



# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks



- Why is application level scheduling important?
  - Different applications may have different runtime characteristics - system level scheduling usually does not take this into account
  - New scheduling approaches and algorithms can easily be tested without having them deployed system-wide
  - Multiple distributed resources without a cross-system meta-scheduler can be utilized effectively





# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks

## KEY DESIGN OBJECTIVES

---

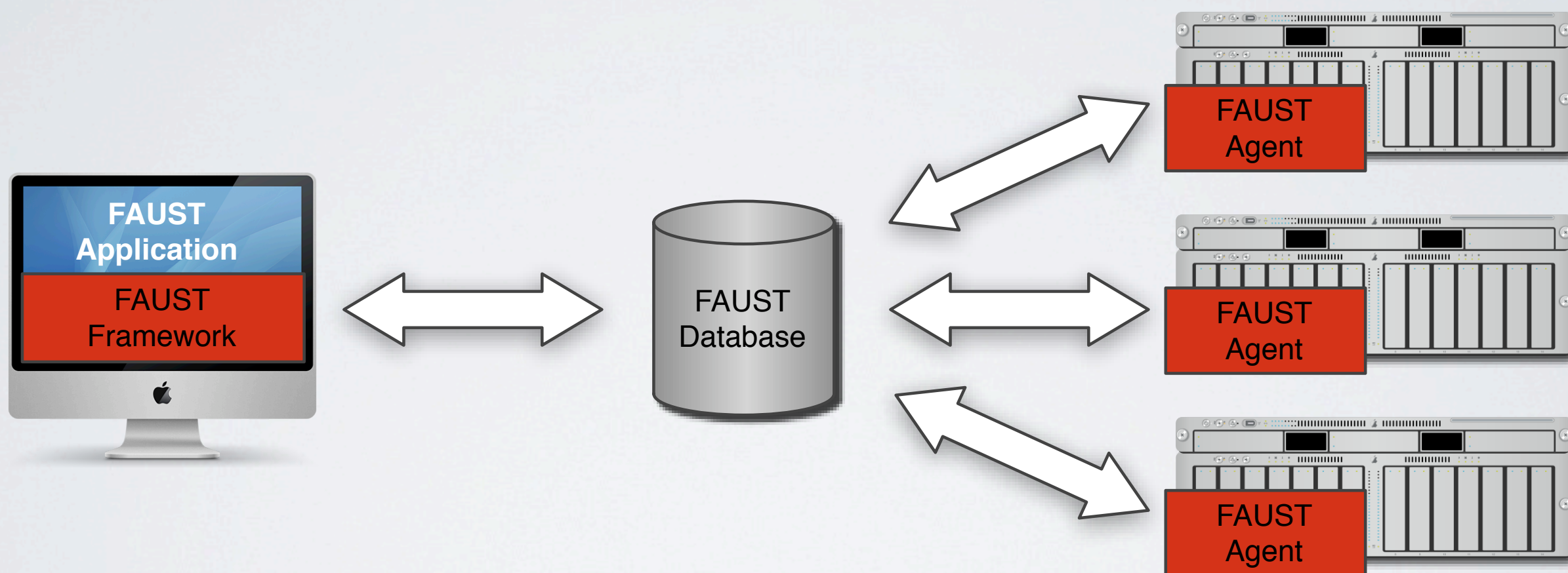
- Create a portable programming *framework* with a SAGA-like interface that can be used to schedule, manage and monitor jobs programmatically
- Provide a transparent mechanism to collect, interpret and store system performance data
- Expose low-level system and scheduling details *only* if desired / requested by the framework user
- Provide a plug-in mechanism that allows the replacement of scheduling algorithms - no hard-wiring



