

Application-Level Interoperability for Grids and Clouds

Shantenu Jha¹²³, Andre Merzky¹

¹CCT, Louisiana State University

²e-Science Institute, Edinburgh

³UC-London

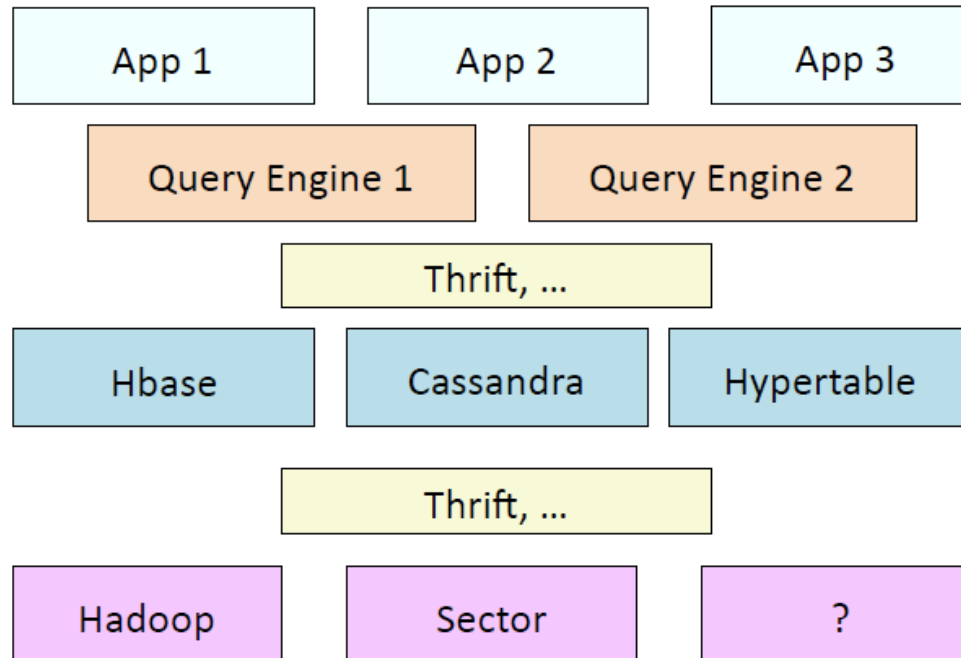
Work done in collaboration with Geoffrey Fox, Kate Stamou, Chris
and Michael Miceli, Hartmut Kaiser

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

<http://saga.cct.lsu.edu/papers>

[http://wiki.esi.ac.uk/Distributed Programming Abstractions](http://wiki.esi.ac.uk/Distributed_Programming_Abstractions)

Initial Interoperability Focus



Courtesy: Rob Grossman

Decouple from vertical stack

Decouple from details of resource provisioning (cf: Walker)

