



CENTER FOR COMPUTATION
& TECHNOLOGY

SAGA

A Simple API for Grid Applications

SAGA API Examples: Shell, Python and C++



omii-uk
www.omii.ac.uk



Agenda

- ▣ SAGA command line tools
- ▣ SAGA Python API
- ▣ SAGA C++ API
- ▣ Examples

Documentation

▣ General information

- ▣ https://svn.cct.lsu.edu/repos/saga-projects/tutorial/general_tutorial
- ▣ <http://saga.cct.lsu.edu/software/cpp/documentation/tutorials/loni-training-2010>

▣ API documentation

▣ Python

- ▣ <http://static.saga.cct.lsu.edu/apidoc/python/latest/>

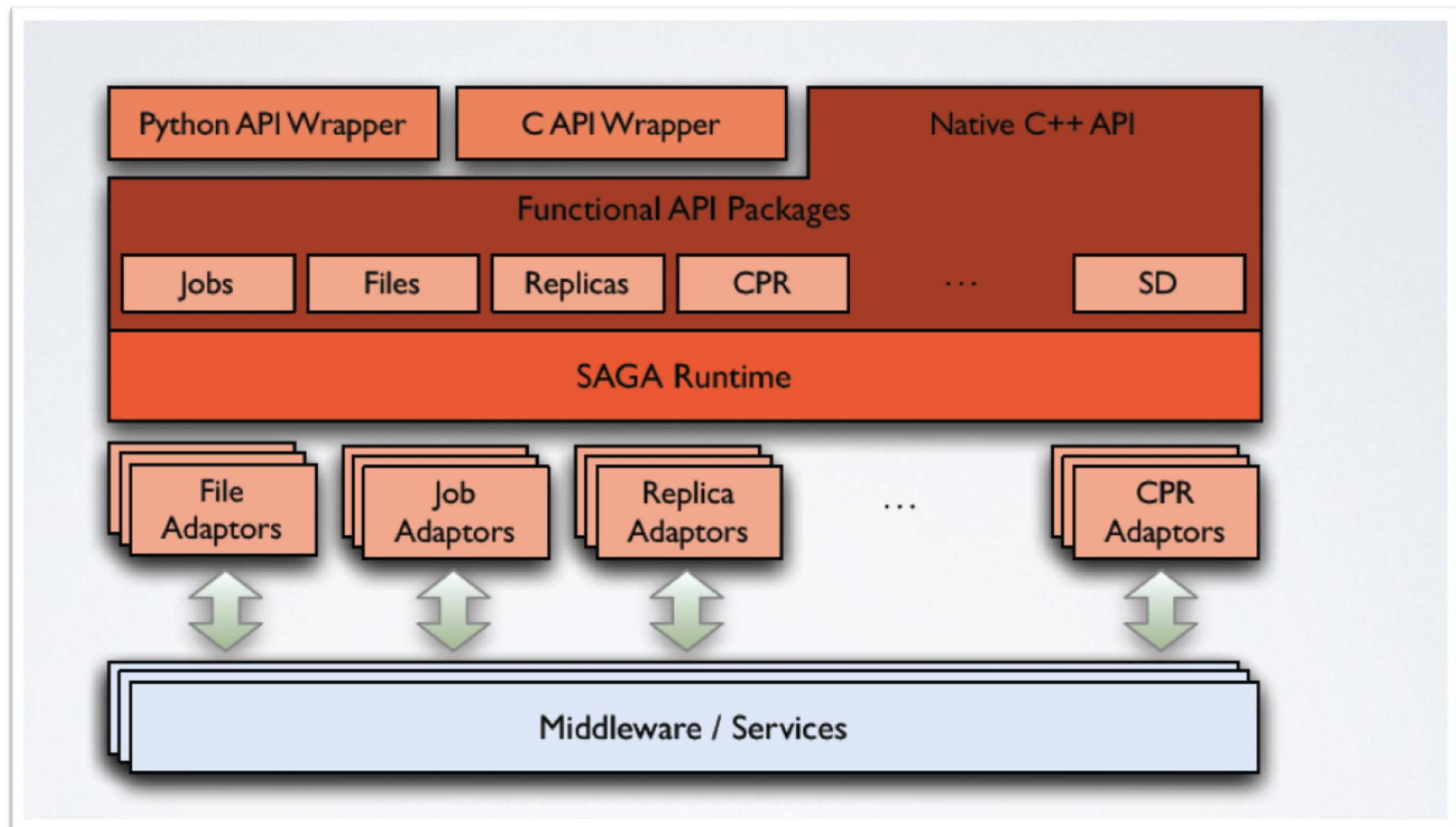
▣ C++

- ▣ <http://static.saga.cct.lsu.edu/apidoc/cpp/latest/>

▣ Programmers manual

- ▣ http://static.saga.cct.lsu.edu/docs/programming_guide/html/saga-programming-guide.html

