SAGA

Introduction to the API and Hands-On Tutorial

http://faust.cct.lsu.edu/trac/saga/wiki/ADSSS09

Ole Weidner, Shantenu Jha

ADSSS'09 Abingdon Sept. 03, 2009

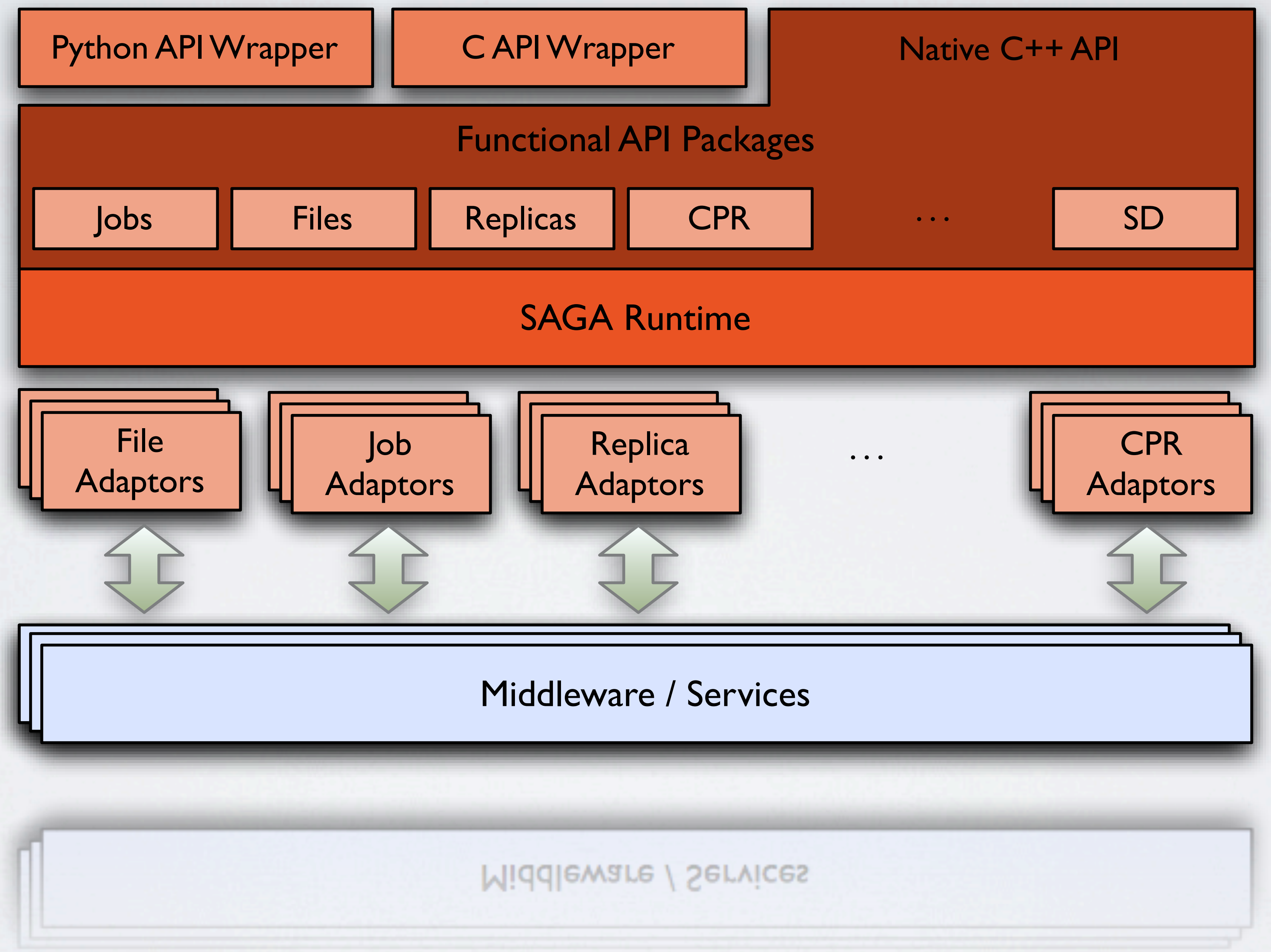
AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
    - Hello (Distributed) World
    - Chaining Jobs
    - Depending Jobs
- Python Language Bindings

