KEK
HIGH ENERGY ACCELERATOR RESEARCH ORGANIZATION

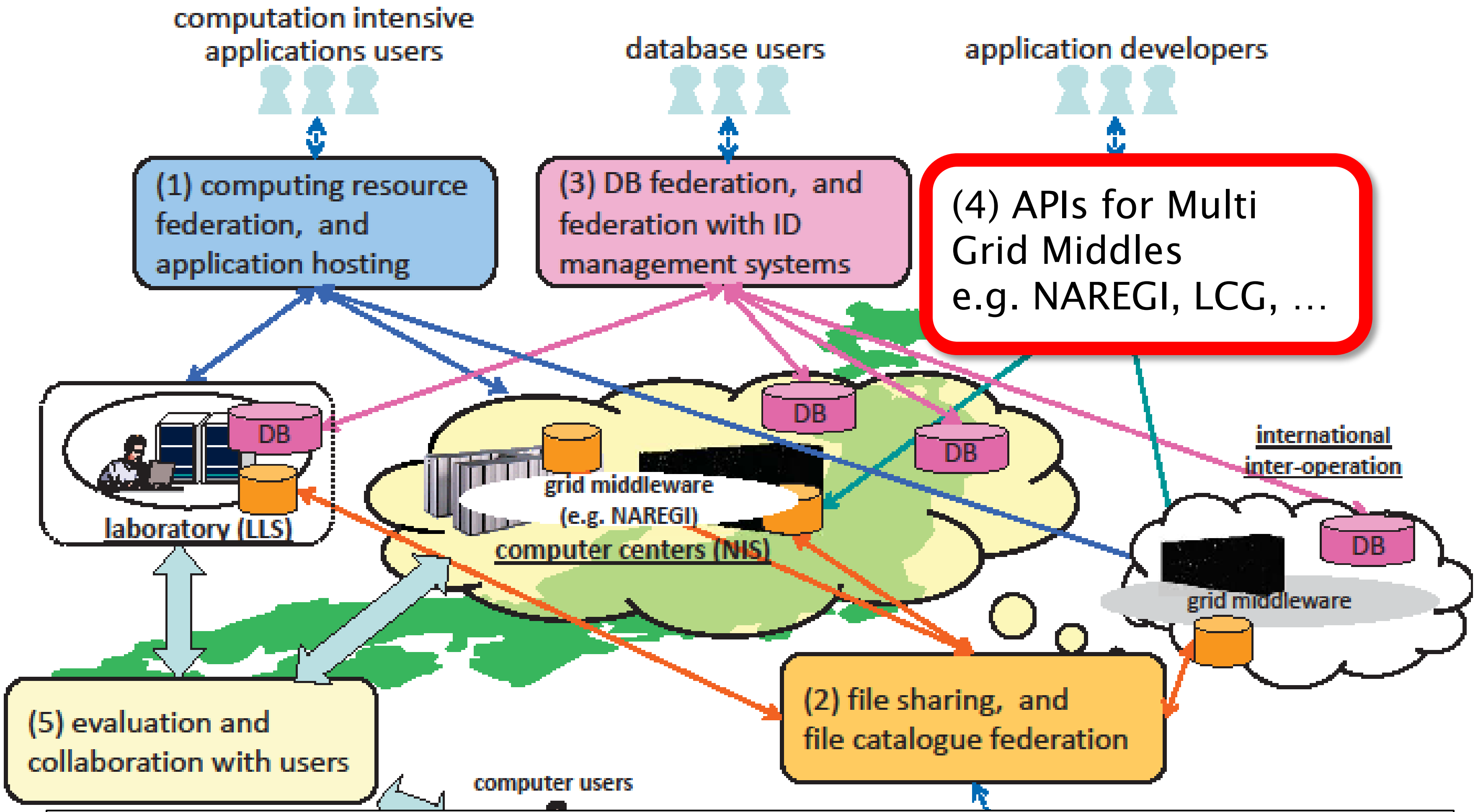
Updates on SAGA related activities since OGF27 on October 2009

