SAGA: An Introductory Lecture

Shantenu Jha, Ole Weidner

http://saga.cct.lsu.edu/

CENTER FOR COMPUTATION & TECHNOLOGY

Outline (1)

e-Science Institute

- Introduction to the Simple API for Grid Applications:
  - Role of adaptors
  - SAGA as an standard
- Understanding Distributed Applications (DA)
  - Differ from HPC or || App?
  - Challenges of DA
  - Rough Taxonomy of Distributed Applications
    i. Distributed Execution Mode Legacy (Implicit)
    ii. Explicit coordination and distribution
    iii. Frameworks: support characteristics or patterns
- Understanding the SAGA Landscape
  - Interface/Specification, C++ Engine, Adaptors
    - Some comments about SAGA as a Standard

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Outline (2)

e-Science Institute

- Using SAGA to develop Distributed Applications
  - SAGA-based DAG Applications (Montage)
  - Frameworks Based:
    - Replica-Exchange
    - Multi-Physics Coupled Simulations,
    - Reservoir Simulations (EnKF)
    - MapReduce
      - Inter-operability across Infrastructure
- SAGA-based Tools and Projects (Advantage of Standards)
  - DESHL, JSAGA, GANGA (gLite)
  - XtreemOS, NAREGI/KEK

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Critical Perspectives

- Distributed CI: Is the whole greater than the sum of the parts? Less?
- Managing data and independent applications across multiple resources is (increasingly) hard
- The number of applications that can utilize multiple distributed application – sequentially, concurrently or asynchronously
- Challenges qualitatively and quantitatively set to get worse:
  - Increasing complexity, heterogeneity and scale

CENTER FOR COMPUTATION & TECHNOLOGY

A take on "What are Grids?"

e-Science Institute

In general Grids are Distributed Systems characterized by:

- Heterogeneous
  - Operating Systems, Libraries, Software stack
  - Middleware, Service versions and Semantics
  - Administrative policies – Access, Usage
- Collective Degrees-of-freedom (DoF):
  - (Can be) Dynamic, depending on relevant time-scales
  - Different levels of control
  - Production Grid Service with high QoS non-trivial
- Even without Heterogeneity and Collective DoF, Distributed Systems are challenging to operate and difficult to develop, deploy and execute applications

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: A Quick Tour

e-Science Institute

- There exists a lack of:
  - Programmatic approaches that provide common distributed functionality at the correct level of abstraction for applications
  - Ability to hide underlying complexity of infrastructure, varying semantics, heterogeneity and changes from the application
- Simple, integrated, stable, uniform and high-level interface
  - Simple: 80:20 restricted scope
  - Integrated: Similar semantics & style across commonly used distributed functional requirements
  - Stable: Standard
  - Uniform: Same interface for different distributed systems
- SAGA: Provides the high-level abstraction that application developers need that will work across different distributed systems
  - Shields details of lower level middle-ware and system issues
  - Enables the details of distribution to be left out

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA Example: File Copy

e-Science Institute

```
#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;
  }
}
```

The interface is simple and the actual function calls remain the same

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA Example: Read File

e-Science Institute

```
01: // Read the first 10 bytes of a file if file size > 10 bytes
02: //
03: saga::file my_file ("griftp://gridhub/~/result.dat);
04:
05: off_t size = my_file.get_size ();
06:
07: if ( size > 10 )
08: {
09:   char buffer[11];
10:   long bufflen;
11:
12:   my_file.read (10, buffer, &bufflen);
13:
14:   if ( bufflen == 10 )
15:   {
16:     std::cout << buffer << std::endl;
17:   }
18: }
```

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA Example: Job Submission

e-Science Institute

```
01: // Submitting a simple job and wait for completion
02: //
03: saga::job_description jobdef;
04: jobdef.set_attribute ("Executable", "job.sh");
05:
06: saga::job_service js;
07: saga::job job = js.create_job ("remote.host.net", jobdef);
08:
09: job.run();
10:
11: while( job.get_state() == saga::job::Running )
12: {
13:   std::cout << "Job running with ID: "
14:             << job.get_attribute("JobID") << std::endl;
15:   sleep(1);
16: }
```

The interface is simple and the actual function calls remain the same

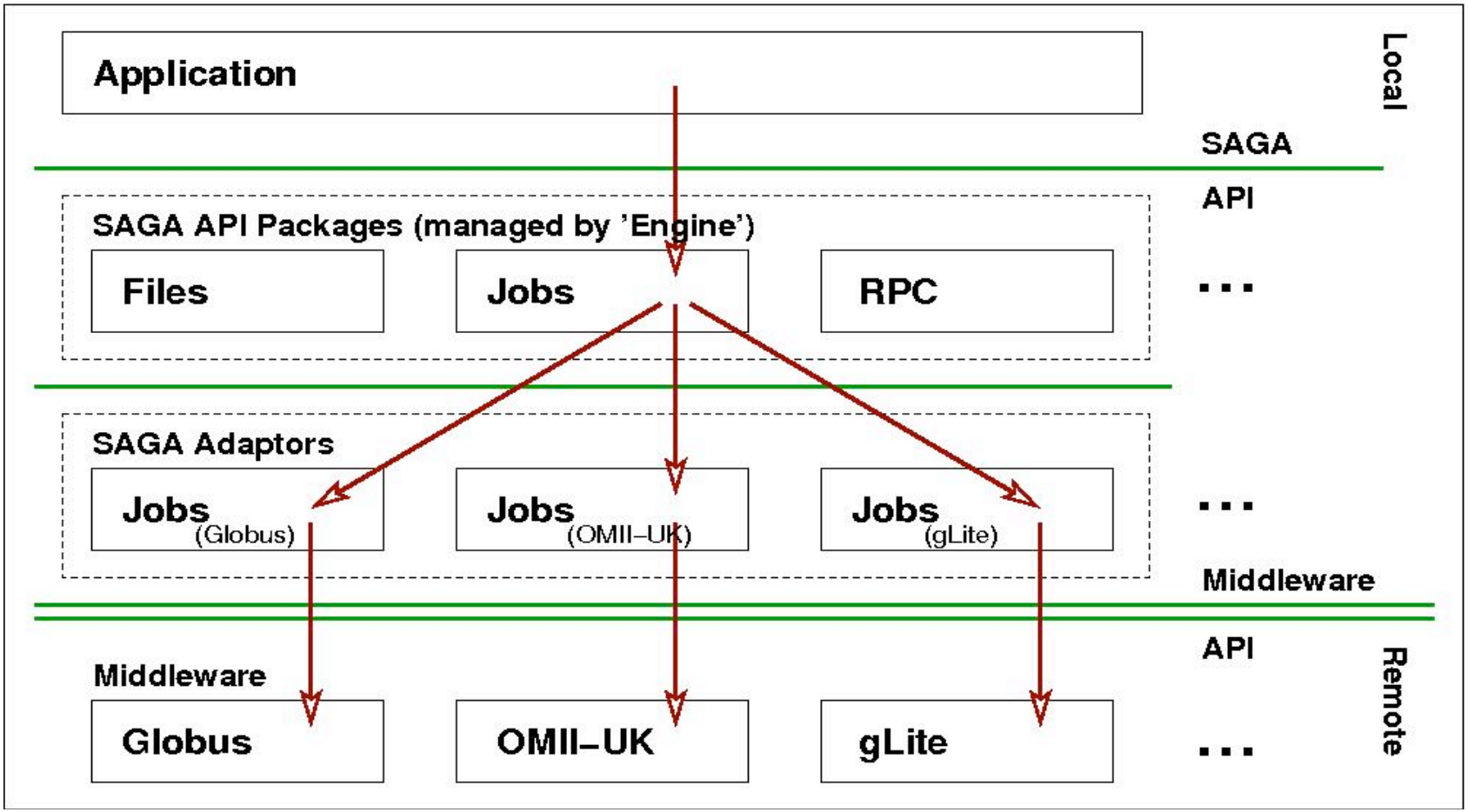
saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: Job Submission

Role of Adaptors (middleware binding)

e-Science Institute

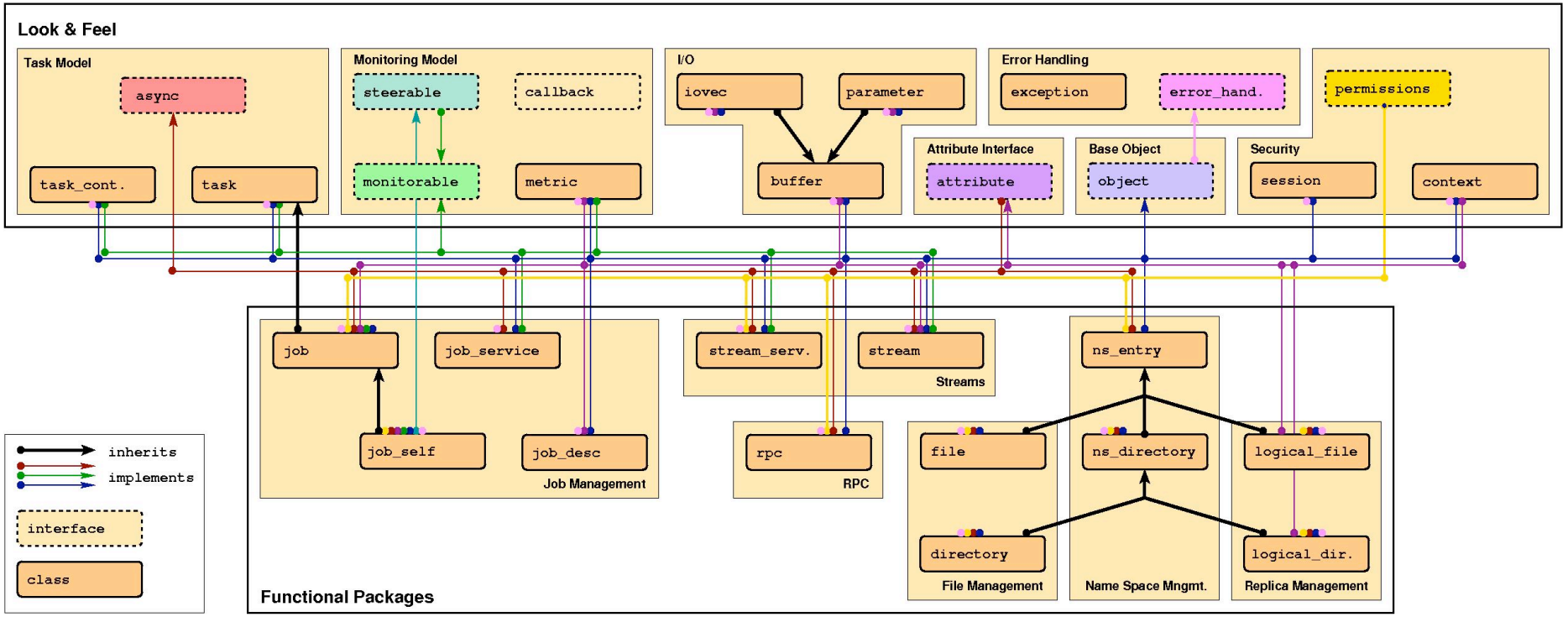


saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: Class Diagram

e-Science Institute



In the works: CPR, Information Services, Messaging, DAI,...

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA API: Towards a Standard

Standards promote Interoperability

e-Science Institute

- The need for standard programming interface
  - Trade-off “Go it alone” versus “Community” model
  - Reinventing the wheel again, yet again, & then again
  - MPI a useful analogy of community standard
    - Vendors (Resource Provider), Software developers, users..
    - Social/historic parallels also important
      - Time to adoption, after specification ....
  - OGF the natural choice (SAGA-RG, SAGA-WG)
    - Spin-off of the Applications Research Group
    - Driven by EU (German/Dutch), UK, US
    - Design derived from 23 Use Cases
      - different projects, applications and functionality
      - biological, coastal modelling, visualization
  - Will discuss the advantage of SAGA as a standard specification

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Developing DA with SAGA: Parallel Programming Analogy

e-Science Institute

- Distributed Development today, || programming pre-MPI
  - Bewildering # of ways of doing the basic stuff (e.g., job submission)
- MPI was a “success” in that it helped many new applications
  - MPI is simple (most things can be done with 6-8 calls)
  - MPI was a standard (stable and portable code!)
- SAGA conception & trajectory similar to MPI
  - SAGA focusses on simplicity (common vs corner case)
  - OGF specification; on path to becoming a standard
- Therefore, SAGA's Measure(s) of success:
  - Does SAGA enable “new” distributed applications?
  - Does it enable effective development of “old” distributed application

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Outline (1)

e-Science Institute

- Critical Perspective on Large-Scale Distributed Applications and Production Cyber-Infrastructure (CI)
- Understanding Production CI
- Understanding Distributed Applications (DA)
  - Differ from HPC or || App?
  - Challenges of DA
  - Rough Taxonomy of Distributed Applications
    i. Distributed Execution Mode Legacy (Implicit)
    ii. Explicit coordination and distribution
    iii. Frameworks: support characteristics or patterns
- Understanding the SAGA Landscape
  - Interface/Specification, C++ Engine, Adaptors

saga

Developing Distributed Applications: Challenges

- Complex Underlying infrastructure characteristics:
  - Dynamical Resource model; Heterogeneous resources
  - Variable Control (or lack thereof)
- Computational Models of Distributed Computing not as well understood as parallel computing
  - Parallel Computing: Focus on performance! For DA?
  - Greater range of DA; no clear taxonomy
- Programming Systems for DA:
  - Incomplete? Customization? Extensibility?
  - What should end-user control? Must control? Should not? What should the system support? Role of tools?
- No universal answers!
  - Inter-play of Application, Infrastructure, Usage Mode

saga

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Interplay: Application-Infrastructure-Usage Mode

Usage Mode	Number of Users
Batch Computing on Individual Resource	850
Exploratory and Application Porting	650
Science Gateway Access	500
Workflow, Ensemble and Paramater Sweep	250
Remote Interactive Steering and Visualization	35
Tight-Coupled Distributed Computation	10

TABLE I

TeraGrid Usage Mode distribution for 2006, the latest year for which data is available. Notice the small proportions of users/applications that utilize multiple resources collectively – concurrently or otherwise

Scale-out?

Extensibility?

Application Type	Characteristics & Examples
Simulation	CPU-intensive, Large number of independent jobs. e.g., Physics Monte Carlo event simulation
Production Processing	Significant I/O of data from remote sources e.g, Processing of physics raw event data
Complex Workflow	Use of VO specific higher-level services; Dependencies between tasks e.g., Analysis, Text mining
Real-time Response	Short runs; Semi-guaranteed response times, e.g., Grid operations and monitoring
Small-scale Parallelism	Allocation of multiple CPUs simultaneously; Use of MPI libraries, e.g., Protein analysis, MD

TABLE II

TYPES OF APPLICATION RUNNING ON THE OPEN SCIENCE GRID

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Distributed Applications (1)

How do they differ from traditional HPC applications?

- Performance Models:
  - Not “peak utilization” e.g., # of jobs
  - Manage scalability, faults, heterogeneity
- Usage Modes:
  - The same application has multiple usage modes
  - How applications are run, deployed and utilized is often determined by the infrastructure
- Static versus Dynamic Execution:
  - Varying resource conditions; distributed applications should be dynamic
- Skillful Decomposition vs Composition

CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Distributed Applications (2)

- The Future is Distributed:
  - Basic Trends in Computing
    - Decoupling & Delocalization of Data Production-Consumption
    - Components of computation
  - Distributed services/peopleNature of Scientific Applications, Discovery/Research
    - Parallelism is important, but exclusively for ~20%
    - Working examples of where if you can distribute effectively, you can choose methods not critical on ||

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Distributed Applications(3)

- Distributed Applications Require:
  - Distribution of compute-data-people (manage)
  - Coordination over Multiple & Distributed sites:
    - Logically or physically distributed
    - Scale-up and Scale-out
    - Peta/Exa/Atta - Scientific Applications requiring multiple-runs, ensembles, workflows etc.
  - Ability to develop simple, novel or effective distributed applications lags behind

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Distributed Applications (4) Development Objectives

- Interoperability: Ability to work across multiple distributed resources
- Scale-Out: The ability to utilize multiple distributed resources concurrently
- Extensibility: Support new patterns/abstractions, different programming systems and Infrastructure
- Adaptivity: Response to fluctuations in dynamic resource and availability of dynamic data
- Simplicity: Accommodate above distributed concerns at different levels *easily…*

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Developing Distributed Applications (2)

- Challenge: How to develop DA effectively and efficiently, with interoperability, scalability, adaptivity, extensibility, and simplicity as first-class concerns?
  - Understand what can be done? What are the gaps?
  - Understand characteristics and requirements DA
- Vectors: Axes representing application characteristics, the value of which help us understand the application and what influences the design/constraints of solutions, tools & programming systems that can be used
  - What makes distributed applications hard to develop?
  - What makes distributed infrastructure hard to use?

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Taxonomy of Distributed Application Development (1)

- Distributing a *legacy* Application, or invoking simple distributed functionality
  - Implicitly distributed
    - Legacy applications, e.g., NGS via portals
  - Mechanisms to support distributed execution anywhere
    - Typically no-coupling or coordination between tasks
      - e.g., 1000 “simple” job submission on 10 clusters
      - e.g., Condor-based submission
    - Low resource specificity or control required

CENTER FOR COMPUTATION & TECHNOLOGY

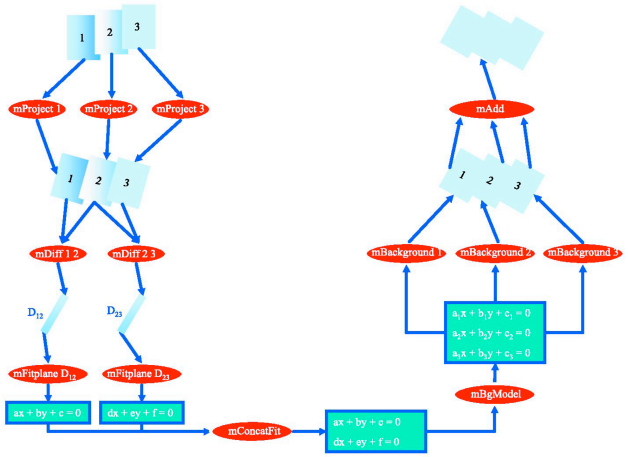
Taxonomy of Distributed Applications Development (2)

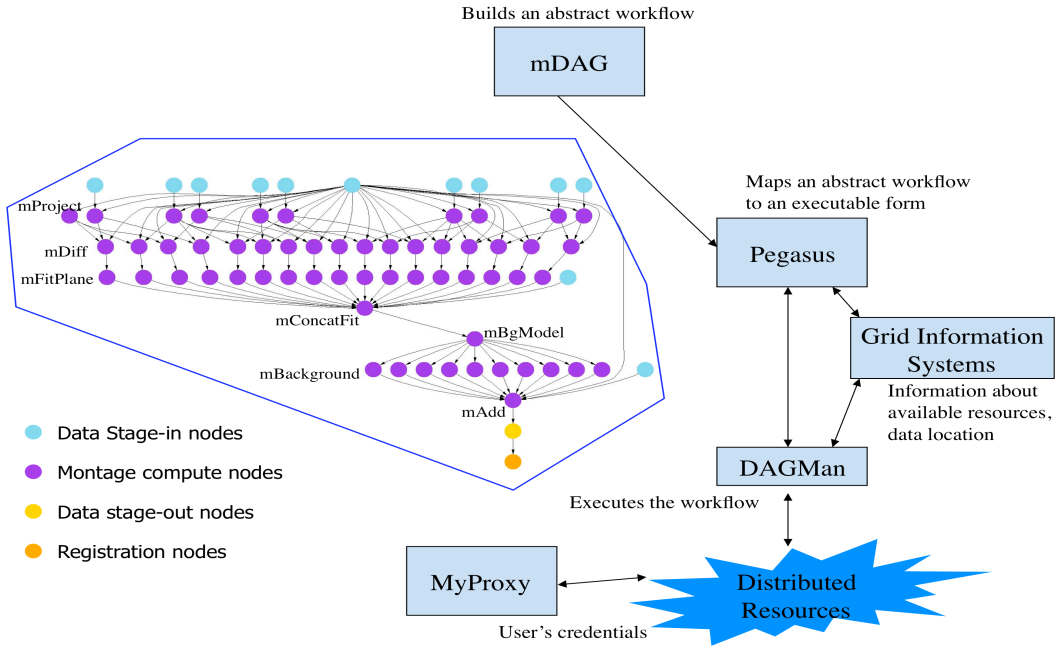
- Developing Applications with Multiple Components
  - *Naturally* decomposed; aggregate (coordination)
    - E.g. DDDAS, sensor-based application, DAG-based
  - Decompose a Large problem into sub-components
  - Designing *De Novo* Distributed Application
    - E.g. GridSAT (Wolski), Distributed Reasoning (Bal)
- Coupling (Freq., Vol.) components set coordination strategy
  - Low flexibility in placement, ordering, resource selection; or Communication (e.g. MPIg)
  - High flexibility in placement, scheduling, ordering, (e.g. Replica-Exchange)

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Montage: DAG-based Workflow Application Exemplar





saga

CENTER FOR COMPUTATION & TECHNOLOGY

Taxonomy of Distributed Application Development (3): Frameworks

- Frameworks: Logical structure for Capturing Application Requirements, Characteristics & Patterns
- Pattern: Commonly recurring modes of computation
  - Programming, Deployment, Execution, Data-access..
- Abstraction: Mechanism to support patterns and application characteristics
- Frameworks designed to either:
  - Support Patterns: MapReduce, Master-Worker, Hierarchical Job-Submission
  - Provide the abstractions and/or support the requirements & characteristics of applications

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Distributed: Implicit vs Explicit ?

