

Monitoring and Steering Capabilities in the Cactus Code

Author: Thomas Radke

This webpage describes in detail the application monitoring and steering capabilities that are currently implemented in the [Cactus Code Simulation Framework](#). It does that by means of a concrete example: a live Cactus run that is running as a perpetual simulation on a dedicated Cactus maintainers' machine. For this example, any user who has a standard webbrowser at hand can log into this simulation, monitor its current state, and is also able to steer certain parameters.

The URL to connect to the simulation is <http://cactus.cct.lsu.edu:5555/>. More information on the Cactus pertual wavetoy simulation can be found on the [Cactus Wave Toy Demo webpage](#).

The Cactus Webserver Thorn HTTPD

The Cactus simulation framework provides a generic code module (in Cactus terminology: a *thorn*) named HTTPD. This thorn implements a standard HTTP-1.0 webserver. It is written in C and uses standard UNIX socket communication calls on the network I/O layer.

If this thorn is compiled into a Cactus configuration and activated at runtime, it serves as an ordinary HTTP webserver embedded in a running Cactus simulation: it listens on a chosen UNIX port for incoming HTTP requests from external clients and sends back appropriate HTTP replies. The port number to use can be chosen by the user in the parameter file at simulation startup; port hunting may be used to automatically search for an available port (within a certain port range) if the one specified in the parameter file is already in use. Thorn HTTPD runs in sequential mode, ie. for a parallel simulation it will only be active on the first processor (with ID 0). The URL for external clients to connect to thorn HTTPD follows the general scheme `hostname:port`.

When the simulation starts, thorn HTTPD will print, after some other messages describing some startup parameters, a line to `stdout` containing the URL of the Cactus HTTP webserver. Since at least the port number (and on a cluster usually also the machine name where the simulation has been scheduled to) is known only at runtime, this line needs to be `grep`'ed for in each run's `stdout` logfile in order to find out how to connect to a specific simulation.

```
~/Cactus/par> mpirun -np 4 ../exe/cactus_WaveDemo WaveDemo.par
```

```
-----  
---
```

```
      10  
1    0101      *****  
01   1010 10   The Cactus Code V4.0  
1010 1101 011   www.cactuscode.org
```

```
1001 100101 *****
00010101
100011 (c) Copyright The Authors
0100 GNU Licensed. No Warranty
0101
```


```
Cactus version: 4.0.b13
Compile date:   Mar 22 2004 (16:20:23)
Run date:      Mar 23 2004 (10:56:35)
Run host:      cactus.cct.lsu.edu
Executable:    ../exe/cactus_WaveDemo
Parameter file: WaveDemo.par
```

--
...

Server started on http://cactus.cct.lsu.edu:5555/

...

Right after thorn HTTPD has been started, it registers a callback function which will be called periodically during the main evolution after every iteration by the Cactus scheduler. This callback will then check for pending incoming HTTP requests and, if there are any, process them in order.

If Cactus was configured to use POSIX threads, thorn HTTPD callback function is spawned in a separate thread and will thus run concurrently to the main simulation thread. It is assumed that HTTPD doesn't require a lot of CPU power so that it will interfere only very little with the actual simulation in progress (*non-obtrusive application monitoring*). This feature greatly improves the response time to answer HTTP client requests in the case where the average elapsed time for a single iteration is large (in the order of several seconds or more).

Basic Application Monitoring with HTTPD

Thorn HTTPD provides some standard information about the current Cactus job on a [Simulation Homepage](#):

1. static metadata
 - descriptive name simulation (if specified by the user in the parameter file)

- the name and build date of the Cactus executable, along with the version ID of the Cactus flesh source code (release number)
 - the name of the parameter file
 - the user login name who started this Cactus simulation
 - the number of processors used
 - the hostname of processor 0 (where thorn HTTPD is running on)
2. dynamic metadata (updated each time when a client's HTTP request is being processed by thorn HTTPD)
- the current iteration number
 - the simulation's current physical time
 - elapsed time since simulation startup
 - estimated time per iteration
 - estimated time to completion

The URL of a Cactus job's *Simulation Homepage* corresponds to the `hostname:port` address on which thorn HTTPD is listening for client requests.

HTTPD's Application Programming Interface

Thorn HTTPD not only provides a basic set of webpages with monitoring information but also defines a (Cactus-specific) API which can be used by other code modules to easily and generically extend the served web space of a simulation by additional embedded monitoring capabilities, including simulation steering.

Thorns can register with HTTPD their own HTTP reply callbacks to process incoming client requests for a certain URL (as a sub-URL of the main Simulation Homepage). The API also defines a set of utility routines to parse HTML query strings (for parameter extraction) and to compose user-defined HTTP replies (eg. to send back an ordinary HTML webpage, to authenticate a user, to set/remove HTTP cookies, etc.).

Extended Application Monitoring with HTTPD and HTTPDExtra

In addition to the [Simulation Homepage](#), thorn HTTPD itself already provides a number of individual webpages with Cactus-specific monitoring and steering functionality:

- a [Cactus Control](#) page which serves as a simulation control panel for this run (see below)
- a [Thorns](#) page with detailed information about the Cactus flesh and individual compiled-in and activated thorns
- a [Parameters](#) page with Cactus parameter information and control
- a [Groups and Variables](#) page with information about Cactus grid variables and groups

A companion thorn of HTTPD, thorn HTTPDExtra, uses HTTPD's API to implement its own set of webpages providing even more useful monitoring information:

- a [Message Board](#) which may be thought of as a *collaborative simulation notepad* where users can exchange short text messages
- a [list of registered output files](#) which can be clicked on for download by any client (eg. for some preliminary data analysis)
- a [Viewport](#) containing a list of registered image output files with pre-visualisations of selected Cactus variables - a simple example of remote data visualisation in ongoing simulations.
- [Processor Information](#) specifying the processor layout (for a parallel Cactus simulation) and MPI-related properties
- [Timer Information](#) as a table with the current values of all active Cactus-intrinsic timers (measuring accumulated `gettimeofday` and `getrusage` times for all scheduled routines)

Application Steering with HTTPD

By convention the *Simulation Homepage* and all other webpages served by HTTPD and other thorns are publicly accessible - everybody can log into the simulation and monitor its current state, view visualisations of intermediate data, download output files etc.

In addition to that, thorn HTTPD also provides remote steering capabilities: certain parameters (which are marked in Cactus as *steerable at runtime*) can be changed, the execution of a running simulation can be switched to single-step mode or set to continue until a given iteration number, physical simulation time, or another user-defined breakpoint condition is met. A simulation can also be triggered to (gracefully) terminate immediately after the current evolution timestep.

At simulation startup a user database with username/password/encryption information is created from the corresponding HTTPD user/password parameter settings in the parameter file. The encryption schemes currently implemented are `none` for plaintext passwords and `crypt` for passwords encrypted with the `crypt(3)` function.

Before any parameters can be set or the execution mode can be changed, a user has to authenticate her-/himself first on the [Cactus Control Panel page](#). It uses the standard username/password authentication method as implemented in the HTTP protocol (`Authorization` header field). All incoming HTTP requests for application steering webpages, such as the [Control and Status Page](#) or the [Check/Modify Parameters Pages](#) are then checked before further processing by matching the HTTP `Authorization` header field information against the corresponding entry in the user database.

The authentication information for the public Cactus WaveToy demo running on <http://cactus.cct.lsu.edu:5555> is `anon` for both the username and password.