RUTGERS

RUTGERS

P*: A Model of Pilot-Abstractions

Andre Luckow, Mark Santcroos, Ole Weidner, Andre Merzky, Pradeep Mantha, Shantenu Jha

IEEE eScience 2012
Chicago, 11 October 2012

RUTGERS

ACM HPDC 2012

Towards a Common Model for Pilot-Jobs

Andre Luckow, Ole Weidner,
Andre Merzky, Sharath Maddineni
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA
{aluckow, oweidern, amerzky,
smaddini}@cct.lsu.edu

Mark Santcroos
Bioinformatics Laboratory
Academic Medical Center
University of Amsterdam
Amsterdam, The Netherlands
m.a.santcroos@amc.uva.nl

Shantenu Jha
CAC/ECE
Rutgers University
Piscataway, NJ
shantenu.jha@rutgers.edu

RUTGERS

IEEE eScience 2012

P*: A Model of Pilot-Abstractions

Andre Luckow[1], Mark Santcroos[2,1], Ole Weidner[1], Andre Merzky[1], Pradeep Mantha[1], Shantenu Jha[3,1*]

[1] *Center for Computation & Technology, Louisiana State University, USA*
[2] *Bioinformatics Laboratory, Academic Medical Center, University of Amsterdam, The Netherlands*
[3] *Rutgers University, Piscataway, NJ 08854, USA*
[*] *Contact Author:* shantenu.jha@rutgers.edu

RUTGERS

NECS12 Workshop

Pilot Abstractions for Compute, Data, and Network

Mark Santcroos
Academic Medical Center
University of Amsterdam
The Netherlands
m.a.santcroos@amc.uva.nl

Silvia Delgado Olabarriaga
Academic Medical Center
University of Amsterdam
The Netherlands
s.d.olabarriaga@amc.uva.nl

Daniel S. Katz
University of Chicago &
Argonne National Laboratory
USA
d.katz@ieee.org

Shantenu Jha
Rutgers University
USA
shantenu.jha@rutgers.edu

Landscape of DCI and Applications: Empirical Assertions

- DCI is complex: Heterogeneous software, access-layers, policy..
- The space of possible Distributed Applications (DA) is large (and rich), but the number of effective and extensible DA small
- There are missing abstractions and poor implementations
  - Many local solutions, lack of end-to-solutions, especially tools
  - Missing conceptual frameworks and systems approach to tools
- Accept, if not embrace “distributedness”
  - Manage it, but also exploit it
  - The ability to reason: distributed performance, decomposition or aggregation!

Context

- Scientific computing applications demand (computer) resources to execute

Application

- Can be a simple script calling multiple instances of a program
- Can be a complex workflow management system that dynamically executes tasks based on a workflow description
- Resource requirements may vary over time
  - Internal dynamics of an application
  - Different solvers
  - Adaptive algorithms

Compute Resources

- Heterogeneous
  - Local clusters
  - High Throughput Computing Grids
  - High Performance Computing Clusters
  - Desktop Grids
  - IaaS (clouds)
- Capacity is fluctuating
  - Growing, shrinking
  - Changing in load and capabilities

Storage / Data Resources

- Local storage systems / data sources
- Different protocols / access mechanisms
- Distributed experiment data
- Data may be replicated
- Persistent / volatile
- Dynamic

Decoupling

Applications that are capable of using tools, abstractions and services that decouple payload management and resource assignment have been more successful at efficiently utilizing distributed resources.