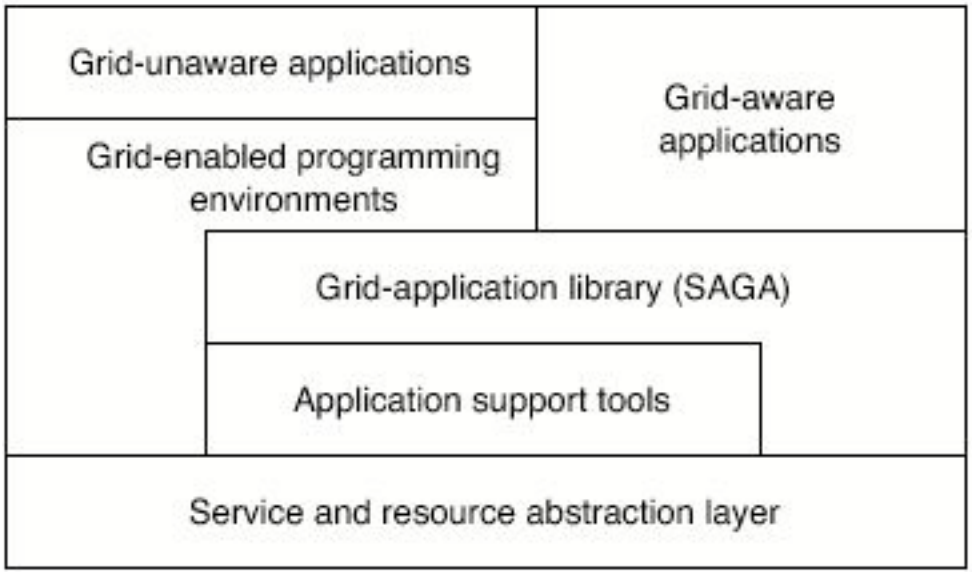
- Most applications/application classes can be either. Which approach (implicit vs explicit) is used depends:
  - How the application is used?
    - Need to control/marshall more than one resource?
  - Why distributed resources are being used?
  - How much can be kept out of the application?
    - Can't predict in advance?
    - Not obvious what to do, application-specific metric
- *Unless necessary, applications should not be explicitly distributed*

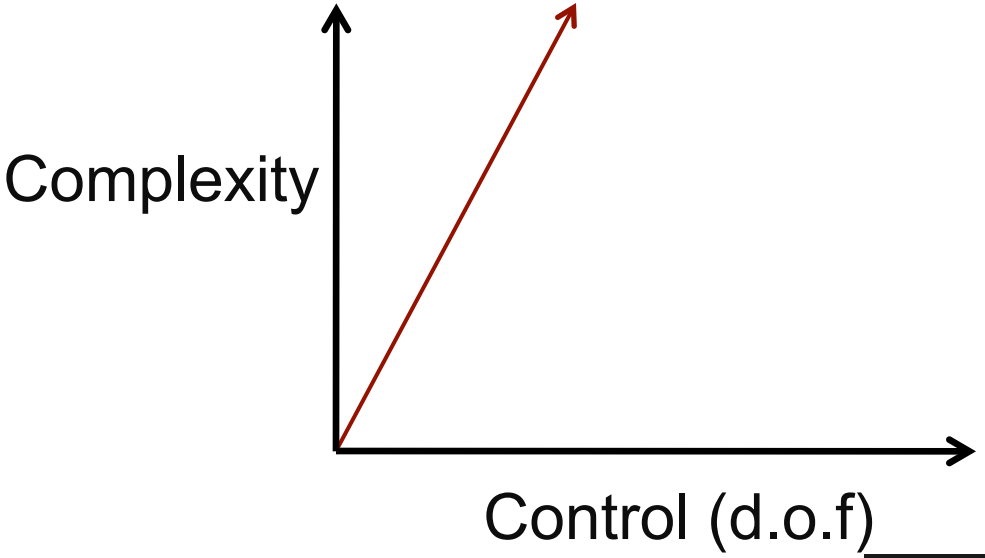
saga

CENTER FOR COMPUTATION & TECHNOLOGY

Application Class vs Type

Application Class (Cluster)	Properties (Values of Application Vectors)	Application Examples	General Concepts
Loosely Coupled Applications	Independent execution units; loosely coupled; data transfer through files.	Montage	Each execution unit is independently developed, and only exposes an I/O. Only an executable instance of each execution unit may be available. This application class involves a loose coupling between such executable instances.
Tightly Coupled Homogeneous Applications	Execution units interact via messaging or other mechanisms; strong dependency between units	Single Num. Rel. Simulation	
Tightly Coupled Heterogeneous Applications	Execution units interact via files; strong dependency between units; units may be implemented using different programming libraries	Molecular Dynamics Simulation (with separate components implementing different capabilities – such as force and velocity calculations, position of particles, etc)	Trivial generalization of MPI to WAN using MPICH-G2 and variants thereof, should also be considered here.
Loose Coupling of Tightly Coupled Applications	Many applications need to be used in conjunction with other applications; they may not necessarily have been designed for this *ab initio*. Alternatively, certain commonly used algorithms call for replacing single long-time running simulations with multiple shorter time-duration simulations – but with possible infrequent communication between the individual simulations.	Replica Exchange simulations for protein folding; multi-physics scientific applications	There are two-levels of communication; the first is internal to a single job/task (think monolithic MPI job), the second level is communication between the jobs/talks. The latter is less frequent and thereby more tolerant of latency and delays. Coordination of the many tasks/jobs is challenging.
Event-Oriented Applications	Execution units employ a publish/subscribe mechanism to coordinate; generally loose coupling between units	Distributed database search	





- No unique mapping between Application Class & Type.. .
- The same application can *often* be either explict or implicitly distributed

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Outline (1)

e-Science Institute

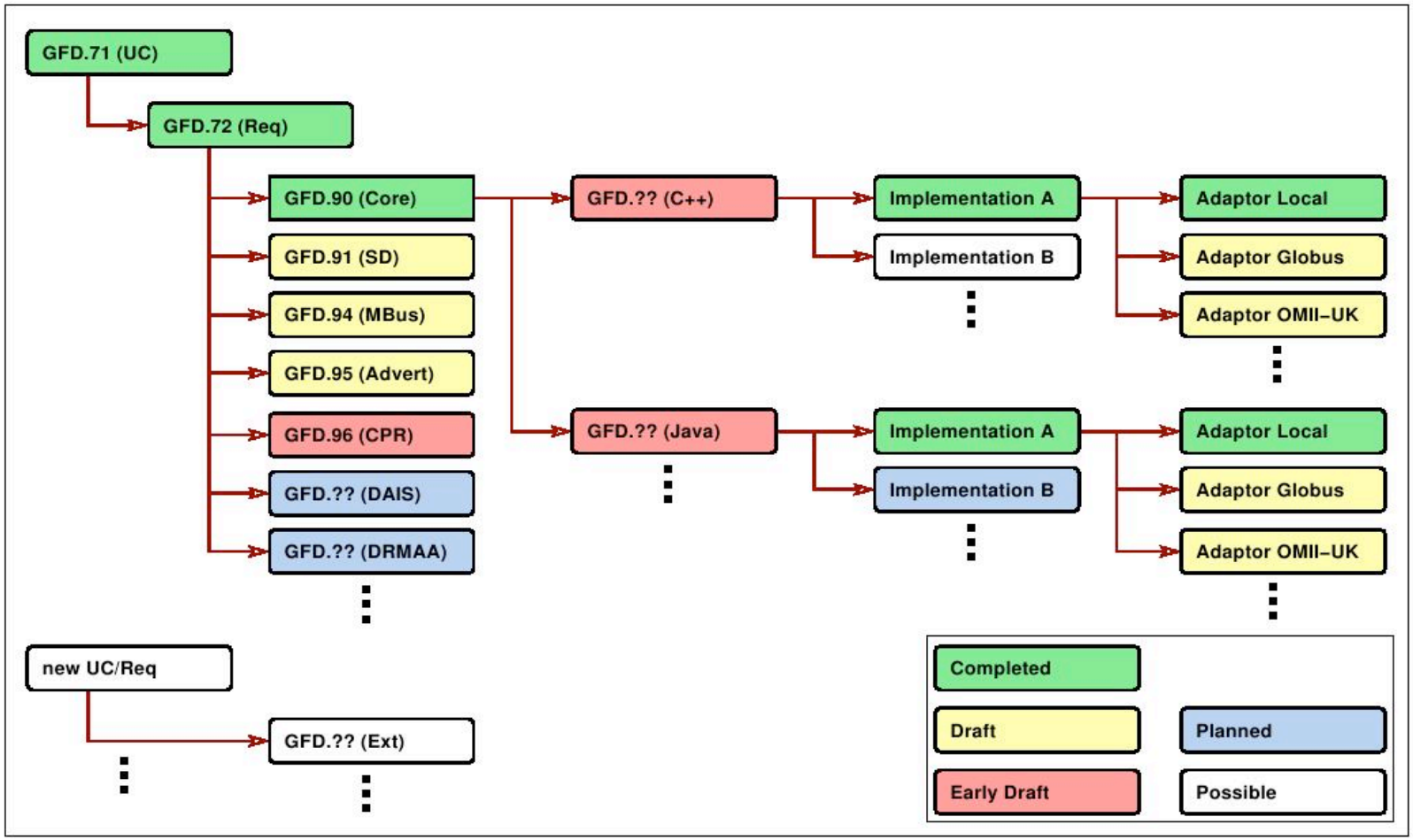
- Critical Perspective on Large-Scale Distributed Applications and Production Cyber-Infrastructure (CI)
- Understanding Production CI
- Understanding Distributed Applications (DA)
  - Differ from HPC or || App?
  - Challenges of DA
  - Rough Taxonomy of Distributed Applications
    1. Distributed Execution Mode Legacy (Implicit)
    2. Explicit coordination and distribution
    3. Frameworks: support characteristics or patterns
- Understanding the SAGA Landscape
  - Interface/Specification, C++ Engine, Adaptors

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: The Standard Landscape

e-Science Institute

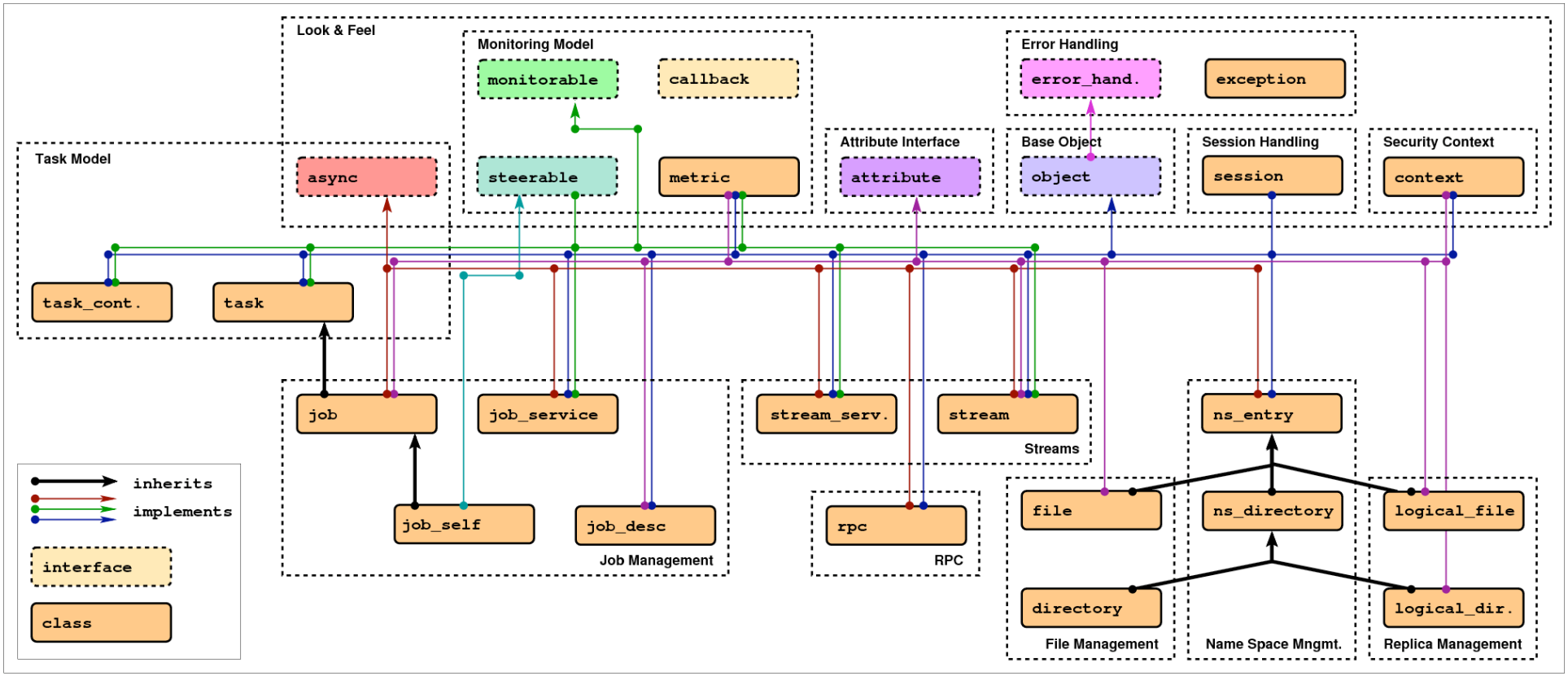


saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA v1.0 Hierarchy - Full Glory

e-Science Institute



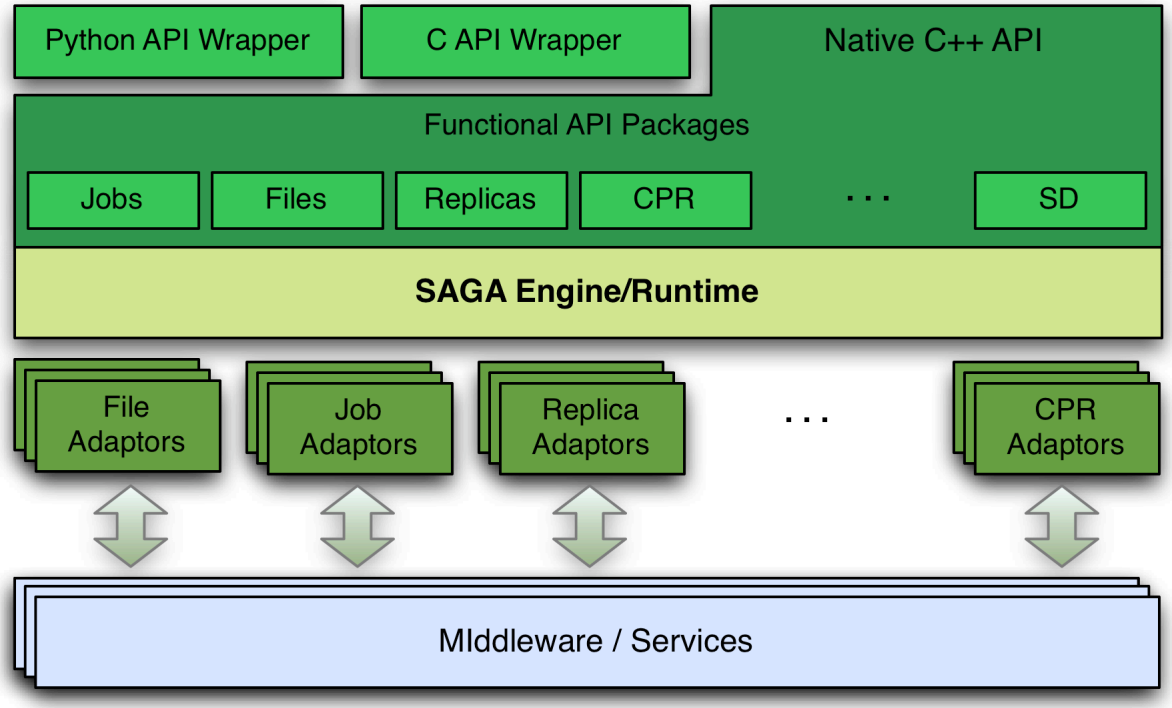
GridRPC - A rendering of *GridRPC*

saga

CENTER FOR COMPUTATION & TECHNOLOGY

And in pictures..

e-Science Institute



saga

CENTER FOR COMPUTATION
& TECHNOLOGY

SAGA Implementation: Requirements

- Non-trivial set of requirements:
  - Allow heterogenous middleware to co-exist
  - Cope with evolving grid environments; dyn resources
  - Future SAGA API extensions
  - Portable, syntactically and semantically platform independent; permit latency hiding mechanisms
  - Ease of deployment, configuration, multiple-language support, documentation etc.
  - Provide synchronous, asynchronous & task versions

  Portability, modularity, flexibility, adaptabilty, extensibility

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA Implementation: Extensibility

e-Science Institute

- Horizontal Extensibility – API Packages
  - Current packages:
    - file management, job management, remote procedure calls, replica management, data streaming
    - Steering, information services, checkpoint in pipeline
- Vertical Extensibility – Middleware Bindings
  - Different adaptors for different middleware
  - Set of ‘local’ adaptors
- Extensibility for Optimization and Features
  - Bulk optimization, modular design

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Is SAGA Simple?