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

INTRODUCTION TO THE API



API PACKAGE: JOB

- http://saga.cct.lsu.edu/cpp/apidoc/namespacesaga_1_1job.html
- Allows definition, submission, management and monitoring of interactive and batch jobs
- Consists of three main classes
  - saga::job::description - used to describe a saga job
  - saga::job::service - represents a (remote) computing resource
  - saga::job::job - represents the job itself
- Currently the following job package adaptors are available:
  - Fork (local), Globus GRAM2, SSH, OMII GridSAM, Condor, Amazon EC2, Platform LSF

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

API PACKAGE: JOB CONT.

```
try {

    saga::url js_url (“gram://gatekeeper.lonestar.tacc.teragrid.org:2119/jobmanager-lsf”);

    saga::job::description jd;
    jd.set_attribute (attributes::description_executable, “/home/oweidner/tests/heat_transfer”);
    jd.set_attribute (attributes::description_number_of_processes, "2");
    jd.set_attribute (attributes::description_queue, "checkpoint");

    saga::job::service js (js_url);
    saga::job::job my_job = js.create_job (jd);

    my_job.run();

    std::cout << "Job ID    : " << my_job.get_job_id() << std::endl;
    std::cout << “Job STATE : " << my_job.get_state() << std::endl;

    my_job.suspend();
    my_job.resume();

    my_job.cancel();
}

catch (saga::exception const & e) {

    std::cerr << “Ooops: “ << e.what() << std::endl;
}
```

API PACKAGE: FILE

- http://saga.cct.lsu.edu/cpp/apidoc/namespacesaga_1_1filesystem.html
- Can be used to traverse, modify, read and write local and remote filesystems
- Consists of two main classes:
  - saga::filesystem::directory
  - saga::filesystem::file
- Currently the following job package adaptors are available:
  - Local FS, Globus GridFTP, SSH, Hadoop Distributed Filesystem (HDFS), CloudStore KFS, OpenCloud Sector-Sphere

API PACKAGE: FILE CONT.

```
try {

    saga::url file_url (“gsiftp://queenbee.loni-lsu.teragrid.org:2811//home/oweidner/.bashrc”);
    saga::filesystem::file f (file_url, saga::filesystem::Read);

    while ( true )
    {
        saga::size_t const n = 1024*64;
        saga::uint8_t data_buf[n+1];
        for ( unsigned int i = 0; i <= n; ++i ) { data_buf[i] = '\0'; }

        if ( f.read (saga::buffer (data_buf, n), n) )
        {
            // output buffer content
            std::cout << data_buf << std::flush;
        }
        else
        {
            break;
        }
    }
}
catch (saga::exception const & e) {

    std::cerr << “Ooops: “ << e.what() << std::endl;
}
```

API PACKAGE: ADVERT

- http://saga.cct.lsu.edu/cpp/apidoc/namespacesaga_1_1advert.html
- An *advert service* can be used for persistent hierarchical storage of application level information and saga objects
- Semantics and usage-mode is defined by the application:
  - e.g. result storage, synchronization of application components, . . .
- API is very similar to the file package:
  - saga::advert::directory - represents the hierarchical structure
  - saga::advert::entry - represents an advert object
- Currently the following advert package adaptors are available:
  - PostgreSQL / SQLite3

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

API PACKAGE: ADVERT CONT.

```
try {

    saga::url advert_url ("advert://macpro01.cct.lsu.edu//users/oweidner/project01/result");
    saga::advert::entry e (file_url, saga::advert::ReadWrite | saga::advert::Create)

    e.set_attribute("Iteration", "120");
    e.set_attribute("Dataset", "sim_42");

    // You can store just a string
    e.store_string("123.33f");

    // or a saga object, e.g. a file
    saga::url file_url ("gsiftp://queenbee.loni-lsu.teragrid.org:2811//home/oweidner/.bashrc");
    saga::filesystem::file f (file_url, saga::filesystem::Read);
    e.store_object(f);
}
catch (saga::exception const & e) {

    std::cerr << "Ooops: " << e.what() << std::endl;
}
```