SAGA: Architecture



API Levels

	Local Adaptors	Globus Adaptors	SSH Adaptors	...
	file://localhost/... any://localhost/...	gram://remotehost/... any://remotehost/...	ssh://remotehost/... any://remotehost/...	
Shell	saga-file copy src dest			
	saga-job run rm cmd			
Python	import saga.filesystem dir.copy(src, dest)			
	import saga.job js.run(cmd)			
C++	using saga::filesystem::directory; dir.copy(src, dest)			
	using saga::job::job; job.run(cmd);			

Command line tools

- SAGA comes with simple command line tools that allow to access basic package functionality.
- The source code is very simple and a great starting point to explore the SAGA package APIs:
 - `saga-file` `$SAGA_ROOT/saga/tools/clutils/file/`
 - `saga-job` `$SAGA_ROOT/saga/tools/clutils/job/`
 - `saga-advert` `$SAGA_ROOT/saga/tools/clutils/advert/`
 - `saga-shell` `$SAGA_ROOT/saga/tools/shell/`

Command line tools

- ▣ 'Shell bindings'
 - ▣ Package specific (file, job, advert, replica)
- ▣ SAGA shell
 - ▣ All in one solution
 - ▣ Filesystem navigation (filesystem, advert, replica)
 - ▣ Job launching
 - ▣ Scripting

Command line tool: saga-file

- Supported protocols
 - Depends on SAGA adaptors
 - Also available: Globus GridFTP, Curl (subset), KFS, Amazon EC2, Opencloud (Sector/Sphere), Hadoop (HDFS)
- Supported commands:

Command	Arguments
copy	<url from> <url to>
move	<url from> <url to>
remove	<url>
cat	<url>
list_dir	<url>

Command line tool: saga-job

- ▣ Supported protocols
 - ▣ Depends on SAGA adaptors
 - ▣ Also available: Globus Gram, Condor, OMI-GridSAM, LSF, Amazon EC2, Opencloud (Sector/Sphere)
- ▣ Supported commands:

Command	Arguments
run	<rm url> <command> <arguments>
submit	<rm url> <command> <arguments>
state	<rm url> <jobid>
suspend	<rm url> <jobid>
resume	<rm url> <jobid>
cancel	<rm url> <jobid>

Command line tool: saga-advert

- ▣ What is it?
 - ▣ Central data store with
 - ▣ Hierarchical keys
 - ▣ Attributes
 - ▣ Filesystem like structure
- ▣ Supported protocols
 - ▣ Depends on SAGA adaptors
 - ▣ Local adaptor:
 - ▣ Local backend: SQLite3
 - ▣ Remote backend: PostgreSQL
 - ▣ Also available: Hadoop H-Base, Hypertable

Command line tool: saga-advert

Command	Arguments
list_directory	<advert-url> <pattern>
add_directory remove_directory	<advert-url>
add_entry remove_entry	<advert-url>
store_string	<advert-url> <string>
retrieve_string	<advert-url>
list_attributes	<advert-url>
set_attribute	<advert-url> <key> <value>
remove_attribute	<advert-url> <key>

Command line tool: saga-shell

- ▣ All in one of all command line tools as mentioned earlier
- ▣ Keeps context in between commands
- ▣ Navigate (remote) filesystems (advert, replica too!)
- ▣ Launch (remote) jobs, uses io redirection to access in/out
- ▣ All commands are implemented using SAGA

Command line tool: saga-shell

Type	Commands
File system navigation	pwd, ls, mv, cp, cd, mkdir, rmdir, touch, cat
Job package	run, suspend, resume, kill, status, ps
replica	rep_find, rep_list, rep_add, rep_remove, rep_update, rep_replicate
environment	setenv, getenv, env
permissions	add_proxy, remove_proxy

Python API Example: File Package

▣ Copy a file

```
using saga
src = url(' ... ')
dst = url(' ... ')
f = filesystem.file(src, filesystem.ReadWrite)
f.copy(dst)
```

Python API Example: File Package

▣ Get a directory file listing

using saga

```
src = url(' ... ')
```

```
d = filesystem.directory(src)
```

```
names = d.list('*')
```

```
for name in names:
```

```
    ns = saga.name_space.entry(name)
```

```
    if ns.is_dir():      print 'd ', name
```

```
    elif ns.is_link():  print '->', ns.read_link()
```

```
    else:               print ' ', name
```

Python API Example: Job Package #1

▣ Submit a job

```
import saga
```

```
js_url = url("fork://localhost/")  
job_service = job.service(js_url)  
job_desc = job.description()  
job_desc.executable = "touch"  
job_desc.arguments = ["-a", "...filename..."]  
my_job = job_service.create_job(job_desc)  
my_job.run()
```


