saga

A Simple API for Grid Applications

SAGA: Tutorial

Ole Weidner and Shantenu Jha

saga

A Simple API for Grid Applications

Getting Started

- Three component tutorial:
  - Introduction to the API via “grid-shell” and cl-utils and (partial) programmers manual (Ole)
  - SAGA to build frameworks and applications (Jha)
  - Mess around with code, try programming manual
- Aim is:
  - Give you a feel for actual SAGA API..
  - SAGA as way to build distributed application

saga

A Simple API for Grid Applications

Getting Started

- http://saga.cct.lsu.edu is your friend!
- Can download from sourceforge.. .recommend SVN
  - https://svn.cct.lsu.edu/repos/saga
  - we'll take a very quick tour of the svn
- Note also saga-projects:
  - https://svn.cct.lsu.edu/repos/saga-projects
  - frameworks/applications (progress)

saga

A Simple API for Grid Applications

api example: saga-file copy

[tools/clutils/file/copy_impl.cpp]

```
saga::filesystem::file file (saga::url(sourceURL), saga::filesystem::Read);
file.copy(saga::url(targetURL));
```

saga

A Simple API for Grid Applications

api example: saga-file list dir

[tools/clutils/file/list_dir_impl.cpp]

```
saga::filesystem::directory dir(directoryURL);
std::vector<saga::url> entries = dir.list();

for ( std::size_t i = 0; i < entries.size (); ++i )
{
  std::cout <<  "   " << entries[i].get_string ();

  if ( dir.is_dir (entries[i]) )
    std::cout << "/";

  if ( dir.is_link (entries[i]) )
  {
    std::cout <<  " -> ";
    std::cout <<  dir.read_link (entries[i]).get_string ();
  }

  std::cout << std::endl;
}
```

saga

A Simple API for Grid Applications

api example: saga-file cat

[tools/clutils/file/cat_impl.cpp]

```
saga::filesystem::file f (saga::url(fileURL), saga::filesystem::Read);

while ( true )
{
  saga::size_t const n = 1024*64;
  saga::uint8_t data[n+1];

  for ( unsigned int i = 0; i <= n; ++i ) { data[i] = '\0'; }

  // read a chunk into the buffer
  if ( f.read (saga::buffer (data, n), n) )
    std::cout << data;
  else
  {
    break;
  }
}
```

saga

A Simple API for Grid Applications

api example: saga-file remove

[tools/clutils/file/remove_impl.cpp]

```
saga::filesystem::file file (saga::url(fileURL), saga::filesystem::ReadWrite);
file.remove();
```

saga

A Simple API for Grid Applications

api example: saga-advert

[tools/clutils/advert/*.cpp]

```
saga::url ad(advertURL);
advert::entry e(ad, saga::advert::Create);
```

```
saga::url ad(advertURL);
advert::entry e(ad);

std::vector<std::string> keys(e.list_attributes());
if (!keys.empty()) {
  std::for_each(keys.begin(), keys.end(), print_attributes(e));
}
```

```
saga::url ad(advertURL);
advert::entry e (ad, saga::advert::ReadWrite);

e.set_attribute(key, val);
```

saga

A Simple API for Grid Applications

api example: saga-replica

[tools/clutils/replica/*.cpp]

```
saga::url lfn(lfnURL);
replica::logical_file logfile (lfn, saga::replica::Write);

logfile.add_location(pfnURL);
```

```
saga::url lfn(lfnURL);
replica::logical_file logfile (lfn);

std::vector<saga::url> locations(logfile.list_locations());
std::vector<saga::url>::const_iterator it;
if (!locations.empty()) {
  for(it = locations.begin(); it != locations.end(); ++it)
 {
    std::cout << "  " << (*it) << std::endl;
 }
}
```

saga

A Simple API for Grid Applications

api example: saga-job

[tools/clutils/job/job_submit_impl.cpp]

```
saga::url js_url(service_url);

saga::job::description jd;
jd.set_attribute (saga::job::attributes::description_executable, exe);
jd.set_attribute (saga::job::attributes::description_interactive,
                  saga::attributes::common_true);
jd.set_vector_attribute (saga::job::attributes::description_arguments,
                         argvec);

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

// create io streams for job io
saga::job::ostream in  = job.get_stdin  ();
saga::job::istream out = job.get_stdout ();
saga::job::istream err = job.get_stderr ();

job.run ();

...
```

saga

A Simple API for Grid Applications

api example: saga-job (I/O)