MORE API PACKAGES

- saga::replica - Replica management
  - Adaptors: Globus RLS, PostgreSQL / SQLite3
- saga::stream - Stream client and server
  - Adaptors: BSD Sockets
- saga::sd - Service discovery
  - Adaptors: default SD
- saga::cpr - Checkpoint and recover
  - Adaptors: default CPR / MiGOL

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

- Open Source - released under the Boost Software License 1.0
- Implemented as a set of libraries
  - SAGA Core - A light-weight engine / runtime that dispatches calls from the API to the appropriate middle-ware adaptors
  - SAGA functional packages - Groups of API calls for: jobs, files, service discovery, advert services, RPC, replicas, CPR, ... (extensible)
  - SAGA language wrappers - Thin Python and C layers on top of the native C++ API
  - SAGA middle-ware adaptors - Take care of the API call execution on the middle-ware
- Can be configured / packaged to suit your individual needs!

PREREQUISITES

- In order to build & install SAGA you need the following:
  - A UNIX operating system (Linux, MacOS, etc.)
  - A C++ Compiler (preferably gcc >= 3.4)
  - The Boost C++ Libraries (>= 1.33.1) from http://boost.org
  - Python (if you want to build the Python language bindings)
- Adaptors may have additional requirements
  - PostgreSQL / SQLite client libraries
  - Globus / Condor / LSF installations
  - etc ...

WHERE TO GET IT

- Download the latest source release (~8 week release cycle) from:

  http://saga.cct.lsu.edu/cpp/download

- Checkout the latest source from Subversion:

  svn co https://svn.cct.lsu.edu/repos/saga/trunk

HOW TO BUILD SAGA

- Configure/make - based build system. It's as simple as:

  ```
  ./configure --prefix=/usr/local && make install
  ```

- Top-level configure/make recursively calls configures/makes for each adaptor, language bindings, etc...
- Fault tolerant: if one of the sub-level packages can't be configured (missing prerequisites, etc...) it is simply skipped
- Sub-level packages can be built individually and even outside the source tree
- XCode and VisualStudio project files available for developers