Pilot Jobs

- A Pilot-Job is an abstraction that uses a placeholder job as a container for a set of compute tasks
- An instance of that placeholder job is commonly referred to as Pilot-Job or pilot

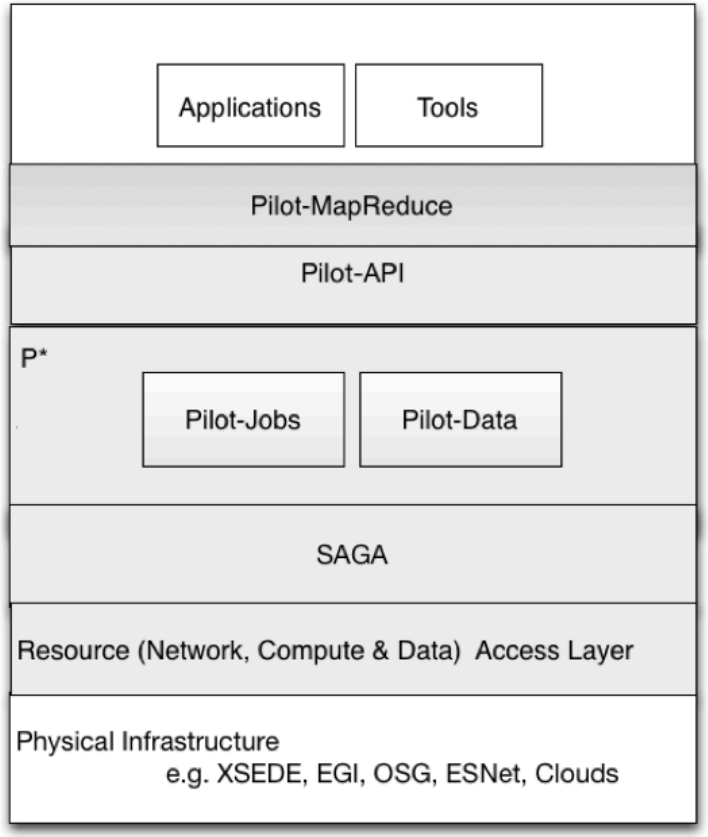
Pick your flavor …

- DIANE
- DIRAC
- CONDOR Glide-In
