

-----  
Cactus-Simulationen on the Grid  
-----

Contact Person: Thomas Radke, AEI  
tradke@aei.mpg.de, 0331 / 567 7626

-----  
1. Scenario Overview  
-----

1.1 Background and Purpose

Cactus is an open source problem solving environment designed for scientists and engineers. Its modular structure easily enables parallel computation across different architectures and collaborative code development between different groups. Cactus originated in the academic research community, where it was developed and used over many years by a large international collaboration of physicists and computational scientists.

Cactus is used by the physicists in the Numerical Relativity department of the AEI to numerically simulate extremely massive bodies, such as neutron stars and black holes. An accurate model of such systems requires a solution of the full set of Einstein's equations for general relativity - equations relating the curvature of spacetime to the energy distribution. The overall goal is to deliver accurate signal patterns of sources of gravitational waves which then can be matched against the data measured at the various gravitational wave detector interferometers around the world (eg. GEO600, LIGO, later on LISA).

1.2 More information

- \* Cactus Homepage:  
<http://www.cactuscode.org>
- \* Live Cactus Demonstration:  
<http://cactus.cct.lsu.edu:5555/>
- \* Cactus Visualisation Tools:  
<http://www.cactuscode.org/Community/visualization>

=====  
PLEASE ALSO TAKE A LOOK AT THE FOLLOWING THREE WEBPAGES !  
THEY CONTAIN DETAILED DESCRIPTION OF INDIVIDUAL CACTUS GRID SCENARIOS.  
=====

- \* Aspekte der Metadaten-Verwaltung fuer die Ueberwachung von Cactus-Simulationen:  
<http://gac-grid.org/intranet/workpackages/ap-2/CactusSimulationMonitoring.html>
- \* Aspekte der Dateiverwaltung bei der Generierung und Weiterverarbeitung von Cactus-Simulationsdaten:  
<http://gac-grid.org/intranet/workpackages/ap-3/CactusDataProcessing.html>
- \* Monitoring and Steering Capabilities in the Cactus Code:  
<http://gac-grid.org/intranet/workpackages/ap-6/CactusMonitoringAndSteering.html>

-----  
2. Current Scenario description  
-----

## 2.1 Environment

### 2.1.1 Hardware

#### - Processing

Cactus simulations are mostly run as parallel MPI jobs on supercomputers at various high-performance computing centres in the world.

Relevant computing resources within D-Grid are

- \* AEI: 172 dual-processor node Intel Linux cluster peyote.aei.mpg.de
- \* LRZ: SGI Altix
- \* RZG: IBM SP-4

Users either login directly to the HPC system or to a headnode of a cluster. Cluster compute nodes are often hidden behind a firewall/VPN and therefore not visible to the outside world.

#### - Storage

Each HPC resource usually has one or more shared NFS scratch filesystems (eg. on peyote we have 8 NFS filesystems a 1.2 TBytes). Each node may have a local scratch filesystem (eg. ext3) in addition but it is then usually not shared by other nodes/the headnode.

#### - Network

Cactus simulations run as parallel MPI jobs and therefore need tight coupling between computing elements (high communication bandwidth, low latencies) to run efficiently. This narrows possible hardware resources down to HPC machines, clusters, or compounds of such with a dedicated high-bandwidth interconnect.

#### - Describe special hardware or other hardware resources that are relevant for the scenario.

N/A

### 2.1.2 Software

#### - Describe used software such as operating system, software libraries, e.g. HDF5-plugin for GridFTP, ...

Cactus requires standard UNIX tools such as gmake, cvs, perl for its installation on a given system (which can be any of today's architectures/OS). In order to build executables, C/C++/Fortran90 compilers is required. MPI is needed for parallelisation (MPICH, LAM, vendor-specific, MPICH-G2 are fine).

Additional software packages may be needed optionally: HDF5, BLAS, LAPACK, GSL, PETSc. All of those are available as RPM downloads or can be built and installed from open source.

In order to use future grid-enabled Cactus extensions, Globus 4.x and the GAT (of the most recent version) will become necessary.

- What programming language is used and what compiler/linker version is required?

C/C++/Fortran must be supported.

Cactus already knows a variety of different compiler versions but can also be hand-configured to use a new set.

- How is the program deployed?

The Cactus source code is entirely available via public CVS access.

User extensions (thorns) are also available via CVS, potentially from private repositories, therefore require they individual CVS checkout permissions.

Due to shared library issues, executables must be built on each individual resource.