Go Iwai, KEK/CRC
On behalf of RENKEI subgroup #4
go.iwai@kek.jp

Agenda

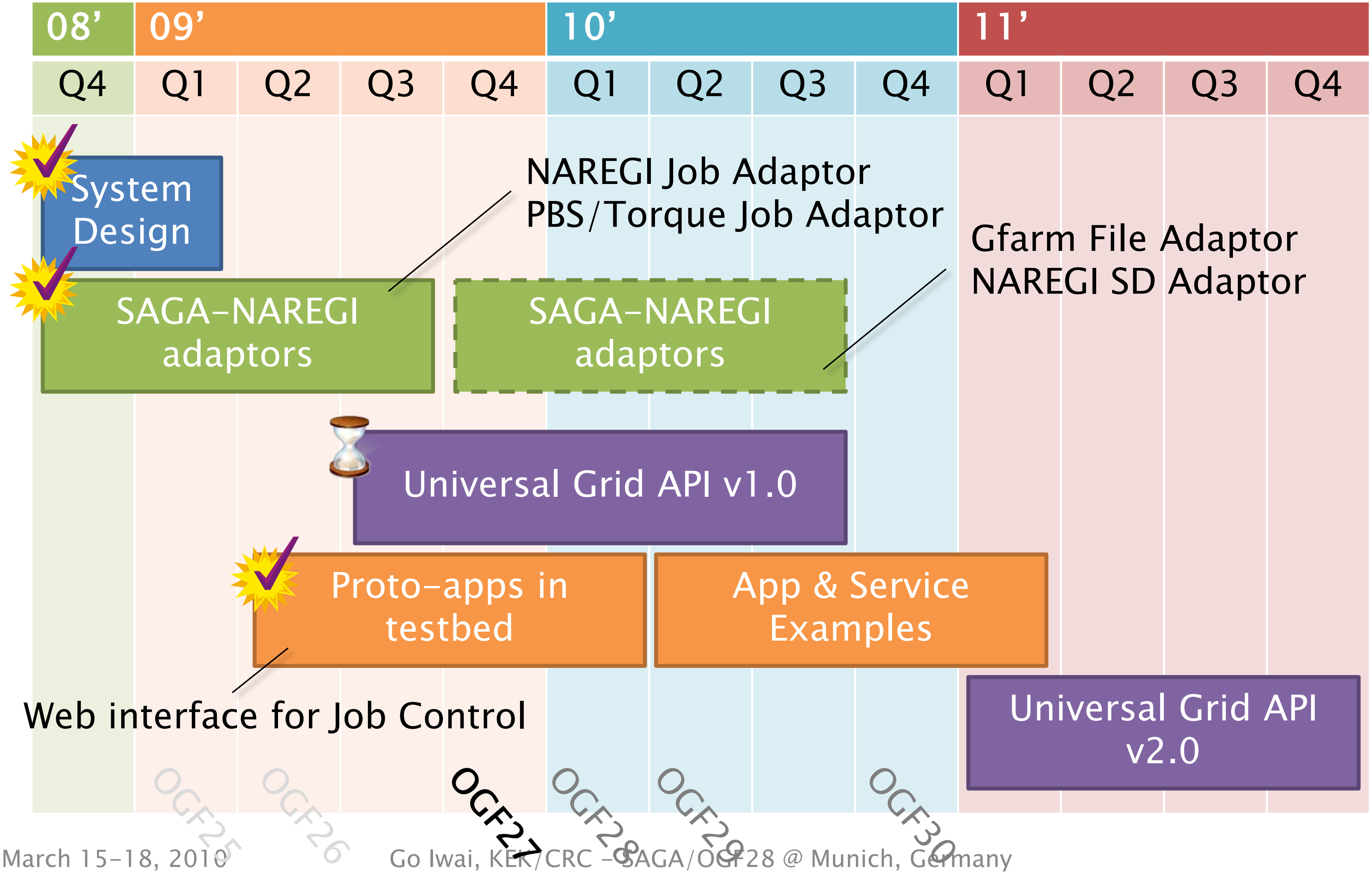
- Status report for SAGA-gfarm (v1&v2) file adaptor
  - Questions and feedback
- Other adaptors to be implemented
  - SAGA-iRODS and SAGA-RNS adaptors
  - Possibility for bridging CSAGA-JSAGA using Python