DEPLOYMENT 2009/2010

NGS
National Grid Service

Distributed
European
Infrastructure for
Supercomputing
Applications

TeraGrid™
EMPOWERING DISCOVERY

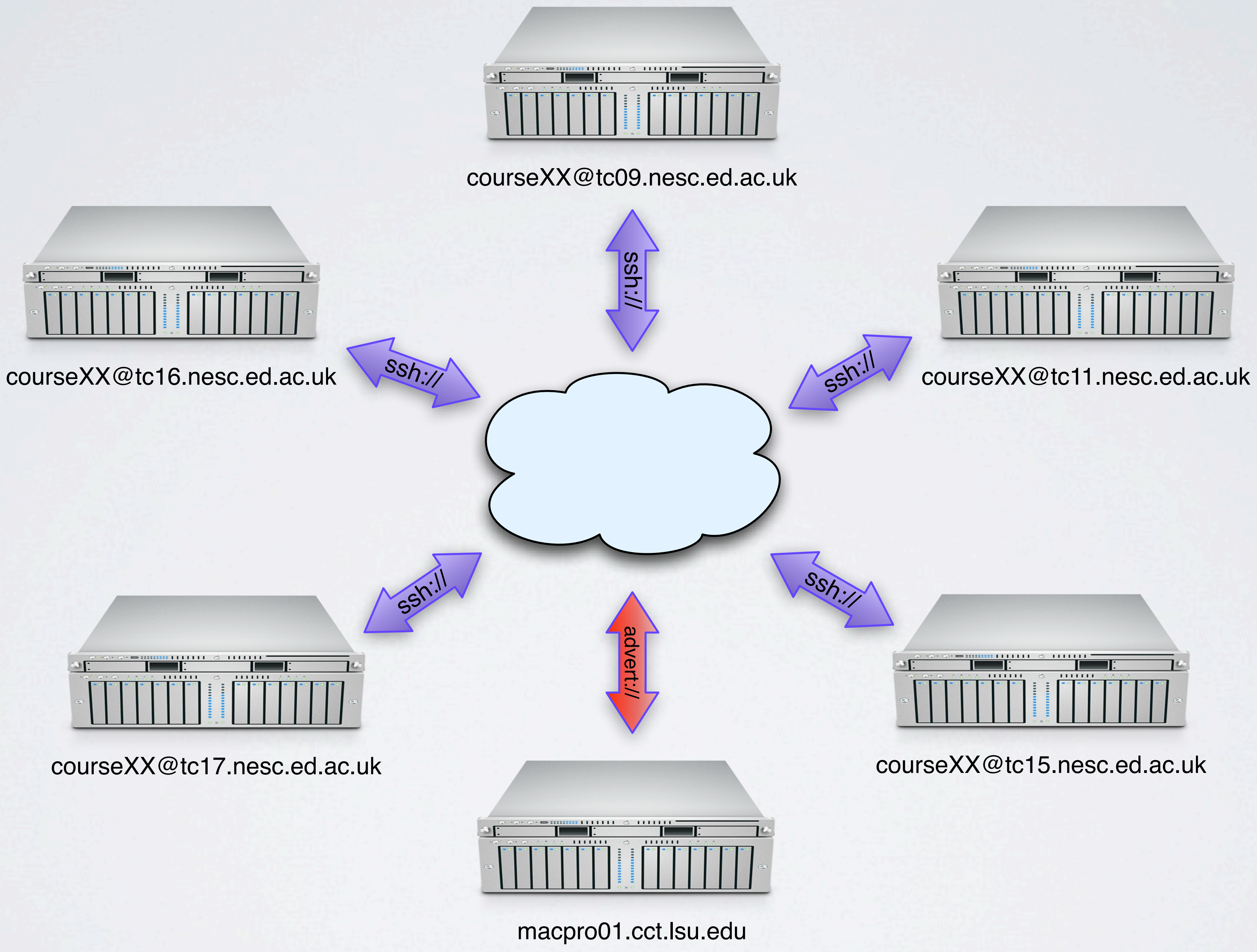
Grid Applications
Grid Middleware
Networking
NAREGI

egee
Enabling Grids
for E-sciencE

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

INFRASTRUCTURE


courseXX@tc09.nesc.ed.ac.uk
ssh://
courseXX@tc16.nesc.ed.ac.uk
ssh://
courseXX@tc11.nesc.ed.ac.uk
ssh://
ssh://
ssh://
advert://
courseXX@tc17.nesc.ed.ac.uk
courseXX@tc15.nesc.ed.ac.uk
macpro01.cct.lsu.edu

INFRASTRUCTURE CONT.

- Login (ssh) to one of the following machines:
  - tc09.nesc.ed.ac.uk
  - tc11.nesc.ed.ac.uk
  - tc15.nesc.ed.ac.uk
  - tc16.nesc.ed.ac.uk
  - tc17.nesc.ed.ac.uk
- Usernames:
  - course01, course02, course03, . . . , course24
- Password: to be announced

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

INFRASTRUCTURE CONT.

- Set up the SAGA environment

  source /usr/local/saga/share/saga/saga-env.sh

```
#!/bin/bash

export SAGA_LOCATION=/usr/local/saga/
export LD_LIBRARY_PATH=${SAGA_LOCATION}/lib/:${LD_LIBRARY_PATH}
export PATH=${SAGA_LOCATION}/bin:${PATH}

export PYTHONPATH=${SAGA_LOCATION}/lib/python2.6/site-packages/:${PYTHONPATH}
```