- How is the program compiled?

GNU configure & make (Cactus comes with predefined config and makefiles which can be controlled via a machine-specific configuration options file)

- State the program license and any commercial 3rd party licenses.

Apart from compiler licenses, no other licenses are involved in building and running Cactus executables.

## 2.2 User Interaction

Describe the user interaction necessary for starting the program and additional interaction with a running program.

### 2.2.1 Initiation

- Describe how the program is started and any steps needed before the actual initiation.

HPC resources are mostly managed by batch systems (eg. Torque on peyote).

Users describe their Cactus job in a batch script

- executable name (supplied by the user)
- parameter file (supplied by the user)
- number of processors
- requested runtime

and then send it off to the queuing system.

- compilation (cf. Section 2.1.2),

Users log into the HPC resource, checkout/update their Cactus source tree, reconfigure/rebuild their Cactus configuration (gmake) - this is usually done manually and probably cannot be fully automatised per se. Although, for automated Cactus tests, we wrote a perl script which executes all of the above and below steps.

- Where is the program executed?  
on the HPC resource as a parallel batch job

- How is the program initiated?  
from the remote shell command line

## 2.2.2 Monitoring/Steering/Visualization during the run-time of the program

for more detailed descriptions please refer to  
<http://gac-grid.org/intranet/workpackages/ap-6/CactusMonitoringAndSteering.html>

- What type of data is produced by the program during run-time used for monitoring/steering/visualization?
  - \* temporary stdout/stderr logfiles managed by the queuing system (i.e. stored as temporary files in a PBS spooldir on a certain compute node)
  - \* final stdout/stderr logfiles stored in a user's directory on the HPC access node
  - \* output datafiles (ASCII, HDF5) can be monitored/visualised while they are still being written/appendd
  - \* for monitoring/steering, Cactus provides a service (rather than output data)
    - see below
- What methods/tools exists for accessing data produced by the program during run-time?
  - \* Thorn HTTPD is a webserver integrated into a Cactus simulation. It provides real-time information about the current simulation; various parameter settings/variables can be monitored and even steered (if applicable).

The HTTPD service speaks HTTP and uses plain TCP/IP socket communication, it can be accessed by a URL <"hostname":"port"> where "hostname" is the name of the root node of the simulation (which maybe be hidden behind a firewall/VPN and therefore is visible only from the headnode), and "port" is the port number to connect to.

HTTPD currently has no encryption, no security, no workarounds for firewalls/VPN.

Via a Cactus-specific API other thorns can register themself under a unique sub-URL and serve their own HTML pages. This is used to provide both monitoring (eg. pre-visualisation of Cactus variables as JPEG images embedded in a webpage) and steering functionality (eg. HTML forms to modify Cactus runtime parameters, pause/terminate a simulation).
  - \* Thorn IOStreamedHDF5 is an I/O thorn which can stream data in HDF5 file format over a TCP/IP socket connection to remote clients.

The same applies to this thorn regarding accessibility.
  - \* Thorn Announce registers a Cactus simulation with a (remote) Cactus portal server. Communication is based on sending XML messages.
    - . shall these interfaces be used for automated monitoring/steering?

HTTPD and IOStreamedHDF5 are mostly for interactive monitoring/steering. Announce should automatically provide monitoring information to a portal.
- Does your application support any standard for monitoring/steering?

The monitoring/steering functionality is application-specific; the underlying protocols used (HTTP/HTML, HDF5, XML messages) are communication standards.
- Describe any security measures related to program access for monitoring/steering/visualization.

Currently there is no access restriction to login to a running Cactus job via

HTTPD so anyone who knows the URL can connect and monitor the simulation. In order to steer it, one must authenticate via a crypt(3) username/password. Both are specified in the Cactus parameter file which can be accessed already via the monitoring functions.

In the future, general access should also require a user authorisation, possible using a more secure method than crypt(3) (eg. a grid certificates).

If possible, the same authentication methods should be used by the visualisation services to make them more secure.

- Who can access the running program OR run-time produced monitoring data?

It must be possible to define groups of users with certain permissions (monitoring/steering) dynamically (eg. in the parameter file or even at runtime by steering the corresponding runtime parameter). The owner of the job should always be allowed to monitor and steer a simulation.

- From where can run-time produced monitoring data be accessed?

From anywhere.

- How is the program termination detected?

Normal termination can be announced to the Cactus portal via thorn Announce. Both normal and anormal program termination can be detected by querying the job management system for the given job ID (qstat).