- Swift Coasters
- ToPoS
- Falkon
- PanDA
- Nimrod/G
- BigJob
- …

P*: a model

- A model for Pilot Abstraction
- Minimal but Complete
- Establish terminology
- Provide a framework for comparison

Positioning of P*



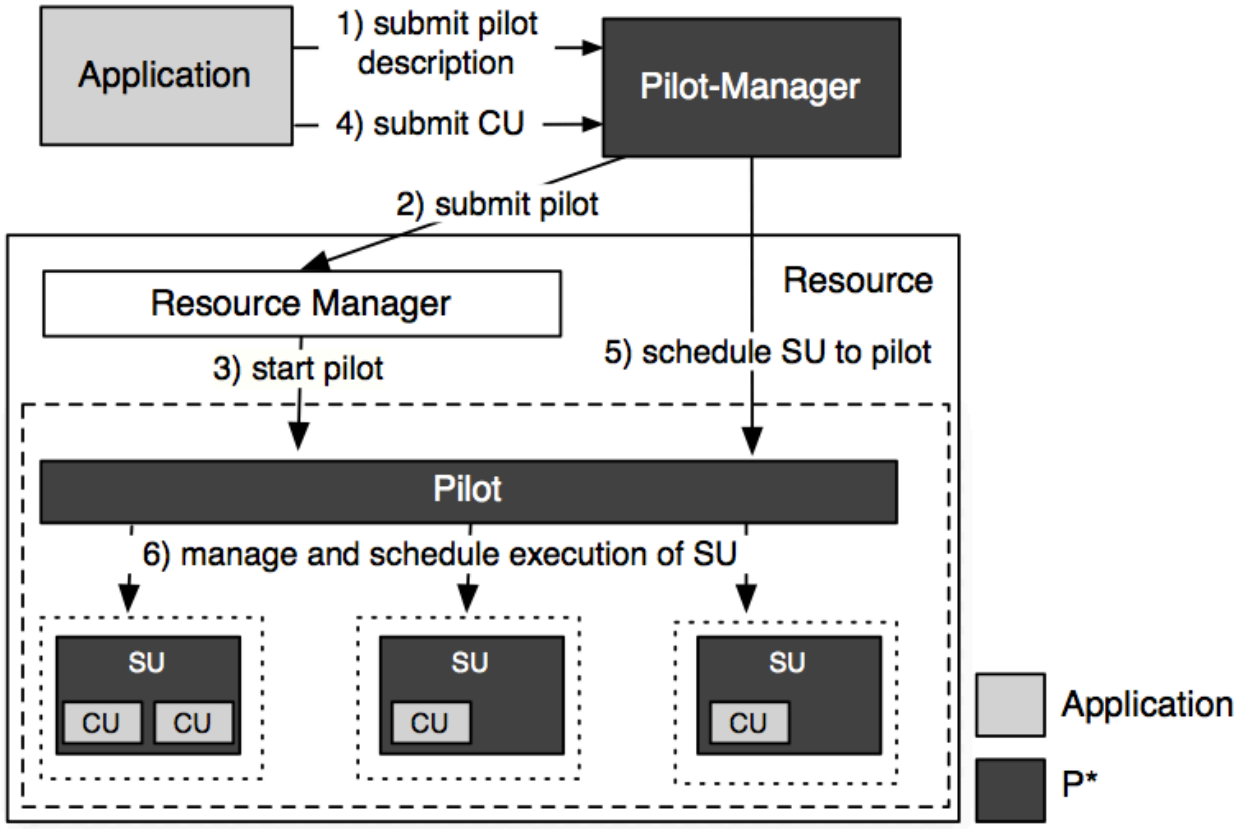
Pilot-Abstraction for Dynamic Distributed Data

