RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Introduction to SAGA-BigJob

Piloting Many Simulations on Many Supercomputers

RADICAL TEAM

http://radical.rutgers.edu

"Many Simulations" Scenarios: Context

- A Single "Application" is broken into many smaller tasks
  - Naturally decomposed or "by design" (algorithmic and infrastructural)
  - Varying task duration: hours to days to weeks
  - Often these tasks are "coupled":
    - General concept for data sharing, synchronization, other dependency
- Coupling between these tasks
  - Uncoupled Tasks, Loosely-Coupled Tasks, Sequential dependencies..
  - Homogenous or Heterogeneous
  - Varying rate of coupling between tasks
  - Regular versus Irregular synchronization:
    - Temporal, Spatial
  - Ad hoc (pair-wise) exchange
    - No a priori determined exchange partners

Common Characteristics and Requirements

- Many requirements, but focus on two common and critical requirements
- R1: Adaptive Application Formulation
  - Flexible composition
- R2: Dynamic Resource Utilization
  - Growing/shinking resource pool
- Need abstractions that:
  - A1: Decouple workload and resource management
  - A2: Provide these capabilities in an extensible and interoperable way
- Need to provide these abstractions as well engineered tools
  - Employ CI best practices
  - Software sustainability
- Is there a single simple tool that can support the above requirements across all machines on XSEDE, OSG independent of application type?

Outline

PART-I

- Scalable approaches to many simulations on many machines
  - Introducing the Pilot Abstraction
- SAGA-BigJob: A simple, extensible and interoperable PJS
- SAGA: The Interoperability Layer
- BigJob: Case Studies of varying the coupling between tasks

PART-II

- Hands on Tutorial

PART-III

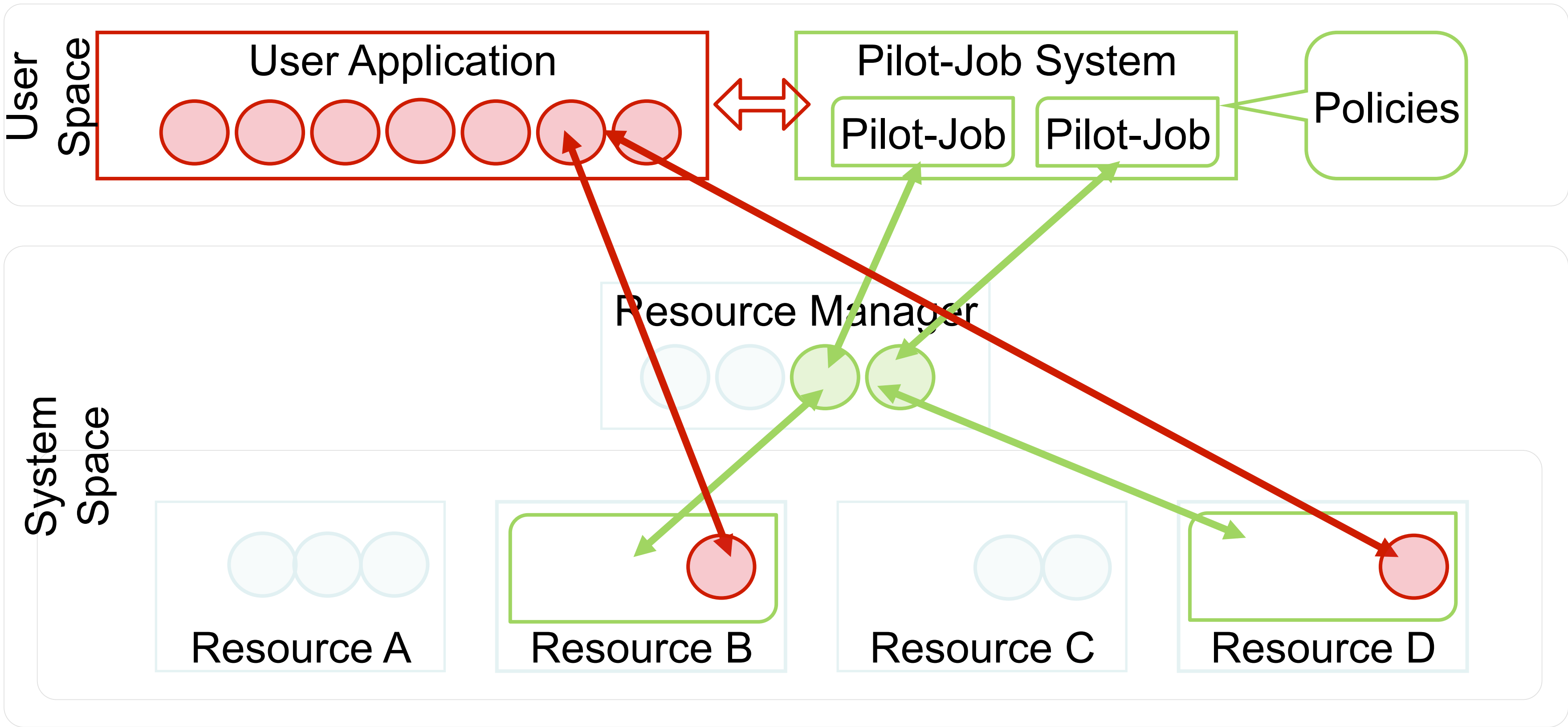
- Theory and Practice of Pilot Abstractions:
  - P*: A Model for Pilot Abstractions
  - BigJob: An implementation of P* Model
- Building higher-level capabilities on Pilot Abstractions
  - Pilot-MapReduce, Replica-Exchange

RUTGERS

PART-I: Introducing Pilot-Abstractions

Introduction to Pilot-Abstraction

- Working definition: a system that generalizes a placeholder job to provide multi-level scheduling to allow application-level control over the system scheduler via a scheduling overlay



Introduction to Pilot-Abstraction (2)

- Working definitions:
  - A system that generalizes a placeholder job to provide multi-level scheduling to allow application-level control over the system scheduler via a scheduling overlay
  - *".. defined as an abstraction that generalizes the reoccurring concept of utilizing a placeholder job as a container for a set of compute tasks; an instance of that placeholder job is referred to as Pilot-Job or pilot."*

- Advantages of Pilot-Abstractions:
  - The Perfect Pilot: Decouples workload from resource management
  - Flexible Resource Management
    - Enables the fine-grained (ie "slicing and dicing") of resources
    - Tighter temporal control and other advantages of application-level Scheduling (avoid limitations of system-level only scheduling)
  - Move control, extensibility and flexibility "upwards"
    - Build higher-level capabilities without explicit resource management

Pilot-Jobs Systems (PJS): Five Myths

- PJS do not need well defined architecture, model and semantics, or, PJs are such a simple concept, they don't need more “attention”
  - Not to confuse “simple to use” with simple to design”
- PJS are only about meta-scheduling (reducing queuing delays) on HTC, or, PJS unfairly game HPC queuing
  - There are interesting usage modes beyond “cycle stealing”
- PJS have to be tied to specific DCI; DCI are tied to specific PJ
  - Extensibility and interoperability have been difficult to establish
- ***PJS are passive (system) tools, as opposed to user-space, active and extensible components of a CI***
  - PJs can be user-controlled “programmable” elements
- PJS do not help with next-generation “data-intensive” applications
  - PJ for NGS O(10-100) GB per task on existing DCI

Landscape of Pilot-Job Systems

- There are many PJS offerings, often semantically distinct
  - PanDA, DIANE, DIRAC, Condor Glide-In, SWIFT, ToPoS Falkon, BigJob…
    - *Why do you think there has been a proliferation of PJs?*

- Difference in the execution models of the PJ
  - We know “what” pilot-jobs do, but the “how” remains less clear
    - How to map tasks to pilot-jobs? How to choose/map optimal resource?
    - How to “slice and dice” resources?