- It depends: It is certainly not simple to implement!
  - Grids are complex and the complexity needs to be addressed somewhere, by someone!
  - Pain using the middleware goes into the SAGA engine and adaptors.
- But it is simple to use!!
  - Functional Packages (specific calls), Look & Feel
  - Somewhat like MPI - most users only need a very small subset of calls

saga

CENTER FOR COMPUTATION
& TECHNOLOGY

Implementations

- OGF Standard: Two independent implementations of a specification are required
- LSU: C++
  - V1.0 release Sep 15
  - Uptake by NAREGI/KEK, gLite (SD)
- Java
  - VU (Amsterdam):
    - Part of the OMII-UK Project
  - JSAGA
  - DEISA (DESHL)

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: C++ Status and Timelines

http://saga.cct.lsu.edu

e-Science Institute

- C++: Currently at v1.3.1 (July 2009)
  - Monthly release cycles for 2008; moving to Qtr cycle
  - Release V1.0 September 15 2008 (OGF24)
    - Both (OMII) JAVA and C++
      - Will mirror V1.0 of the specification
      - Cleaned up Build system for Linux, Mac & Win
  - Adaptors:
    - Globus RLS, GRAM, GridSAM, Gridftp (available)
    - ssh, libcurl (in progress)
    - Condor, LSF
    - CloudStore (KFS), HDFS
  - Extension packages:
    - Service Discovery (document in public comment)
    - Checkpoint and Recovery (Migol)
- Python bindings to the C++ available
- User Manual and Programmer's Guide (beta version available)

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Outline (2)

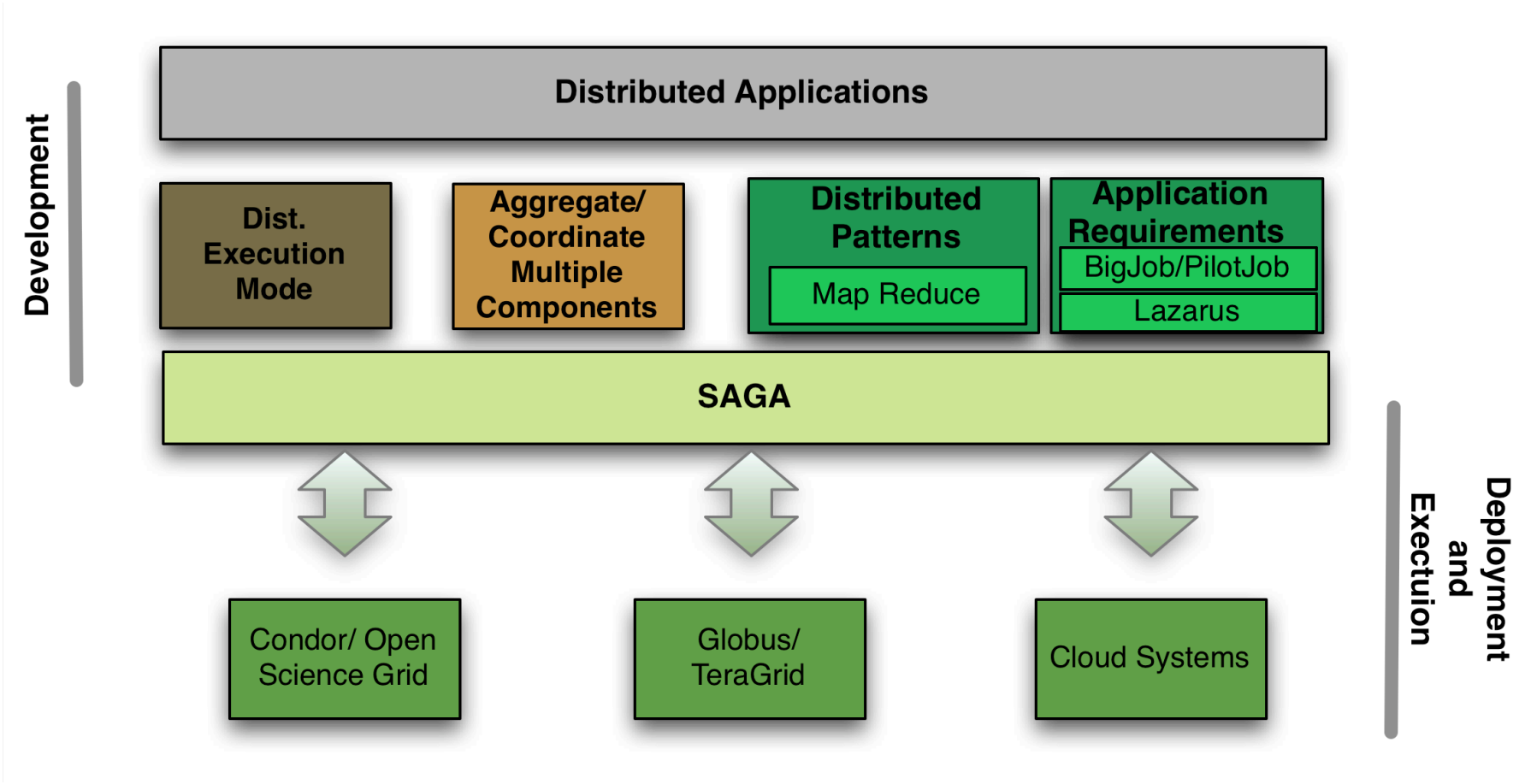
e-Science Institute

- Using SAGA to develop Distributed Applications
  - SAGA-based DAG Applications (Montage)
  - Frameworks Based:
    - Replica-Exchange
    - Multi-Physics Coupled Simulations,
    - Reservoir Simulations (EnKF)
    - MapReduce
      - Interoperabilty across Infrastructure
- SAGA-based Tools and Projects (Advantage of Standards)
  - DESHL, JSAGA, GANGA (gLite)
  - XtreemOS, NAREGI/KEK

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

SAGA and Distributed Applications



saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

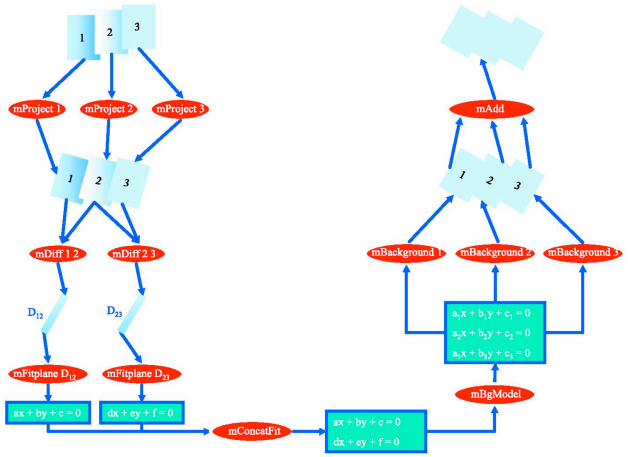
SAGA: Application Types

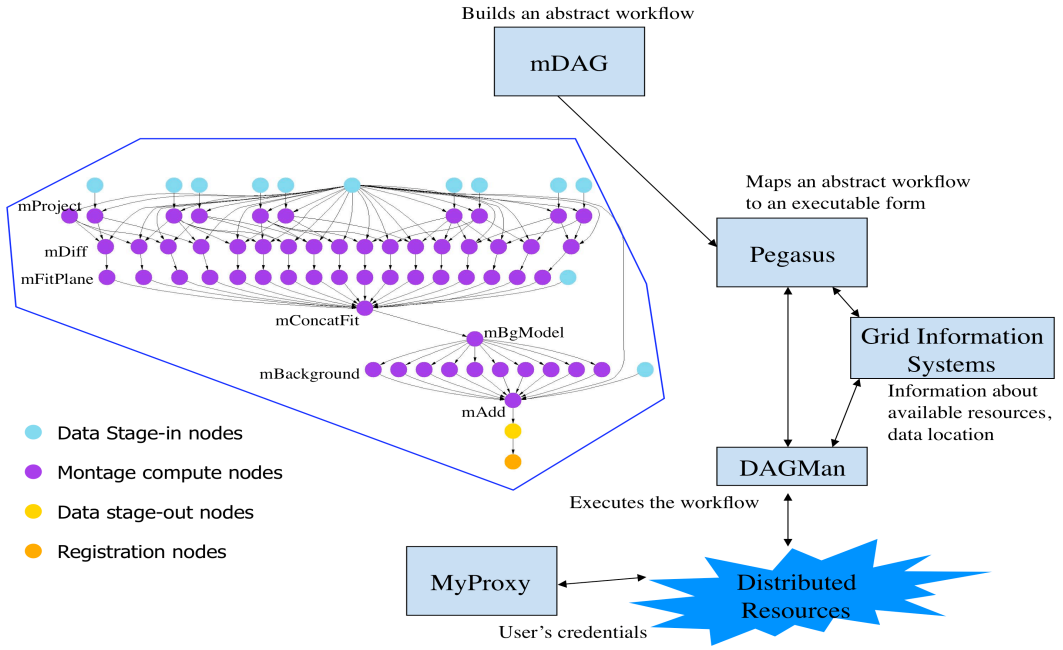
- Example of Distributed Execution Mode:
  - Implicitly Distributed
    - 1000 job submissions on the TG
  - SAGA shell example/tutorial
- Example of Explicit Coordination and Distribution
  - Explicitly Distributed
  - DAG-based Workflows
  - EnKF-HM application
- Example of SAGA-based Frameworks
  - MapReduce, Pilot-Jobs

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Montage: DAG-based Workflow Application Exemplar



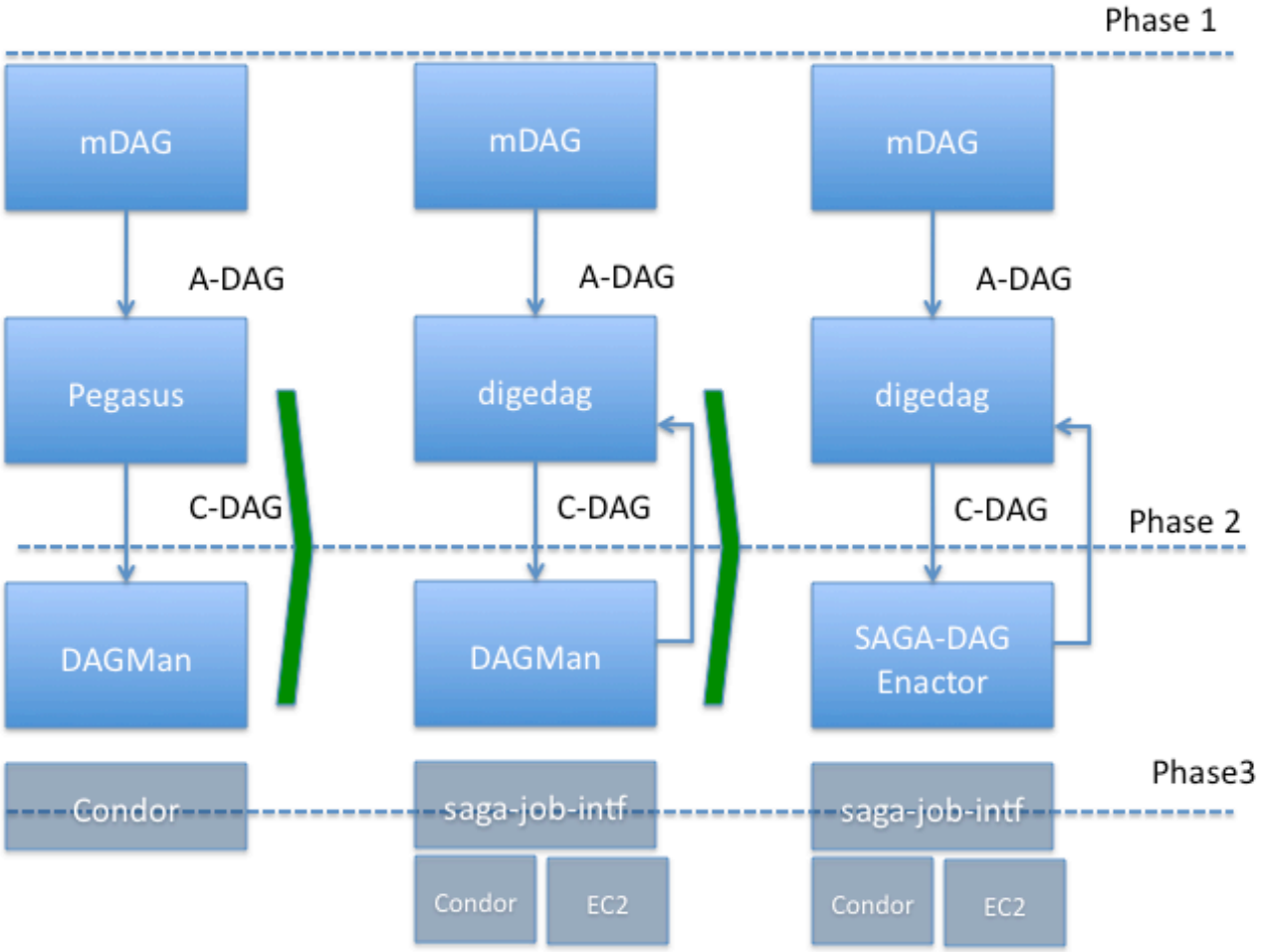


saga

CENTER FOR COMPUTATION & TECHNOLOGY

DAG based Workflow Applications Extensibility Approach

Application Development Phase



Generation & Exec. Planning Phase

Execution Phase

saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA-based DAG Execution Preserving Performance

#	resources	middleware	walltime	std-dev.	diff to local
1	l	f	68.7 s	9.4 s	– –
2	l	s	131.3 s	8.7 s	62.6 s
3	l	c	155.0 s	16.6 s	86.3 s
4	l	f, s	89.8 s	5.7 s	21.1 s
5	l	f, c	117.7 s	17.7 s	49.0 s
6	l	s, c	133.5 s	32.5 s	64.8 s
7	l	f, s, c	144.8 s	18.3 s	76.1 s
8	q	s	491.6 s	50.6 s	422.9 s
9	e	a	354.2 s	23.3 s	285.5 s
10	e, q	s, a	363.6 s	60.9 s	294.0 s
11	l, q, e	f, s, a	409.6 s	60.9 s	340.9 s
12	l	d	168.8 s[6]	5.3 s	100.1 s
11	p	d	309.7 s	41.5 s	241.0 s

TABLE II: Execution measurements
resources: ***l****=local,* ***p****=Purdue,* ***q****=Queen Bee,* ***e****=aws/EC2*
middleware: ***f****=fork/SAGA,* ***s****=ssh/SAGA,* ***a****=aws/SAGA,*
c*=Condor/SAGA,****d****=Condor/DAGMan,* ***p****=Pegasus*

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

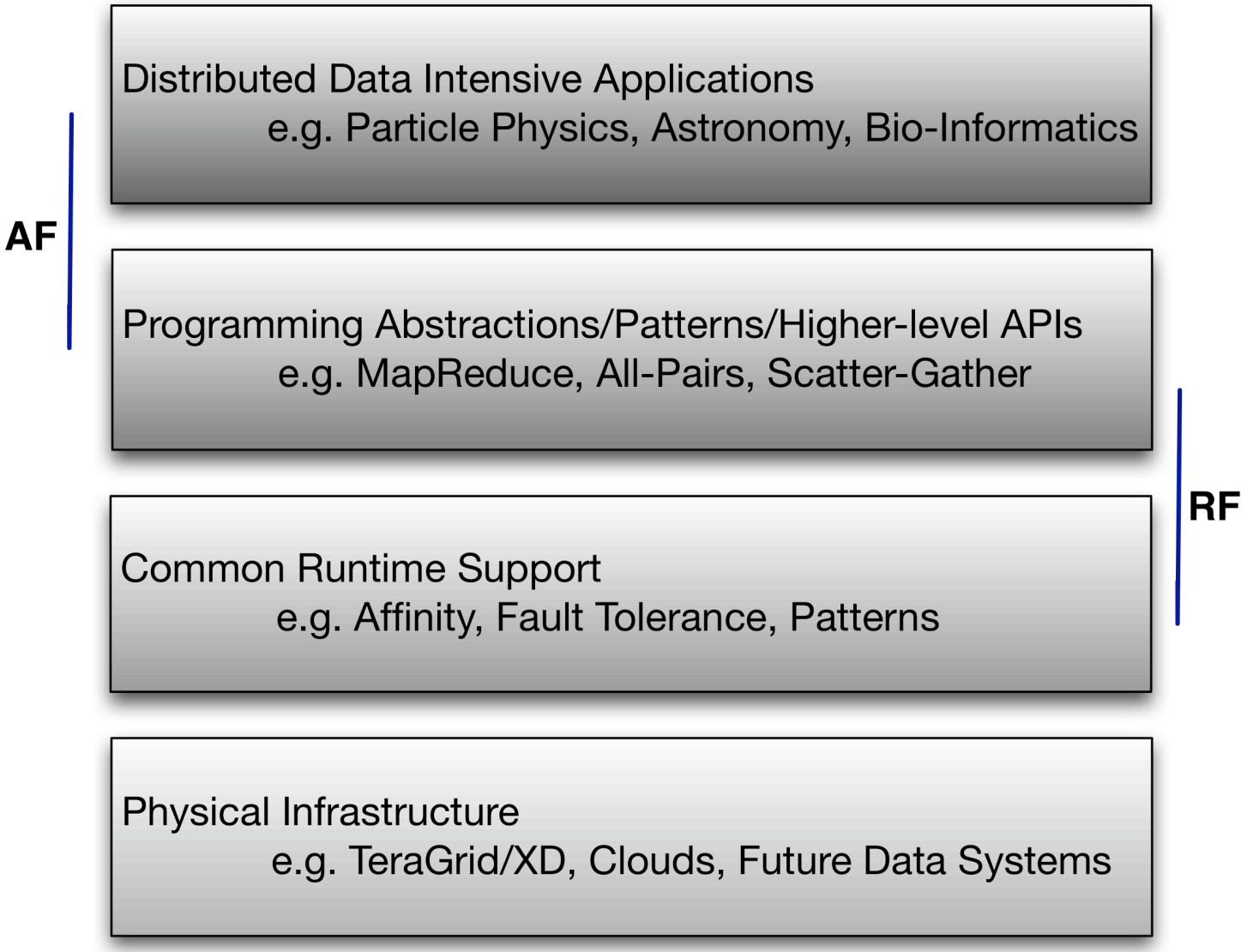
SAGA: Frameworks

- Frameworks: Logical structure for Capturing Application Requirements, Characteristics & Patterns
- Frameworks designed to either:
  - Support Patterns:
    - Programming, run-time, data-access
    - MapReduce, Master-Worker, H-J Submission
  - Provide the abstractions and/or support the requirements & characteristics of applications
- Applications can be more than 1 type
  - 3 Development Classes are not mutually exclusive
  - Can have a framework that helps in the explicit coordination