- Similar heterogeneity for data infrastructure
- Application level capabilities on logical level and not on file level
- Fundamentally task placement is independent of data placement
- Dynamic decision for data
- Other considerations (varying data sources, data rates, etc)

Elements of the P* model

- Pilot-Compute (PC)
- Pilot-Data (PD)
- Compute Unit (CU)
- Data Unit (DU)
- Scheduling Unit (SU)
- Pilot-Manager (PM)

P* Model of Pilot Abstractions

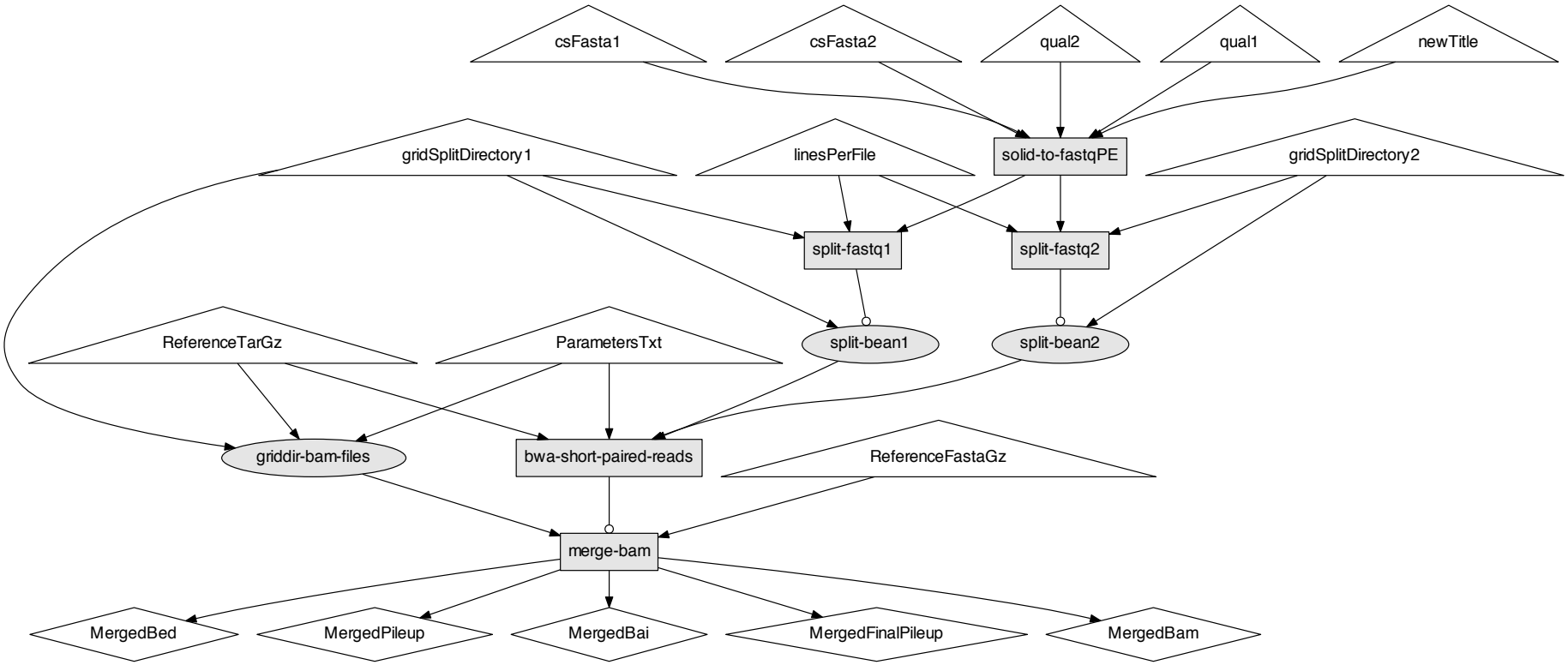


Pilot-API

- Python implementation of P* model
- Exhibits the decoupling of Workload and Resources

RUTGERS

NGS Workflow



Element: Pilot Compute(PC)

- The placeholder entity that gets submitted to a resource
- Often has the role of an agent:
  - collects information
  - manages the resources allocated
  - exchanges data
- Executes application code

PilotCompute Creation

```
# Instantiate a PilotComputeService running on localhost
pilot_compute_service =
                          PilotComputeService('localhost')

# Define a PilotComputeDescription
pilot_compute_desc = {
      'service_url': 'cream://gb-ce-amc.amc.nl/
                      cream-pbs-medium'
}

# Create a PilotCompute through the PilotComputeService
pilot_compute_service.create_pilot(pilot_compute_desc)
```

Element: Pilot Data(PD)

