Being OptimallyLazy: 
The case for integrating SAGA with Condor

Shantenu Jha\textsuperscript{12*}, João Abecasis\textsuperscript{1} 
\textsuperscript{1} Center for Computation and Technology, LSU 
\textsuperscript{2} Department of Computer Science, LSU

*Also with NeSC (Edin) and UC-London
If Computer Scientists are from Mars....

• e-Science is about Applications, ideally new and exciting..
• Applications need to use Infrastructure seamlessly
  – “e-Infrastructure that will enable novel and different research” linked with e-Science
  – Handle current heterogeneity and future developments
• Development and deployment of “novel” applications
  – Complex, dynamic, qualitative and quantitative use
  – Application-Level Interoperability, Scheduling
• Lack of distributed programming abstractions (DPA) for applications is one (of several) important barriers...
A Network Performance Aware Application

- Capable of acquiring application-specific network characteristic data
- Determine ideal migration target across heterogeneous systems without changes in the application code
- Reusable and extensible
Hard-to-predict Runtime Characteristics

Life Beyond DAGMAN OR Homage to the “Known Unknowns”
SAGA: A Quick Tour

• A lack of:
  • Programming interface that provides common distributed functionality with the correct level of abstractions?
  • Ability to hide underlying complexities, varying semantics, heterogeneities and changes from application program(ers)
• Simple, integrated, stable, uniform, high-level interface
• Simplicity partly arises from scope being restricted (80/20)
• Like MapReduce – leave details of distribution etc. out
• Measure(s) of success:
  • Does SAGA enable quick development of “new” distributed applications?
    • Does it enable greater functionality using less code?
  • Can programming patterns/abstraction (all-pairs, map-reduce) be supported?
  • Will it enable applications across different systems? (ALI)
Copy a File: Globus GASS

```c
int copy_file (char const* source, char const* target) {
    globus_url_t source_url;
    globus_io_handle_t dest_io_handle;
    globus_ftp_client_operationattr_t source_ftp_attr;
    globus_result_t result;
    globus_gass_transfer_requestattr_t source_gass_attr;
    globus_gass_copy_attr_t source_gass_copy_attr;
    globus_gass_copy_handle_t gass_copy_handle;
    globus_gass_copy_handleattr_t gass_copy_handleattr;
    globus_ftp_client_handleattr_t ftp_handleattr;
    globus_io_attr_t io_attr;
    int output_file = -1;

    if (globus_url_parse (source_URL, &source_url) != GLOBUS_SUCCESS) {  // 1
        printf("can not parse source_URL \"%s\"\n", source_URL);
        return (-1);
    }

    if (source_url.scheme_type != GLOBUS_URL_SCHEME_GSIFTP &
        source_url.scheme_type != GLOBUS_URL_SCHEME_FTP &
        source_url.scheme_type != GLOBUS_URL_SCHEME_HTTP &
        source_url.scheme_type != GLOBUS_URL_SCHEME_HTTPS ) {  // 2
        printf("can not copy from %s - wrong prot\n", source_URL);
        return (-1);
    }

    globus_gass_copy_handleattr_init (&gass_copy_handleattr);
    globus_gass_copy_attr_init (&source_gass_copy_attr);
    globus_ftp_client_handleattr_init (&ftp_handleattr);
    globus_io_fileattr_init (&io_attr);
    globus_gass_copy_attr_set_io (&source_gass_copy_attr, &io_attr);
    globus_gass_copy_handleattr_set_ftp_attr (
        &gass_copy_handleattr, &ftp_handleattr);
    globus_gass_copy_handle_init (&gass_copy_handle, &gass_copy_handleattr);

    if (source_url.scheme_type == GLOBUS_URL_SCHEME_GSIFTP ||
        source_url.scheme_type == GLOBUS_URL_SCHEME_FTP ) {  // 3
        globus_ftp_client_operationattr_init (&source_ftp_attr);
        globus_gass_copy_attr_set_ftp (&source_gass_copy_attr, &source_ftp_attr);
    } else {  // 4
        globus_gass_transfer_requestattr_init (&source_gass_attr, source_url.scheme);
        globus_gass_copy_attr_set_gass(&source_gass_copy_attr, &source_gass_attr);
    }

    output_file = globus_libc_open ((char*) target,
        O_WRONLY | O_TRUNC | O_CREAT,
        S_IRUSR  | S_IWUSR | S_IRGRP |
        S_IWGRP);
    if ( output_file == -1 ) {  // 5
        printf("could not open the file \"%s\"\n", target);
        return (-1);
    }

    /* convert stdout to be a globus_io_handle */
    if ( globus_io_file_posix_convert (output_file, 0, &dest_io_handle)
        != GLOBUS_SUCCESS) {  // 6
        printf("Error converting the file handle\n");
        return (-1);
    }

    result = globus_gass_copy_register_url_to_handle {
        &gass_copy_handle, (char*)source_URL,
        &source_gass_copy_attr, &dest_io_handle, my_callback, NULL};
    if ( result != GLOBUS_SUCCESS ) {  // 7
        printf("error: %s", globus_object_printable_to_string (globus_error_get (result)));
        return (-1);
    }

    globus_url_destroy (&source_url);
    return (0);
}
```
SAGA Example: Copy a File
High-level, uniform

```cpp
#include <string>
#include <saga/saga.hpp>

void copy_file(std::string source_url, std::string target_url) {
    try {
        saga::file f(source_url);
        f.copy(target_url);
    }
    catch (saga::exception const &e) {
        std::cerr << e.what() << std::endl;
    }
}
```
// Submitting a simple job and wait for completion