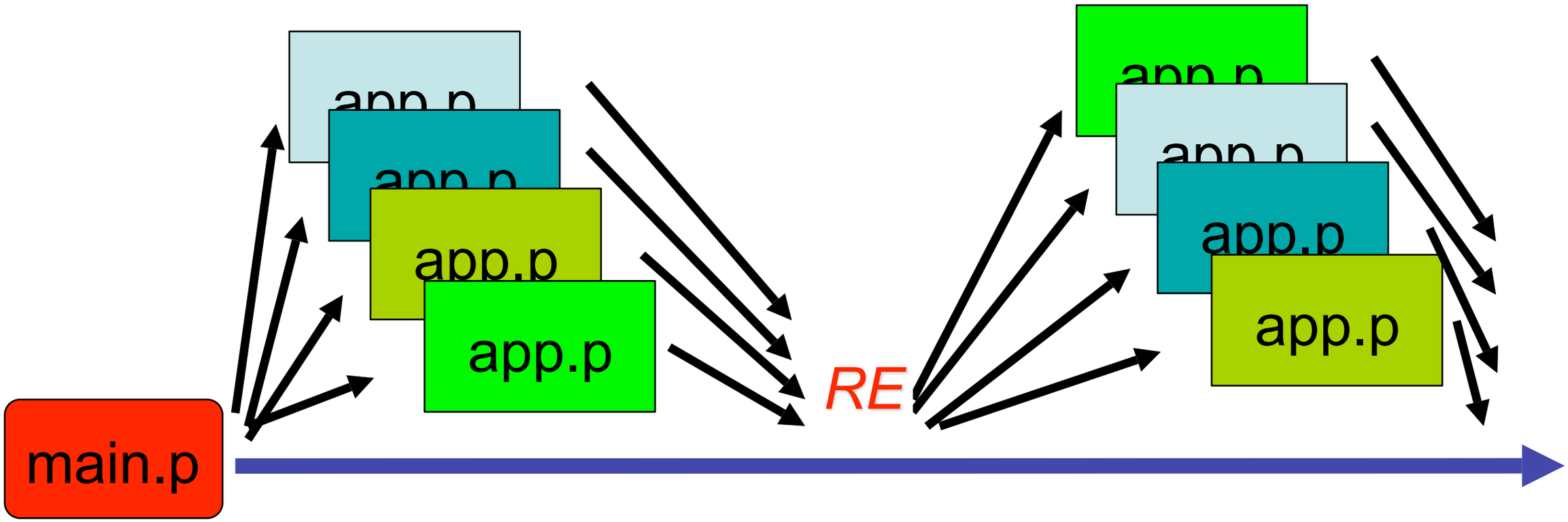
[tools/clutils/job/job_submit_impl.cpp]

```
...

while ( true )
{
  char buffer[1024*64];

  // get stdout
  out.read (buffer, sizeof (buffer));
  if ( out.gcount () > 0 )
    std::cout <<  std::string (buffer, out.gcount ()) << std::flush;

// get stderr
err.read (buffer, sizeof (buffer));
if ( err.gcount () > 0 )
  std::cerr <<  std::string (buffer, err.gcount ()) << std::flush;

if ( out.fail () || err.fail () )
  break;
```

saga

A Simple API for Grid Applications

Replica Exchange



A. “main.py” is the application manager implemented with SAGA and “app.py” is any application program that is launched and monitored by “main.py”
B. A different color for each “app.py” represents a different replica with the assigned id, and black arrows represents inputs and outputs
C. In this example, “app.py” generates N random numbers and stores them into its output file
D. Based on outputs (i.e., random numbers) replica exchange is attempted by “main.py”