- The placeholder entity that represents a storage resource (reservation)
- Can have the role of an agent:
  - collects information
  - manages the resources allocated
- Physically stores the data

PilotData Creation

```
# Instantiate a PilotDataService
pilot_data_service = PilotDataService()

# Define a PilotDataDescription
pilot_data_description = {
      'service_url': 'srm://tbn18.nikhef.nl/
                      home/mark/pilotdata/'
}

# Create a PilotData
pilot_data_service.create_pilot(pilot_data_description)
```

Element: Compute Unit (CU)

- Is defined by the application
- Encapsulates a self-contained piece of work that is submitted to the PJ framework
- E.g.:
  - task, job, rpc, web service call, etc.

ComputeUnit

```
# Define ComputeUnitDescription for solid-to-fastqPE
compute_unit_description = {
   'executable':  'solid-to-fastqPE',
   'arguments':   [<...APPLICATION_ARGUMENTS...>],
   'input_data':  [solid_du],
   'output_data': [fwd_fastq_du, rev_fastq_du],
}

# Submitting solid-to-fastqPE ComputeUnit
cd_service.submit_compute_unit
                          (compute_unit_description)
```

Element: Data Unit (DU)

- Is defined by the application
- Encapsulates a self-contained piece of logical data that is submitted to the PJ framework
- E.g.:
  - file, chunk, database, etc.

