The Cactus Framework: Design, Applications and Future Directions

Gabrielle Allen
gallen@cct.lsu.edu
Center for Computation & Technology
Departments of Computer Science & Physics
Louisiana State University

Cactus Code

- Freely available, modular, portable and manageable environment for collaboratively developing parallel, high-performance multi-dimensional simulations (Component-based)
- Developed for Numerical Relativity, but now general framework for parallel computing (CFD, astro, climate, chem eng, quantum gravity, …)
- Finite difference, AMR (Carpet, Samrai, Grace), new FE/FV, multipatch
- Active user and developer communities, main development now at LSU and AEI.
- Open source, documentation, etc
Computational Science Needs

- Requires incredible mix of technologies & expertise!
- Many scientific/engineering components
  – Physics, astrophysics, CFD, engineering, ...
- Many numerical algorithm components
  – Finite difference? Finite volume? Finite elements?
  – Elliptic equations: multigrid, Krylov subspace, ...
  – Mesh refinement
- Many different computational components
  – Parallelism (HPF, MPI, PVM, ???)
  – Multipatch / multiscale / multimodel
  – Architecture (MPP, DSM, Vector, PC Clusters, FPGA, ???)
  – I/O (generate TBs/simulation, checkpointing…)
  – Visualization of all that comes out!
- New technologies and needc
  – Grid computing, Steering, data archives
  – Semantics, data driven, networks
Design Issues for Cactus

- Usable for many different physical systems
- Collaborative
- Portable
- Large scale
- High throughput
- Easy to understand and interpret results
- Community Driven, Open Source
- Supported and developed
- Produce believed results
- Flexible
- Reproducible
- Have generic computational toolkits
- Incorporate other packages/technologies
- Access to latest technologies
- Easy to use/program (Empire code ported in 2 weeks)
- Preparing for future (DDDAS, multimodel, multiscale, coupling, petascale)

Large Scale

- Typical run needs > 100GB of memory:
  - 200 Grid Functions
  - 400x400x400 grid
- Typical run makes 3000 iterations with 6000 Flops per grid point:
  - 600 TeraFlops !!
- Output of just one Grid Function at just one time step
  - 500 MB (1 TB for 15GF each 50 timesteps)
- One simulation longer than queue times
  - Need 10-50 hours
- Computing time is a valuable resource
  - One simulation: 5000 to 20000 SUs
  - Need to make each simulation count

Parallelism
Optimization
Parallel/Fast I/O, Data Management, Visualization
Checkpointing
Interactive monitoring, steering, visualization, portals
Cactus Structure

Core “flesh” with plug-in “thorns”

Cactus Flesh

- Written in ANSI C
- Independent of all thorns
- Contains flexible build system, parameter parsing, rule based scheduler, ...
- After initialization acts as utility/service library which thorns call for information or to request some action (e.g. parameter steering)
- Contains abstracted APIs for:
  - Parallel operations, IO and checkpointing, reduction operations, interpolation operations, timers. (APIs designed for science needs)
- All actual functionality provided by (swappable) thorns
Cactus Thorns (Components)

- Can be written in C, C++, Fortran 77, Fortran 90, (Java, Perl, Python)
- Separate libraries encapsulating some functionality
- To keep distinction between functionality and implementation of functionality each thorn declares it provides a certain “implementation”
- Different thorns can provide the same “implementations”
- Thorn dependencies expressed in terms of “implementations”, so that thorns providing same “implementation” are interchangeable.

Thorn Specification

- Each thorn contains configuration files which specify interface with Flesh and other thorns
- Configuration files converted at compile time (Fortran) into a set of routines the Flesh can call for thorn information
  - Scheduling directives
  - Variable definitions
  - Function definitions
  - Parameter definitions
  - Configuration details
- Configuration files have well defined language, can use as basis to build interoperability with other component based frameworks
Program Flow

Stop if errors
Read Parameter File
Activate Thrtle
Initialise Scheduler

Check consistency
Initialise Simulation
Evolve Simulation
Terminate Simulation

Initialise Simulation

Check parameters make sense

Set up coordinates
Initial data
Evolve Simulation

Numerical Methods

- Most Cactus application codes use finite differences on structured meshes
- Parallel driver thorns: Unigrid, FMR (Carpet), AMR (Grace, SAMRAI), finite volume/element on structured meshes
- Method of lines thorn
- Elliptic solver interface (PETSc, SOR, Multigrid, Trilinos)
- Multipatch now with Carpet driver
- Unstructured mesh support being added.
• Support for IO in different formats
  – 2-d slices as jpegs
  – N-d ASCII data
  – N-d data in IEEElO format
  – N-d data in HDF5 format (to disk or streamed)
  – Panda parallel IO
  – Isosurfaces, MPEGS
• Checkpoint/restart (move to any new machine)

