL, and C code;
- the outcome of our collaborative research efforts with PSNC and other institutes.

CaKernel is not

- designed to be a generic solution;
- in its final form yet.

LSU Center for Computation & Technology

Design of CaKernel

CaKernel contains 3 major parts:

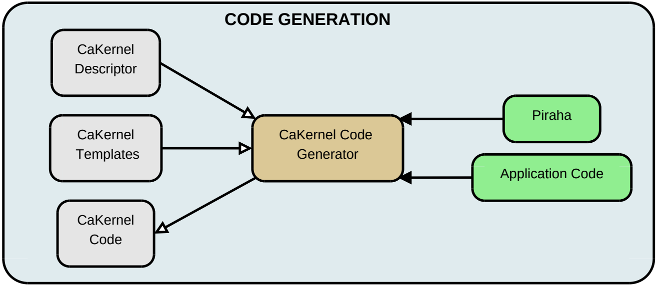
- **CaKernel Descriptor**: is used to declare the variables that will be needed in the computation, and identify a few relevant properties;
- **CaKernel Templates**: are sets of templates which are highly optimized for particular types of computational tasks and optimization strategies;
- **CaKernel Code Generator**: is used to parse the descriptors and automatically generate header files by referring to CaKernel templates. The descriptor parser and code generator are built on Piraha (http://code.google.com/p/piraha-peg/).

Grid Abstractions behind Cactus & CaKernel

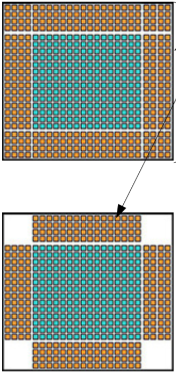
- **Grid Hierarchy (GH)** represents the distributed adaptive GH. In Cactus, grid operations are usually handled by a driver thorn to create, operate and destroy hierarchical grid structures.
- **Grid Function (GF)** represents a distributed data structure that represents the variables in an application. The application developers are responsible for providing proper routines to do initialization, boundary updates, etc.
- **Grid Geometry (GG)** represents the coordinates, bounding boxes, and bounding box lists of the computational domain. Operations on the GG, such as union, intersection, refine, and coarsen are usually implemented in a driver thorn as well.

CaKernel Code Generation

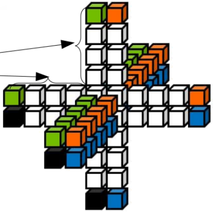
The **CaKernel code generator** parses the CaKernel descriptor and automatically generate CaKernel code from a set of highly optimized templates.



3D Stencil Computation



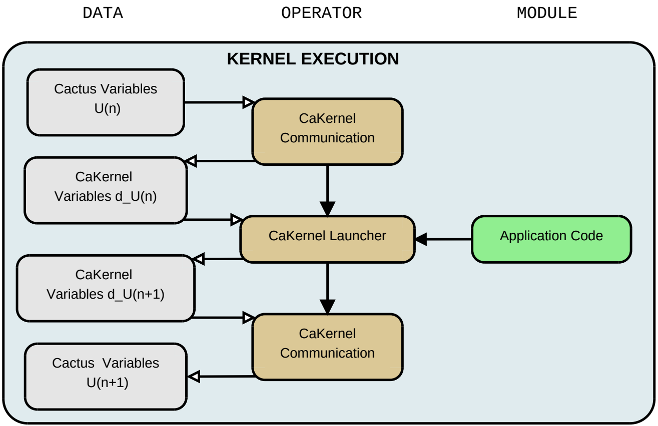
```
CCTK_CUDA_KERNEL UPDATE_VELOCITY
TYPE=3DBLOCK
STENCIL="1,1,1,1,1,1"
TILE="16,16,16"
{
CCTK_CUDA_KERNEL_VARIABLE CACHED=YES \
    INTENT=SEPARATEINOUT
{
vx, vy, vz
} "VELOCITY"
CCTK_CUDA_KERNEL_VARIABLE CACHED=YES\
    INTENT=IN
{
p
} "PRESSURE"
CCTK_CUDA_KERNEL_PARAMETER
{
density
} "DENSITY"
}
```



(credit to P. Micikevicius from NVIDIA)

Code Workflow

Cactus variables $\mathbf{U}(n)$ are evolved to the next time step $\mathbf{U}(n+1)$ during the execution stage.