saga

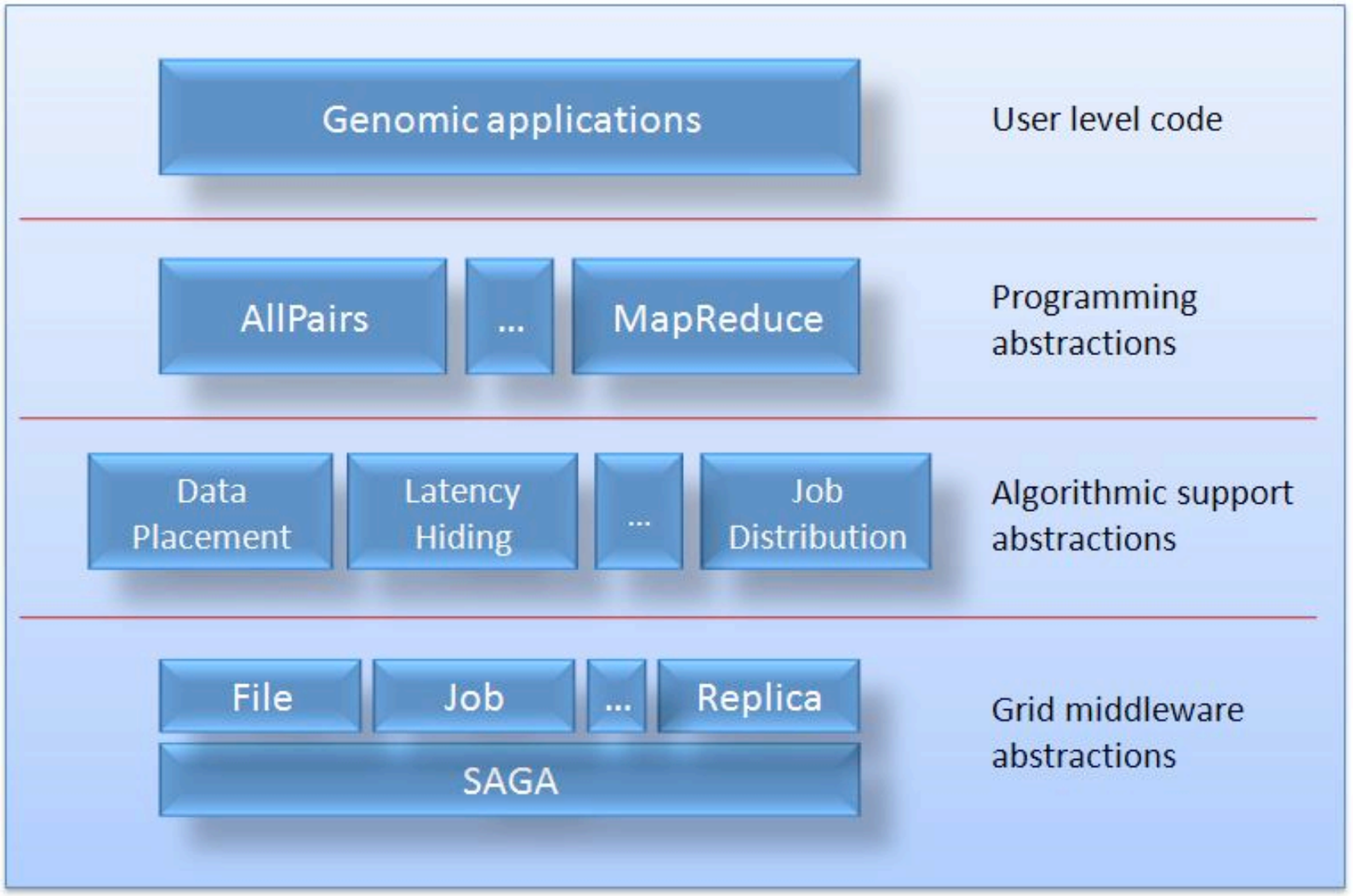
A Simple API for Grid Applications

Replica Exchange

- Scientific problems such as *protein-folding*
- Replica job submission, monitoring, information gathering and exchange for the next replica job submission
- An important distributed application *pattern:*
  - master-worker
  - pair-wise exchange
    - loosely coupled ensemble of tighly-coupled

saga
A Simple API for Grid Applications

SAGA MapReduce ..