Input DataUnit

```
# Define DataUnitDescription for  short read files
solid_dud = {'file_urls': [
      'file:///Test42/Test42_F3.csfasta.bz2'),
      'file:///Test42/Test42_F3_QV.qual.bz2'),
      'file:///Test42/Test42_R3.csfasta.bz2'),
      'file:///Test42/Test42_R3_QV.qual.bz2')]
}

# Submit the DataUnit, this will initiate a transfer.
solid_du = cd_service.submit_data_unit(solid_dud)
```

Output DataUnit

```
# DataUnit for forward fastq output file
data_unit_description = {
        'file_urls': ['Test42_fwd_read.fastq.gz']
}

fwd_fastq_du = DataUnit(data_unit_description)
```

Element: Scheduling Unit (SU)

- Represents a DataUnit / ComputeUnit internal to the runtime system
- Allows for aggregation and division of application level payload

Mapping

P* Element	BigJob	DIANE	Condor-G/ Glide-in
Pilot-Manager	BigJob Manager	RunMaster	condor_master condor_collector condor_negotiator condor_schedd
Pilot	BigJob Agent	Worker Agent	condor_master condor_startd
Compute Unit (CU)	Task	Task	Job
Scheduling Unit (SU)	Sub-Job	Task	Job

Three services

- PilotComputeService
  - Manages a pool of compute resources
- PilotDataService
  - Manages a pool of storage resources
- ComputeDataService
  - Manages the application's workload

PilotComputeService

- **PilotComputeService** is responsible for creating and managing the PilotComputes
- It is the application's interface to the Pilot-Manager in the P* Model
- Created from **PilotComputeDescription**
- Creates a **PilotCompute**

PilotDataService

- **PilotDataService** is responsible for creating and managing the PilotDatas
- It is the application's interface to the Pilot-Manager in the P* Model
- Created from **PilotDataDescription**
- Creates a **PilotData**

ComputeDataService

- The **ComputeUnitDescription** is a task description based on SAGA Job Description
- It offers the application to describe a **ComputeUnit** in an abstract way that is dealt with by the Pilot-Manager

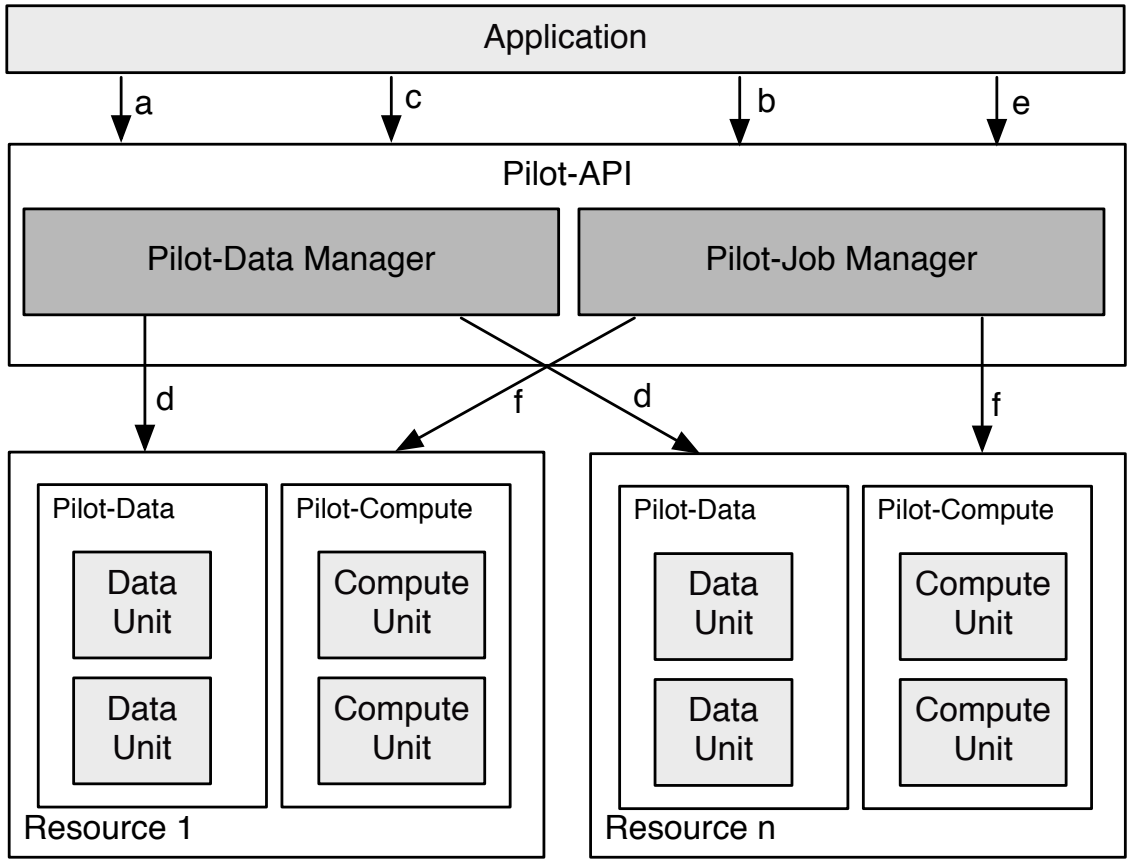
ComputeDataService

```
# Instantiate a ComputeDataService
cd_service = ComputeDataService()

# Connect PilotComputeService to ComputeDataService
cd_service.add_pilot_compute_service
                                    (pilot_compute_service)

# Connect PilotDataService to ComputeDataService
cd_service.add_pilot_data_service(pilot_data_service)

# Submitting solid-to-fastqPE ComputeUnit
cd_service.submit_compute_unit
                                (compute_unit_description)
```