Kranc Code Generation Package

Kranc is

- a suite of Mathematica packages with a computer algebra toolbox for numerical relativists.
- a prototyping system for physicists or mathematicians handling very complicated systems of partial differential equations. Kranc can generate entire Cactus based codes starting from a high level set of partial differential equations.
- available at `http://kranccode.org/`

Integration with Cactus

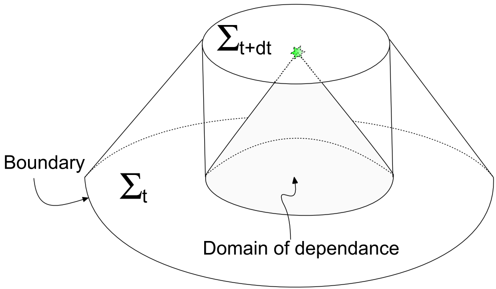
Kranc is closely coupled with Cactus by

- Defining the grid functions which the simulation will use.
- Performing a user-specified calculation at each point of the grid.
- Computing the right hand sides of evolution equations so that the time integrator can compute the evolved variables at the next time step.

Potential Applications

Various kinds of initial value boundary problems:

$$
\begin{aligned}
\partial_t u^a &= f(u^a), \\
u^a\big|_{t=0} &= g(u^a), \\
u^a\big|_{\partial\Sigma} &= h(u^a)
\end{aligned}
$$



Kranc in Action: Wave Equation (I)

Mathematical expression:

$$\partial_t^2 u = \delta^{ab} \partial_b \partial_a u$$

Rewritten in 1st order in time:

$$\begin{aligned} \partial_t u &= \rho, \\ \partial_t \rho &= \delta^{ab} \partial_b \partial_a u \end{aligned}$$

Input to Kranc:

$$\begin{aligned} dot[u] &\rightarrow rho, \\ dot[rho] &\rightarrow KD[ua, ub] PD[u[la], lb] \end{aligned}$$

Kranc in Action: Wave Equation (II)

Generated C code to calculate the RHS:

```
/* Precompute derivatives (new style) */
    PDstandardNth11u = PDstandardNth11(u, i, j, k);
    PDstandardNth22u = PDstandardNth22(u, i, j, k);
    PDstandardNth33u = PDstandardNth33(u, i, j, k);
/* Calculate temporaries and grid functions */
    urhsL = rhoL;
    rhorhsL = PDstandardNth11u + PDstandardNth22u
            + PDstandardNth33u;
/* Copy local copies back to grid functions */
    rhorhs[index] = rhorhsL;
    urhs[index] = urhsL;
```

Kranc - CaKernel GPU Code Optimization

The CaKernel parts of Chemora use Kranc-provided and run-time-available information to generate efficient GPU executables from the numerical kernel.

- Stencils and dynamic tile selection (loop tiling, heuristic approaches)
- Lightweight kernel generation (dynamic code generation, index arithmetic simplification)
- Fat kernel detection (loop fusion, code reconstruction, dynamic adjustment of the number of threads)
- Integrated performance monitoring (PAPI, NVIDIA Cupti, kernel identification)

Equation Description Language

- Very high level, Latex-like syntax for domain scientists
- Description: variables, equations, initial/boundary conditions, parameters, analysis quatities
- A layer between real physics problems to numerical implementations

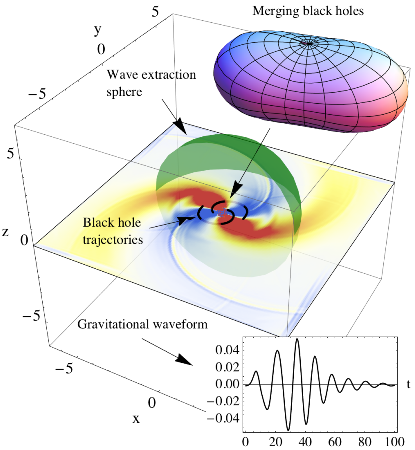
```
begin calculation Init
  u   = 0
  rho = A exp(-1/2 (r/W)**2)
  v_i = 0
end calculation

begin calculation RHS
  D_t u   = rho
  D_t rho = delta^ij D_i v_j
  D_t v_i = D_i rho
end calculation

begin calculation Energy
  eps = 1/2 (rho**2 + delta^ij v_i v_j)
end calculation
...
```

Binary Blackhole Simulation

Modeling gravitational waves from binary blackhole system.



Lid Driven Cavity (LDC) Problem

The LDC problem describes a initially stationary fluid contained in a square cavity with a moving lid whose velocity is tangent to the lid surface. It is a standard test case for the numerical solvers of the **Incompressible Navier-Stokes equations**.