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

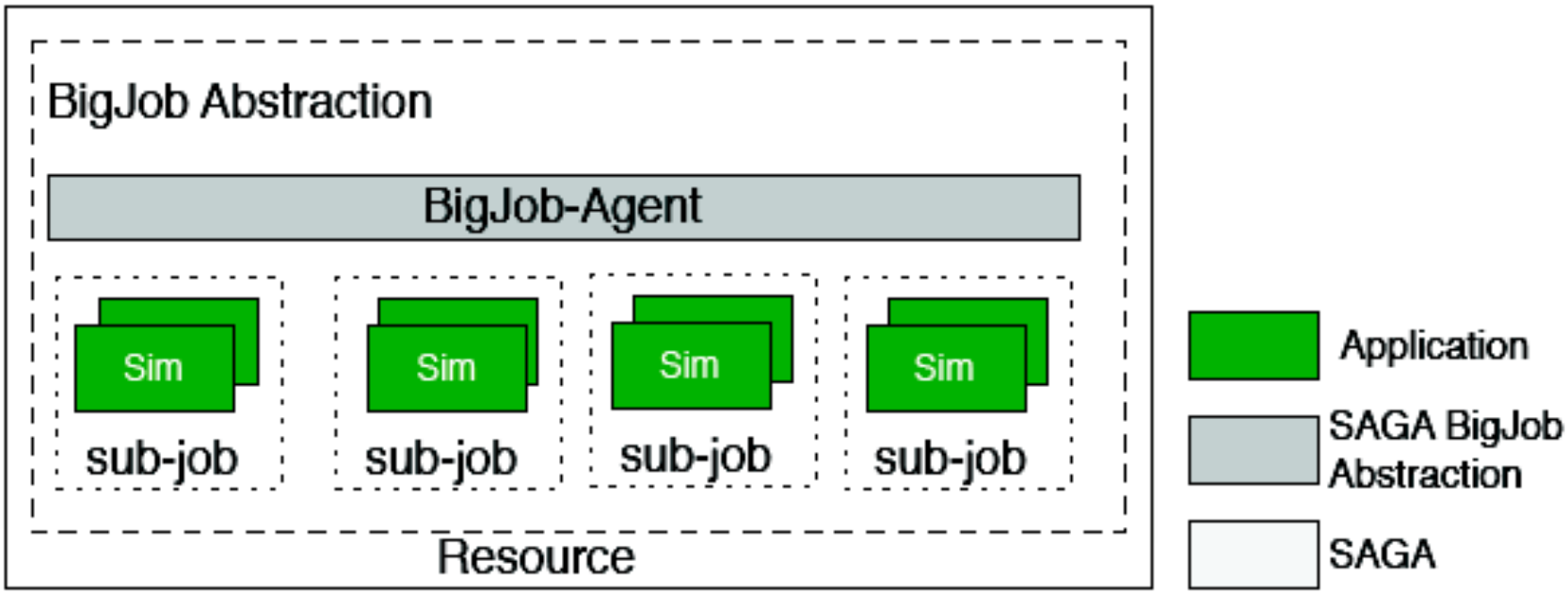
SAGA-based frameworks: Logical ordering



saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

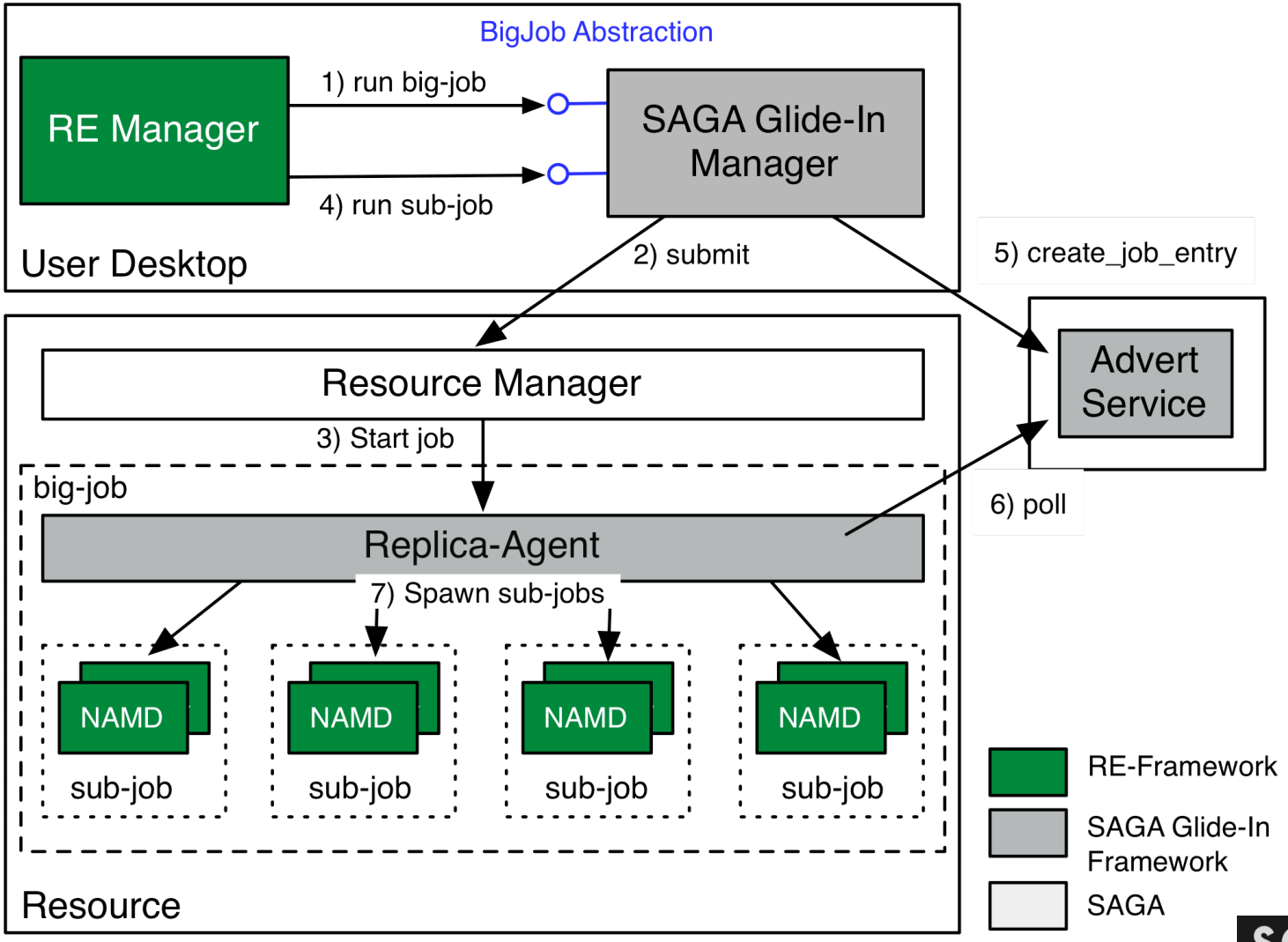
Abstractions for Distributed Computing (1)
BigJob: Container Task



saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

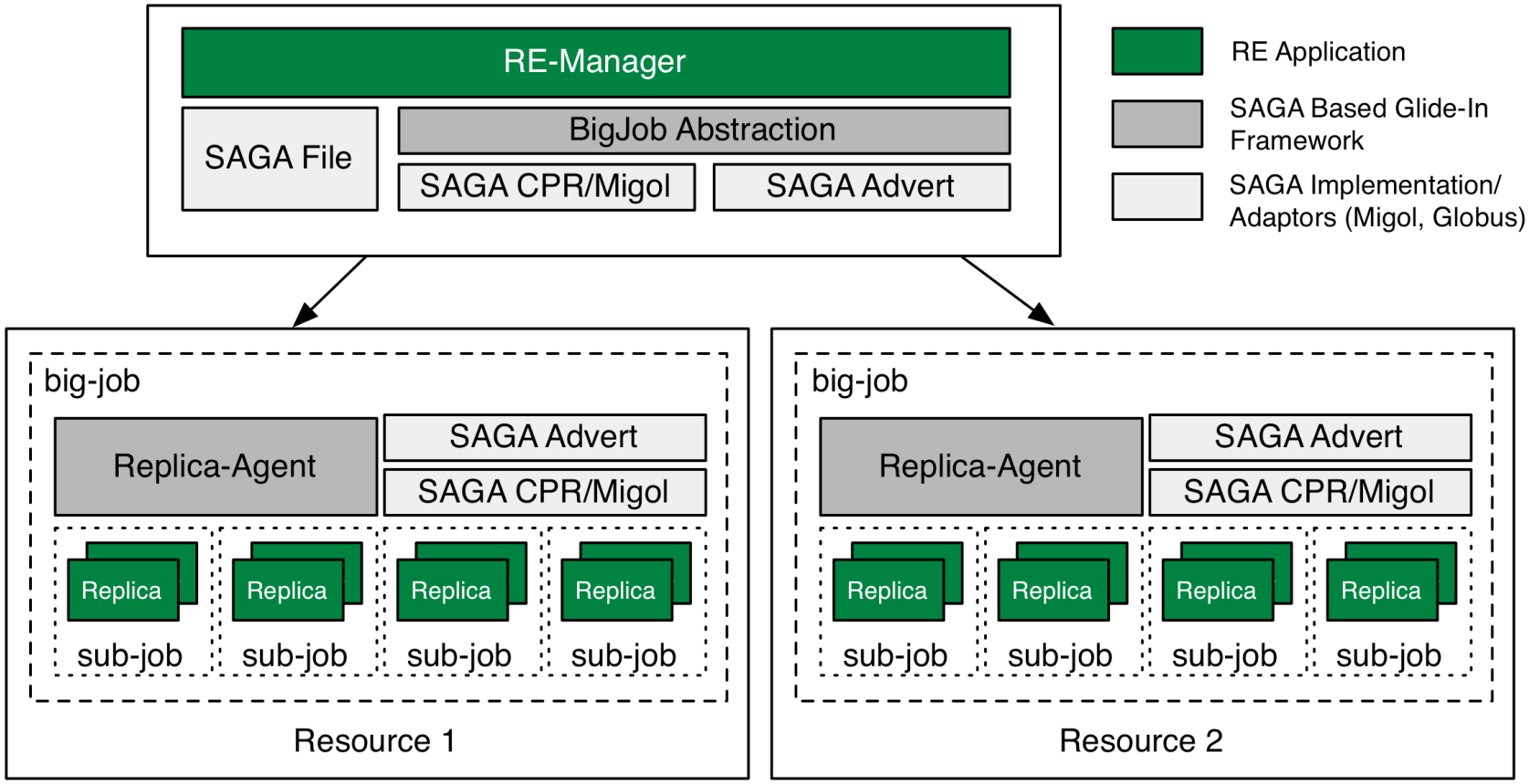
Abstractions for Distributed Computing (2) SAGA Pilot-Job (Glide-In)



saga

CENTER FOR COMPUTATION & TECHNOLOGY

Coordinate Deployment & Scheduling of Multiple Pilot-Jobs



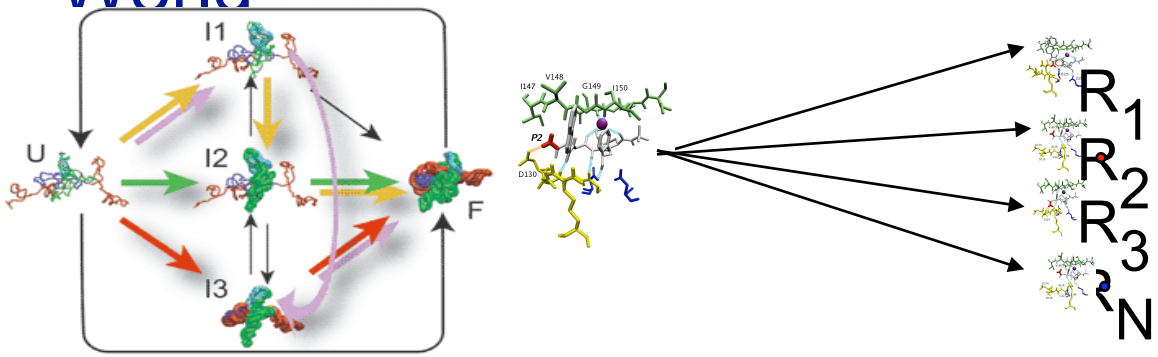
saga

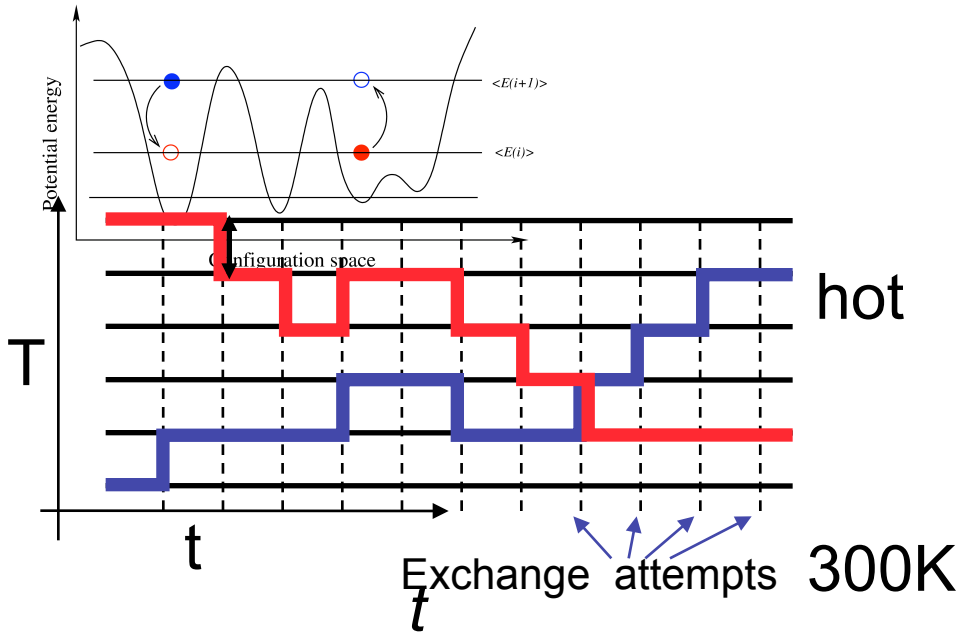
CENTER FOR COMPUTATION & TECHNOLOGY

Replica Exchange: Hello Distributed World

e-Science Institute

- 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





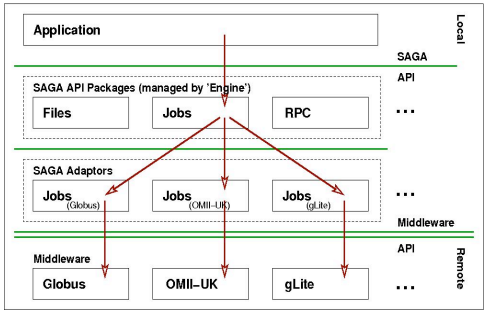
saga

CENTER FOR COMPUTATION & TECHNOLOGY

RE: Programming Requirements

e-Science Institute

- RE can be implemented using following “primitives”
  - Read job description
    - # of processors, replicas, determine resources
  - Submit jobs (local and remote)
    - Move files, job launch
  - Checkpoint and re-launch simulations
    - Exchange, RPC (logic to swap or not)

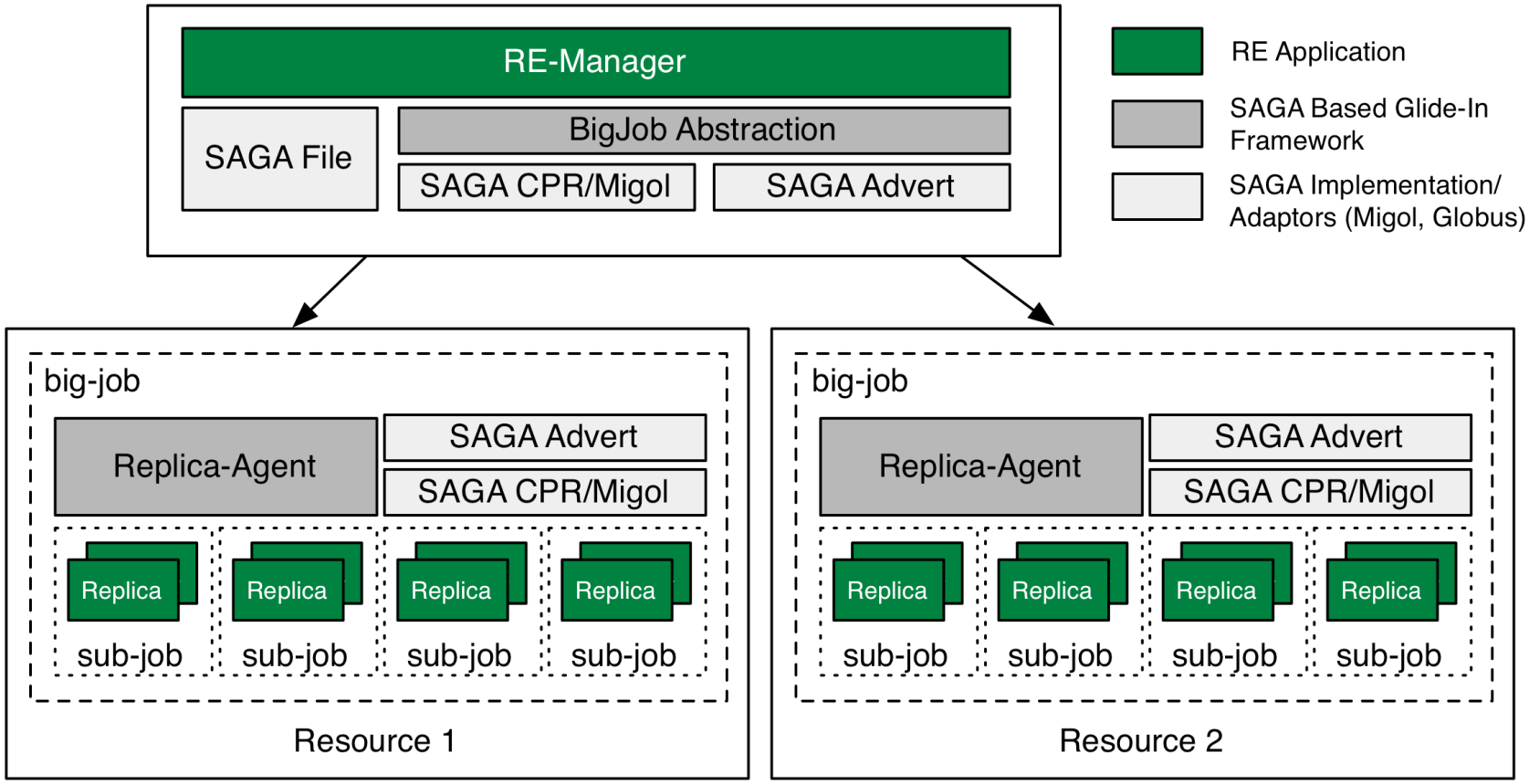


- Implement above using “Grid primitives” provided by SAGA
- Separated “distributed” logic from “simulation” logic
  - Independent of underlying code/engine
  - Science kernel is independent of details of distributed resource mgmt
  - Need to coordinate resources integrated from Desktop to Supercomputers!!