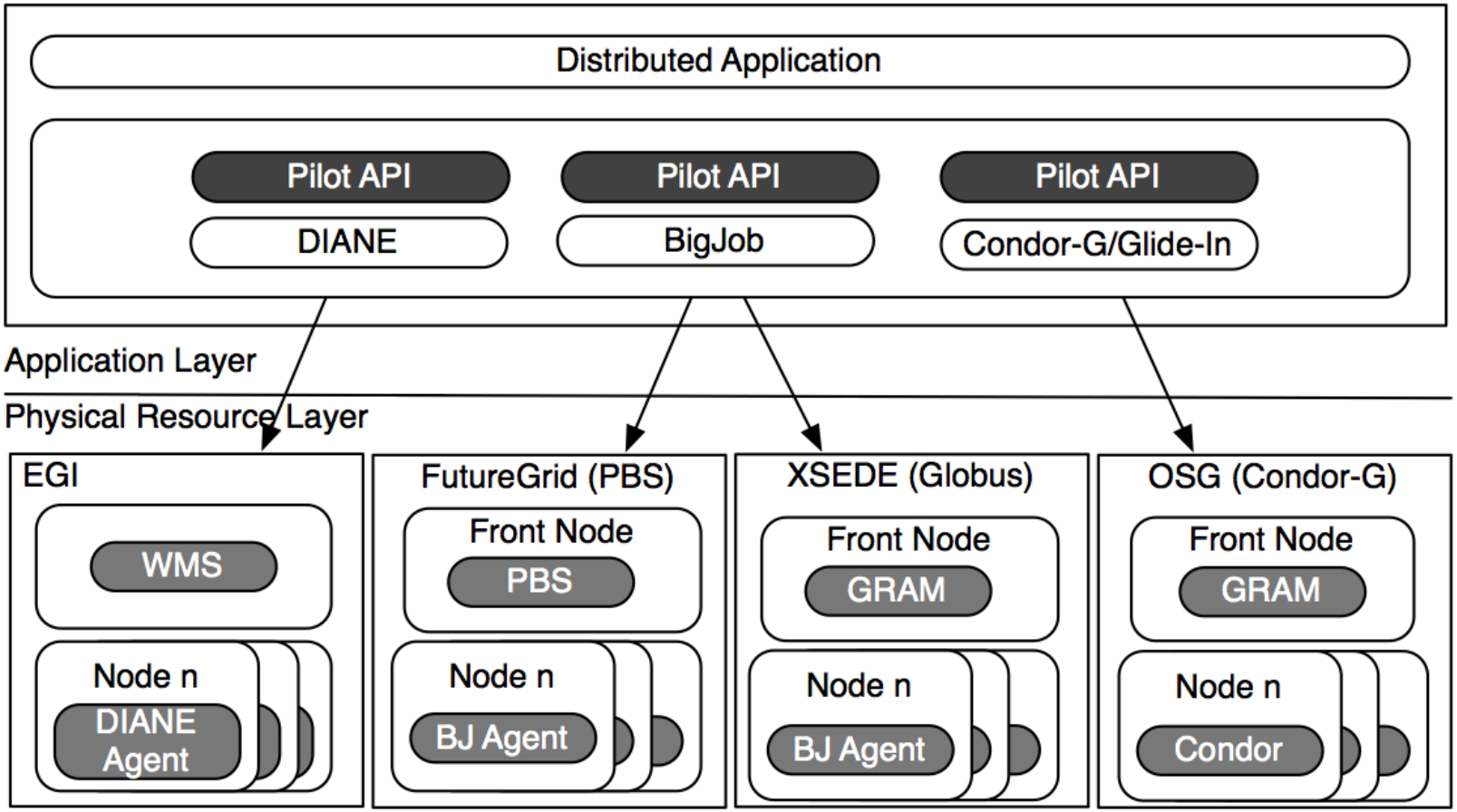
Characteristics and other properties

Properties	**BigJob**	**DIANE**	**Condor-G/ Glide-in**
Coordination	M/W	M/W	M/W
Communication	Advert Service	CORBA	TCP
Scheduling	FIFO, custom	FIFO, custom	Matchmaking, priority-based scheduler
Agent Submission	API	GANGA Submission Script	Condor CLI
End User Environment	API	API and M/W Framework	CLI Tools
Fault Tolerance	Error propagation	Error propagation, retries	Error propagation, retries
Resource Abstraction	SAGA	GANGA/ SAGA	Globus
Security	Multiple (GSI, User/-Pass.)	Multiple (GSI)	Multiple (GSI, Kerberos)