saga::job_description jobdef;
jobdef.set_attribute("Executable", "job.sh");

saga::job_service js;
saga::job job = js.create_job("remote.host.net", jobdef);
job.run();

while( job.get_state() == saga::job::Running )
{
    std::cout << "Job running with ID: "
    << job.get_attribute("JobID") << std::endl;
    sleep(1);
}

}
SAGA: Class Diagram

In the works: CPR, Information Services, Service Discovery, Messaging....
SAGA API: Towards a Standard
Standards help Interoperability

• The need for a standard programming interface
  – “Go it alone” versus “Community” model
  – Reinventing the wheel again, yet again, and again
  – MPI as a useful analogy of community standard
  – OGF the natural choice; establish SAGA-RG

• Open Grid Forum: Design derived from 23 Use Cases
  – Different projects, applications and functionality
    • Biological, Coastal-modelling, visualization ..
  – Functional Areas: Job Mgmt, Data Management (Files, Logical Files...), Streams & ...
  – Non-functional Areas: Asynchronous, QoS, Bulk

• Interface is language independent, object-oriented and each sub-system is independent, SIDL (extensible)
The SAGA Landscape

- SAGA Core (GFD-90)
  - extensions
  - extensions

- Java binding
- C++ binding
- C binding
- Python ...

- Implementation A
- Implementation B
- Implementation C
- Implementation D

- adaptors
- adaptors
- adaptors
SAGA
C++ (LSU)
SAGA: Job Submission
Role of Adaptors (middleware binding)
Replica Exchange Algorithm

- Task Level Parallelism
  - Embarrassingly distributable!
  - Loosely coupled
- Create replicas of initial configuration
- Spawn 'N' replicas over different machine
- Run for time $t$; Attempt configuration swap
- Run for further time $t$; Repeat till finish

Exchange attempts 300K
RE: Programming Requirements

- RE can be implemented using following “primitives”
  - Read job description
    - # of processors, replicas, determine resources
  - Submit jobs
    - Move files, job launch
  - Checkpoint and re-launch simulations
    - Exchange, RPC (to swap or not)
- Implement above using “grid primitives” provided by SAGA
  - Separated “distributed” logic from “simulation” logic
    - Independent of underlying code/engine
    - Science kernel is independent of details of distributed resource management
- Need to use resources integrated from Desktop to Supercomputers!!
“Grid” Universe

- All handled in your submit file
- Supports a number of “back end” types:
  - Globus: GT2, GT4
  - NorduGrid
  - UNICORE
  - Condor
  - PBS
  - LSF
  - EC2
  - NQS

http://www.cs.wisc.edu/condor
Condor APIs

• Command line tools
  the *de facto* API

• SOAP
  The API

• GAHP
  An ASCII communication protocol
Application

SAGA

File
Job
Replica
RPC
RPC

CLI

Condor
Application

SAGA

File  Job  Replica  RPC  RPC

GAHP

Condor
class saga::job::service
class saga::job::description

Executable, Arguments, Environment, Input,

Output, Error, FileTransfer
class saga::job::job
class saga::job::job

• run()
• cancel()
• wait()
• suspend()
• resume()
• condor_submit
• condor_rm
• (condor_run)
• condor_hold
• condor_release
#include <saga.hpp>

int main()
{
    using namespace std;
    using namespace saga;
    using namespace saga::job::attributes;

    job::service js("condor://gg201.cct.lsu.edu");

    job::description jd;
    jd.set_attribute(description_executable, "/bin/sleep");
    jd.set_vector_attribute(description_arguments, vector<string>(1, "1000"));

    job::job job = js.create_job(jd);

    job.run();
}
Acknowledgments

• Funding Agencies:
  – US NSF/ La BoR
    • Cybertools projects
  – UK EPSRC:
    • OMII-UK “OMII SAGA” project
    • Theme “Distributed Programming Abstractions”
  – CCT Internal Funds
  – NIH and Google*
  * Keep Clicking on those advertisements!

http://saga.cct.lsu.edu
Acknowledgments: The SAGA Team

Hartmut Kaiser
 Andre Merzky
 Ole Weidner

Thilo Kielmann
 Ceriel Jacobs
 Kees Verstop

+ Many other students