**External Packages**

• IO API: HDF5, FlexIO, jpeg, mpeg, NetCDF, Panda
• Elliptic/Solver subsystem: LAPACK, BLAS, PETSc, Trilinos, FFTW
• Timer API: PAPI
• Driver API: MPI, Grace, SAMRAI, PARAMESH
  – New programming models
• Grid: GAT, MPICH-G2, (Globus)
Toolkits

- Cactus Computational Toolkit
  - Core thorns which provides many basic utilities including: Boundaries, I/O methods, Reduction/interpolation operations, Coordinates, Parallel drivers, Elliptic solvers
- Einstein Toolkit
- CFD Toolkit
Cactus CFD Toolkit

- Unified numerics on arbitrary discretizations (FDM/FVM/FEM) and meshes (Structured/Multi-block/Unstructured)
- Multi-scale and Multi-physics Framework
- Coupling multi-model simulations
- Plethora of solution algorithms: e.g. Artificial compressibility, SIMPLE, Fractional step method (incomp.), LDFSS (comp.)
- Variety of solvers to exploit the advantages and avoid limitations of specific numerical solvers (PETSc/Trilinos/MUMPS).
- Examples & Test suites: benchmarks for numerical validation/verification and petascale applications.
- Interdisciplinary applications: toolkit is common "language" for collaboration

**Application Areas:** Biology, Mech/Chem/Civil/Environ./Petrol. Engg., Oceanography, Computer Sci., Mathematics

---

**Easy to Use and Program**

- Program in favorite language (C,C++,F90,F77)
- Hidden parallelism and optimization
- Computational Toolkits
- Good error, warning, info reporting
- Modularity !! Transparent interfaces with other modules (protection from other modules)
- Extensive parameter checking
- Work in the same way on different machines
- Interface with favorite visualization packages
- Documentation
Portable

- Solid and flexible make system, supported on all tried platforms including Windows, MacOSX, XT3, BG/L, Earth Simulator, PlayStation 2, X-Box, iPAQ.

Cactus Grid Scenarios

- Dynamic, adaptive apps …
- Distributed MPI
- SC2001: Spawner thorn, any analysis method can be sent to another resource for computation
- SC2002: Cactus TaskFarm, designed for distributing MPI apps on the Grid
Dynamic Adaptive Distributed Computation

Cactus + MPICH-G2
Communications dynamically adapt to application and environment
Any Cactus application
Scaling: 15% -> 85%

“Gordon Bell Prize”
(With U. Chicago/Northern, Supercomputing 2001, Denver)

Grid Application Toolkit (GAT)

- Abstract programming interface between applications and Grid services
- Designed for applications (move file, run remote task, migrate, write to remote file)
- Led to GGF Simple API for Grid Applications

<table>
<thead>
<tr>
<th>Default Adapters</th>
<th>Basic functionality, will work on single isolated machine (e.g. cp, fork/exec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Globus Adapters</td>
<td>Core Globus Functionality: GRAM, MDS, GT, RLS, GridFTP</td>
</tr>
<tr>
<td>GridLab Adapters</td>
<td>GRMS, Mercury, Delphos, iGird</td>
</tr>
<tr>
<td>Under Develop</td>
<td>Scp, DRMAA, Condor, SGE, SRB, Curl, RFT</td>
</tr>
</tbody>
</table>

Main result from GridLab project

www.gridlab.org/GAT
Remote Viz & Steering

Based on GridSphere JSR168 compliant portal framework
Portlets for parameter file preparing, comparison, managing
Simulation staging, managing, monitoring
Link to data archives, viz etc.

Cactus Portal

Changing steerable parameters
• Parameters
• Physics, algorithms
• Performance

Any Viz Client: LCA Vision, OpenDX
Streaming HDF5 Autodownsampling
Notification and Information

Notification and Information

Looking Forward

- Generalized component descriptions for thorns
- Interoperability with other frameworks
- Supporting multiscale, multimodel scenarios
- New applications with coastal, CFD
- Grid computing and data archives
- Methodologies for petascale computing
- Keep developments driven by physics