REsources liNKage for E-sclence (RENKEI)

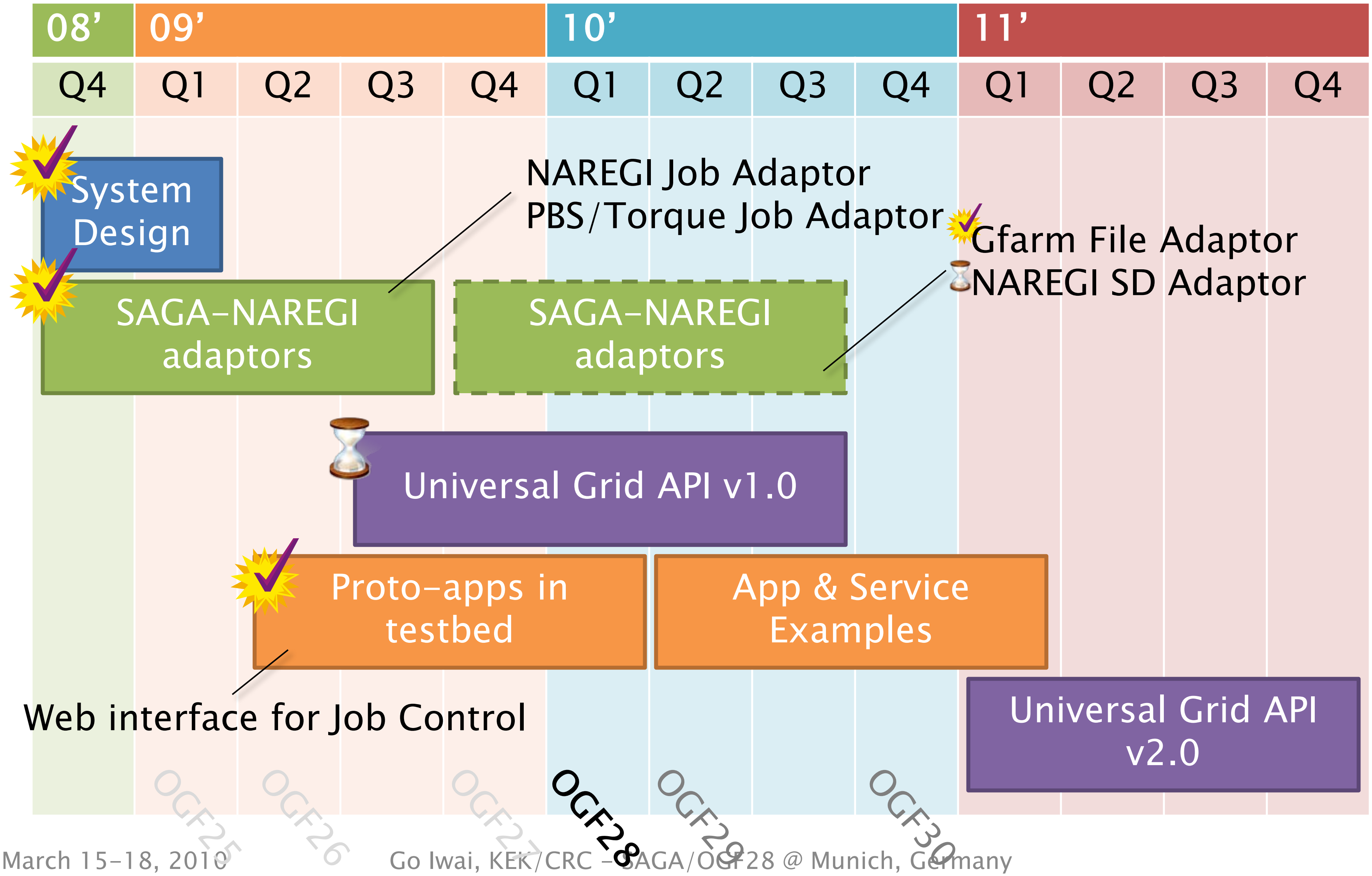

computation intensive applications users
database users
application developers
(1) computing resource federation, and application hosting
(3) DB federation, and federation with ID management systems
(4) APIs for Multi Grid Middles e.g. NAREGI, LCG, …
DB
laboratory (LLS)
grid middleware (e.g. NAREGI)
computer centers (NIS)
international inter-operation
grid middleware
(5) evaluation and collaboration with users
(2) file sharing, and file catalogue federation
computer users

This activity is funded by MEXT as a part of RENKEI project which develops seamless linkage of resources in the Grids and the local one for e-Science.

Development timeframe and status in RENKEI/subg4 at OGF27



Development timeframe and status in RENKEI/subg4 at OGF28



Agenda

- Status report for SAGA-gfarm (v1&v2) file adaptor
  - Questions and feedback
- Other adaptors to be implemented
  - SAGA-iRODS and SAGA-RNS adaptors
  - Possibility for bridging CSAGA-JSAGA using Python

Status

- Just done for adaptor implementation
  - Both Gfarm v1 and v2
    - Gfarm is a distributed file system and the storage component of NAREGI
      - Gfarm was an independent product originally
    - Small differences between v1 and v2
    - The current version of NAREGI support V1
      - The future versions support V2
- Now in the testing process
  - Release in this month (March 2010) if success

Software requirement

* Both products are verified on CentOS 5.3

Gfarm v1
NAREGI specific version

Gfarm v2