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Coordinate Deployment & Scheduling of Multiple Big-jobs (Pilot-Jobs)



saga

CENTER FOR COMPUTATION & TECHNOLOGY

Scale-Out: Dynamic Execution & Resource Aggregation

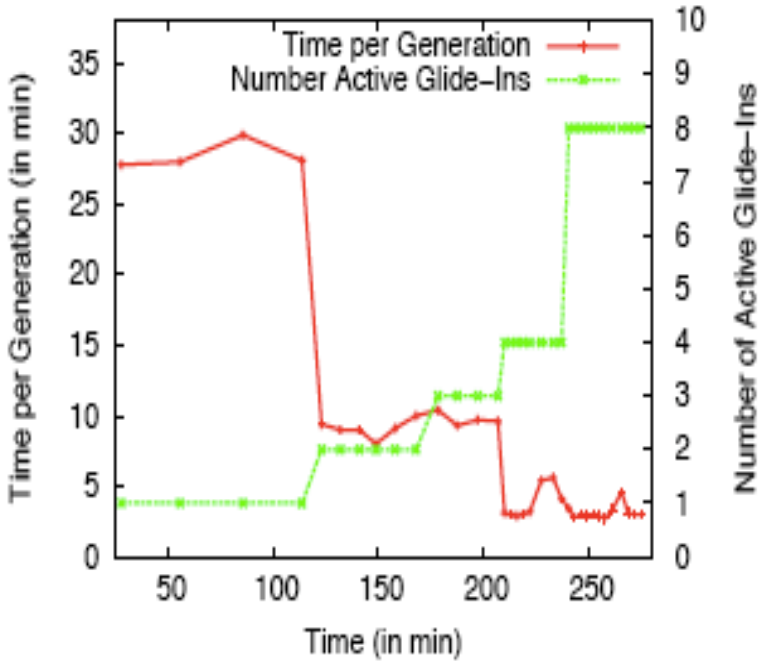


Figure 5: **The plots show the time-series of the average times between exchange attempts (in red and using the left-hand y axis) and the number of active Glide-Ins over a six-hour run on the TeraGrid.**

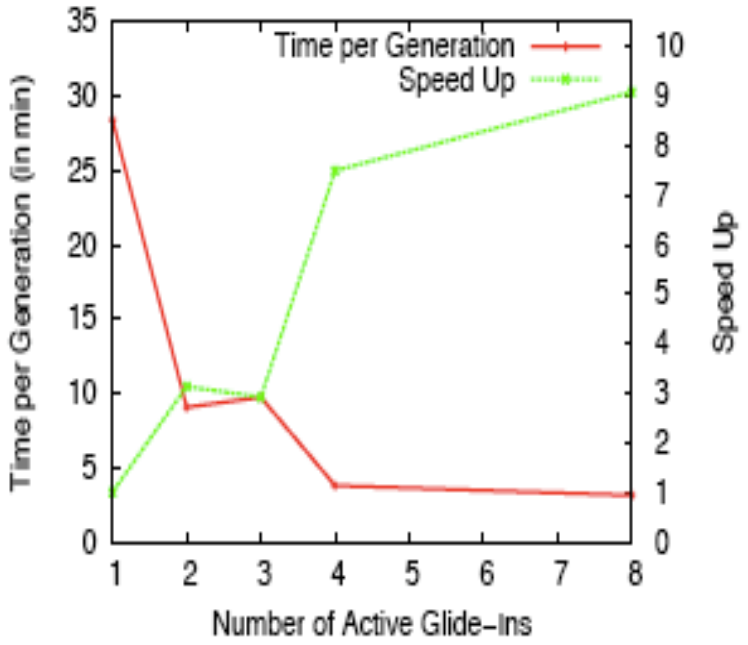


Figure 6: **The plot in red (using the left-hand y axis) illustrates how the average time between exchange attempts decreases as the number of Glide-Ins increases. The plot in green shows the speedup.**

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Scale-Out: Performance Enhancements

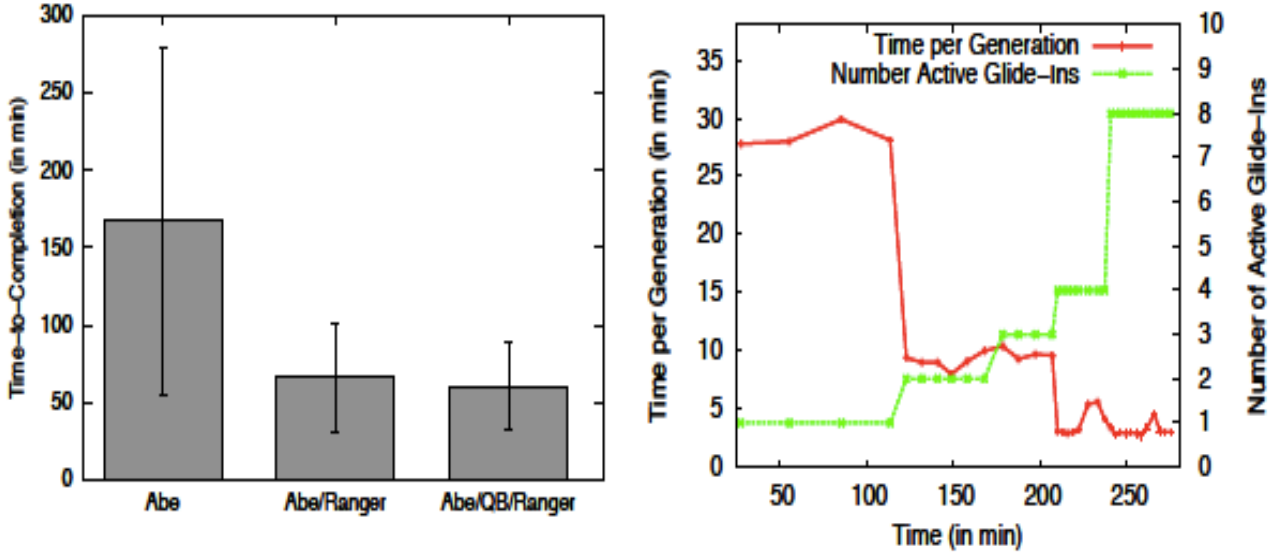


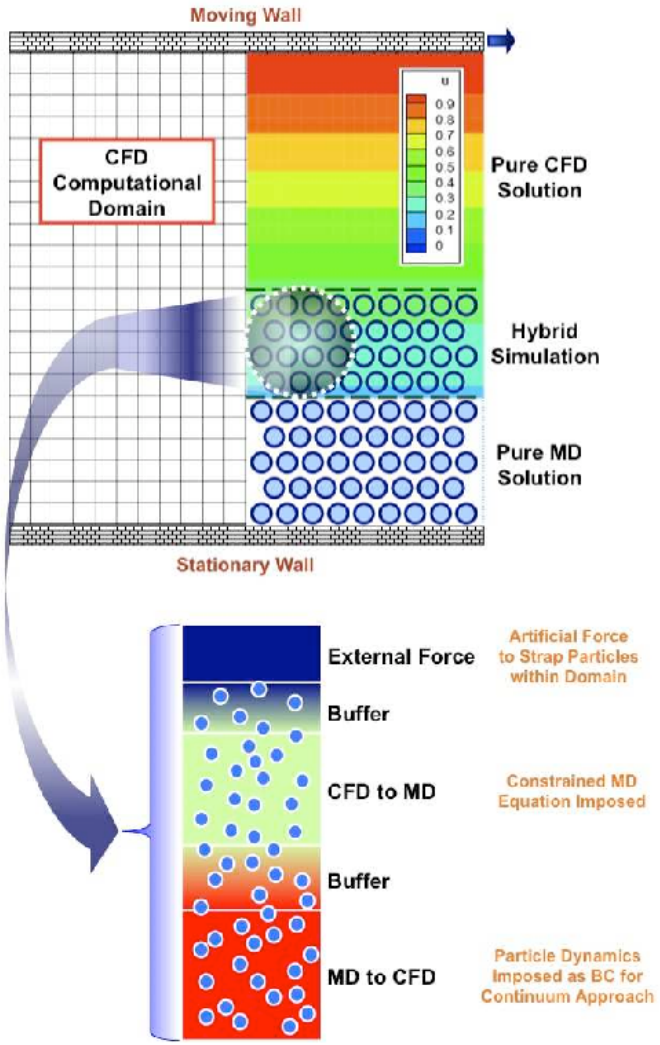
Figure 3: Performance data providing conclusive evidence that SAGA can be used to lower the time-to-solution, as the number of resources that can be used increases. SAGA provides the ability to use multiple resources in a simple and scalable fashion. The total number of computer-cycles used do not necessarily increase, i.e., time-to-completeness is not at the expense of efficiency. The lower figure contains plots which show the time-series of the average times between exchange attempts (upper line using the left-hand y axis) and the number of active Glide-Ins over a 6hr run.

saga

CENTER FOR COMPUTATION & TECHNOLOGY

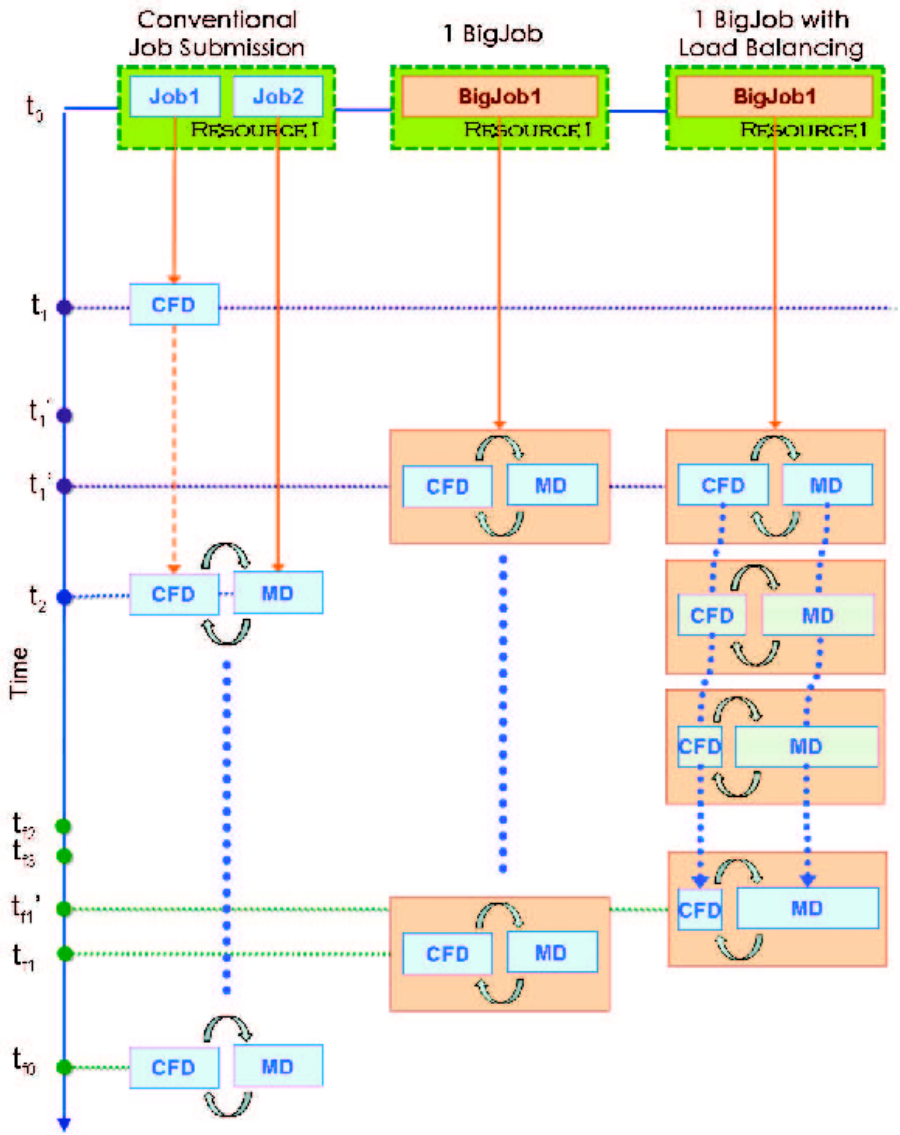
Multi-Physics Runtime Frameworks

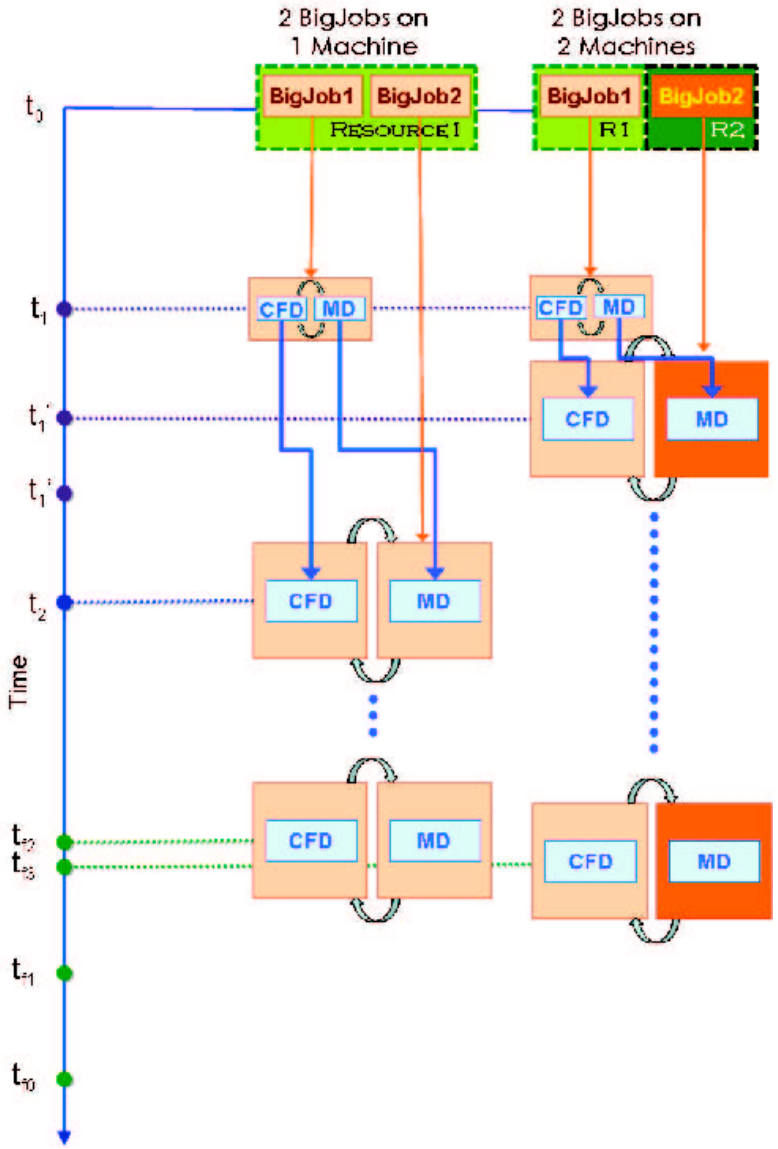
- Coupled Multi-Physics require two distinct, but concurrent simulations
- Can co-scheduling be avoided?
  - Adaptive execution model the answer is yes
- Load-balancing required. Capability comes for free!
- First demonstrated multi-platform Pilot-Job:
  - TG(MD) – Condor (CFD)



saga

LSU
CENTER FOR COMPUTATION
& TECHNOLOGY





saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Multiple BigJobs: Basic Performance

16-16	Baseline Simulation	BigJob (no LB)	BigJob (LB)
Waiting on the Queue	3000 (4136)	10834 (4593)	15201 (10968)
Inactive Mode	10300 (15327)	4 (0)	4 (0)
Active Runtime	1672 (246)	1641 (162)	1370 (98)
Total Time	14973 (19213)	12480 (4430)	16577 (11004)
8-24	Baseline Simulation	BigJob (no LB)	BigJob (LB)
Waiting on the Queue	1900 (1873)	3372 (6450)	10344 (6767)
Inactive Mode	5486 (10318)	4 (0)	4 (0)
Active Runtime	1171 (175)	1169 (72)	1280 (81)
Total Time	8557 (11882)	4545 (6457)	11629 (6720)

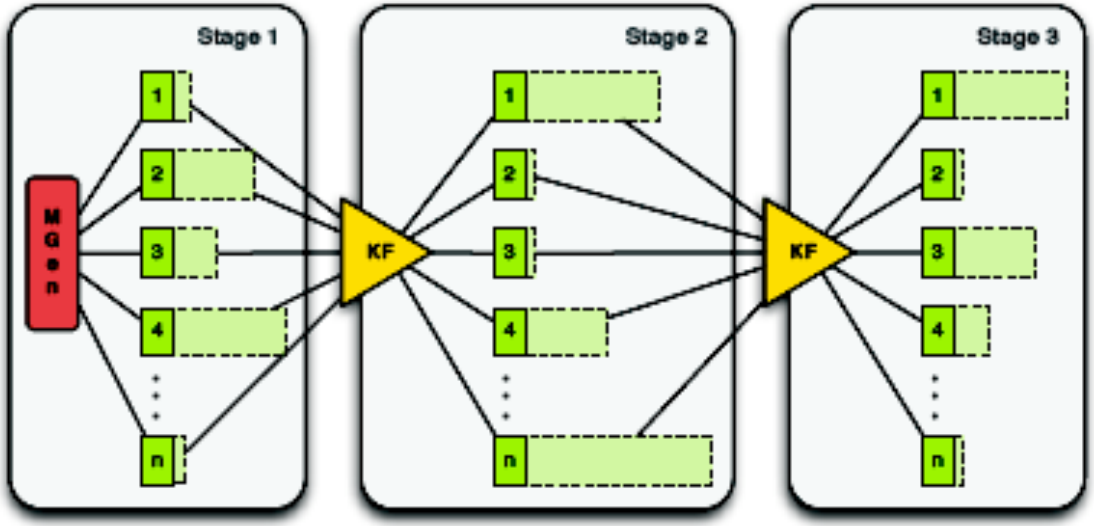
saga

CENTER FOR COMPUTATION & TECHNOLOGY

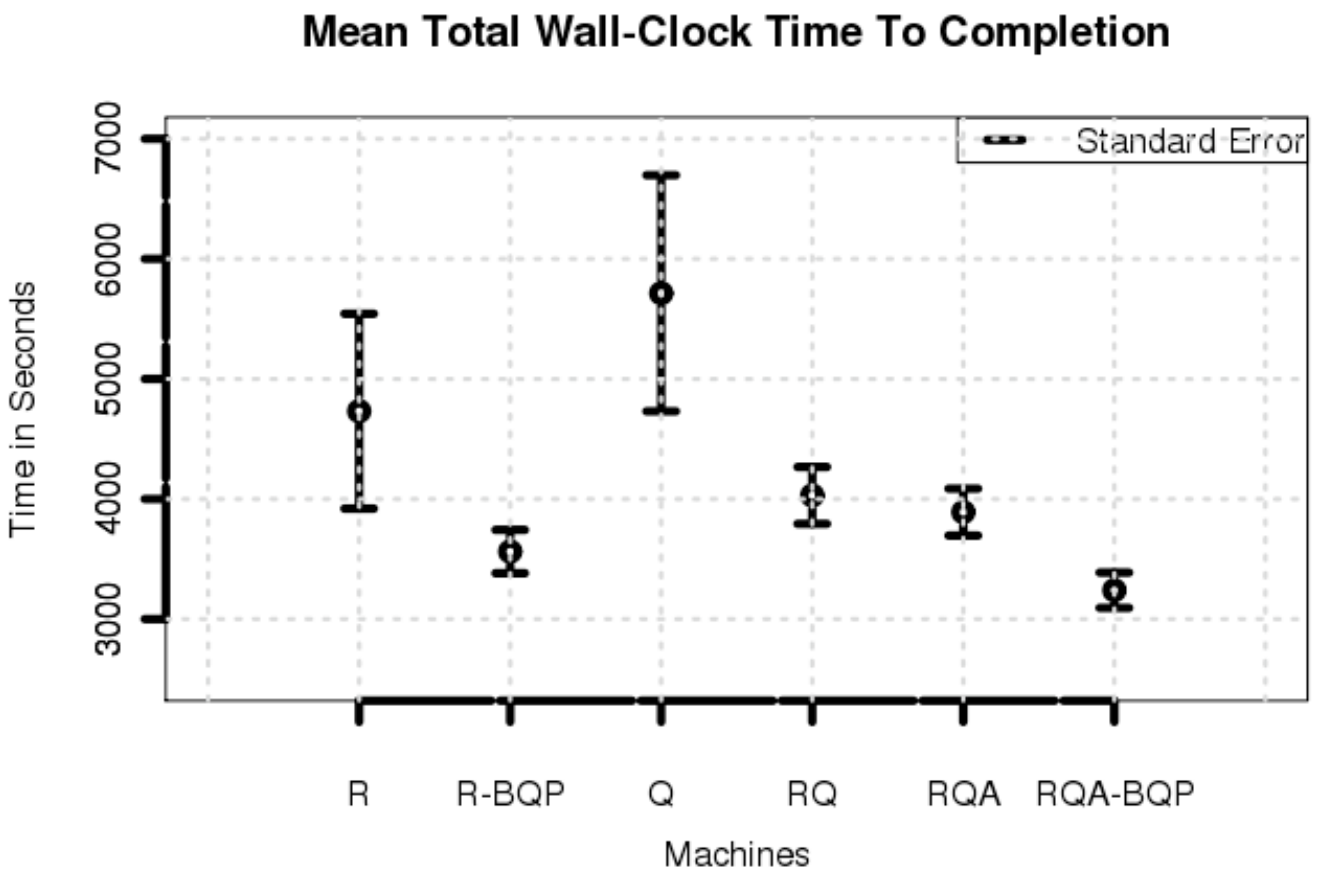
Ensemble Kalman Filters

Heterogeneous Sub-Tasks

- Ensemble Kalman filters (EnKF), are recursive filters to handle large, noisy data; use the EnKF for history matching and reservoir characterization
- EnKF is a particularly interesting case of irregular, hard-to-predict run time characteristics:



saga



But Why does BQP Help?