- How much monitoring data and how often is monitoring data transferred during a program run (min/max/avg)?

Thorn HTTPD and thorn Announce periodically update (eg. every so many iterations) text-based information which isn't very large (a few kBytes). Thorn IOStreamedHDF5 may periodically send HDF5 data to (potentially multiple) connected visualisation clients (in the order of some MBytes). For long-term runs, a user may want to download some output files via HTTPD which then can amount up to some 100MBytes.

- Does your program generate metadata and stores this externally (e.g. in a catalog)?

Parameters in the parameter file contain the most important (static) metadata for any given Cactus simulation. Other static and dynamic imulation metadata is provided by HTTPD (although not stored) and Announce which sends it off to a Cactus portal via XML messages. There is is stored, currently in a portal-internal mySQL database (no access by other services, internal format).

- Who accesses this metadata? From where? Does your program access metadata generated by other programs?

The parameter file is provided by the users.

Dynamically generated metadata provided by the HTTPD service is accessible by anyone with a standard webbrowser. There usually are access restrictions due to firewalls/VPN.

Via the portal, anyone (with given authorisation) can access all registered Cactus simulations and their metadata from anywhere via a standard webbrowser.

- How many executions/jobs must be monitored/steered in parallel? By how many users?

As an example, on the peyote compute cluster at AEI, there are typically 10-30 Cactus jobs running at the same time. All of them may be monitored

and/or steered by their owners, potentially by more people (usually not more than 4 at the same time).

## 2.3 Input

### 2.3.1 Parameters

A Cactus parameter file (text-based, < 10 kBytes) is provided by the user. For most Cactus simulations this is the only input data.

### 2.3.2 Input data

- How is the input data prepared?

Some Cactus simulations require input data files: eg. a checkpoint from a previous simulation that should be recovered from, or filereader files to import certain variables into the simulation to start. Such files can be very large, depending on the underlying grid size of the simulation (up to 1 TBytes).

Currently the user makes sure that such files are copied onto the local filesystem (a checkpoint directory name is specified in the parameter file). The source of the checkpoint/filereader files can be the same or some other HPC resource where they were created during a previous simulation (HDF5 files are platform-independent so it doesn't matter where they were created).

- Where is the input data stored? Describe all central and distributed locations.

must be accessible from the local file system of the HPC resource

- Are file-names known in advance (before the program is started)?  
No.

- Are data locations (directory, server, ...) known in advance?  
The input/output directories are specified in the Cactus parameter file.

- Describe the different ways data is accessed.  
UNIX file I/O; HDF5 library

- Non-file based data access (XML, database, ...) should include description of  
N/A

- How much data is accessed at each run?

see above: parameter files are small, checkpoint/filereader files are large  
files are always read entirely

- Is it possible that a data set/file is accessed multiple times over a short period of time?

An initial data checkpoint may be used to start a parameter study: multiple Cactus simulations recover from the same set of checkpoint files. This doesn't happen very often though.

- How many users are using the same data simultaneously?  
Are these users geographically distributed?

The input data is user-specific. Checkpoint/filereader files may be distributed

in the future when doing distributed Cactus simulations across multiple resources.

- Elaborate on the use of metadata related to input data.

The Cactus parameter file contains all the static metadata for a simulation. It basically contains text-based key/value pairs where values can be of type boolean, integer, floating-point, string and keyword (fixed set of possible string values) in a plain ASCII text file, plus arbitrary comments. Usually each simulation has its own parameter file.

### 2.3.3 Additional Notes

N/A

## 2.4 Output

### 2.4.1 Output data

- Where is the output data stored? Describe all centralized or distributed locations.

Simulation output data is stored on the file system(s) of the HPC resource. For some files (eg. the temporary PBS logfiles, temporary checkpoint files) this would ideally be the node-internal scratch file system (because it has the fastest access), most of them are directly stored on shared NFS filesystems though so that they don't need to be moved there afterwards.

- How is the output data structured?

Cactus writes ASCII and HDF5 data files, one per output variable; there can be many output variables specified in the parameter file (up to 100). New timesteps are appended to existing output files (this means that the files get changed while the simulation is still running).

HDF5 files are usually chunked, i.e. each processor writes its portion into a separate file. For postprocessing the entire set of all chunked files is required.

For distributed simulations (as planned in a later project stage) data would be written distributed across resources.