- Conceptual & practical barriers to extensibility (& interoperability)
  - The landscape of PJS reflects, in addition to PJS specifics, the broader eco-system of distributed middleware & infrastructure
  - Software Engineering issues, interfaces, standardization

- Data remains a dependent variable, not a primary variable
  - Introduce the concept of Pilot-data

Pilot-Job Paradigm

Based upon analysis of several Pilot-Job implementations

Architecture: Three distinct logical elements:

- **Workload Manager**: Responsible for making available the tasks to the executor alongside the needed data and retrieving results
- **Task Executor**: Responsible for executing the tasks while managing their data.
- **Communication and Coordination** (C-C): Patterns allow for and regulate the interaction between (and within) these two components.

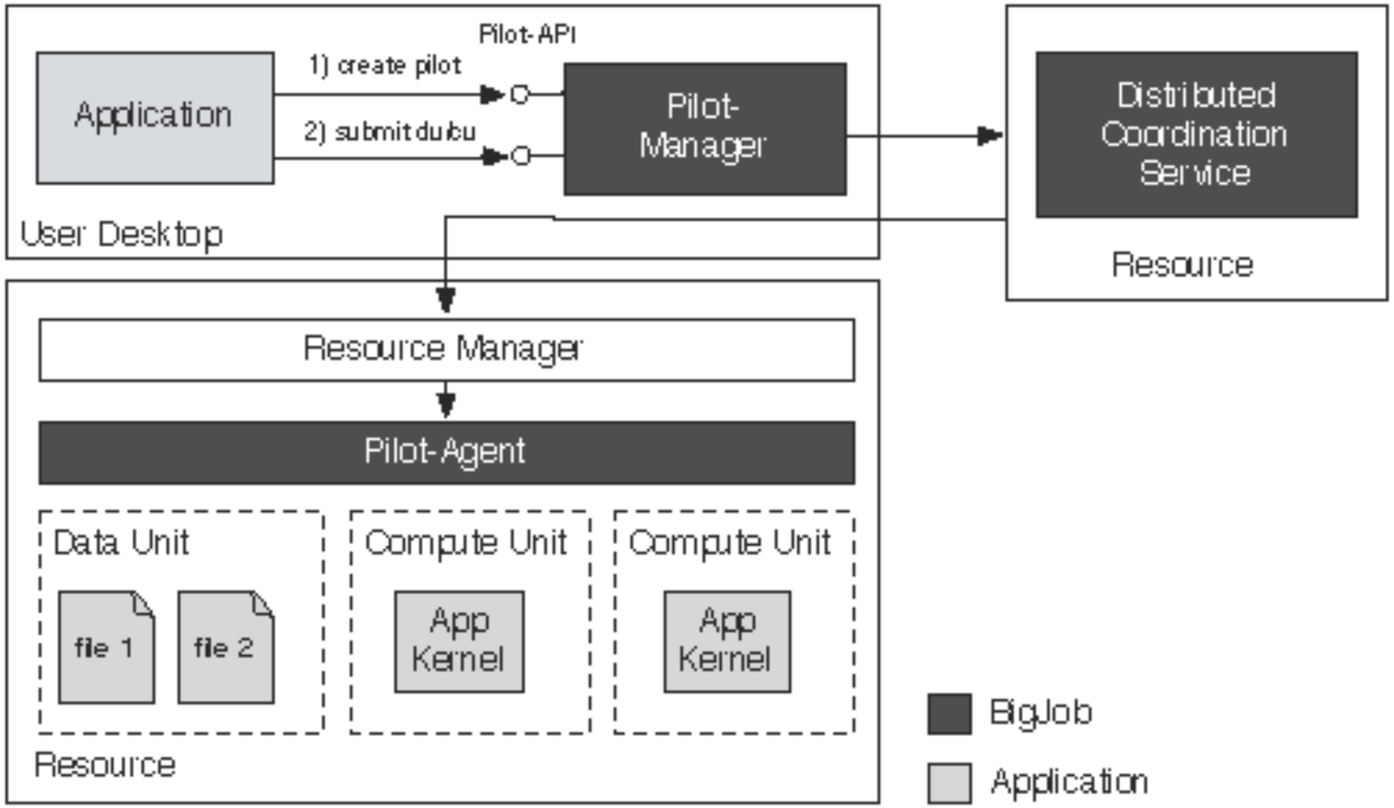
Execution Patterns: Based on multi-level scheduling and late-binding

- **Multi-level scheduling**. Tasks of a workload are scheduled on one or more pilots and the pilots are then scheduled on a given resource

Capability/Functionality: A system that generalizes a placeholder job to provide multi-level scheduling to allow application-level control over the system scheduler via a scheduling overlay

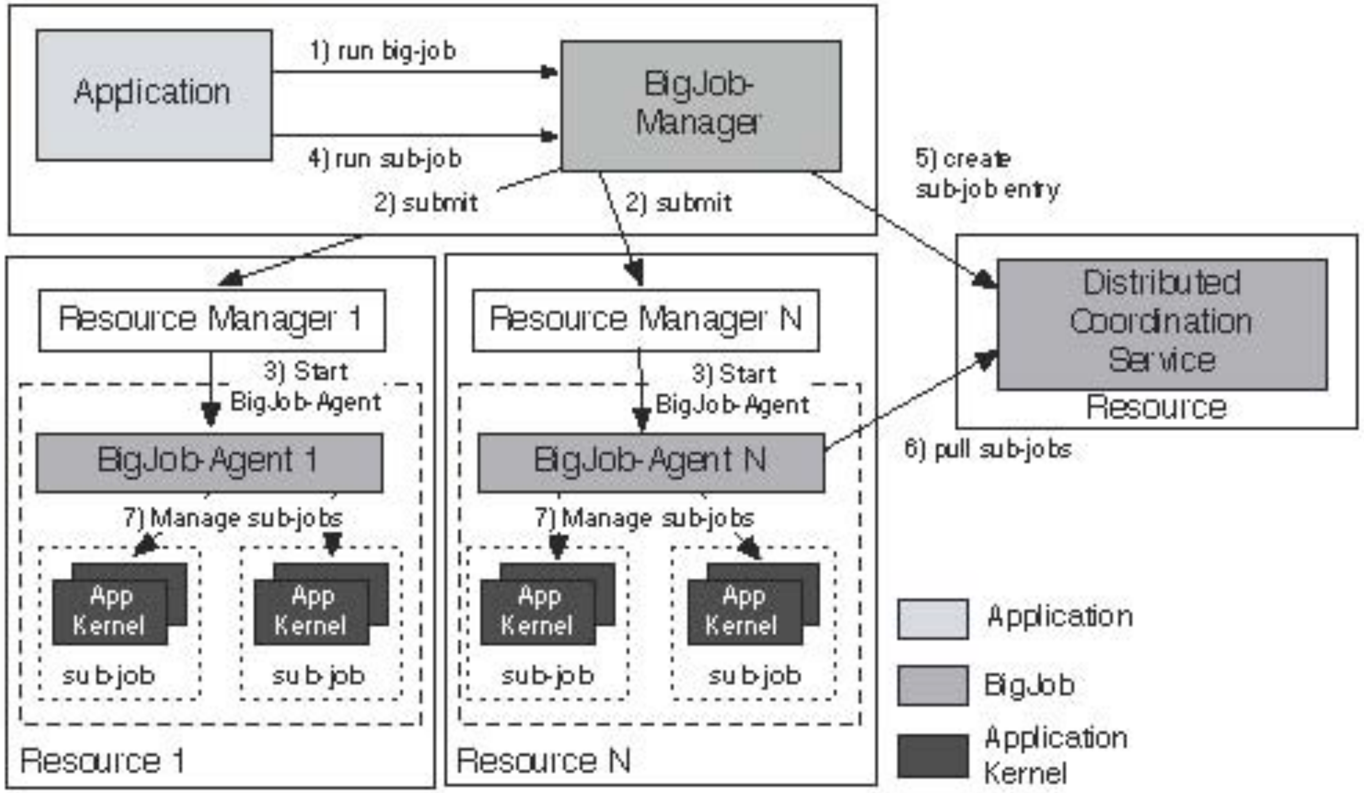
BigJob: A simple, extensible and interoperable Pilot-Job System