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

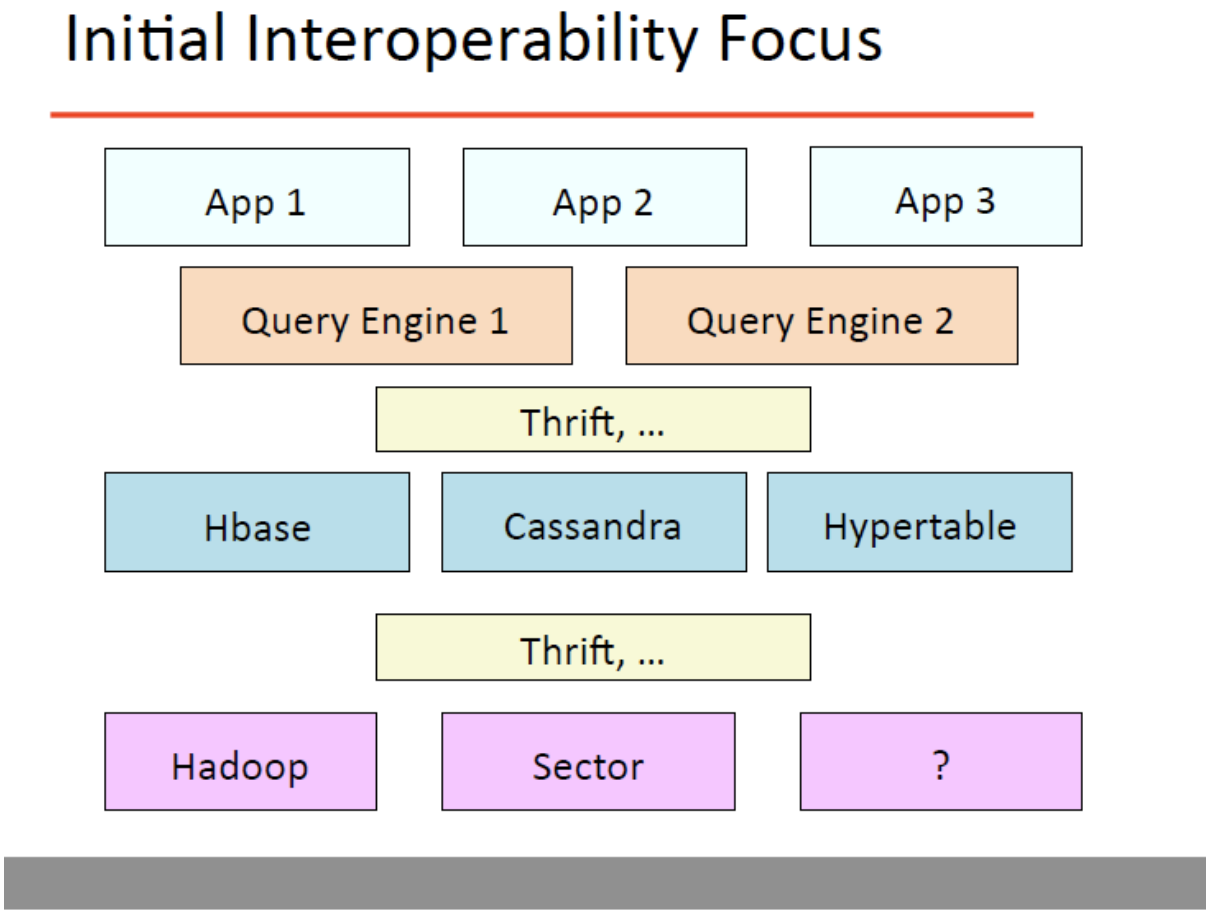
Forward Vs Reverse Scheduling
Adv of Ensemble of Loosely-Coupled

- Distributed Applications: When formulated using Loosely-Coupled Tasks:
  - Application keeps work ready to be fired, when resource becomes available (ideal case)
  - Match resources to computational requirement (Forward)
    - Take a workload; find a set of resources
  - Find a set of resources (first-to-run); tune workload
    - Reverse Scheduling: Support for Dynamic Execution of Applications is underlying paradigm
    - Late binding to resource and optimal resource configuration (not just resource selection)
    - Unbalanced, irregular workload

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Interoperabilty: Motivation



Slide adapted: Grossman

Decouple from vertical stack
Decouple from details of resource provisioning
Couple horizontal stacks

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Application-level Interoperability

Cloud-Cloud; Cloud-Grid

- Application-level (ALI) vs. System-level Interoperability (SLI)
- Infrastructure Independence is Pre-requisite for ALI
  - Programming Model should be Infrastructure & SLA agnostic:
    - Same application, priced differently, for same performance
    - Same application, priced same, for different performance
  - Availability zone
  - VM motion
  - Data-transfer cost vs. no cost
- Grids AND Clouds:
  - Hybrid & Heterogeneous workload: data-compute affinity differ
  - Complex data-flow dependency: need runtime determination

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Power of Google's MapReduce?

- MapReduce an important development but..
  - Currently tied to infrastructure
    - MapReduce + BigTable + GFS or
    - MapReduce + Hbase + HDFS (Yahoo)
  - Limited number of scientific applications that use this simple (though powerful) pattern
    - Other patterns?

Is it possible to provide the power of MapReduce and other patterns in an infrastructure independent

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: Role of Adaptors
Use over different infrastructure

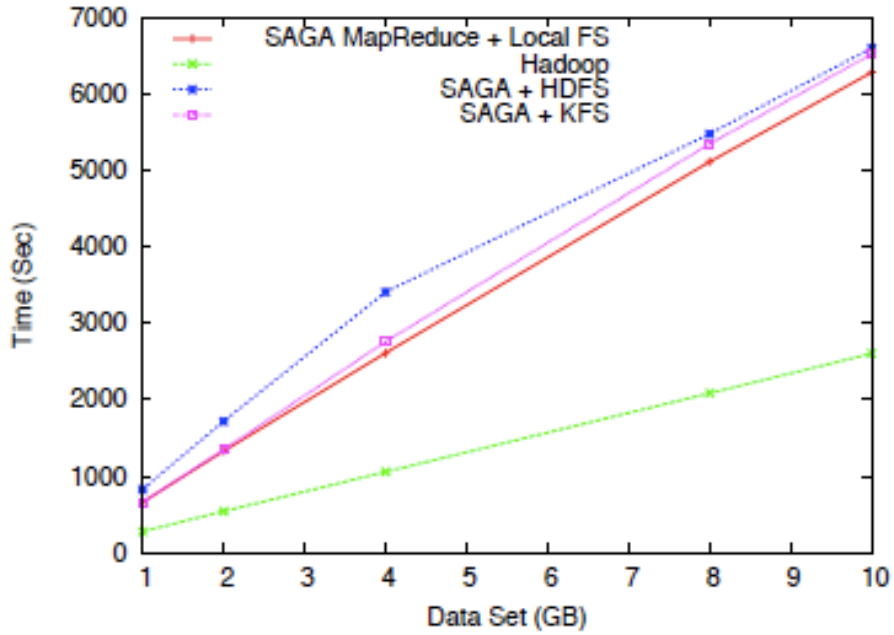


Fig. 5: T_c for SAGA-MapReduce using one worker (local to the master) for different configurations. The label "Hadoop" represents Yahoo's MapReduce implementation; T_c for Hadoop is without chunking, which takes several hundred sec for larger data-sets. The "SAGA MapReduce + Local FS" corresponds to the use of the local FS on Linux clusters, while the label "SAGA + HDFS" corresponds to the use of HDFS on the clusters. Due to simplicity, of the Local FS, its performance beats distributed FS when used in local mode.

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Controlling Relative Compute-Data Placement

Configuration		data size	work-load/worker	T_c
compute	data	(GB)	(GB/W)	(sec)
local	local-FS	1	0.1	466
distributed	local-FS	1	0.1	320
distributed	DFS	1	0.1	273.55
local	local-FS	2	0.25	673
distributed	local-FS	2	0.25	493
distributed	DFS	2	0.25	466
local	local-FS	4	0.5	1083
distributed	local-FS	4	0.5	912
distributed	DFS	4	0.5	848

TABLE I: Table showing T_c for different configurations of compute and data. The two compute configurations correspond to the situation where all workers are either placed locally or workers are distributed across two different resources. The data configurations arise when using a single local FS or a distributed FS (KFS) with 2 data-servers. It is evident from performance figures that an optimal value arises when distributing both data and compute.

saga

SAGA Job Launch via GRAM gatekeeper

```
{ // contact a GRAM gatekeeper                                          1
  saga::job::service      js;                                          2
  saga::job::description jd;                                           3
  jd.set_attribute (``Executable'', ``/tmp/my_prog'');                 4
  // translate job description to RSL                                  5
  // submit RSL to gatekeeper, and obtain job handle                   6
  saga:job::job j = js.create_job (jd);                                7
  j.run ():                                                            8
  // watch handle until job is finished                                9
  j.wait ();                                                           10
 } // break contact to GRAM                                           11
```

SAGA create a VM instance on a Cloud

```
{// create a VM instance on Eucalyptus/Nimbus/EC2                     1
 saga::job::service      js;                                          2
 saga::job::description jd;                                           3
 jd.set_attribute (``Executable'', ``/tmp/my_prog'');                 4
 // translate job description to ssh command                          5
 // run the ssh command on the VM                                     6
 saga:job::job j = js.create_job (jd);                                7
 j.run ():                                                            8
 // watch command until done                                          9
 j.wait ();                                                           10
} // shut down VM instance                                            11
```

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Interoperability

Number-of-Workers			Size	T_s	T_{spawn}	$T_s - T_{spawn}$
TG	AWS	Eucalyptus	(MB)	(sec)	(sec)	(sec)
-	1	1	10	5.3	3.8	1.5
-	2	2	10	10.7	8.8	1.9
-	1	1	100	6.7	3.8	2.9
-	2	2	100	10.3	7.3	3.0
1	-	1	10	4.7	3.3	1.4
1	-	1	100	6.4	3.4	3.0
2	2	-	10	7.4	5.9	1.5
3	3	-	10	11.6	10.3	1.6
4	4	-	10	13.7	11.6	2.1
5	5	-	10	33.2	29.4	3.8
10	10	-	10	32.2	28.8	2.4

TABLE II: Performance data for different configurations of worker placements on TeraGrid, Eucalyptus-Cloud and EC2. The first set of data establishes Cloud-Cloud interoperability. The second set (rows 5- 11) represent interoperability between Grids-Clouds (EC2). The experimental conditions and measurements are similar to Table 1.

saga

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

SAGA: Extension to Clouds

- Is the API perfect, that nothing really changes?
- Minor change:
  - Grids: Job_service() is not coupled to app lifetime
  - Clouds: Job_service() is coupled to the VM lifetime
    - When job_service() terminates, what happens to VM?
    - When VM terminates, what happens to job_service() ?
    - Changes to the SAGA Resource Discovery and Management Package will address these shortcomings

saga

CENTER FOR COMPUTATION & TECHNOLOGY

Grids vs Clouds?

- Its not Grids vs. Clouds:
  - *"…..such a good job of understanding parallel programming in the 90's … very well prepared for multi-core.." (Ken Kennedy)*
    - Hard problems in distributed applications & computing
      - e.g., coordination of data & computation
    - These remain, even if we change names!
  - HL-A & support for Usage modes
    - Specific infrastructure, Clouds vs. Grids is *less critical*
  - SAGA-MapReduce: Interoperability & Extensibility
    - Correct abstractions, interfaces are in place..

saga

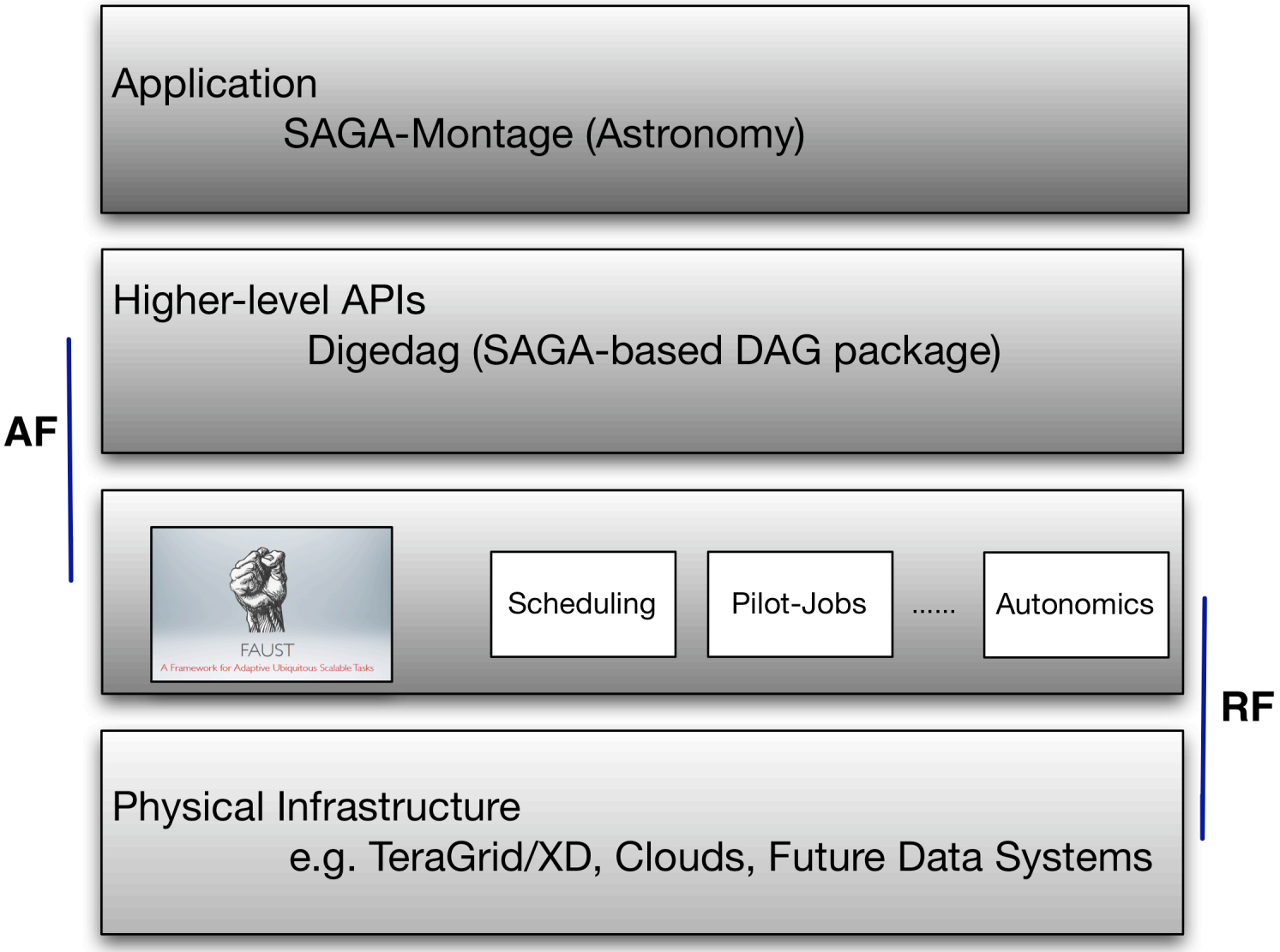
CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Distributed Applications: Development Objectives Redux

- Interoperability: Ability to work across multiple distributed resources
- Scale-Out: The ability to utilize multiple distributed resources concurrently
- Extensibility: Support new patterns/abstractions, different programming systems and Infrastructure
- Adaptivity: Response to fluctuations in dynamic resource and availability of dynamic data
- Simplicity: Accommodate above distributed concerns at different levels *easily…*

saga

SAGA-based frameworks: Logical ordering



FAUST
A Framework for Adaptive Ubiquitous Scalable Tasks

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

A Fresh Perspective?