Couple horizontal stacks

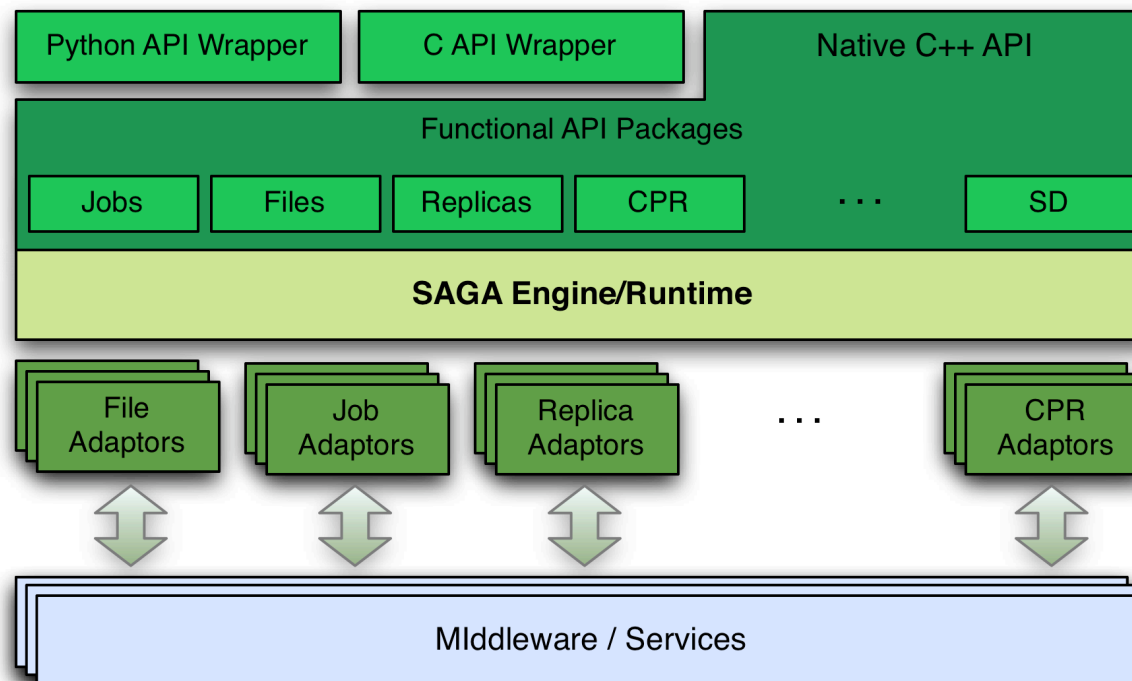
Application-level Interoperability

Cloud-Cloud; Cloud-Grid

- Application-level vs. System-level Interoperability
- ALL: Infrastructure Independence is Pre-requisite
 - Programming Model should be Infrastructure/SLA agnostic
 - Same availability zone?; data-transfer cost vs. no cost
 - Same application, priced differently, for same performance
 - Same application, priced same, for different performance
- Seamless transition of existing program/application
- Grids AND Clouds:
 - Hybrid & Heterogeneous workload: data-compute affinity differ
 - Complex data-flow dependency: need runtime determine

How can scientific applications be developed to utilize a broad range of DS, without vendor lock-in or disruption, yet with the flexibility and performance that scientific applications demand?

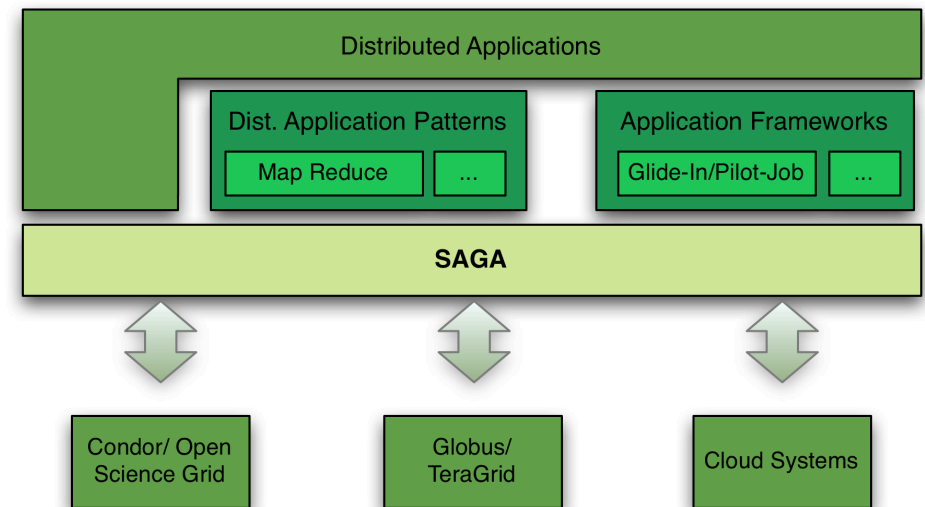
SAGA: In a Nutshell



SAGA provides an infrastructure independent, application-level interface to create distributed applications

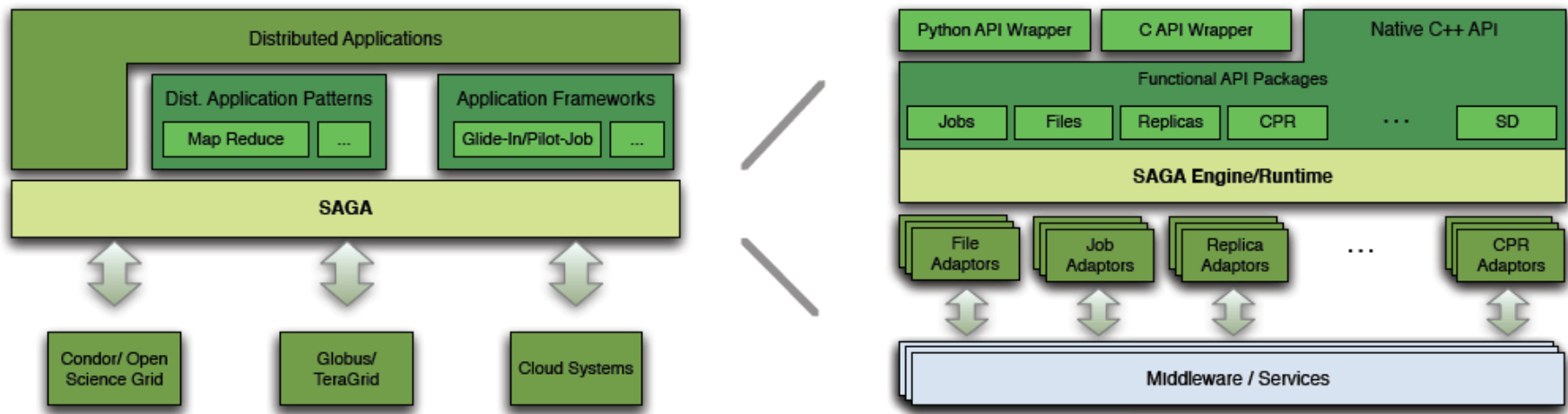
SAGA and Distributed Applications

- *Legacy* application -- distributed execution modes
 - Replica-Exchange MD
 - *Novel* first-principles applications
- Distributed application using patterns
 - MapReduce
- Applications using Frameworks
 - Frameworks can use patterns