General release of Gfarm not for NAREGI

Software	Version	Software	Version
OS	CentOS 5.3 or openSUSE 10.3 (Required by NAREGI)	OS	Linux – no dependency on distribution
Apache Ant	1.6.1 or later	Apache Ant	1.6.1 or later
Compiler	GCC C & C++ 3.4.6 or later	Compiler	GCC C & C++ 3.4.6 or later
Java SE SDK	Sun Java SE SDK 1.5 or later (1.6 recommended)	Java SE SDK	Sun Java SE SDK 1.5 or later (1.6 recommended)
JDBC Compliant DB	e.g. PostgreSQL 8.0 or later	JDBC Compliant DB	PostgreSQL 8.0 or later
Perl	5.005 or later	Perl	5.005 or later
xinetd	No dependency on version	xinetd	No dependency on version

What can we do with Gfarm adaptor

- File operations through the adaptor are provided
- Conceal the complicated mechanism behind, e.g. globus, gfarm, postgresql, etc
- SAGA adaptors for NAREGI now covers “job” and “file”
  - Service Discovery will be considered
  - RPC?

```
import saga
src = saga.url(“gfarm2://gfarm.org/tmp/src.dat”)
dst = saga.url(“file:///tmp/dst.dat”)
f = saga.filesystem.file(src)
f.copy(dst)
```

Agenda

- Status report for SAGA-gfarm (v1&v2) file adaptor
  - Questions and feedback
- Other adaptors to be implemented
  - SAGA-iRODS and SAGA-RNS adaptors
  - Possibility for bridging CSAGA-JSAGA using Python

`saga::name_space::directory::get_url()`

- What is expected behavior?
  - SAGA engine adds "/" at the end of URL if there is no "/" at the end for a `url` object
    - E.g. `scheme://example.org/ogf` to `scheme://example.org/ogf/`
  - `get_url()` does so

- Not found in GFD90
  - Is this behavior caused by a specification or implementation dependent?