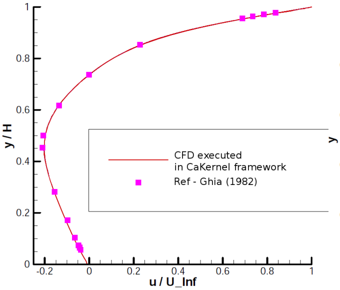
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla \phi + \nu \nabla^2 \mathbf{u} + \mathbf{f} \quad (1)$$

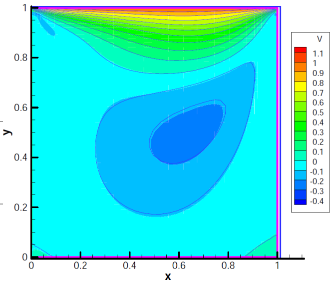
$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

where $\mathbf{u}$ is the velocity field, ν is the kinematic viscosity, $\mathbf{f}$ is the body force, ϕ is the modified pressure (pressure over density).

Verification & Validation

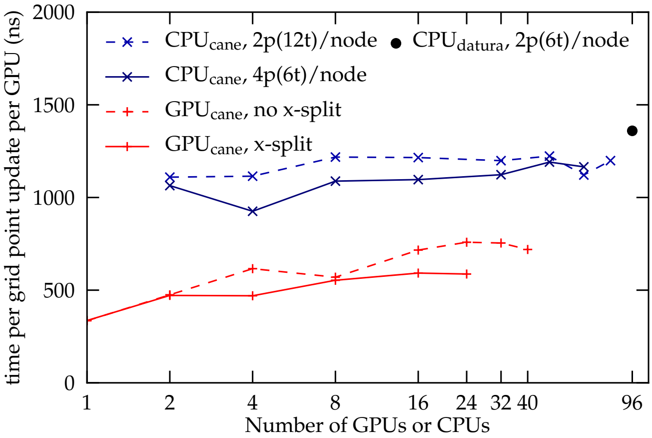
We solved the LDC problem with a Reynolds number of 100. A comparison of the X component of the velocity field in the midsection along the Y axis with those measured by Ghia et al. (1982) is shown below.





Scaling of Chemora (CFD Example)

Scaling results of the CFD example on a local cluster.



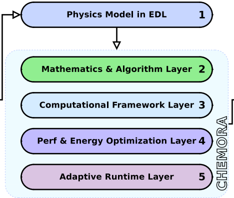
Conclusions

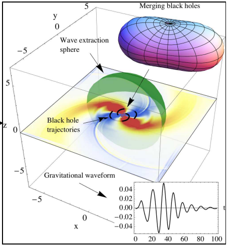
Targeting Blue Waters and future heterogeneous supercomputers, we designed and implemented Chemora that

- takes a multi-layered approach to enable optimized code for solving complex equations from a high level input,
- is built upon existing computational infrastructures to enable a smooth transition to the next generation computing resources,
- enables synergistic multidisciplinary collaboratoins,
- could be used in a wide spectrum of scientific applications.

Future Plans

```
...
dir^a        = sign(beta^a)
detgt        = 1
gtu^ab       = detgt**-1 * (-1 * gt13**2 * gt22 + 2 * gt12 * gt13 * gt23 + -1
               * gt11 * gt23**2 + -1 * gt12**2 * gt33 + gt11 * gt22 * gt33) *
               (inverse(gt)ab)
Gtl_abc      = 1/2 * ((PDstandardNth_c gt_ba) + -1 * (PDstandardNth_a gt_bc) +
               (PDstandardNth_b
               gt_ca))
Gtlu_ab^c    = gtu^cd *
               Gtl_abd
Gt^a_bc      = gtu^ad *
               Gtl_dbc
Xtn^i        = gtu^jk *
               Gt^i_jk
Rt_ij        = 1/2 * (PDstandardNth_j Xt^k) * gt_ki + 1/2 * (PDstandardNth_i
               Xt^k) * gt_kj + -1/2 * (PDstandardNth_lm gt_ij) * gtu^lm + 1/2
               * Xtn^k * Gtl_ijk + 1/2 * Xtn^k * Gtl_jik + Gt^k_jl * Gtlu_ik^l
               + Gt^k_il * Gtlu_jk^l + Gt^k_il *
               Gtlu_kj^l
...
```





Technical Support

- this work was performed using the computational resources of XSEDE, Blue Waters at NCSA, LSU/LONI, and PSNC.