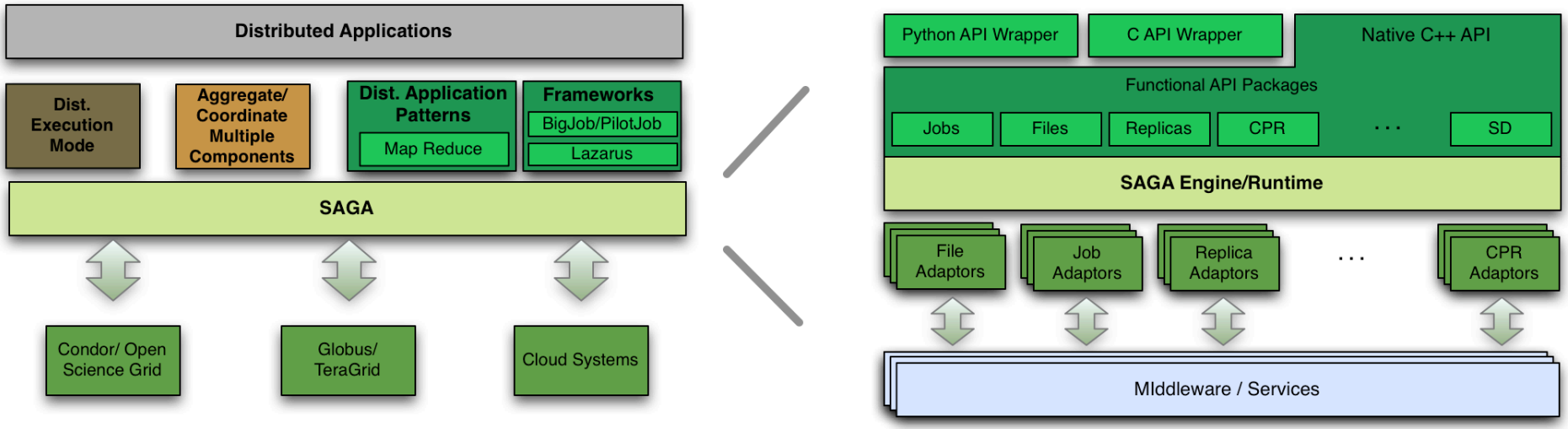
Application	*Interoperabilty*	*Scale-Out*	*Extensibility*	*Adaptivity*	*Simplicity*
SAGA-Montage	HPC-Condor-Clouds	Yes	FAUST	-	Yes
Replica-Exchange	Multiple TG nodes	Yes	Yes	Yes	Yes
Multi-Physics	Multiple HPC, Condor	Yes	Load-Balancing	Yes	Yes
Reservoir Simulation (EnKF)	Multiple TG-Condor-Cloud	Yes	Yes (BQP)	Yes	Yes
SAGA-MapReduce	TG-Clouds (ECP+EC2)	Yes	Yes	Yes (C2D)	Lower Performance

Table 3: **Fresh Perspective on Distributed Applications and CyberInfrastructure**

Interoperability *Ability to work across multiple distributed resources*
Scale-Out *Ability to use multiple distributed resources concurrently*
Extensibility *Support new patterns/abstractions, different programming systems and Infrastructure*
Adaptivity *Response to fluctuations in dynamic resources and availabilty of data*
Simplicity *Ease of above, at different levels and without sacrificing functionality or performance*

saga

SAGA: Bridging the Gap between Infrastructure and Applications



CENTER FOR COMPUTATION & TECHNOLOGY

SAGA for your Distributed Application?

e-Science Institute

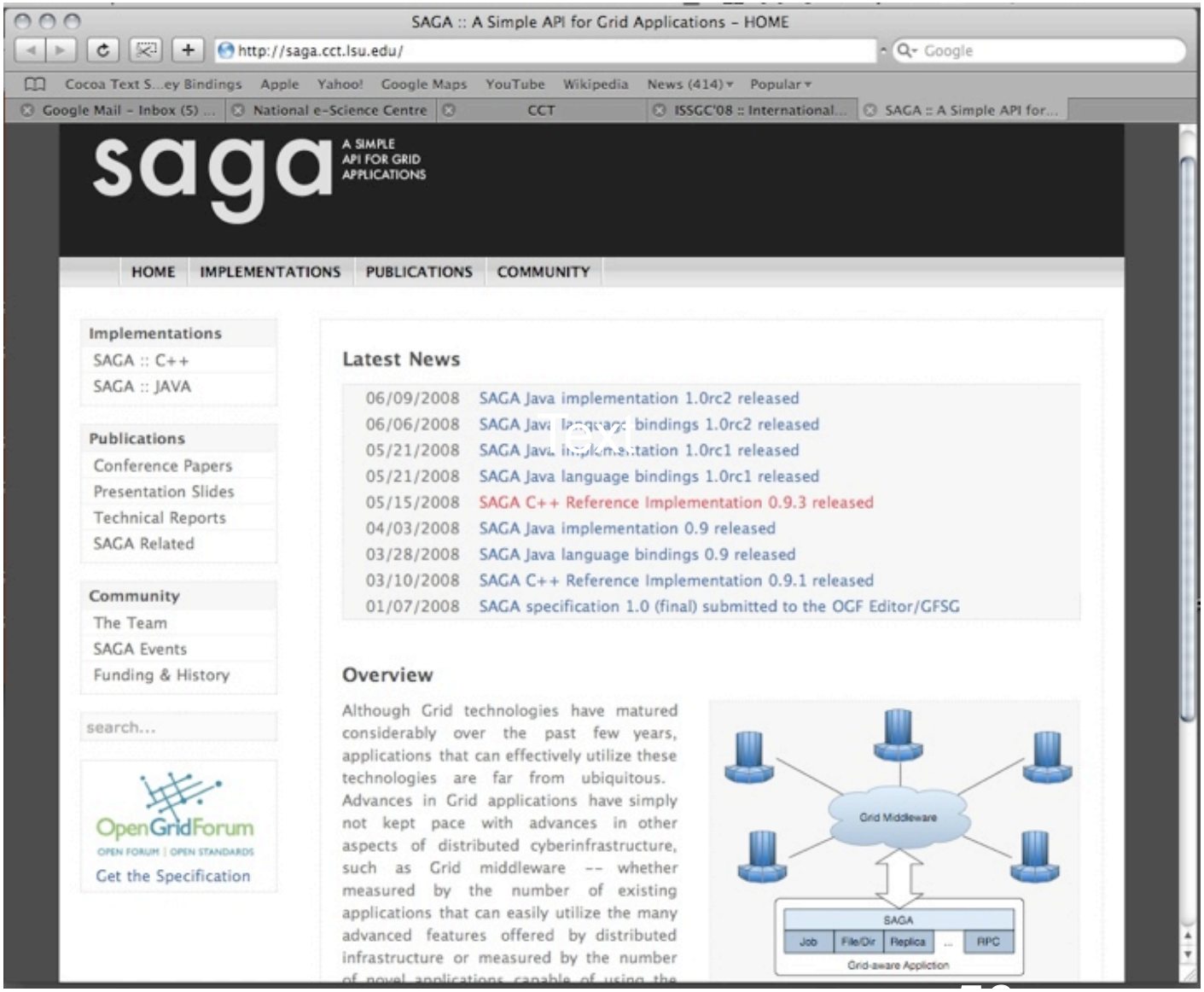
- Should you develop using SAGA or not? (As always) it depends:
  - Advantages:
    - Simple yet powerful API, which supports many:
      - Applications types, but not all kinds of applications
      - Programming patterns, models (eg superscalar)
      - Usage modes (eg parameter sweep, task-farming..)
    - Application are Infrastructure independent
    - Provides extensibility (eg Condor, fault-tolerance)
    - Interfaces well with other software (eg BQP)
  - Disadvantages:
    - Overhead (trivial IMHO); “script it *vs* just do it”
    - Deployment of SAGA is currently non-trivial
      - sagalite and work with RP's to ease deployment
    - Stable & Simple interface, but restricted functional scope

saga

CENTER FOR COMPUTATION
& TECHNOLOGY

http://saga.cct.lsu.edu

e-Science Institute



saga

CENTER FOR COMPUTATION & TECHNOLOGY

SAGA-based Tools and Projects

e-Science Institute

- DESHL
  - DEISA-based Shell and Workflow library
- JSAGA from IN2P3 (Lyon)
  - http://grid.in2p3.fr/jsaga/index.html
- GANGA/gLite
- XtreemOS
- NAREGI/KEK
- SD Specification
  - With gLite adaptors

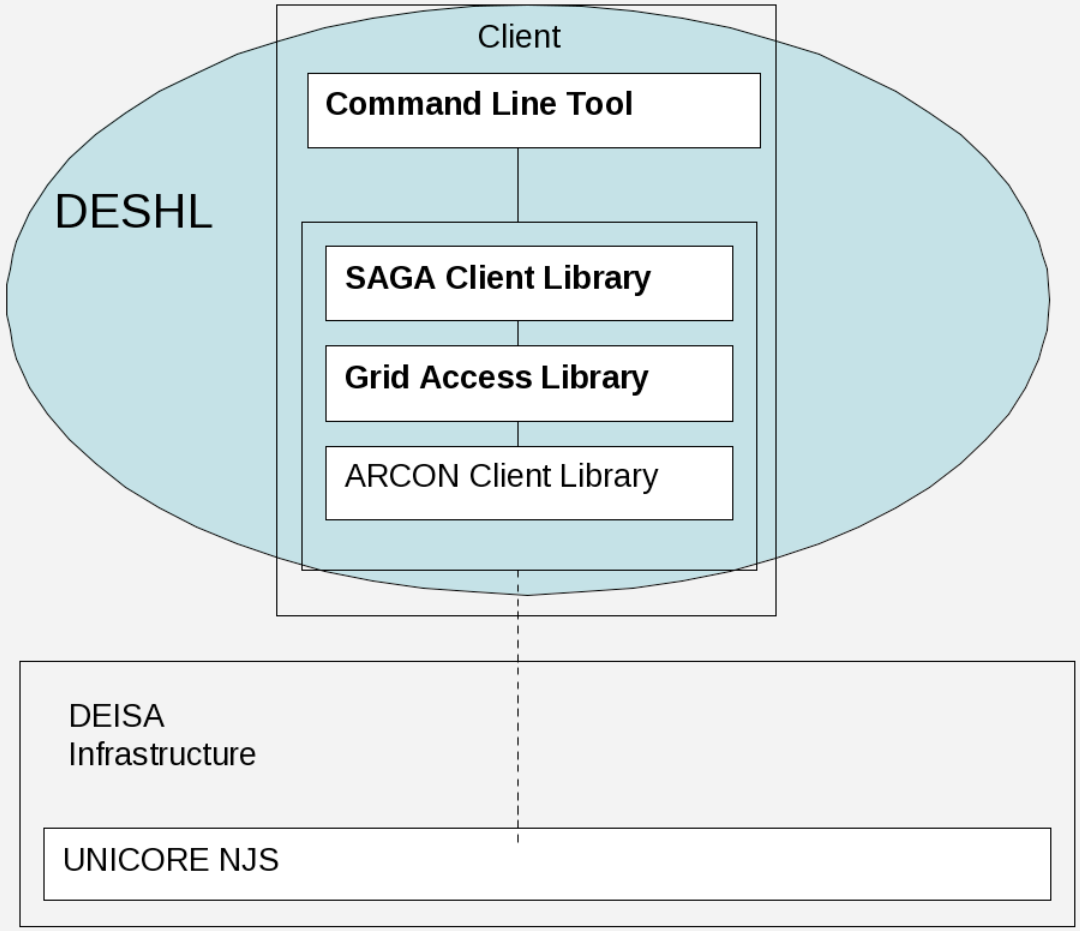
Advantage of Standards

saga

DESHL v4.0 Architecture

Distributed European Infrastructure for Supercomputing Applications

Layered Design sitting on top of existing UNICORE infrastructure



DESHL Client Overview

Distributed European Infrastructure for Supercomputing Applications

- Supplied as a Java command-line application
- Follows Open Grid Service Architecture (OGSA) specification for job batch submission and management
- Uses SAGA (Simple API For Grid Applications) directives for job specification
- Where appropriate, DESHL commands follow the Open Group Technical Standard for Batch Environment Systems
- Layered design to protect against future changes in underlying infrastructure
- Sits on top of existing UNICORE infrastructure

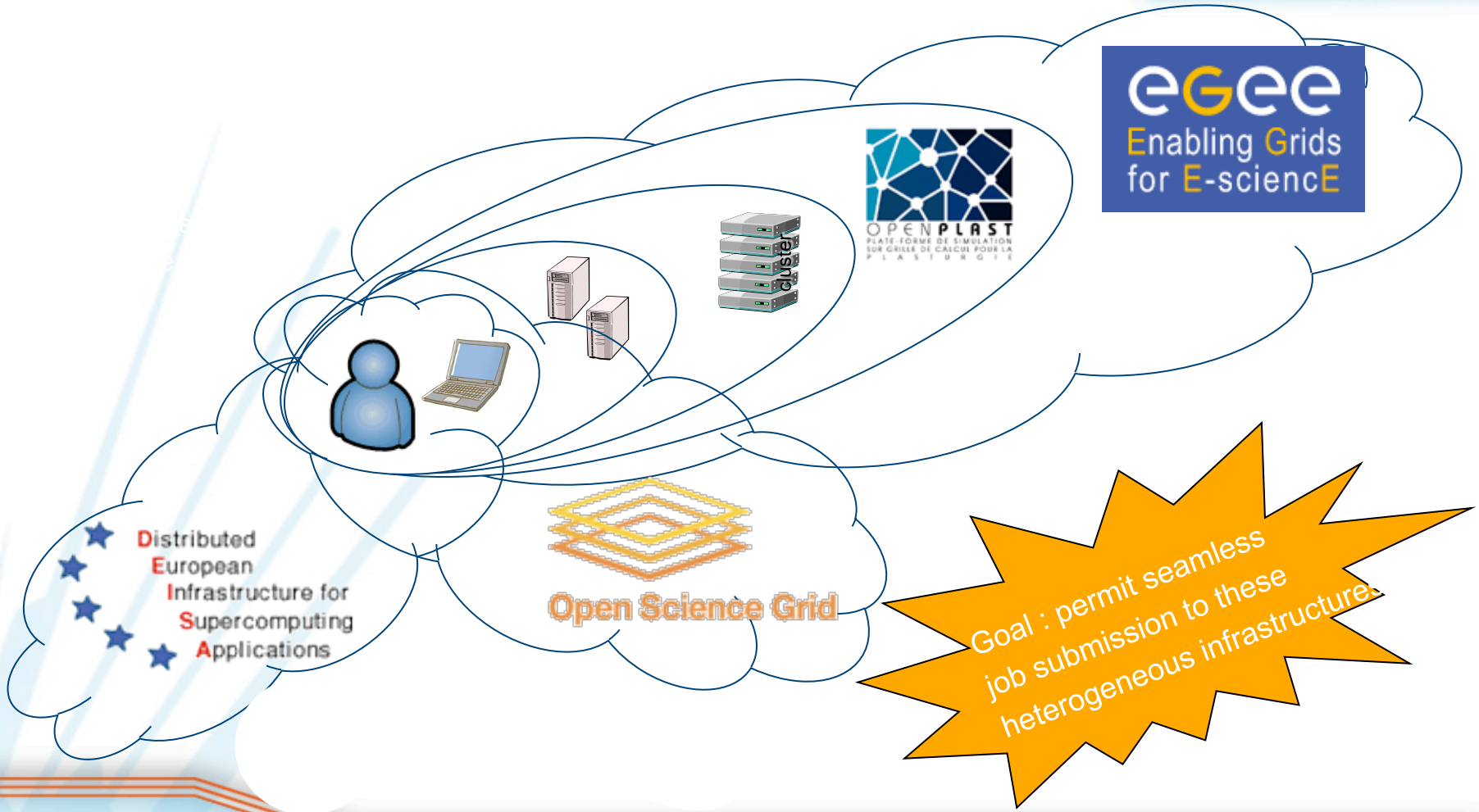
CENTER FOR COMPUTATION & TECHNOLOGY

JSAGA

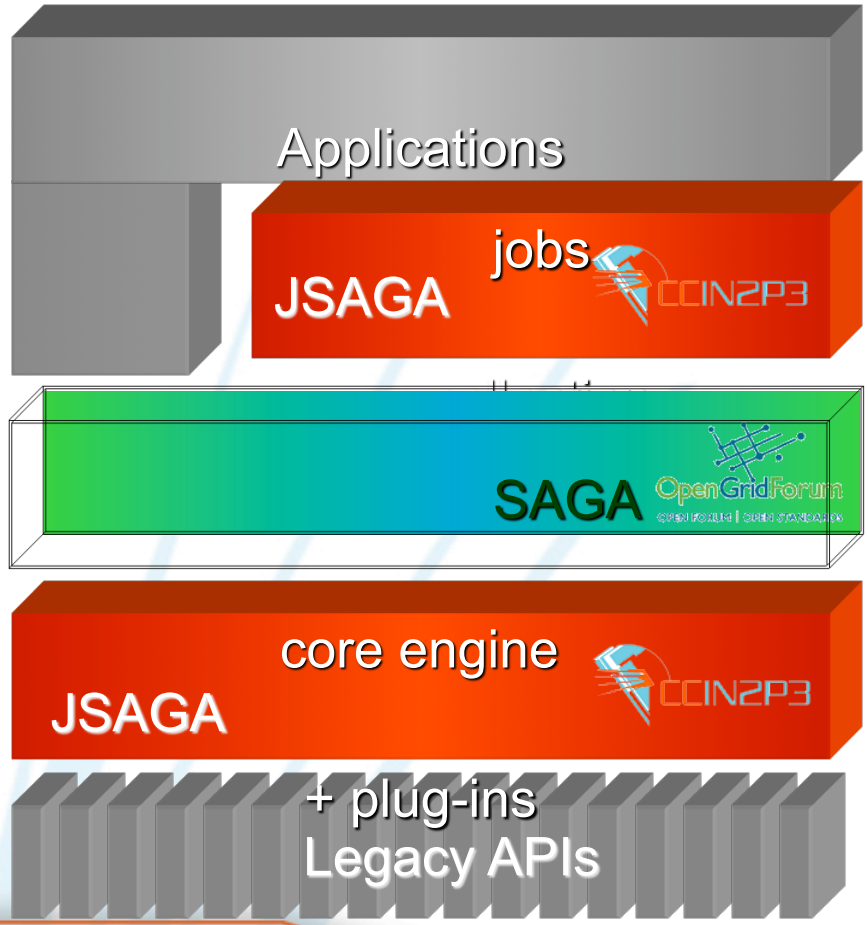
e-Science Institute

saga

JSAGA use-cases and goal



Both implementer and user of SAGA

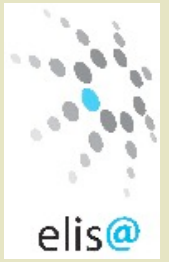


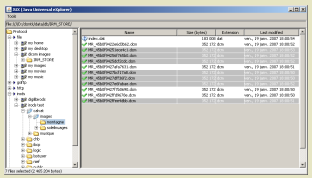
JSAGA **uses SAGA** in a module, which hides heterogeneity of **grid infrastructures**

