SIMPLIFYING COMPLEX SOFTWARE ASSEMBLY
THE COMPONENT RETRIEVAL LANGUAGE AND IMPLEMENTATION

Presenter:
Eric Seidel
Dept. of Computer Science
City College of New York
eric@eseidel.org

Co-authors:
Gabrielle Allen, Steven Brandt, Frank Löffler, and Erik Schnetter
Center for Computation & Technology
Louisiana State University
COMPONENT FRAMEWORKS

• Set of individual software modules coordinated by glue framework
  • Each component (module) performs a specific task and encapsulates a set of related functions data
  • Frameworks can range from having a few components to many
  • Components communicate via interfaces
• Used for various purposes, HPC examples include
  • Cactus Framework
  • CCA Frameworks (e.g. Caffeine)
  • Domain specific frameworks (e.g. Earth System Modeling Framework)
CACTUS

- Component Framework
  - Over 500 unique components
  - Distributed around the world
- Flesh
  - Core application
- Thorns
  - Independent modules
  - Perform actual computation
- High Performance Computing
  - Massively parallel
  - Runs on high end supercomputer clusters
- Supports many applications
  - Numerical Relativity
  - Quantum Gravity
  - Computational Fluid Dynamics

www.cactuscode.org
CACTUS WORKFLOW

• Managed using “Thornlists”
• Plaintext list of thorns required for a specific configuration
• Used to checkout, update, build, and test the source code
EINSTEIN TOOLKIT

• Toolkit for relativistic astrophysical simulations
• Developed using Cactus
  • Comprised of 135 thorns
  • Initial Data, Evolution/Analysis methods, Utilities
• First official release 2 months ago

www.einsteintoolkit.org
MOTIVATION

• Distributed Software Frameworks are hard to assemble and manage
  • Einstein Toolkit comprised of 135 individual components
  • Very tedious to manually checkout or update
    • Large barrier to entry for new users
VERSION CONTROL SYSTEMS

- Used to track revisions in source code
- Concurrent Versions System (cvs)
  - Released in 1990
  - Uses client-server model
    - Server stores full history of repository
    - Clients retrieve specific revision
- Subversion (svn)
  - Released in 2000
  - Successor to cvs
  - Also uses client-server model
- Git
  - Released in 2005
  - Uses distributed model
    - Everyone has copy of full history
GET CACTUS

- Designed to checkout and update Cactus thorns and flesh
- Specific to Cactus Framework
- Originally designed for CVS
  - SVN and git added later
- Still difficult to distribute the framework
  - Users must edit the thornlist

!REPOSITORY_TYPE pserver
!REPOSITORY_LOCATION cvs.cactuscode.org
!REPOSITORY_NAME /cactusdevcvs
!REPOSITORY_USER eric9

CactusBase/Boundary
CactusBase/CartGrid3D
CactusBase/CoordBase
CactusBase/IOASCII
CactusBase/IOBasic
CactusBase/IOUtil
CactusBase/InitBase
CactusBase/LocalInterp
• Designed to fix problems with original GetCactus script
• Provides unified, tool agnostic syntax
• Abstracts authentication procedures
• General-Purpose
  • No longer specific to Cactus

COMPONENT RETRIEVAL LANGUAGE

# NAME is an alphanumeric or '.' character

DOCUMENT : DIRECTIVES ;

DIRECTIVE : DEFINE NAME '=' PATH EOL
  | CHECKOUT '=' COMPONENTLIST EOL
  | CHECKOUT '==' EOL COMPONENTLIST EOL
  | REPO_LOC '==' LOC EOL
  | AUTH_LOC '==' LOC EOL
  | PATH_DIRECTIVE '==' PATH EOL
    # !REPO_PATH, !CHECKOUT, !TARGET,
    # !ANON_PASS, !NAME
  | NAME_DIRECTIVE '=' NAME EOL
    # !CRL_VERSION, !AUTH_USER,
    # !ANON_USER, !TYPE
  ;

DIRECTIVES : DIRECTIVE
  | DIRECTIVES DIRECTIVE
  ;