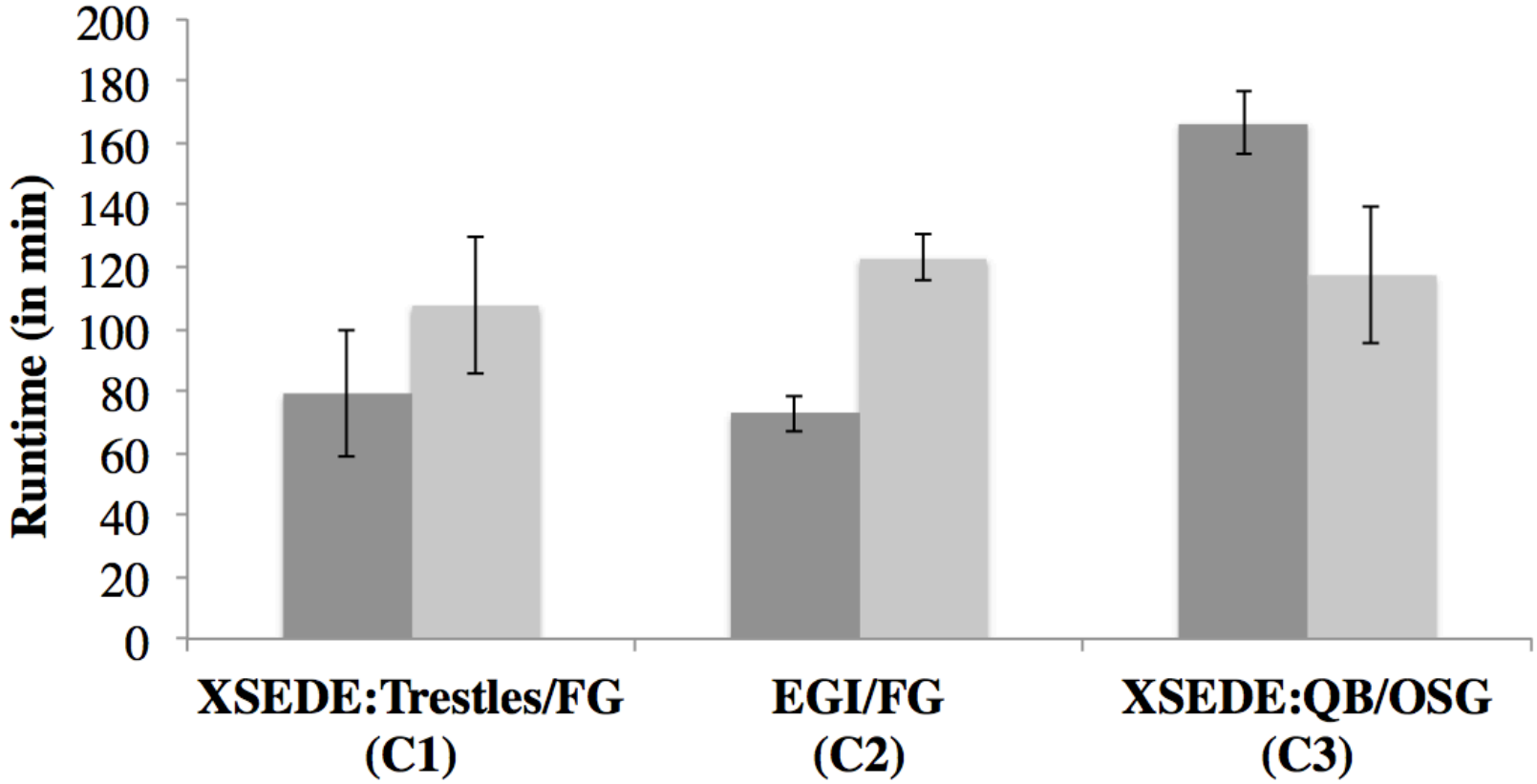
> 1000 words


Application
a
c
b
e
Pilot-API
Pilot-Data Manager
Pilot-Job Manager
d
f
d
f
Pilot-Data
Data Unit
Data Unit
Pilot-Compute
Compute Unit
Compute Unit
Resource 1
Pilot-Data
Data Unit
Data Unit
Pilot-Compute
Compute Unit
Compute Unit
Resource n

Pilot-API and PJ frameworks



PJ Framework Interoperability



Conclusions

- Established P* as an abstraction for supporting dynamic execution
- Common framework for comparisons
- Well defined Pilot-API for application developers

Discussion

- Applicability to workflow paradigm
- Extension with networking
- Clouds are an important and useful development but not a panacea

RUTGERS

Acknowledgements

This work is funded by NSF CHE-1125332 (Cyber-enabled Discovery and Innovation), HPCOPS NSF-OCI 0710874 award, NSF-ExTENCI (OCI-1007115) and NIH Grant Number P20RR016456 from the NIH National Center For Research Resources

SJ acknowledges the e-Science Institute, Edinburgh for supporting the research theme. “Distributed Programming Abstractions” & 3DPAS

MS is sponsored by the program of BiG Grid, the Dutch e-Science Grid, which is financially supported by the Netherlands Organisation for Scientific Research, NWO

This work has also been made possible thanks to computer resources provided by TeraGrid TRAC award TG-MCB090174 (Jha) and BiG Grid

This document was developed with support from the US NSF under Grant No. 0910812 to Indiana University for “FutureGrid: An Experimental, High-Performance Grid Test-bed”