BigJob: Architecture

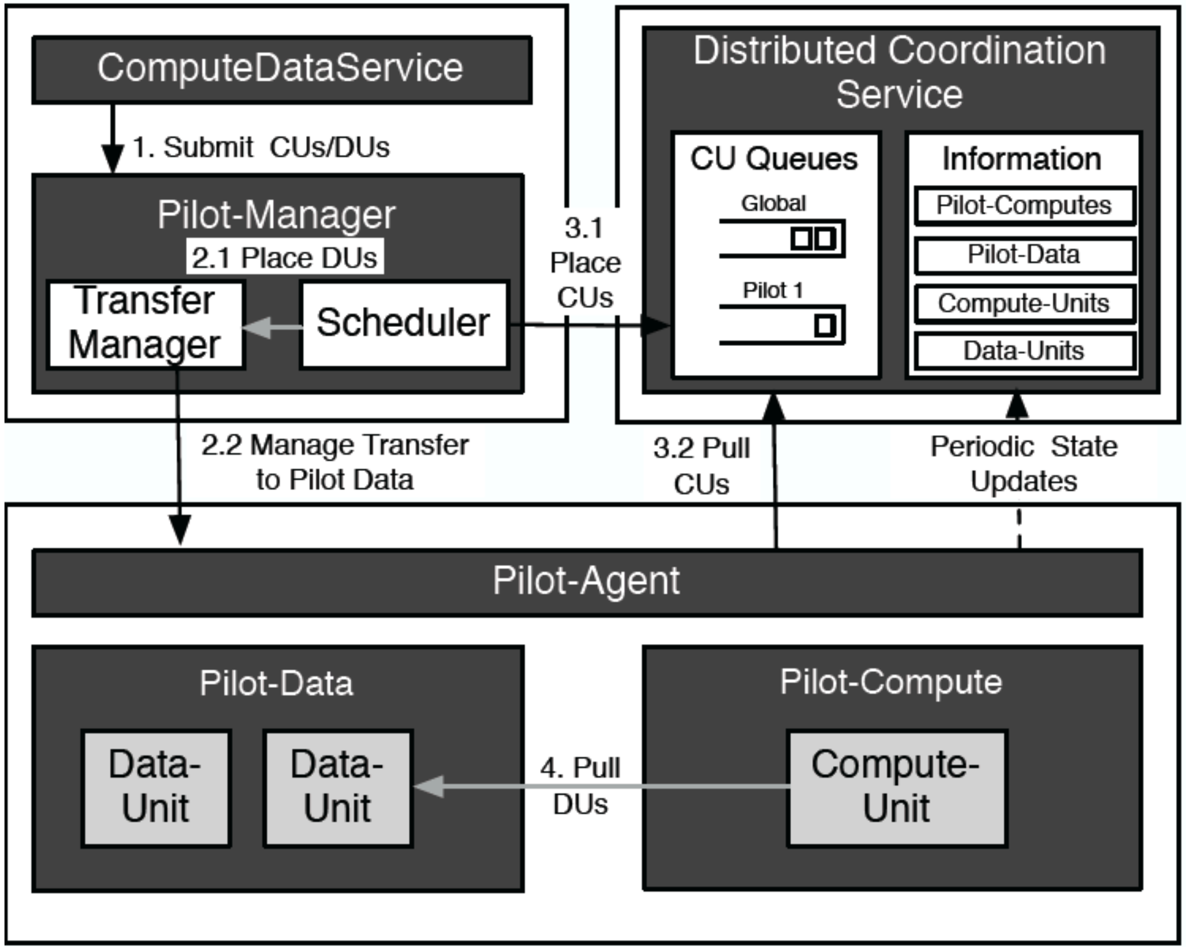


BigJob: Distributed View

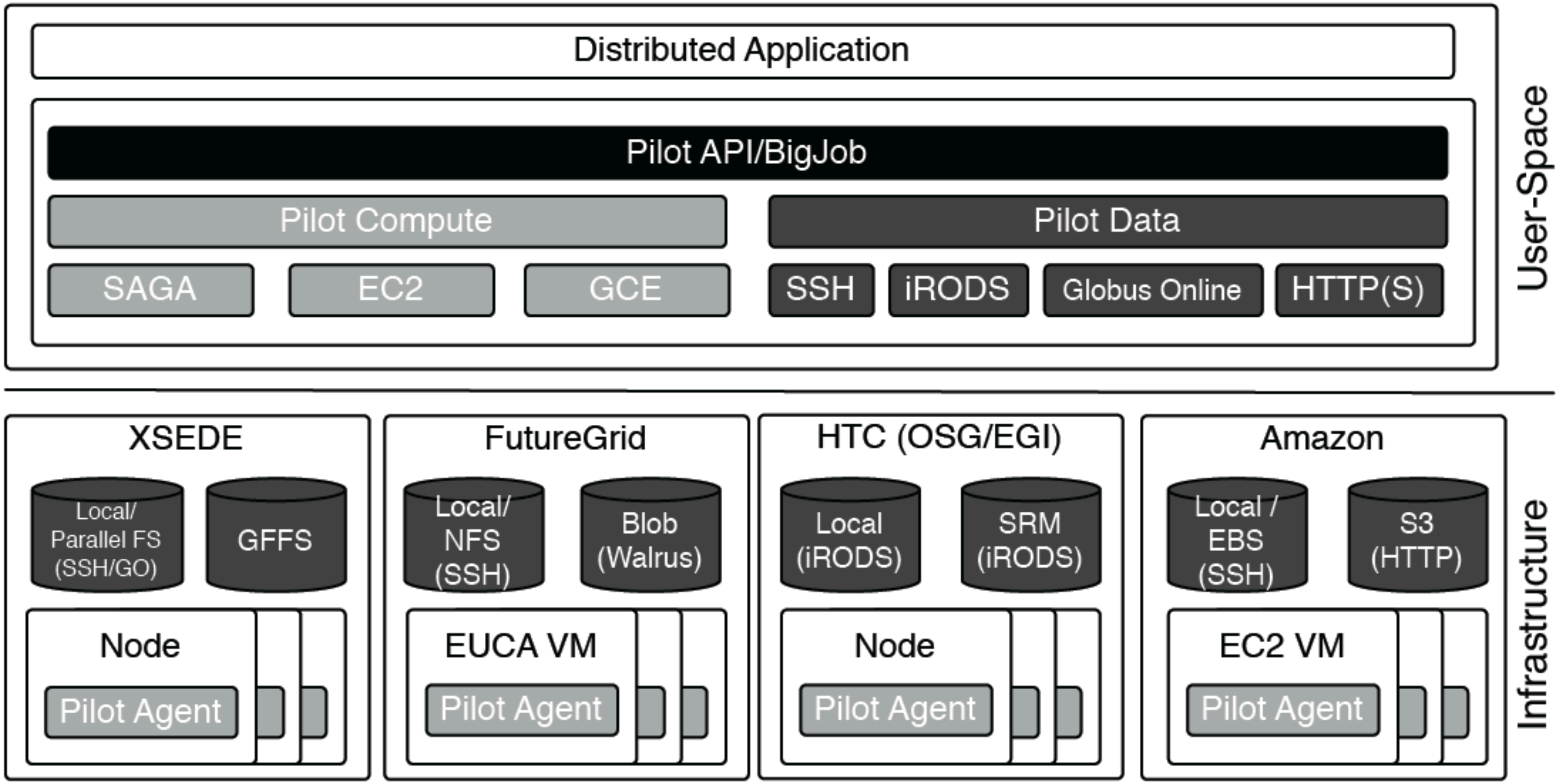
(Strongly encourage you to try after the tutorial)



BigJob Workload Management

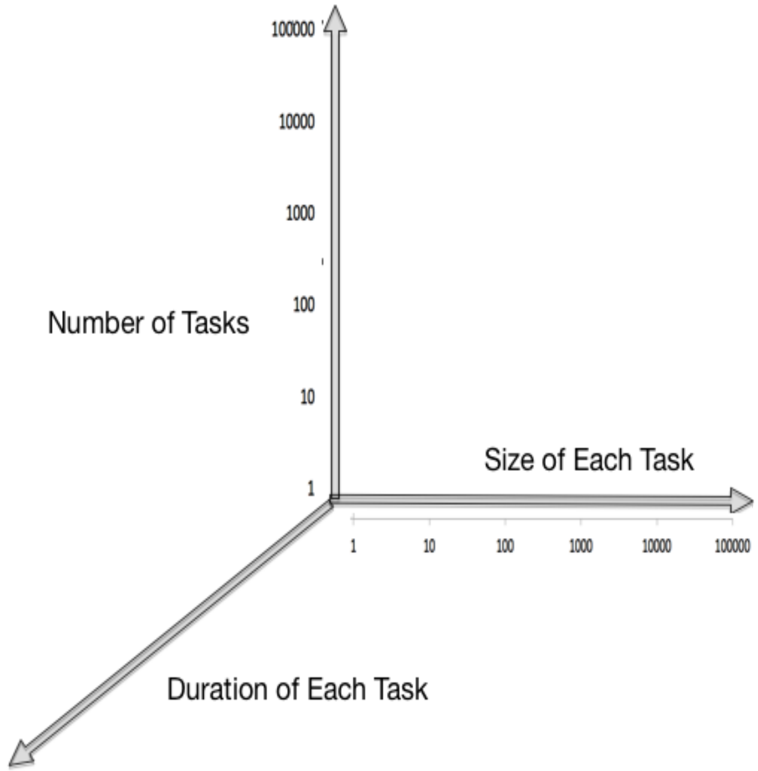


BigJob: Resource Interoperability



Design Objective: Scaling Along Many Dimensions

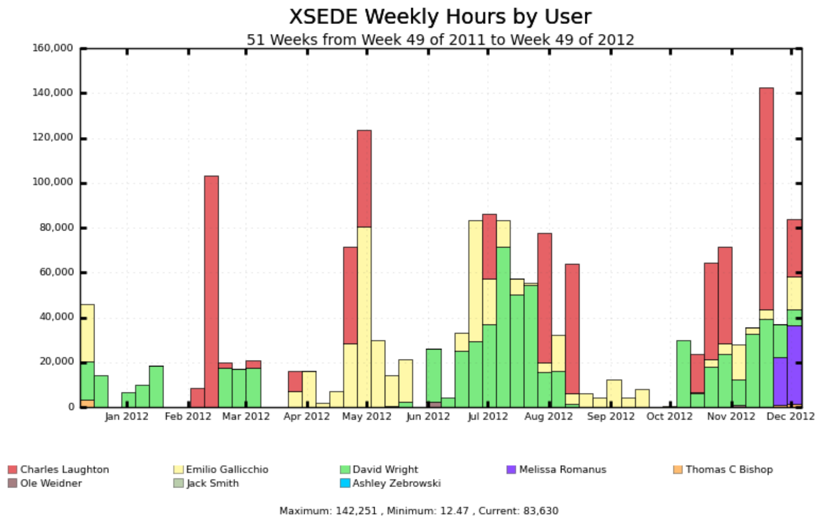
- Scaling Dimension #1: Size of each task
  - *Scale-up*
  - 1 core per task,128/512…..
- Scaling Dimension #2: Total Number of Concurrent tasks
  - *Scale-Out*
  - Enhanced (MD) sampling O(1000), statistical errors O(10^6) tasks,
- Scaling Dimension #3: Total Number of Tasks/per unit time
  - Real time processing O(1000)
- Scaling Dimension #4: Number of Resources Used
  - *Scale-Across*
  - Execute on Grids, Clouds, Clusters