Filenames follow a Cactus I/O naming convention (eg. contain the output variable's name, potentially the iteration number and other info). Rather than guessing individual filenames it is safer to just recall the list of output directories (as specified in the parameter file) in a metadata management.

- Describe what happens when the program finishes? How are the results used?

The ASCII output files are medium-sized and thus can be copied to the user's machine for postprocessing. The HDF5 output files usually are so large that they have to remain at the output location and postprocessed remotely.

So far all files are manually copied/moved/deleted by the user when needed.

- Describe the different ways data is created/changed.

UNIX file I/O; HDF5 library; datafiles are appended during the simulation and not changed afterwards

- Non-file based data access (XML, database, ...)

N/A

- How much data is written by the program at each run?

ASCII output files are medium size (up to 100 MBytes).  
HDF5 output files are large (up some 100 GBytes).  
The total amount of output data can be very large (up to some TBytes).

- Describe the parameters which influence the amount of data and number of files/data sets generated.  
This depends on the grid size of the simulation, the number of timesteps, the number of output variables.

- Elaborate on the use of metadata related to output data.  
The stdout/stderr logfiles can grow up to some 10 MBytes but should be regarded as static simulation metadata.  
Live simulation metadata is provided by the monitoring services HTTPD and Announce in a non-file based format (see above).  
Output files also keep some metadata (eg. variable name, timestep) as comments (in ASCII files) or attributes (in HDF5 files).

#### 2.4.2 Additional Notes

Users may need to move output data from their original location (each a shared NFS scratch filesystem) to some other place (eg. a long-term storage filesystem). Metadata catalogues should then be updated accordingly.

#### 2.5 Information resources

Currently there exists a prototype version of a Cactus portal:  
<http://portal.cct.lsu.edu>.

Users can get a login on that portal, query Cactus simulations that have been accounted with thorn Announce, connect to running simulations via thorn HTTPD and monitor/steer them.

The protocols that are used by HTTPD and Announce are described above.

#### 2.6 Data Stream Management

N/A

#### 2.7 Resource Security and Access Restriction

see above sections

#### 2.8 Additional Information

Give additional information not covered by the sections above.

- How long (avg) does the scenario execute (minutes, hours, days)?  
Most Cactus simulations are in the order of couple hours or a few days runtime. Large simulations can run for some weeks and therefore require checkpointing & recovery to bridge queue time limits.

- How often will the scenario be executed?

N/A

- Are the executions time-critical?

No. But for interactive monitoring/steering/visualisation a certain response time would be reasonable.

-----  
3. Future Scenario and AstroGrid-D Usage  
-----

### 3.0 General goals

Our group already has external computing time allocations at various HPC sites in Germany and worldwide so the goal is not necessarily to get more allocations but instead make it easier to use the existing ones more efficiently.

Currently this isn't quite the case because people find it difficult to use remote machines (what are the available HPC resources, special login machines, how do I get an account, what's my login there, which password?). A Cactus simulation portal for users should help to solve these issues.

Via a Cactus portal as a common user interface it should be possible to keep a better overview on Cactus simulations that a user her/himself and other users in the community have done already (eg. by searching in a simulation metadata catalogue). The portal could also serve as a single point-of-contact to monitor and steer Cactus simulations running remotely on the Grid, allowing for better collaboration between scientists in the community.

Finally, postprocessing and visualisation of Cactus simulation output data is very cumbersome and resource-consuming: currently all the output data have to be copied to the local machine for visualisation (which is in most cases not possible due to network bandwidth and local storage constraints). Standard methods for efficient partial remote file access are envisioned to make remote data analysis possible and more efficient.

### 3.2 Environment

- Are there any constraints due to your participation in other projects or international collaborations?

So far all of the HPC resources (especially the ones in the U.S.) are committed to using Globus 3.x/4.x as the underlying Grid middleware. Based on that, we are committed to using the Grid Application Toolkit (GAT) as a Grid application API, and the GridSphere portal framework as standard to build a web-based Cactus portal.

Currently AEI scientists use NCSA alliance, DOE, and GermanGrid certificates which are widely accepted.

### 3.3 User Interaction

- Which parts should be automated?

When users start a Cactus job (manually), the simulation should register itself with the Cactus portal and provide the static and dynamic metadata for online monitoring/steering and later retrieval. This includes access also to temporary data output (eg. logfiles) while the simulation is still running. A simulation metadata catalogue should keep all the simulation metadata ready for querying/updating by arbitrary users.

- Which user interface are you planning to use?

A GridSphere-based Cactus portal as user interface.