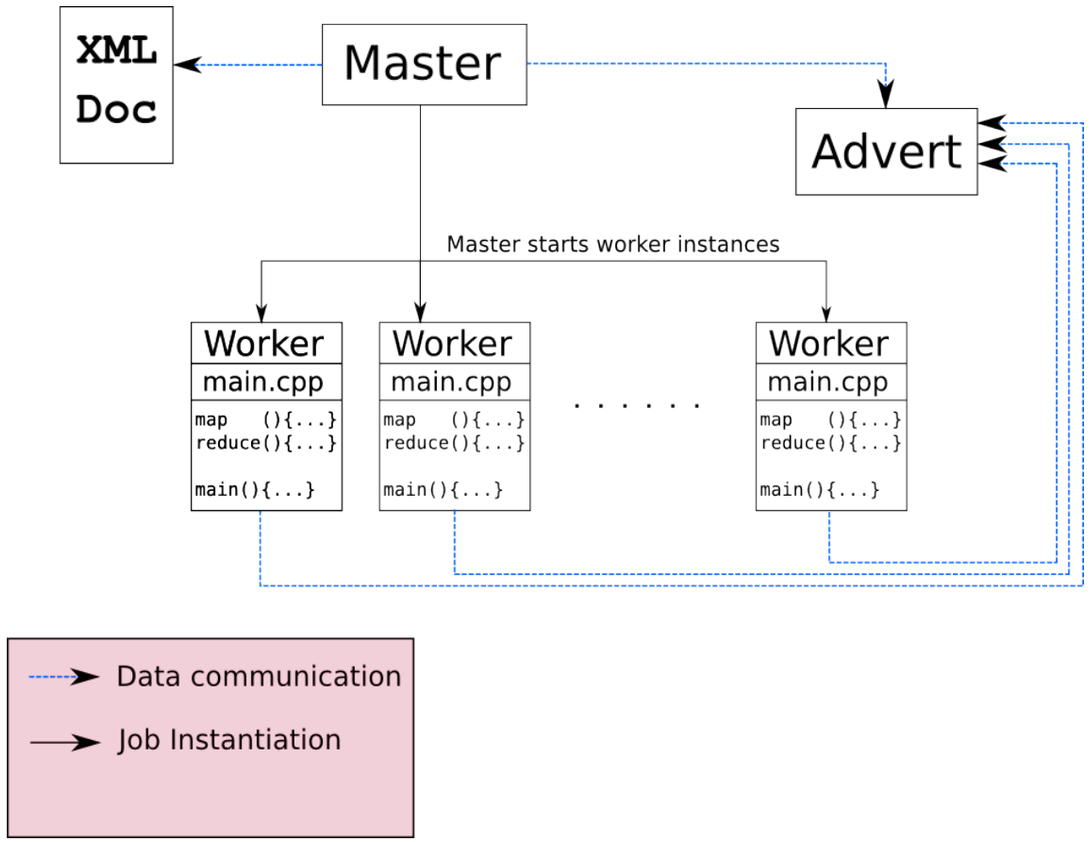


- MapReduce: A data-parallel programming model
- Still work in progresss.. we are adding support for active data (bitdew?).. for data-intensive

saga

A Simple API for Grid Applications

MapReduce: The SAGA Formulation



- Google' MapReduce is dependent on their File System
- We create the MapReduce PM using MW patterns implemented using SAGA; can be used independent of the underlying infrastructure! And is general (you add the resources, you orchestrate)