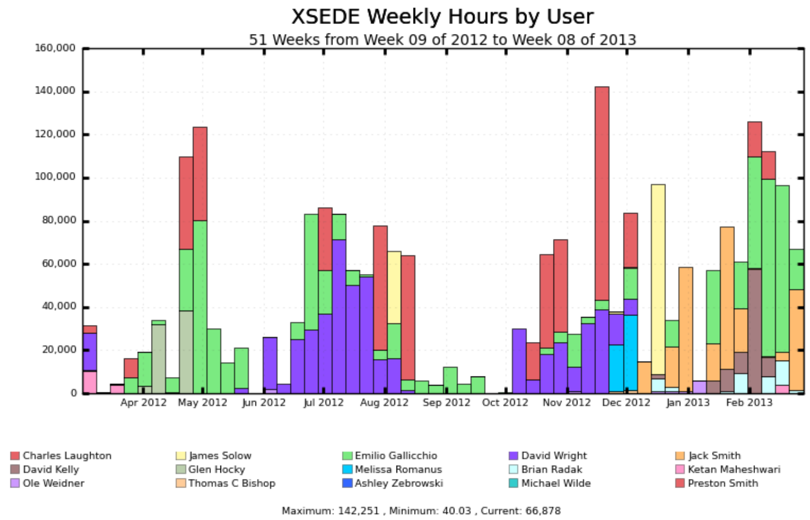


"Coarse-Grained" BigJob Performance

- Number of zero-payload tasks that BJ can dispatch per second:
  - Distributed: O(10)
  - Locally: > O(10)
- Number of Pilots (Pilot-Agents) that can be marshaled
  - Locally/Distributed: O(100)
- Typical number of tasks per Pilot-Agent:
  - Locally/distributed: O(1000)
- Number of tasks concurrently managed = Number of Pilot-Agents x tasks per each agent :
  - O(100) x O(1000)
- (Obviously) The above depends upon data per task:
  - BigJob has been used over O(1)--O(10^9) bytes/task, for tasks of duration O(1) second to O(10^5) seconds

BigJob: (Partial) Usage on XSEDE Machines





> 10M SUs/year (and increasing) on XSEDE machines

SAGA: Interoperability Layer for BigJob

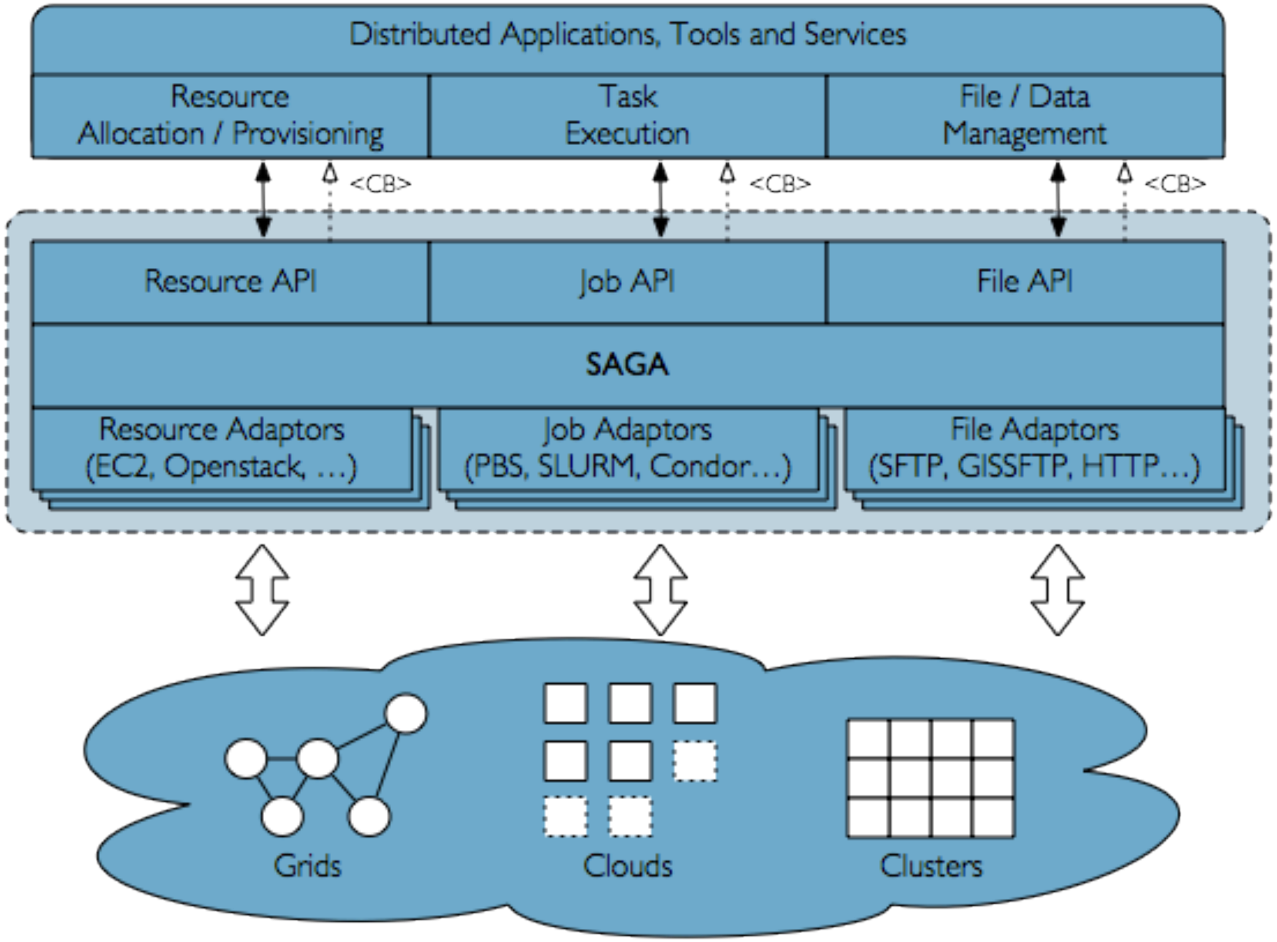
http://saga-project.org
http://saga-project.github.io/saga-python

A Light-Weight Python Access Layer for Distributed Computing Infrastructure

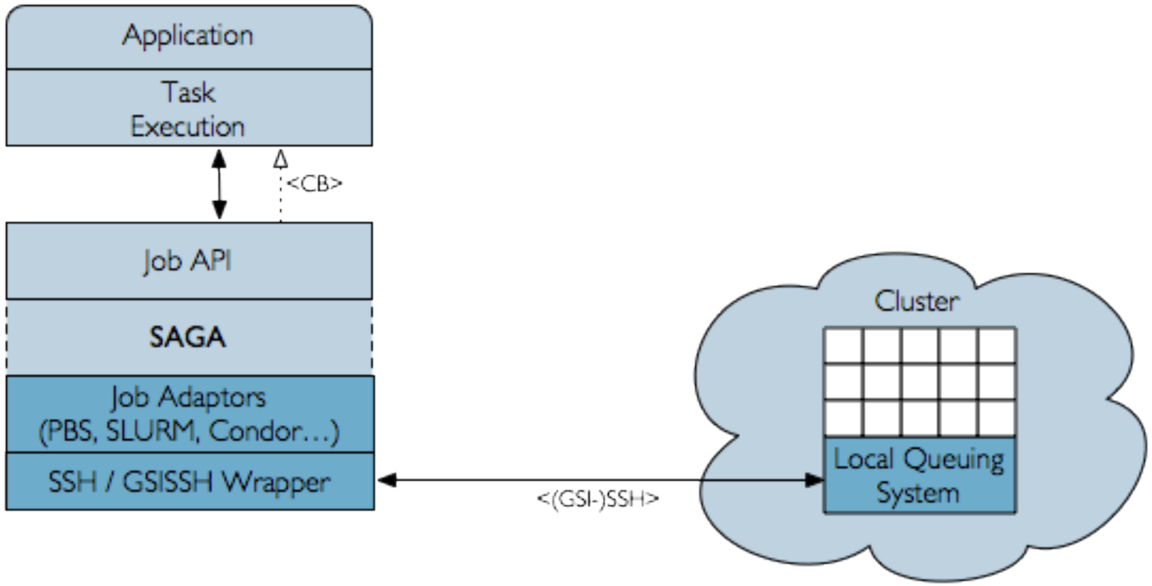
Overview

- SAGA: Simple API for Distributed (“Grid”) Applications
  - Application level standardized (Open Grid Forum GFD.90) API
  - Application is a broad term: “one person’s application is another person’s tool (building block)”.
- SAGA-Python:
  - Native Python implementation of Open Grid Forum GFD.90
  - Allows access to different middleware / services through a unified interface
  - Provides access via different backend plug-ins (“adaptors”)
  - SAGA-Python provides both a common API, but also unified semantics across heterogeneous middleware:
    - Transparent Remote operations (SSH / GSISSH tunneling)
    - Asynchronous operations
    - Callbacks
    - Error Handling