Inconsistent behavior in get_xxx()

```
For file adaptor: file://localhost/etc/X11/
  get_url()  [file://localhost/etc/X11/]
  get_cwd()  [file://localhost/etc]
  get_name() [X11]

For globus adaptor: gridftp://example.orig/etc/X11/
  get_url()  [gridftp://example.org/etc/X11/]
  get_cwd()  [gridftp://example.org/etc/X11/]
  get_name() [/]
```

- Since we are using the existing implementation as a reference, these inconsistency confused us very much
- At p.204 in GFD90
  - `get_url() = get_cwd() + '/' + get_name()`

saga::name_space::directory::get_cwd()

- CWD (Current Working Directory) is same with a returned value by pwd command
  - So – we suppose that `directory("file://localhost/etc/X11/").get_cwd()` should return `file://localhost/etc/X11` not `file://localhost/etc`, which should be returned by `dirname`
- As far as we can see behavior of `default_file` adaptor, it is assumed `dirname` command in UNIX
- If so – it is more natural to have an API `get_dirname()`

- At p.206 in GFD90
  - Outputs: cwd -- current working directory
- At p.204 in GFD90
  - `get_url() = get_cwd() + '/' + get_name()`

In case of the root directory

- If directory is instantiated with the root directory such as "`scheme://example.org/`", what strings should be returned?
  - Is following example correct?
    - "`scheme://example.org/`" for `get_cwd()`
    - "/" for `get_name()`

`saga::url`

- Although many APIs require `saga::url` for their arguments,
  - We are a bit confused which kind of URL is actually required
    - 1) `scheme://example.org/dir/file.ext`
    - 2) `file.ext`
    - 3) Both
- E.g. which kind of URL should be inserted into `std::vector<saga::url>` returned by `saga::name_space::directory::list()`
- The description seems bit unclear in GFD90.

saga::name_space::directory::functions()

- When should we return `IncorrectURL` while invoking directory::functions(), which is given `saga::url`?
- Which are incorrect in these examples of a directory object instantiated with a `url("ssh://example.org/etc/")`
  - 1) If just an entry name is given
    - `d.is_entry(url("hosts"))`
  - 2) If relative or absolute path in same namespace is given
    - `d.is_entry(url("X11/xorg.cof"))`
    - `d.is_entry(url("/boot/vmlinuz.img"))`
  - 3) If hostname is different but scheme is same
    - `d.is_entry(url("ssh://example.com/etc/hosts"))`
  - 4) If scheme and/or hostname is different
    - `d.is_entry(url("gsiftp://example.org/tmp"))`

`saga::name_space::directory::copy()` `saga::name_space::directory::move()`

- What kind of url is assumed for this APIs?
  - 1) Entries that are placed in same directory
  - 2) Entries that exist in same system
  - 3) Entries crossing different system
- Can we do these?
  - See examples in next slides

```
directory dir("file:///home/user/");

// (a) file:///home/user/source.txt => file:///home/user/target.txt
dir.copy(“source.txt”, “target.txt”);

// (b) file:///home/user/source.txt => file:///tmp/target.txt
dir.copy(“source.txt”, “/tmp/target.txt”);

// (c) file:///tmp/source.txt => file:///home/user/target.txt
dir.copy(“/tmp/source.txt”, “target.txt”);

// (d) file:///etc/bash.bashrc => file:///tmp/bash.bashrc
dir.copy(“/etc/bash.bashrc”, “/tmp”);

// (e) file:///home/user/source.txt
//     => gridftp://example.org/home/user/source.txt
dir.copy("source.txt", "gridftp://example.org/home/user/");

// (f) gridftp://example.org/home/user/source.txt
//     => file:///home/user/source.txt
dir.copy("gridftp://example.org/home/user/source.txt", "./");

// (g) gridftp://example.org/home/user/source.txt
//     => ssh://example.com/home/user/source.txt
dir.copy(“gridftp://example.org/home/user/source.txt”,
         "ssh://example.com/home/user/source.txt");
```

saga::filesystem::move(target)