JSAGA **implements SAGA** to hide heterogeneity of **middlewares**

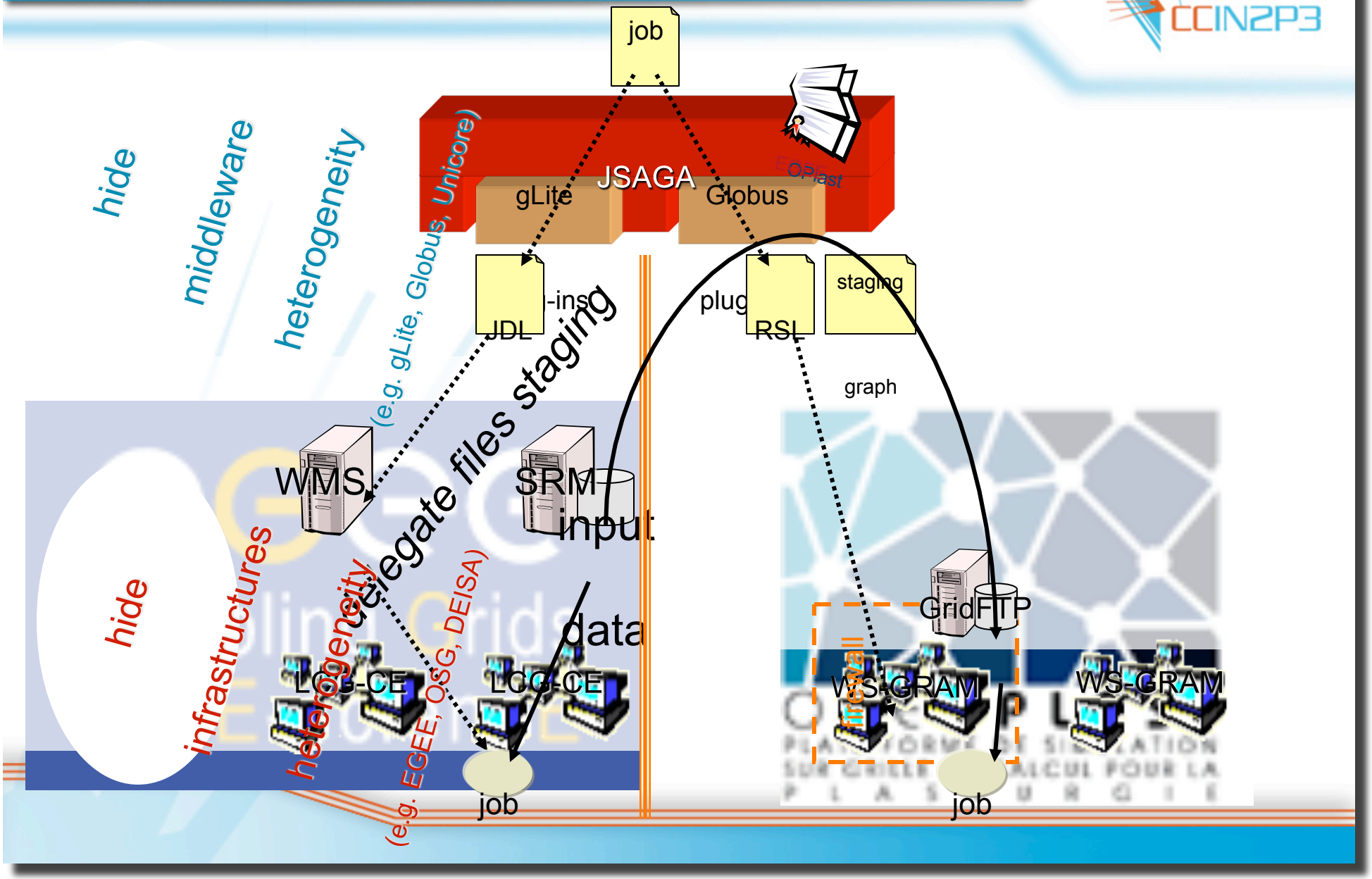
Projects using JSAGA

- Elis@
  - a web portal for submitting jobs to **industrial and research** grid infrastructures
- SimExplorer
  - a set of tools for managing **simulation experiments**
  - includes a workflow engine that submit jobs to heterogeneous distributed computing resources
- JJS
  - a tool for running efficiently **short-life jobs** on EGEE
- JUX
  - a **multi-protocols** file browser





m/w vs. infrastructure heterogeneity



CENTER FOR COMPUTATION & TECHNOLOGY

SAGA XtreemOS

e-Science Institute

- Challenges:
  - Linux applications should run with little (no) modifications
  - Grid applications should run with little (no) modifications
  - XtreemOS functionality must be provided to applications

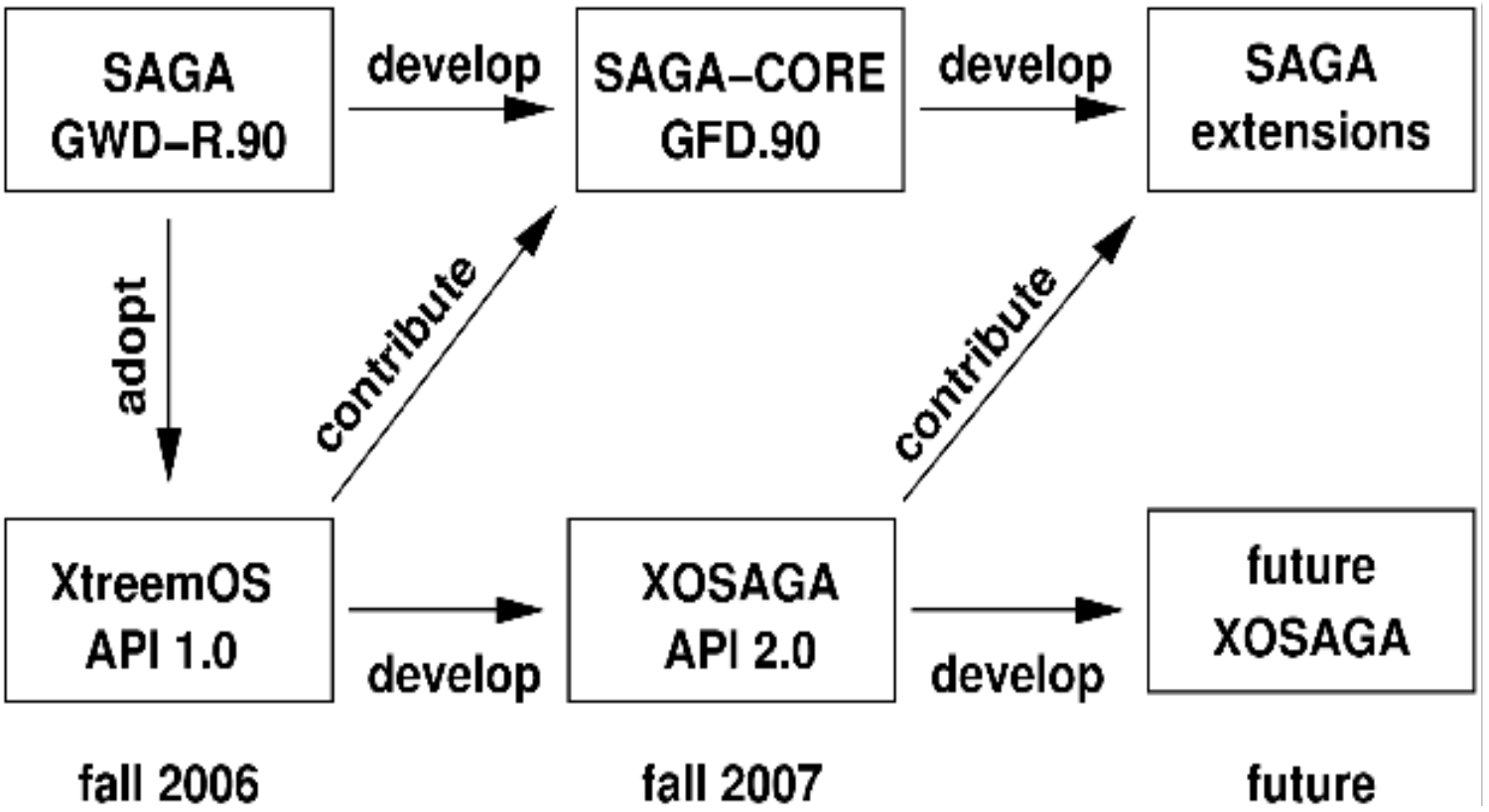
Approach:

- Use OGF-standardized SAGA API, enabling both Linux apps (via POSIX semantics) and grid apps
- Provide extensions packages to SAGA for XtreemOS functionality and in the process contribute to standards

CENTER FOR COMPUTATION & TECHNOLOGY

Co-evolution of SAGA Standard & the XtreemOS API (slide courtesy Thilo Kielmann)

e-Science Institute



An Extension to the SAGA Service Discovery API

Antony Wilson

(Steve Fisher and Paul Livesey)

4th EGEE Users Forum – Catania 2009

e-infrastructure

Service Discovery API

- **The specification for the Service Discovery has been finalised – GFD.144**
  - http://www.ogf.org/documents/GFD.144.pdf
  - Loosely based around the gLite Service Discovery
  - Uses the GLUE (version 1.3) model of a service
    - A Site may host many Services
    - *A* Service has multiple Service Data entries
    - Each Service Data entry is represented by a key and a value
- **APIs in C, C++, Java and Python complete but not releasd**
- **Adapter for gLite under development**
  - Based around GLUE 1.3
    - Once GLUE 2 is finalised the adapter will be updated

Service Discovery API

- **API allows selection based on three filters:**
  - serviceFilter – allows filtering on:
    - type, name, uid, site, url, implementor and relatedService
  - dataFilter – no predefined values
    - Uses keys from Service Data entries
  - authzFilter – authorization, no predefined values, useful values include:
    - vo, dn, group and role (values dependent on adapter)
  - NB if an authzFilter is not provided then one is automatically constructed from the users security context
    - The gLite adapter will provide the VOMS proxy credentials as the default value for the security context
- **Each of the filter strings uses SQL92 syntax**
- **The filters act as if they are part of a WHERE clause**

Service Discovery

- **Selection returns a list of ServiceDescriptions**
  - Each description contains:
    - type, name, uid, site, url, implementor, list of relatedServices and service data (key value pairs)
- **Example:**

```
discoverer = SDFactory.createDiscoverer( )
serviceDescriptions =
  discoverer.listServices("type = 'computing service'", "")
loop serviceDescriptions
  description.getAttribute("name")
    returns the value of the name attribute
```

  - **URL of information system can be passed in with the constructor, or obtained from a conf file**

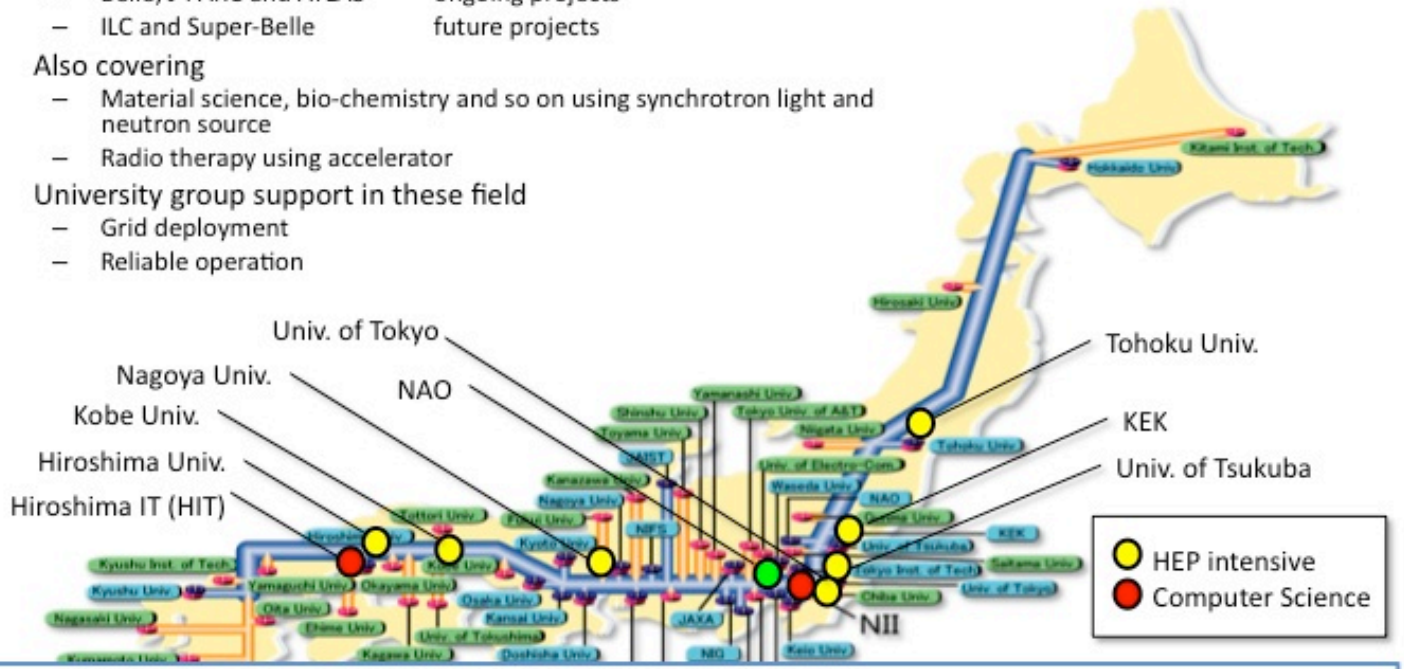
Extended Service Discovery API

- **Problem: The service discovery API only gives basic information as it cannot represent the GLUE model in three tables**
- **Purpose: To navigate the information model starting from a selected service**
- **API will be independent of the underlying information system**
- **Different information systems supported by means of adapters**
  - We will provide a gLite adapter
- **Navigation will be from entity to entity as expressed in the GLUE entity relationship models**

SAGA @ KEK

- KEK is the central laboratory for HEP in Japan
- Major HEP projects:
  - Belle, J-PARC and ATLAS ongoing projects
  - ILC and Super-Belle future projects
- Also covering
  - Material science, bio-chemistry and so on using synchrotron light and neutron source
  - Radio therapy using accelerator
- University group support in these field
  - Grid deployment
  - Reliable operation

SINET3: Production R&E network
10-40 Gbps national backbone
10 Gbps for NYC and LA



Coordination and implementation of infrastructures for particularly Japanese HEP institutes are KEK's responsibility.

Grid Deployment at KEK

- NAREGI middleware is being deployed as the general purpose e-science infrastructure in Japan
- Seamless user environment between the local resource and multiple grid environment should be provided.
  - Otherwise user have to create as many applications as middleware.

Matrix between experiment and middleware

	gLite	NAREGI	Gfarm	SRB	iRODS
Belle	Using	Planning	Using	Using	
Atlas	Using				
Radio therapy	Using	Developing	Planning		
ILC	Using	Planning	Planning		
J-PARC	Planning	Planning	Planning		Testing
Super-Belle	To be decided by 2010				

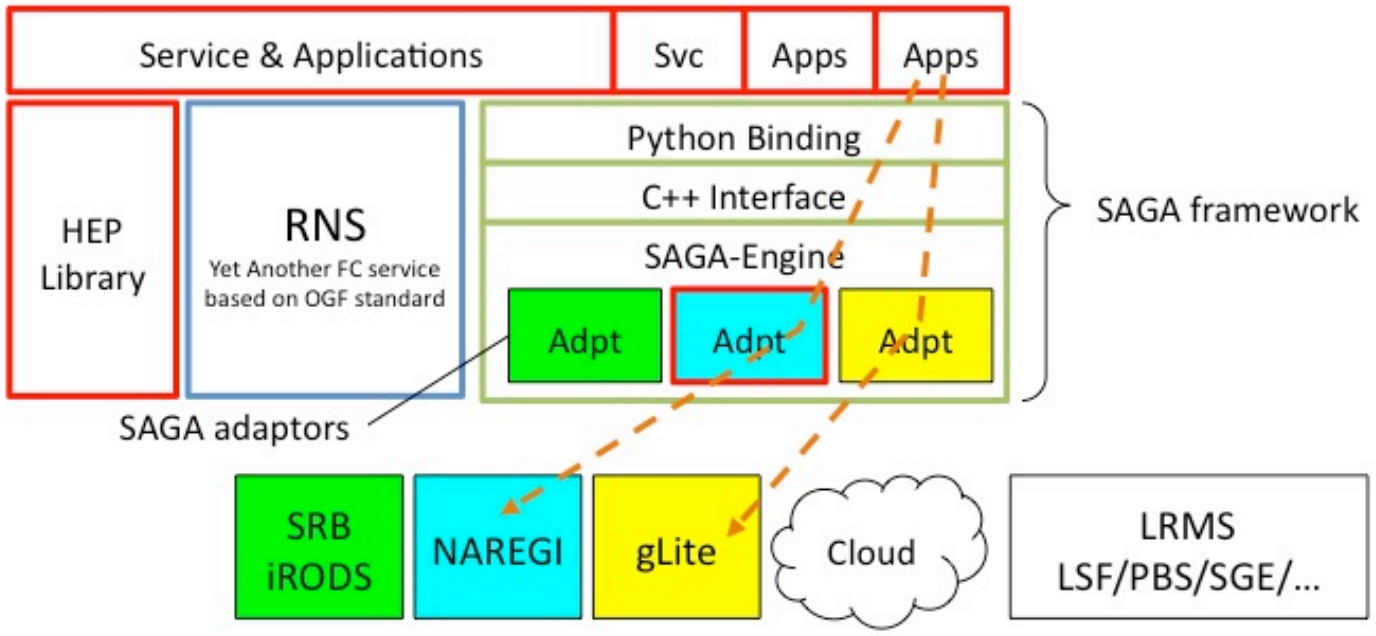
RENKEI Project Aims

Osaka Univ.
Tsukuba Univ.

KEK

SAGA

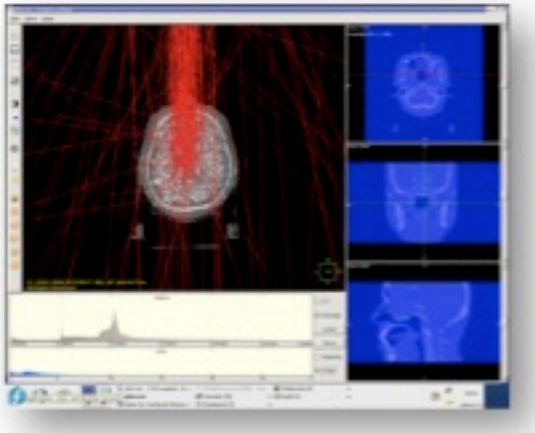
Middleware-independent service & application

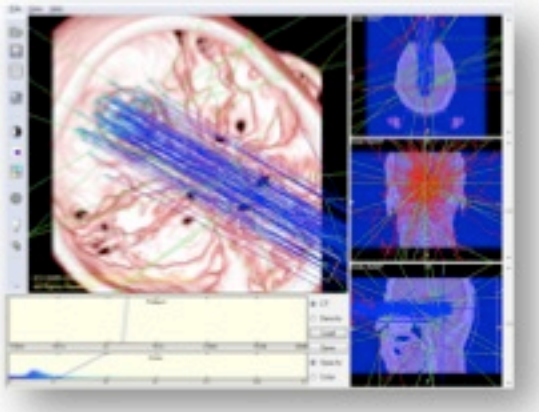


This activity is funded by MEXT as a part of RENKEI project which develops seamless linkage of resources in the Grids and the local one for e-Science.

Practical Examples in Testbed

- Grid environment
  - MW: NAREGI v1.1 released in October 2008
  - VO scale: KEK, NAO, HIT, and NII
- SAGA adaptors:
  - NAREGI adaptor for job completed in April 2009
  - Torque adaptor completed in June 2009
- Demonstration in testbed
  - Particle therapy simulation based on Geant4 as the 1st practical example
  - Resource scale
    - 3 sites: KEK, NAO, HIT
    - CPU: 10 cores
    - OS: CentOS 5.2 x86_64
    - Memory: 2 GB each





Well done!

More application-wise development in 2009

Acknowledgements

Funding Agencies:

UK EPSRC: DPA, OMII-UK , OMII-UK PAL

US NSF: Cybertools, HPCOPS (TeraGrid), GridChem

NIH: LaCATS, INBRE-LBRN

CCT Internal Resources

People:

SAGA D&D: Hartmut Kaiser, Ole Weidner, Andre Merzky, Joohyun Kim, Lukasz Lacinski, João Abecasis, Chris Miceli, Bety Rodriguez-Milla

SAGA Users: Andre Luckow, Yaakoub el-Khamra, Kate Stamou, Cybertools (Abhinav Thota, Jeff, N. Kim), Owain Kenway

Google SoC: Michael Miceli, Saurabh Sehgal, Miklos Erdelyi

Collaborators and Contributors: Steve Fisher & Group, Sylvain Renaud (JSAGA), Go Iwai & Yoshiyuki Watase (KEK)

Some Relevant Links

- http://saga.cct.lsu.edu
- http://www.cct.lsu.edu/~sjha/select_publications
- http://saga.cct.lsu.edu/projects

  (out of date; to be updated)
- TeraGrid-DEISA Interoperabilty Project
  - http://www.teragridforum.org/mediawiki/index.php?title=LONI-TeraGrid-DEISA_Interoperabilty_Project