Schematic

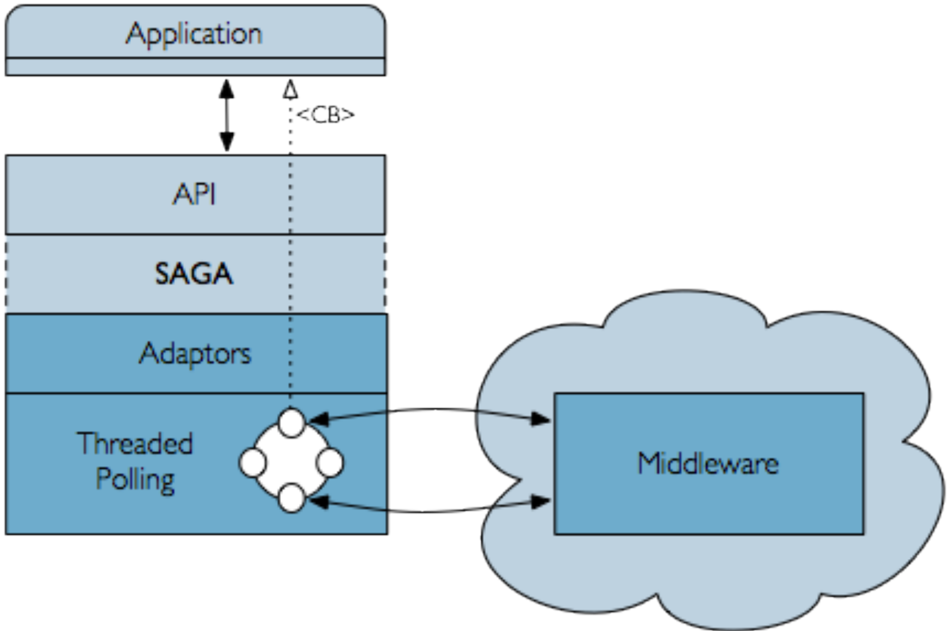

Distributed Applications, Tools and Services
Resource Allocation / Provisioning
Task Execution
File / Data Management
<CB>
<CB>
<CB>
Resource API
Job API
File API
SAGA
Resource Adaptors (EC2, Openstack, ...)
Job Adaptors (PBS, SLURM, Condor...)
File Adaptors (SFTP, GISSFTP, HTTP...)
Grids
Clouds
Clusters

Transparent Remote Operations



- Fast and optimized (GSI-)SSH command transport wrapper can be used by adaptors to access otherwise ‘local-only’ services, like many queuing systems

Transparent CALLBACKS



- Callbacks and asynchronous operations are important concepts in distributed applications but are not supported by all middleware systems
- SAGA-Python moves application-level polling into the adaptor layer and exposes a clean callback interface through the API

Available adaptors

http://saga-project.github.io/saga-python/doc/adaptorssaga.adaptor.index.html

- Job Submission Systems
– SSH, GSISSH, Condor, Condor-G, PBS(-Pro), TORQUE, SGE, SLURM.
– Under development: LSF
- File / Data Management
– SFTP, GSIFTP, HTTP, HTTPS.
– Under development: iRODS (Globus Online)
- Resource Management / Clouds
– Amazon EC2, Openstack (‘libcloud’-based)

http://saga-project.github.io/saga-python/doc/adaptors/saga.adaptor.ec2_resource.html

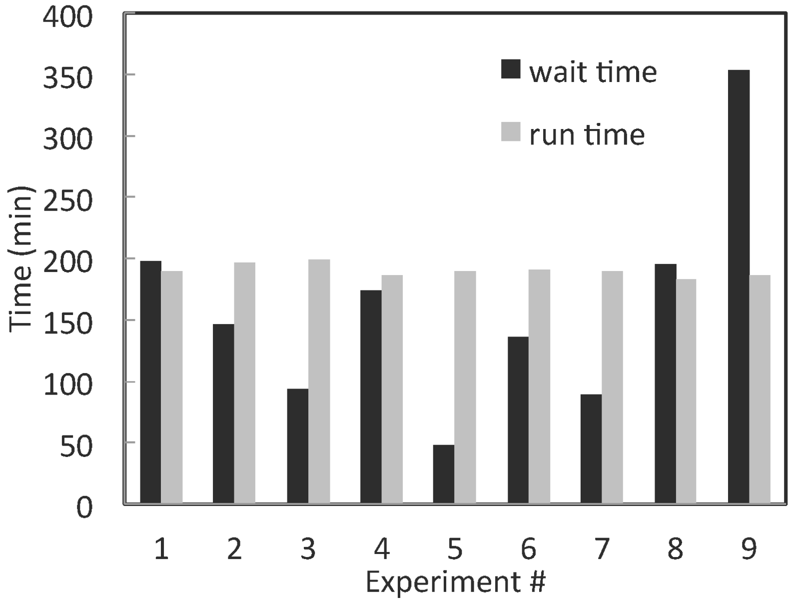
SAGA: Interoperability layer upon which other tools and applications are built

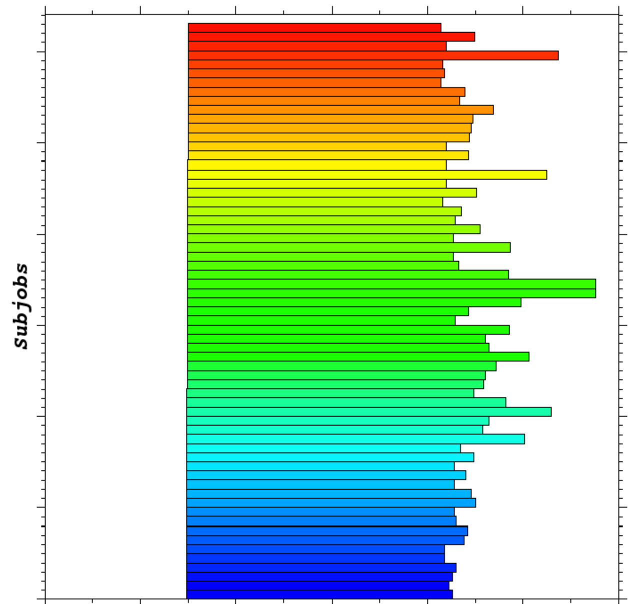
- HOW SAGA is Used?
  - Uniform Access-layer to DCI, e.g, EGI, XSEDE, etc
  - Application “Scripting Layer” to DCI
  - Build tools, middleware services and capabilities e.g. Gateways,
- WHAT is SAGA Used for?
  - Production-grade science and engineering and research tool to design, implement and reason about distributed programming models, systems and applications
- Where is SAGA Used?
  - Pilot-job and Workflow systems, science gateways and web portals
  - Domain-specific (distributed) applications, libraries and frameworks