- We have to move a file while file is opening.
- It should be expressed where offset sets on a file after and before invoking `move()`
- See right for examples
  - Which one is appropriate behavior (a) or (b)?

```
std::string data1 = "0123456789";
saga::const_buffer buffer1(data1.c_str(), data1.length());

std::string data2 = "abcdefghij";
saga::const_buffer buffer2(data2.c_str(), data2.length());

saga::filesystem::file f("file:///home/user/source.txt",
                          saga::filesystem::Create);
// 0....5....10...15...20...25...
//
// ^(offset == 0)

f.write(buffer1, 10);  // (1) write to source.txt at 0
// 0....5....10...15...20...25...
// 0123456789
//           ^(offset == 10)

f.write(buffer1, 10);  // (2) write to source.txt at 10
// 0....5....10...15...20...25...
// 01234567890123456789
//                     ^(offset == 20)

f.move(“target.txt”);  // rename
// offset == (a) 0? (b) 20?

f.write(buffer2, 10);  // (3) write to target.txt
// 0....5....10...15...20...25...
// abcdefghij01234567890          <= (a) at  0 ?
//           ^(offset == 10)
// 01234567890123456789abcdefghij <= (b) at 20 ?
//                               ^(offset == 30)
f.close();
```

`saga::name_space::directory::copy()` `saga::name_space::directory::move()`

- Should we preserve (like a “`cp -p`”) file attributes, e.g. timestamp, ownership, etc

`saga::name_space::entry::remove()` `saga::name_space::entry::close()`

- What do we want to clarify here?
  - At p.214 in GFD90
    - `remove()`: if the instance was not closed before, this call performs a `close()` on the instance, and all notes to `close()` apply.
    - `close()`: any subsequent method call on the object MUST raise an '`IncorrectState`' exception (apart from DESTRUCTOR and `close()`). `close()` can be called multiple times, with no size effects.
  - Does `remove()` raise '`IncorrectState`' after `close()`?
  - Is `remove()` able to called multiple times, same with `close()`?

saga::name_space::directory::open()
saga::name_space::directory::open_dir()

- If we create a template code with saga-cpp-1.3.3/adaptros/generator/generator.pl, both open() and open_dir() are failed with a message below:
  - SAGA(NotImplemented): namespace_dir_cpi.hpp(85): call_wrapper: sync namespace_dir_cpi::open is not implemented

`permissions_allow() & permissons_deny()`

- Question:
  - Can we change a permission to owner and group like `chmod` command in POSIX?

Others

- More questions:
  - Can we obtain timestamp attributes for files and directories?
  - Can we invoke `flush` and `sync` from application side?
  - Can we obtain adaptor-specific exception?
    - For example, if we have exception in `saga::filesystem::file` with default_file adaptor, can we pick up `saga::error` and its messages only for `default_file` adaptor somehow.

Agenda

- Status report for SAGA-gfarm (v1&v2) file adaptor
  - Questions and feedback
- Other adaptors to be implemented
  - SAGA-iRODS and SAGA-RNS adaptors
  - Possibility for bridging CSAGA-JSAGA using Python