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

SAGA-FILE TOOL

- Provides basic functionality of the file package
- Examples:
  - List the contents of a directory
    saga-file list_dir file://localhost//tmp/
  - Get the size of a file
    saga-file get_size file://localhost//etc/passwd
  - Copy a file
    saga-file copy /etc/passwd ssh://tc11//tmp/courseXX_etcpasswd_copy
  - Print the contents of a file
    saga-file cat ssh://tc11//tmp/courseXX_etcpasswd_copy

SAGA-JOB TOOL

- Provides basic functionality of the job package
- Examples:
    - Submit a non-interactive job
      `saga-job submit ssh://localhost /bin/touch /tmp/blah`
    - Run an interactive job
      `saga-job run ssh://tc09 /bin/cat /proc/cpuinfo`

SAGA-ADVERT TOOL

- Provides basic functionality of the advert package
- Examples:
  - List the content of an advert directory
    saga-advert list_directory advert://macpro01.cct.lsu.edu//ADSSS09 ?
  - Create an advert entry
    saga-advert add_entry advert://macpro01.cct.lsu.edu//ADSSS09/aloha
  - Attach an attribute
    saga-advert set_attribute advert://macpro01.cct.lsu.edu//ADSSS09/aloha Foo Bar
  - List all attributes
    saga-advert list_attributes advert://macpro01.cct.lsu.edu//ADSSS09/aloha

HANDS-ON

- Try and run command line tools
  - Copy a file, move it, delete it, read its contents (local / remote)
  - Run a job (/bin/sleep 20), monitor its status (local / remote)
  - Use the advert service to create directories, entries, store data, set attributes

    NOTES:
    - Please stay within advert://macpro01.cct.lsu.edu//ADSSS09/
    - Create a sub-directory for your username, e.g. course24

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

EXAMPLES

- We will go over three different examples
- For each example:
  - Create a subdirectory, e.g. ~/ex01 ~/ex02 ~/ex03
  - Download the Makefile from the tutorial wiki
  - Download the source files form the tutorial wiki
  - Compile
  - Run

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

EXAMPLE I

- Spawn 3 (remote) jobs (/bin/echo) with the 3 words "Hello", distributed", and "world!" as their arguments.
- You can change these three lines:

```
#define HOST1 "fork://localhost"
#define HOST2 "fork://localhost"
#define HOST3 "fork://localhost"
```

- What do you observe when you run it multiple times?

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

SAGA

A Simple API for Grid Applications [http://saga.cct.lsu.edu]

EXAMPLE 2

- Introduces dependencies (ordered execution) between the jobs
- Each job receives the output of the previous job (integer number), increments it by one and passes it to the next job
- Again, change these three lines:

```
#define HOST1 "fork://localhost"
#define HOST2 "fork://localhost"
#define HOST3 "fork://localhost"
```

- Can you come up with other, real-life uses-cases (maybe even from your own field of work/research) for this usage mode?

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
  - Hello (Distributed) World
  - Chaining Jobs
  - Depending Jobs
- Python Language Bindings

EXAMPLE 3

- Initial job will re-spawn itself on a set of different hosts (copy +execute)
- Each instance will increment a number stored in a central result store (Advert DB)
- You have to provide the hosts you want to spawn to on the command line
- Display the result using the saga-advert command line tool

- Again, can you think of a use-case for this usage mode?

AGENDA

- Introduction to the API
- Requirements and Installation
- Tutorial Infrastructure
- Command Line Utilities
- Code Examples (C++)
    - Hello (Distributed) World
    - Chaining Jobs
    - Depending Jobs
- Python Language Bindings

PYTHON

```
$ python
Python 2.6.2 (r262:71600, Aug 28 2009, 21:32:19)
[GCC 3.4.6 20060404 (Red Hat 3.4.6-10)] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> import saga
>>> f = saga.file.file("ssh://tc15//etc/adjtime")
>>> print f.read()
0.0 0 0.0
0

>>> js_url = saga.url("fork://localhost/")
>>> job_service = saga.job.service(js_url)
>>> job_desc = saga.job.description()
>>> job_desc.executable = "/bin/date"
>>> my_job = job_service.create_job(job_desc)
>>> my_job.run()
```

EXERCISE?

THANKS

Questions / Comments ?