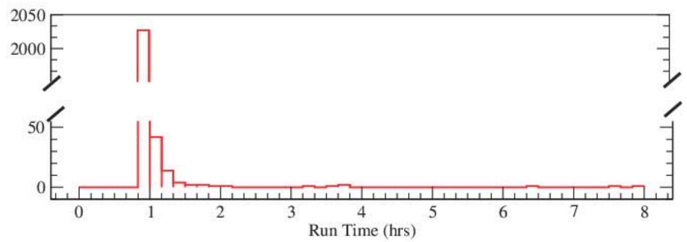
Case Studies: Varying the “coupling” between many (MD) Simulations

HT-HPC on Kraken

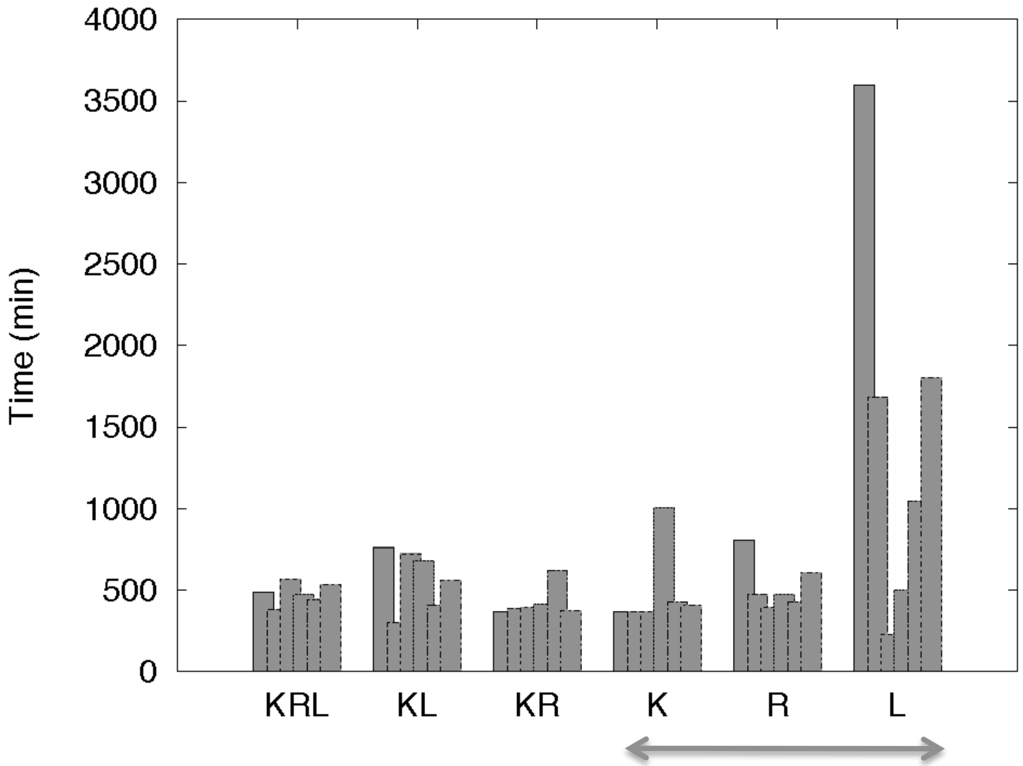
126 ensembles, each of 192 cores = 24192 cores