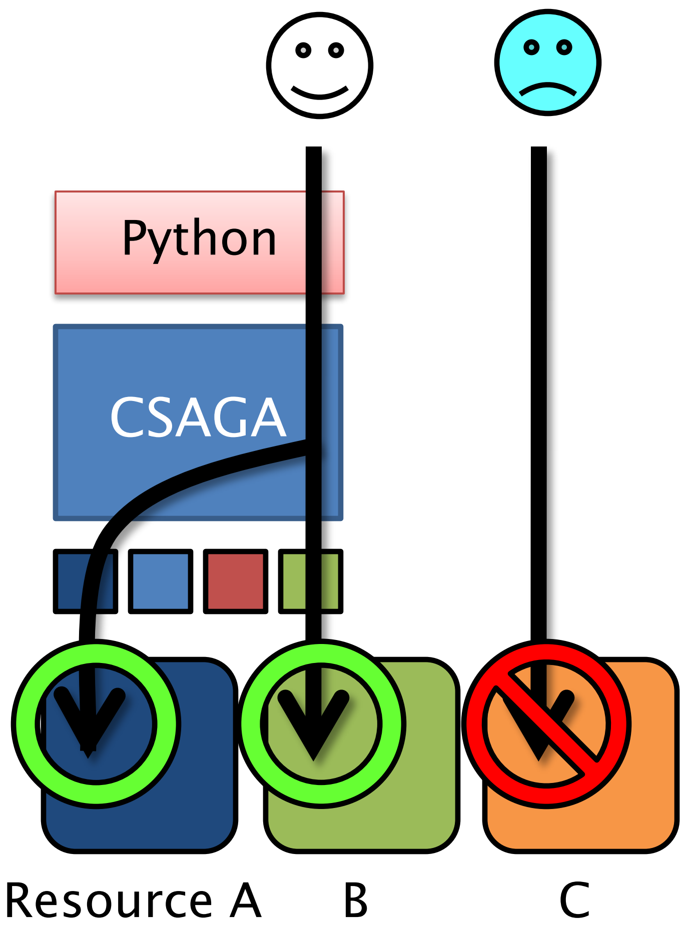
Upcoming adaptors

- SAGA-iRODS
  - ~50% completed
    - Almost done for file & directory functions in saga::filesystem and saga::name_space
    - Not yet finished to manipulate meta data in saga::replica
- SAGA-RNS
  - Just being started
  - RNS v1.1 implemented in Java
    - Natural to implement an adaptor in Java
      - Or invoke Java in C++

```
saga::url
u("irods://localhost/zone/hoge.txt");

// open a name file
saga::filesystem::file f(u);

// seek 200 bytes
f.seek(200, saga::filesystem::Current);

// read 100 byte
saga::mutable_buffer b;
f.read(b, 100);

// print the data
std::cout << b.get_data() << std::endl;
```

How can we make a decision to develop adaptors either for CSAGA or JSAGA ?

"SAGA C++" hereafter referred to as the "CSAGA"

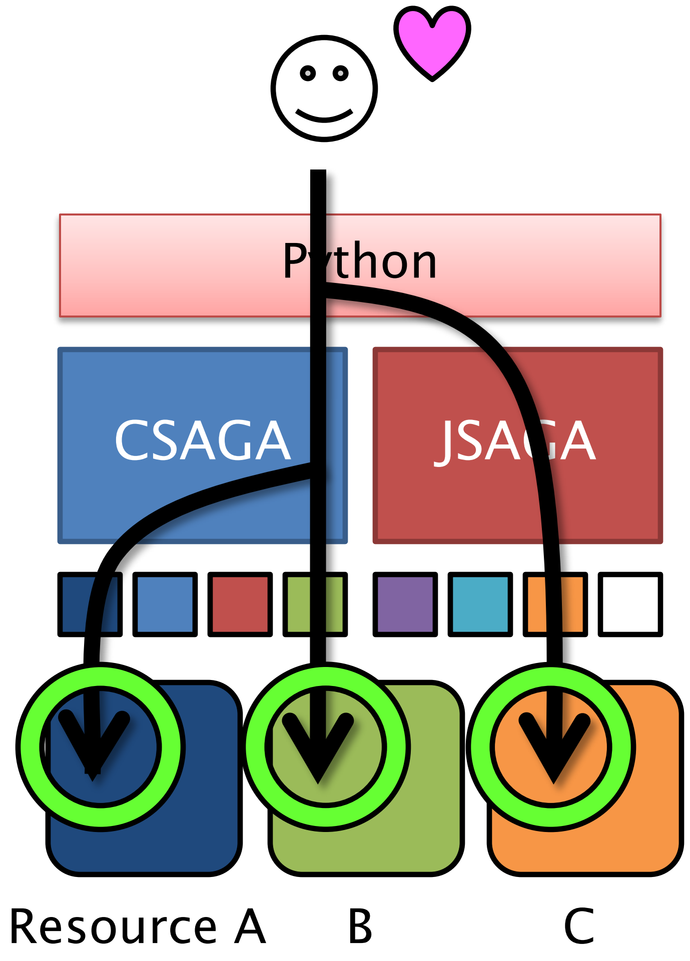


- If we face these situations, e.g.
  - Adaptor C is not yet implemented
  - Adaptor C is implemented both for CSAGA and JSAGA
  - Middleware C is Java based software – easier to implement adaptor in Java