LOC : PSERVER PATH  # CVS repository
  | NAME ':' '/' '/' PATH  # Git/SVN repository
  | NAME '@' NAME ':' PATH  # Git repository
  ;

PATH : NAME
  | '/' NAME
  | PATH '/' NAME
  ;

COMPONENTLIST : PATH
  | COMPONENTLIST EOL PATH ;
SAMPLE CRL FILE

!DEFINE ROOT = Cactus
!DEFINE ARR = $ROOT/arrangements

!TARGET = $ROOT
!TYPE = svn
!AUTH_URL = https://svn.cactuscode.org/flesh/trunk
!URL = http://svn.cactuscode.org/flesh/trunk
!CHECKOUT = Cactus
!NAME = .

!TARGET = $ROOT
!TYPE = svn
!URL = https://svn.cct.lsu.edu/repos/numrel/$1/trunk
!CHECKOUT = simfactory

!TARGET = $ARR
!TYPE = git
!AUTH_URL = git://github.com/ianhinder/Kranc.git
!URL = https://svn.cactuscode.org/arrangements/$1/$2/trunk
!CHECKOUT = KrancNumericalTools/GenericFD

# McLachlan, the spacetime code
!TARGET = $ARR
!TYPE = git
!URL = git://carpetcode.dyndns.org/McLachlan
!AUTH_URL = carpetgit@carpetcode.dyndns.org:McLachlan
!REPO_PATH = $2
• Designed to be very modular
• Currently supports 5 version control systems and http/ftp downloads
• Very easy to add more
• Can take input as local file or URL
• Manages all authentication issues

./GetComponents  http://tinyurl.com/einsteintoolkit-2010-06
AUTHENTICATION

- Authentication handled entirely by VCS tools
- GetComponents stores list of authenticated repositories and users
  - Also tracks repositories with specified anonymous access
- Very secure
  - GetComponents never sees any passwords!
CHECKOUT VS. UPDATE SPEED

<table>
<thead>
<tr>
<th>TeraGrid Resource</th>
<th>Serial Checkout</th>
<th>Parallel Checkout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abe</td>
<td>325</td>
<td>975</td>
</tr>
<tr>
<td>Frost</td>
<td>975</td>
<td>1300</td>
</tr>
<tr>
<td>Kraken</td>
<td>1300</td>
<td>975</td>
</tr>
<tr>
<td>Lincoln</td>
<td>975</td>
<td>650</td>
</tr>
<tr>
<td>LoneStar</td>
<td>650</td>
<td>325</td>
</tr>
<tr>
<td>Longhorn</td>
<td>325</td>
<td>0</td>
</tr>
<tr>
<td>Queen Bee</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ranger</td>
<td>1300</td>
<td>975</td>
</tr>
<tr>
<td>Spur</td>
<td>975</td>
<td>1300</td>
</tr>
<tr>
<td>Steele</td>
<td>1300</td>
<td>975</td>
</tr>
</tbody>
</table>

Wednesday, August 4, 2010
• Generating component lists is still time-consuming and tedious
  • Barrier/impossible for new users
• Don’t need all Einstein Toolkit modules to run a simulation
  • How to determine which components are needed for a particular simulation?
  • e.g. what is needed to model two black holes, or a coastal surge?
COMPONENT DEPENDENCIES

- Dependency tracking could allow custom built simulations
- Specify one component containing data about the simulation
  - Initial values, type of simulation, etc
- Then recursively check component dependencies
COMPONENT DEPENDENCIES
-- WAVETOY EXAMPLE
COMPONENT DEPENDENCIES
-- QUANTUM GRAVITY
COMPONENT DEPENDENCIES
-- EINSTEIN TOOLKIT
DISTRIBUTION

- GetComponents is freely available with an open-source license
- [www.eseidel.org/download/GetComponents](http://www.eseidel.org/download/GetComponents)
- Full documentation available
  - ./GetComponents --man
ACKNOWLEDGEMENTS

• Many thanks to Gabrielle Allen, Steve Brandt, Frank Löffler, and Erik Schnetter