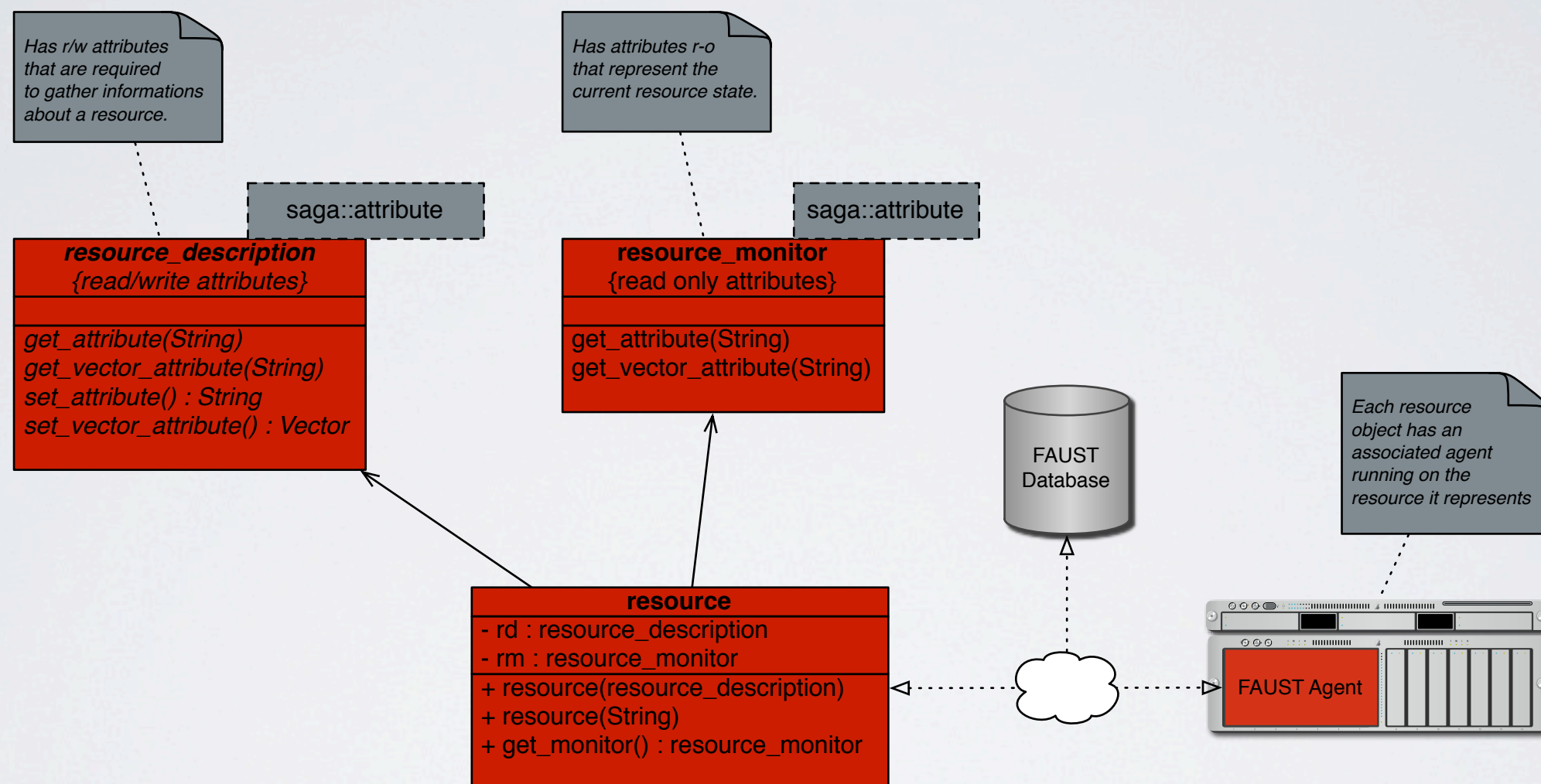
# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks

## FRAMEWORK ARCHITECTURE









### // FAUST API EXAMPLE 1

// Creating a resource from a resource description and  
// instantiate a resource monitor instance.

```
faust::resource_description queenbeeRD;
```

```
queenbeeRD.set_attribute("identifier", "queenbee.loni.org");  
queenbeeRD.set_attribute("faust_agent_submit_url", "gram://qb1.loni.org/jobmanager-fork");  
queenbeeRD.set_attribute("faust_agent_binary_path", "/work/oweidner/FAUST/agent/faust_agent");  
queenbeeRD.set_attribute("saga_root_path", "/work/oweidner/megajobs");
```

```
dir_ids.push_back("my_work_dir");  
dir_path.push_back("/home/");  
dir_dev_space_total_cmd.push_back ("echo \"scale=0; `df . | awk '/\\|/ {print $2}` * 512/1024/1024\\\" | bc");  
dir_dev_space_used_cmd.push_back  ("echo \"scale=0; `df . | awk '/\\|/ {print $3}` * 512/1024/1024\\\" | bc");
```

```
queenbeeRD.set_vector_attribute("dir_id", dir_ids);  
queenbeeRD.set_vector_attribute("dir_path", dir_path);  
queenbeeRD.set_vector_attribute("dir_dev_space_total_cmd", dir_dev_space_total_cmd);  
queenbeeRD.set_vector_attribute("dir_dev_space_used_cmd", dir_dev_space_used_cmd);
```

```
faust::resource queenbee (queenbeeRD, false);  
faust::resource_monitor queenbeeMON = queenbee.get_monitor();
```

```
std::vector<std::string> attribs = queenbeeMON.list_attributes();  
std::vector<std::string>::const_iterator it;  
for(it = attribs.begin(); it != attribs.end(); ++it)  
{  
    std::cout << (*it) << ": " << queenbeeMON.get_attribute(*it) << std::endl;  
}
```



```
// FAUST API EXAMPLE 2
// Reconnecting to a persistent resource agent and
// retrieving the original resource description.

// SCOPE 1
{
    faust::resource queenbee (queenbeeRD, true);

    // instance gets destroyed here but the agent lives
    // on since the persistency flag is set!
}

// SCOPE 2
{
    // use the resource "identifier" string to reconnect
    // to an existing resource agent
    faust::resource queenbee ("queenbee.loni.org");

    faust::resource_description queenbeeRD = queenbee.get_description();
    std::vector<std::string> attribs = queenbee.list_attributes();
    std::vector<std::string>::const_iterator it;
    for(it = attribs.begin(); it != attribs.end(); ++it)
    {
        std::cout << (*it) << ": " << queenbeeRD.get_attribute(*it) << std::endl;
    }
}
```





```
10:31:58 FAUST faust::resource (localhost) [INFO] Checking faust::resource_description for completeness. SUCCESS
10:31:59 FAUST faust::resource (localhost) [INFO] Creating advert endpoint 'advert://macpro01.cct.lsu.edu:5432//FAUST/
10:31:59 FAUST faust::resource (localhost) [INFO] Populating advert endpoint 'advert://macpro01.cct.lsu.edu:5432//FAUST/
10:31:59 FAUST faust::resource (localhost) [INFO] Trying to launch agent 042b4002-4592-4c9c-a526-d1c2c6e8fcb4 via
10:32:1 FAUST faust::resource (localhost) [INFO] Sending command 'PING' to faust_agent instance. SUCCESS Waiting for
10:32:1 FAUST faust::resource (localhost) [INFO] Starting service thread. SUCCESS
10:32:1 FAUST faust::resource (localhost) [INFO] Re-connecting to advert endpoint advert://macpro01.cct.lsu.edu:5432//

10:32:2 FAUST faust::resource (localhost) [INFO] Sending command 'PING' to faust_agent instance. SUCCESS Waiting for
10:32:2 FAUST faust::resource (localhost) [INFO] Retrieving resource description for localhost. SUCCESS
10:32:2 FAUST faust::resource (localhost) [INFO] Starting service thread. SUCCESS

10:32:2 FAUST faust::resource (localhost) [INFO] Setting persistency for endpoint 'localhost' to false. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Sending command 'TERMINATE' to faust_agent instance. SUCCESS Waiting
10:32:3 FAUST faust::resource (localhost) [INFO] Removing advert endpoint ''. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Shutting down service thread. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Shutting down service thread. SUCCESS

10:32:3 FAUST faust::resource (localhost) [INFO] Checking faust::resource_description for completeness. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Creating advert endpoint 'advert://macpro01.cct.lsu.edu:5432//FAUST/
RESOURCES/localhost/'. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Populating advert endpoint 'advert://macpro01.cct.lsu.edu:5432//FAUST/
RESOURCES/localhost/'. SUCCESS
10:32:3 FAUST faust::resource (localhost) [INFO] Trying to launch agent dbf7bd51-6192-483c-bf2f-cf5276bca22c via
fork://localhost/. SUCCESS
10:32:5 FAUST faust::resource (localhost) [INFO] Sending command 'PING' to faust_agent instance. SUCCESS Waiting for
acknowledgement SUCCESS
10:32:5 FAUST faust::resource (localhost) [INFO] Starting service thread. SUCCESS
10:32:5 FAUST faust::resource (localhost) [INFO] Re-connecting to advert endpoint advert://macpro01.cct.lsu.edu:5432//
FAUST/RESOURCES/localhost/.
```





- FAUST uses CMake and builds on all UNIX/Linux/MacOS X platforms
- Requirements: Boost 1.35+, CMake 2.6.2+ and SAGA 1.2.1+
- Checkout FAUST source code from:  
<https://svn.cct.lsu.edu/repos/saga-projects/applications/FAUST>
- Build and Install

```
$ export BOOST_ROOT=<path_to_your_boost_installation>
$ cmake -DSAGA_ROOT=<path_to_your_saga_installation> -DCMAKE_PREFIX=<install_dir> .
$ make
$ make install
```



- Framework / Agent interaction is implemented and works pretty stable on local and remote systems.
- Agents are capable of executing arbitrary shell commands on host machines and write the results back to the database.
- Version 0.1 of the resource API is defined, implemented ready to be used.
- Beta testers wanted!





# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks

## HIGH-PRIORITY TODO ITEMS

---

- Compile a list of resource\_description to resource\_monitor attribute mappings and implement them.
- Implement basic job submission functionality so people can start using FAUST in their projects, even if the scheduling component is not in place yet.



# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks

## ONLINE RESOURCES

---

- API documentation:  
<http://macpro01.cct.lsu.edu/~oweidner/FAUST/>
- SVN repository:  
<https://svn.cct.lsu.edu/repos/saga-projects/applications/FAUST>
- My email address: oweidner @ cct.lsu.edu





# FAUST

A Framework for Adaptive Ubiquitous Scalable Tasks

---

# THANKS

Any Questions / Suggestions ?