Middleware adaptors

Computing resources composed of different middleware

A possibility to work on both implementations



- If we can work on both CSAGA and JSAGA
  - We don't need to develop another for adaptor C in CSAGA (at least while working on Python)
- Many adaptors we want have been found in JSAGA, e.g. gLite-WMS, cream, NAREGI, SRM, LFN, iRODS, SRB
  - The opposite is true
- A portability of JSAGA makes end-users happy
  - Particularly work on their own machines
    - No middleware installation
    - *Setup is always more difficult*
- But…

Local file listing

```
import org

url = org.ogf.saga.url.URLFactory.createURL("file:///")
dir = org.ogf.saga.file.FileFactory.createDirectory(url)
list = dir.list()
for i in range(0, list.size()):
    print list.get(i)
```

```
$ jython jsaga_local_file_list.py
selinux/
srv/
boot/
cdrom/
...snip
```

```
import saga

url = saga.url("file:///")
dir = saga.filesystem.directory(url)
for f in dir.list():
    print(f.get_string())
```

```
$ python csaga_local_file_list.py
boot
etc
proc
sys
...snip
```

Local job submission

```
import org

url = org.ogf.saga.url.URLFactory.createURL("local://localhost")
js = org.ogf.saga.job.JobFactory.createJobService(url)
jd = org.ogf.saga.job.JobFactory.createJobDescription()
jd.setAttribute("Executable", "/bin/date")
jd.setAttribute("Interactive", "true")
job = js.createJob(jd)
job.run()
print(job.getState())
o = job.getStdout()
print(o.read())
```

Jython

```
$ jython jsaga_local_job_submission.py
DONE
84 # decimal number of a ‘T’ as Thursday
```

```
import saga

url = saga.url("fork://localhost")
js = saga.job.service(url)
jd = saga.job.description()
jd.executable = "/bin/date"
jd.interactive = "True"
job = js.create_job(jd)
job.run()
print(job.get_state())
print(job.get_stdout().readlines())
```

python™

```
$ python csaga_local_job_submission.py
Done
[‘Thu Feb 18 12:34:56 CET 2010¥n']
```

CSAGA-JSAGA interface through Python Things to do/what's going on

- Just an idea, not state in implementation
  - Exploring feasibility of concept for this
  - Just working to know that we can realize this or not ☺
- Many differences even in simple examples
  - E.g. To create the job service: `saga.job.JobFactory.createJobService(url)` in JSAGA and `saga.job.service(url)` in CSAGA
- More technical assessments
  - CPython or Jython?
  - Java Native Access (JNA) or Compiled Native Interface (CNI), or any candidates?

Summary

- SAGA-Gfarm adaptor both for version 1 & 2 have been developed
  - Now in testing process – release in Mach 2010
- On going works:
  - SAGA-iRODS: ~50% of implementation
  - SAGA-RNS: Just making a decision to start
    - Considering that which implementation is more beneficial to our communities
  - CSAGA-JSAGA: Just for confirmation we can do this or not
- (Near) Future works:
  - Service Discovery for NAREGI/CIM

Acknowledgement

- Sylvain Reynaud (CC-IN2P3)
  - Many valuable suggestion and support to Jython-JSAGA bridge

- SAGA developer team
  - Day-to-day efforts toward more matured distributed computing environment with SAGA

Many thanks for your attentions!

Contact

Go Iwai	go.iwai@kek.jp
RENKEI subgroups #4 (KEK)	renkei@ml.post.kek.jp