saga

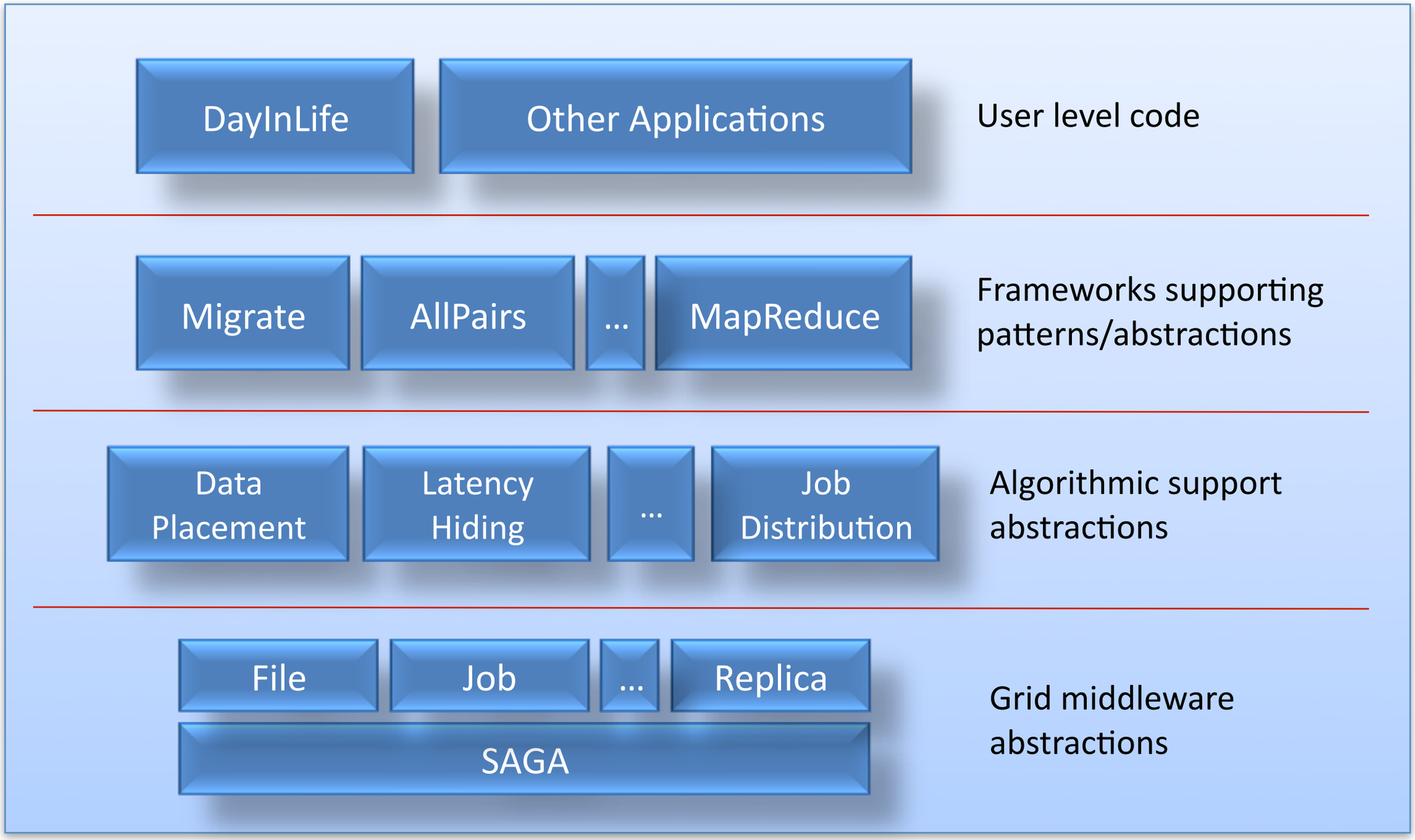
A Simple API for Grid Applications

A Day In the Life of a Distributed Application

- (Self-)migration of an application is a common use case in distributed computing
  - Create a checkpoint
  - Find a new resource
  - Migrate to the new resource
  - Terminate local instance
  - Remote instance restarts from checkpoint
- DayInLife is an example of this scenario encapsulated in a framwork
  - Free application itself from daunting boilerplate code
  - Use framework for file replication, locating output/input files, provide logging support, etc.
- Application itself is:
  - Set expected names for input/output files
  - Read input (file itself is provided by framework)
  - Application specific code: `counter = counter + 1;`
  - Write output (file itself is provided by framework)