Python API Example: Job Package #2

▣ Submit a job

```
import saga
```

```
js_url = saga.url("fork://localhost/")
```

```
job_service = job.service(js_url)
```

```
(job, sin, sout, serr) = job_service.run_job( \
    "/bin/echo -n HELLO SAGA")
```

```
print sout.read()
```

Python API Example: Advert Package

- ▣ Create and modify an advert entry

```
# host A
using saga
name = url(' ... ')
e = advert.entry(name, advert.ReadWrite | advert.Create)
e.set_attribute('started', ' ... ')
```

```
# host B
using saga
name = url(' ... ')
e = advert.entry(name)
started = e.get_attribute('started')
```

C++ API Example: File Package

▣ Copy a file

```
saga::url src (' ... ');  
saga::url dst (' ... ');  
saga::file f(src, saga::filesystem::ReadWrite);  
f.copy(dst);
```

C++ API Example: File Package

- ▣ Get a directory file listing

```
saga::url src (' ... ');
saga::directory d (src);
std::vector<std::string> names = d.list('*');

for (auto it = names.begin(); it != names.end(); ++it) {
    saga::name_space::entry ns (*it);
    if (ns.is_dir())          cout << 'd ' << *it << '\n';
    else if (ns.is_link())    cout << '->' << ns.read_link() << '\n';
    else:                    cout << ' ' << *it << '\n';
}
```

C++ API Example : Job Package #1

▣ Submit a job

```
saga::url js_url("fork://localhost/");
saga::job::service js(js_url);
saga::job::description jd;
js.set_attribute("executable", "touch");

std::vector<std::string> args;
args.push_back("-a");
args.push_back("...filename...");
js.set_vector_attribute("arguments", args);

saga::job::job j = js.create_job(jd);
j.run();
```

C++ API Example : Job Package #2

▣ Submit a job

```
saga::job::service js;  
saga::job::ostream in;  
saga::job::istream out, err;
```

```
saga::job::job j = js.run_job(  
    "/bin/echo -n HELLO SAGA", "fork://localhost/",  
    in, out, err);
```

```
std::string line;  
while (std::getline(line, out))  
    std::cout << out << std::endl;
```

C++ API Example: Advert Package

- Create and modify an advert entry

```
// host A
```

```
saga::url name = url(' ... ')
```

```
saga::advert::entry e(name,  
                      advert.ReadWrite | advert.Create)
```

```
e.set_attribute("started", " ... " )
```

```
// host B
```

```
saga::url name = url(' ... ')
```

```
saga::advert::entry e(name, advert.Read)
```

```
std::string started = e.get_attribute("started")
```

Programmers Guide

- ▣ Set of very small and easy examples, one for each package/paradigm
 - ▣ file_copy, file_copy (async)
 - ▣ Error handling
 - ▣ Attributes
 - ▣ Stream (server/client)

Example 1: hello_world

- ▣ Hello world
 - ▣ Launch 3 jobs on different machines
 - ▣ Execute `"/bin/echo"`
 - ▣ No job dependency
 - ▣ Each job returns its passed input argument
 - ▣ `"Hello"`
 - ▣ `"distributed"`
 - ▣ `"world!"`
 - ▣ Jobs are launched in parallel (in separate threads)
 - ▣ As soon as result is collected it's printed on local console

Example 1: hello_world

□ Hello world

□ Arbitrary sequence of results

- Optimally: "Hello distributed world!"

□ Demonstrates

- How to launch a remote job using SAGA job_service
- Pass arguments using the command line
- Collect result by output redirection

□ The source code can be found here (see 'Example1'):

- <http://faust.cct.lsu.edu/trac/saga/wiki/> **FIXME**
- The example uses localhost to spawn childs
- For remote execution change HOST1, HOST2, HOST3 from "localhost" to "**FIXME**"

Example 2: chaining_jobs

- ▣ Launch 3 jobs on 3 different machines
- ▣ Output of previous job is needed to launch next job
- ▣ Simple sequential execution, but SAGA style
- ▣ Demonstrates
 - ▣ How to launch a job using SAGA job_service
 - ▣ How to feed input to launched job
 - ▣ How to collect output
- ▣ Launched job: `/usr/bin/bc`
- ▣ Increment the number passed as the argument
 - ▣ Pass returned incremented number to next job

Example 3: depending_jobs

- ▣ Coordinating information from advert service
- ▣ Launch a single job sequentially on a set of remote resources
 - ▣ Simulating checkpointing/relaunching on different resource (migration)
- ▣ Maintain a single result value in advert service
 - ▣ Gets written by one job, and read by the next
- ▣ Demonstrates
 - ▣ How to launch remote job using SAGA job, while maintaining environment
 - ▣ Assembling argument lists
- ▣ Result is left in advert service, but accessed afterwards