An extended HTTPD version (security added based on Grid certificates, able to overcome VPN/firewalls) as application interface to monitor/steer a Cactus simulation.

GAT wherever applicable (eg. to announce metadata, partial remote file access).

- Are you planning to use any standard for application monitoring/steering?

For job monitoring we would like to use solutions developed by the DGI/HEP.

For application monitoring/steering we will probably stick with our existing approaches but would like to use existing solutions to overcome firewall/VPN issues.

For remote data visualisation we would like to use DGI/other solutions and/or extend our own approaches for efficient partial remote HDF5 file access.

- Aspects of a Portal / WWW based interface:

. Which portal features are mandatory/optional

these are all mandatory:

\* simple-to-use credential management (eg. using GAMA;

[http://www.gridisphere.org/gridsphere/html/mardigrasworkshop2005/09\\_sdsc\\_gamma.pdf](http://www.gridisphere.org/gridsphere/html/mardigrasworkshop2005/09_sdsc_gamma.pdf))

\* resource/job monitoring

\* application monitoring/steering

\* information service for simulation metadata (browse through all simulations' metadata to find similar parameter settings etc.)

\* metadata catalogue management

these are optional:

\* job management (start/stop)

\* file management

. How are user managed? Where is information about users defined / stored?

\* left open to the portal developers

. Which authentication/authorisation methods are needed ?

\* standard grid authentication/authorisation methods

which must be compatible to local authentication/authorisation methods (eg. shared authentication/authorisation scheme for both the Cactus portal and Cactus-specific application services such as HTTPD)

. Do you want to access specific data services (web services, databases, etc.) via a portal?

N/A

. Are there any existing programs, on which the user interface should be based OR which should be replaced by the portal?

Portlets for a prototype Cactus portal already exist and should be reused and extended if possible. HTTPD should be kept as a portal-independent method for Cactus application monitoring/steering.

. Should there be a central AstroGrid portal OR do you want to set up a portal server for each scenario/application ?

We need a Cactus users portal.

User management may be done via a central AstroGrid portal.

. Does the scenario require any special interfaces OR is it sufficient to use

generic interfaces ?

We do want to use generic interfaces wherever possible. But they must be flexible enough to fit the application's needs.

- Aspects of a generic Grid Application Programming API (GAT)
  - . Which GAT functionality would you like to make use of (eg. job submission, file handling, resource brokering, etc.) ?
    - \* metadata access
    - \* remote file access
    - \* optionally job submission
  - . What programming languages must be supported ? Which platforms ?
    - \* The Cactus application code is C/C++/Fortran90.
    - So the smallest denominator is C. Which is available on all platforms.
  - . Which Grid Middleware should be supported (Globus, Unicore, gLite, etc.) ?
    - \* Globus x.x, everything else optional
  - . For specific GAT functionality, which protocols/packages/tools should be supported ?
    - eg. for job management: clusters with PBS, SGE, Condor
    - \* metadata access: whatever the metadata management developers come up with
    - \* remote file access: partial access to ASCII and HDF5 files
    - \* job submission: PBS, LSF, SGE

### 3.4 Input

- Do you handle input data manually or do you need an automated management of data?
  - \* it's handled manually by the user

### 3.5 Output

- Do you handle output data manually or do you need an automated management of data?
  - \* usually it's handled manually by the user but for postprocessing we want to write scripts which automatically visualise output data from a given simulation; such scripts should be able to query the location and type of data from a metadata catalogue

### 3.6 Additional Information

- How long (avg) does the scenario execute (minutes, hours, days)? Do you aim at a specific speedup?
  - same as described above; since the number of resources doesn't change no speedup is expected for a single simulation
- How often will the scenario be executed?
  - Each simulation is unique.
- Which restrictions of the current approach (as described in section 2) do you want to overcome?
  - see above: VPN/firewall issues, efficient remote file access

## 4. Bigger Picture for the far future

#### 4.1 Organization of Multiple Runs

#### 4.2 Handling relationships between data products

See descriptions above. These are actually goals for the medium-term future.

#### 4.3 Constructing More Complex Runs

Such as distributed Cactus simulations across multiple HPC resources (metacomputing); this is really only a long-term goal of the project.

This will require also an advanced job management: resource brokering (HPC resource with GridMPI installations), co-reservation and allocation. Distributed Cactus jobs are probably too difficult to be started manually by the users so automatic support should be provided through a portal job submission interface. The distributed nature of generated output files will add another layer of complexity to file management services.