saga
A Simple API for Grid Applications

A Day In the Life of a Distributed Application



saga

A Simple API for Grid Applications

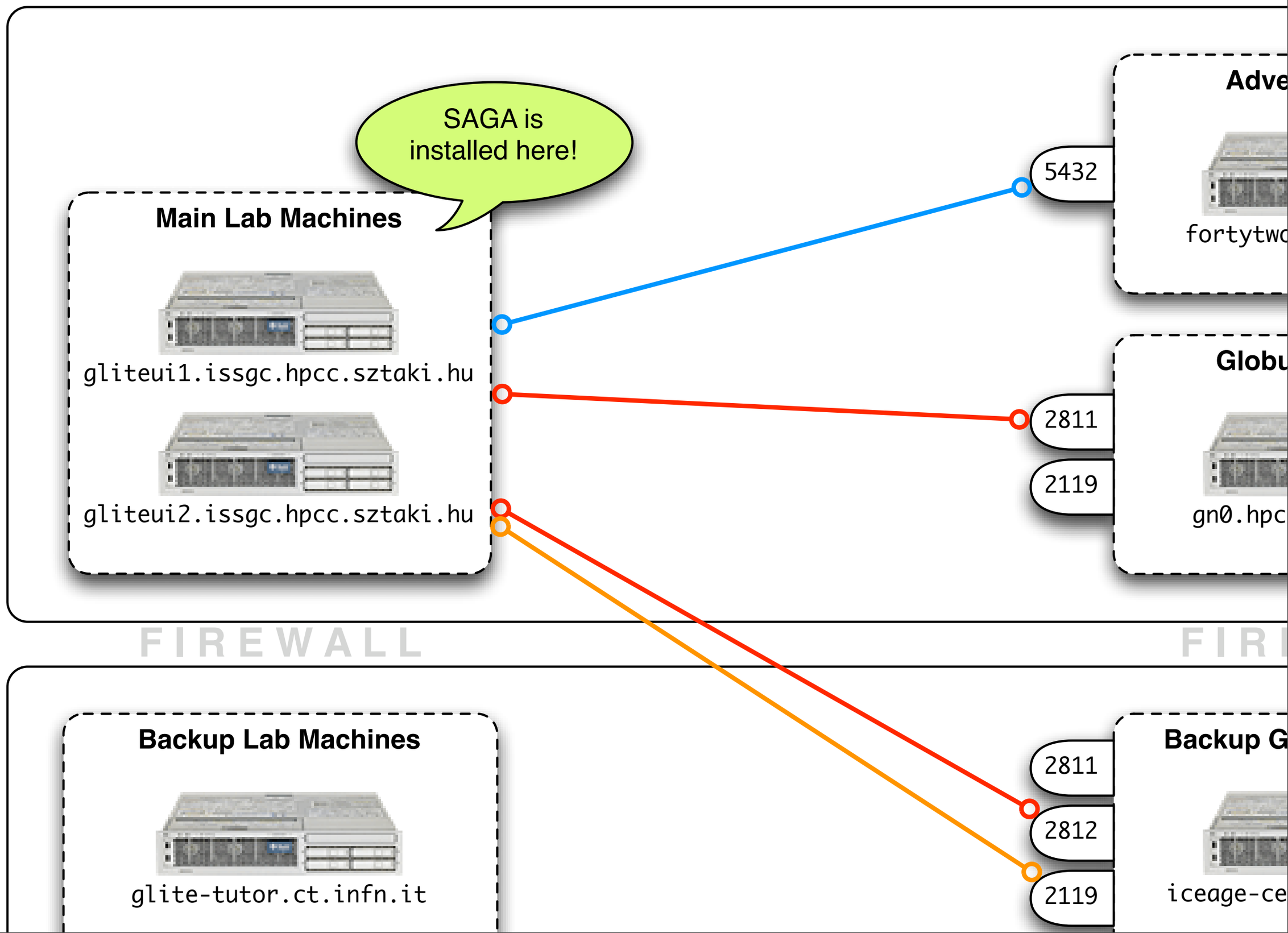
Some Further Excercises

- Take dayinlife example.. run on your own machine(s).. use advert service and python bindings to show machines used on google maps!
- Think of some programming patterns.. (or ask me for some).. code ‘em up using SAGA a la replica exchange (master-worker; scatter-gather)
- Write your own (application) coordinator..
- ....

saga

A Simple API for Grid Applications

Testbed infrastructure



Now repeat

- Look at file: xyz.txt located fadsfaljfk;l