The theoretical underpinnings of developing applications using SAGA are strongly influenced by the DPA theme (eSI, Edinburgh)

SAGA: Unified View



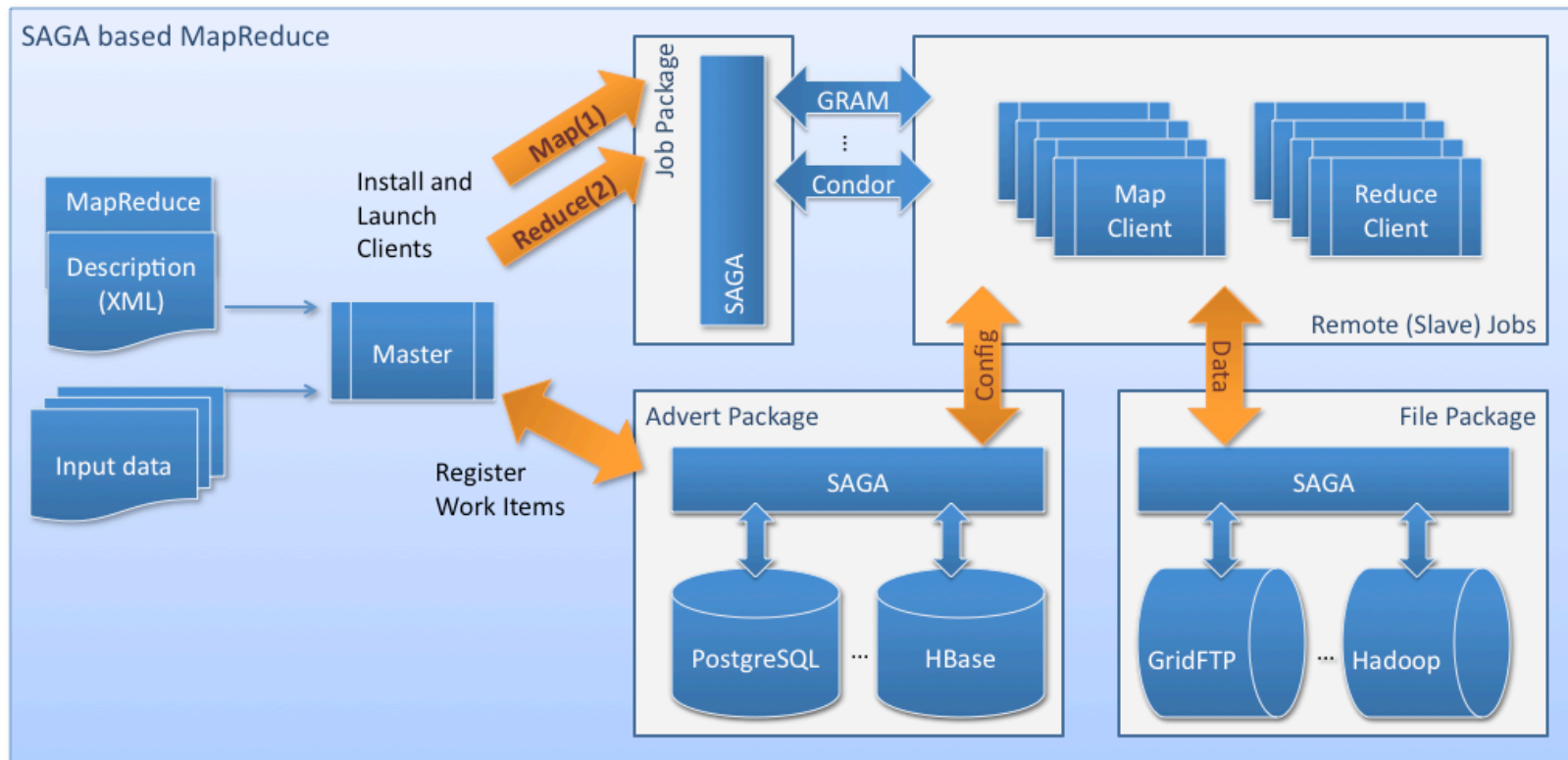
Focus on Application Development and Characteristics, not infrastructure details

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-MapReduce: Control + Architecture



Still using Master-Worker

Workers are SAGA apps which get coordinated via AS

Control the chunk-size, # of workers, placements...

Experiments

Use SAGA-MapReduce concurrently and cooperatively towards the solution of the same problem instance

- Establish Infrastructure Independence:
 - Using HDFS, BigTable, KFS, Hbase,
- Provide control over relative data-compute location
- Grids vs. Clouds: Interop and Performance Issues
 - Vary number of workers (1-10) and dataset size (0.1-10GB)
 - Vary number of workers per VM (1:2, 1:4)
 - Distributed workers across different clouds (ECP & EC2)
 - Distribute workers on TeraGrid, ECP and EC2 (1:1)

SAGA: Role of Adaptors (1)

Use over different infrastructure

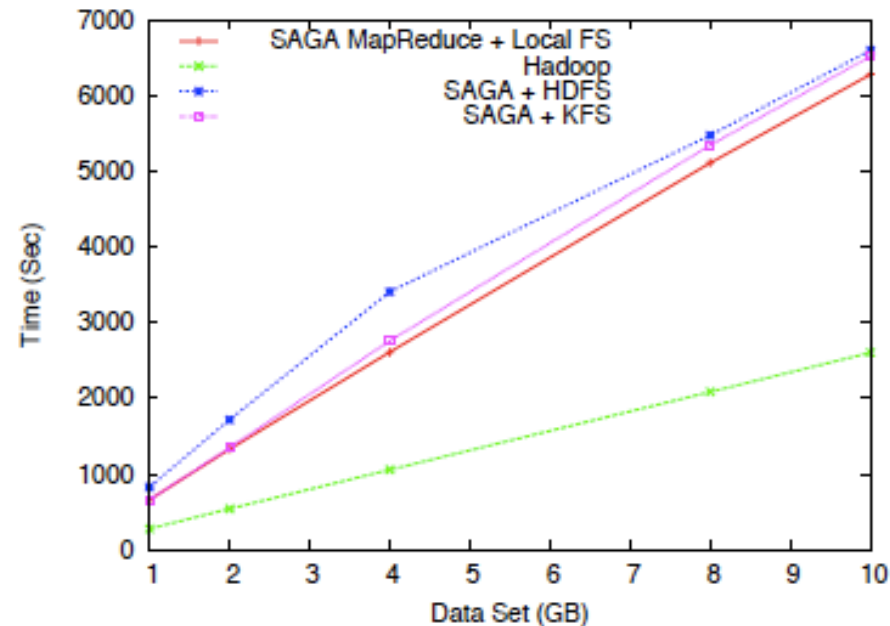


Fig. 5: T_c for SAGA-MapReduce using one worker (local to the master) for different configurations. The label “Hadoop” represents Yahoo’s MapReduce implementation; T_c for Hadoop is without chunking, which takes several hundred sec for larger data-sets. The “SAGA MapReduce + Local FS” corresponds to 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 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
```

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.

Interoperability

TG	Number-of-Workers		Size (MB)	T_s (sec)	T_{spawn} (sec)	$T_s - T_{spawn}$ (sec)
	AWS	Eucalyptus				
-	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: Extension to Clouds

- So is the API so 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

Grid vs. Clouds: Simple Performance

- Grid: Deploy (i) SAGA on Grid, (ii) app binary on Grid, (iii) run
- EC2/ECP: (i) Provision VM instance, (ii) deploy SAGA on instance, (iii) move app binary on instance, (iv) run
- VM startup takes ~45secs (Eucalyptus) and ~100secs (EC2)
 - Size of VM influences start up time, but within fluctuation
 - 1-10sec to assign job to VM
 - VM can be reused for any number of jobs
 - VM can be persistent
 - Vary number of workers assigned to VM
- Switching between Eucalyptus and EC2
 - Job service URL; Adapter configuration; Environment variables

Final Thoughts..

- Its not Grids vs. Clouds:
 - Its about (scientific) Applications, HLA & Usage modes right application level interfaces, HLA, support for usage modes, then Clouds vs. Grids is *less critical*
 - Interop of SAGA-MapReduce shows how – once correct abstractions and interfaces are in place -- much of work in Distributed Computing easily translates to managing different back-end (role of adaptors)
 - “*We did such a good job of understanding parallel programming in the 90’s that we are very well prepared for multi-core..*” (Ken Kennedy)
 - There are hard problems in distributed applications/ computing that remain, even if we change names..

Acknowledgements

- Hartmut Kaiser, Joohyun Kim
- Yaakoub el-Khamra, Andre Luckow
- Kate Stamou, Ole Weidner
- Chris and Michael Miceli [funded by GSOC]
- UK EPSRC: eSI Theme “Distributed Programming Abstractions” & OMII-UK “OMII SAGA” project
- US NSF: Cybertools Project