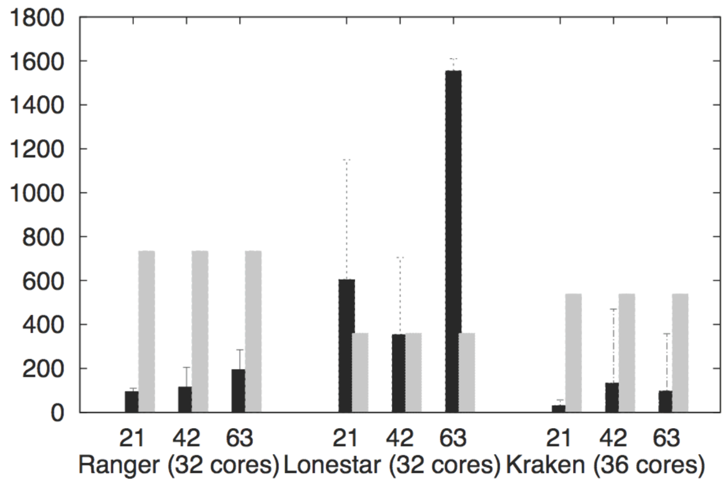






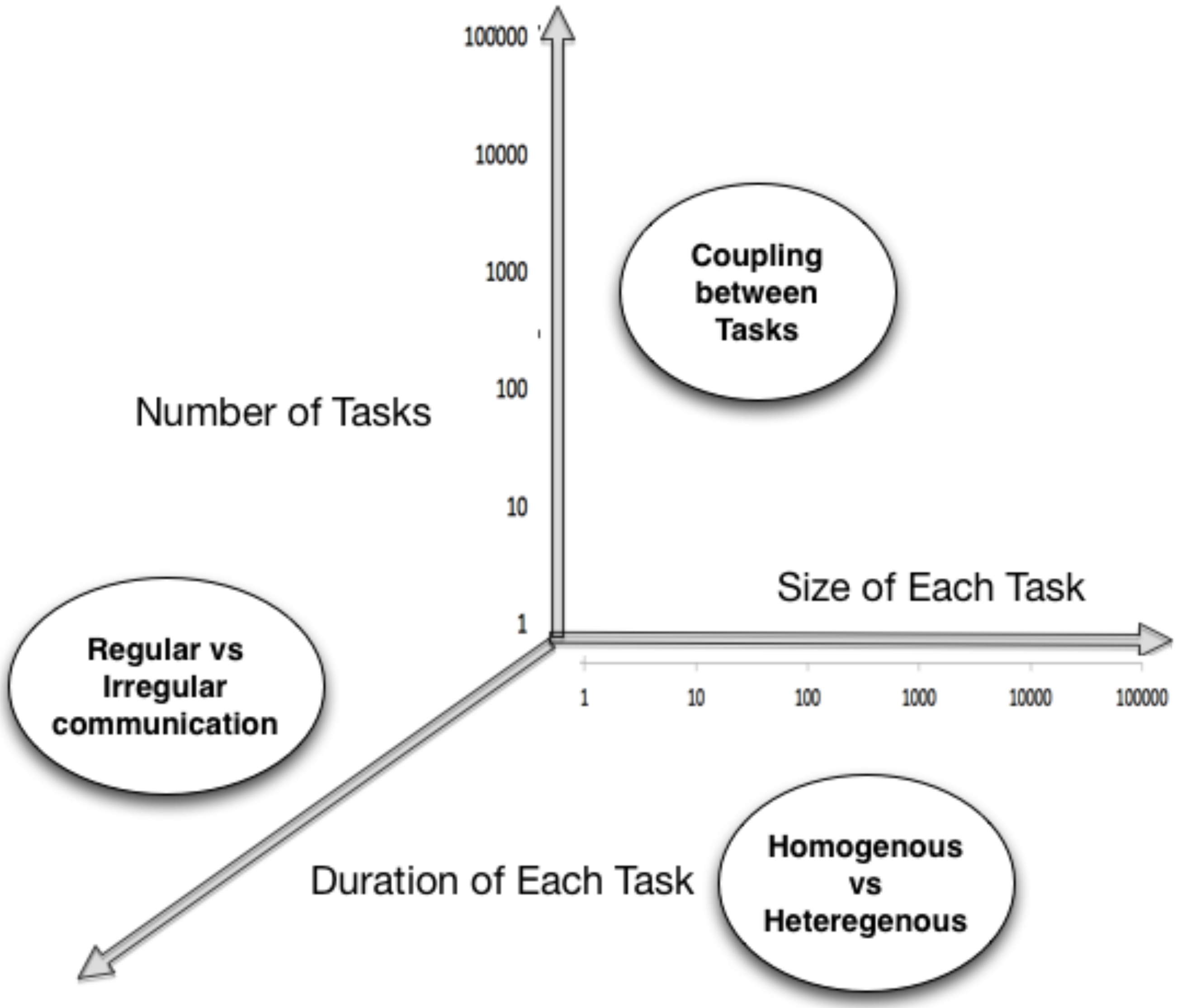
Scale-Out/Across





X-axis: number of tasks (size)

Scaling Along Many Dimensions

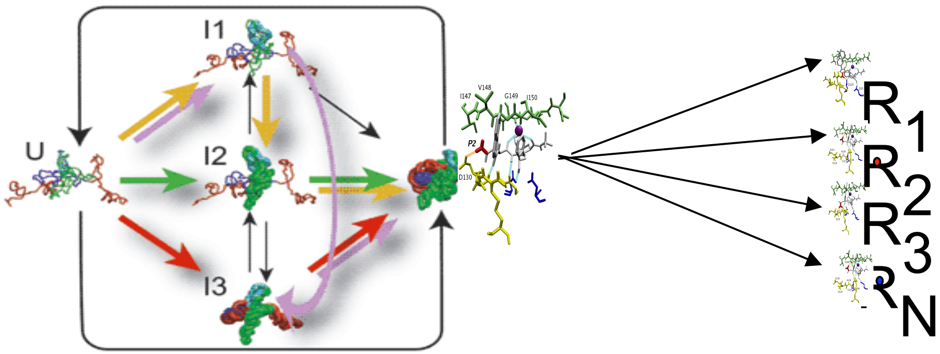


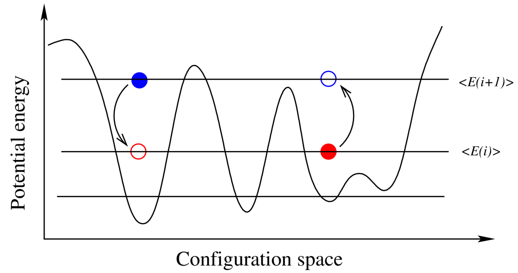
Varying the Coupling between Simulations

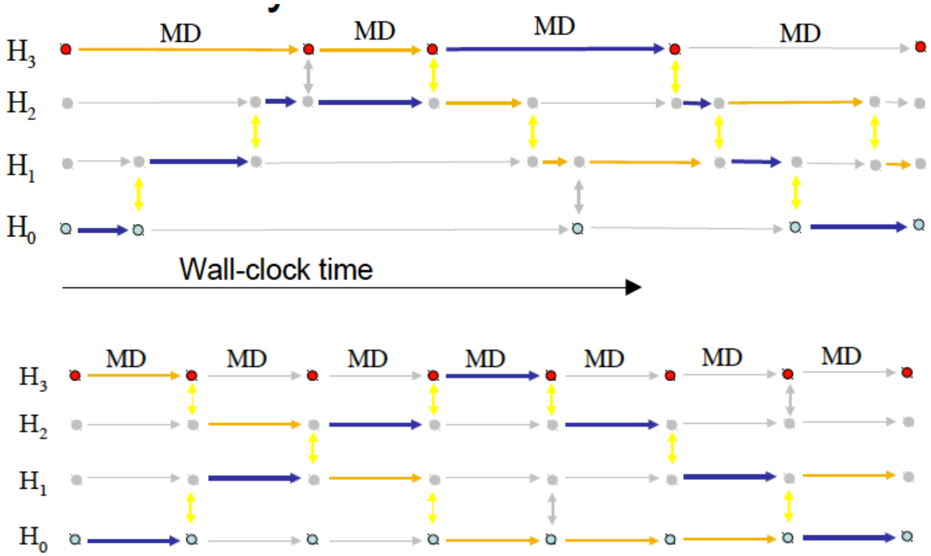
- Large Ensemble of Uncoupled (Molecular) Simulations:
  - Enhanced Sampling of bio-molecular systems
- Multiple-Chaining of Simulations
  - Long time-scale simulations
- Ensemble of Loosely Coupled Simulations
  - Replica-Exchange Class of Algorithms
  - Understand protein-ligand recognition via (multidimensional) replica-exchange

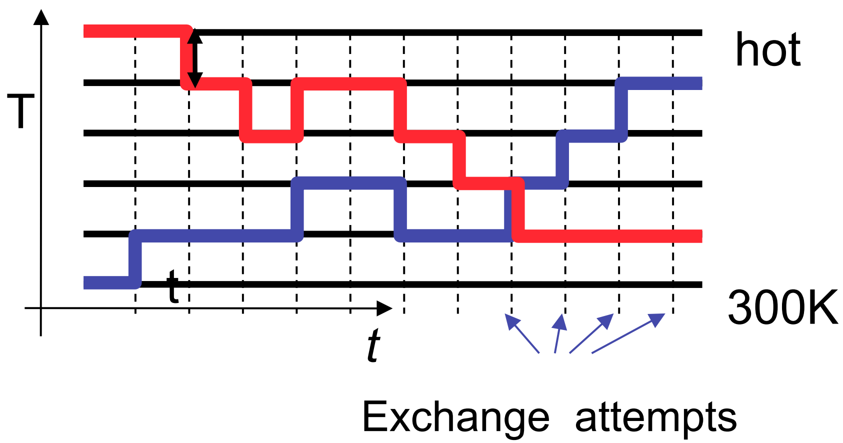
- Requirements:
  - Launch and monitor/manage (order 10^2-10^4) ensemble members
    - Where each ensemble member could be 128-1024 cores
  - Varying degrees of coupling between ensemble members
  - Varying job duration: hours to days to weeks
  - Ad hoc pair wise (a)synchronous data exchange

Replica Exchange Molecular Simulations

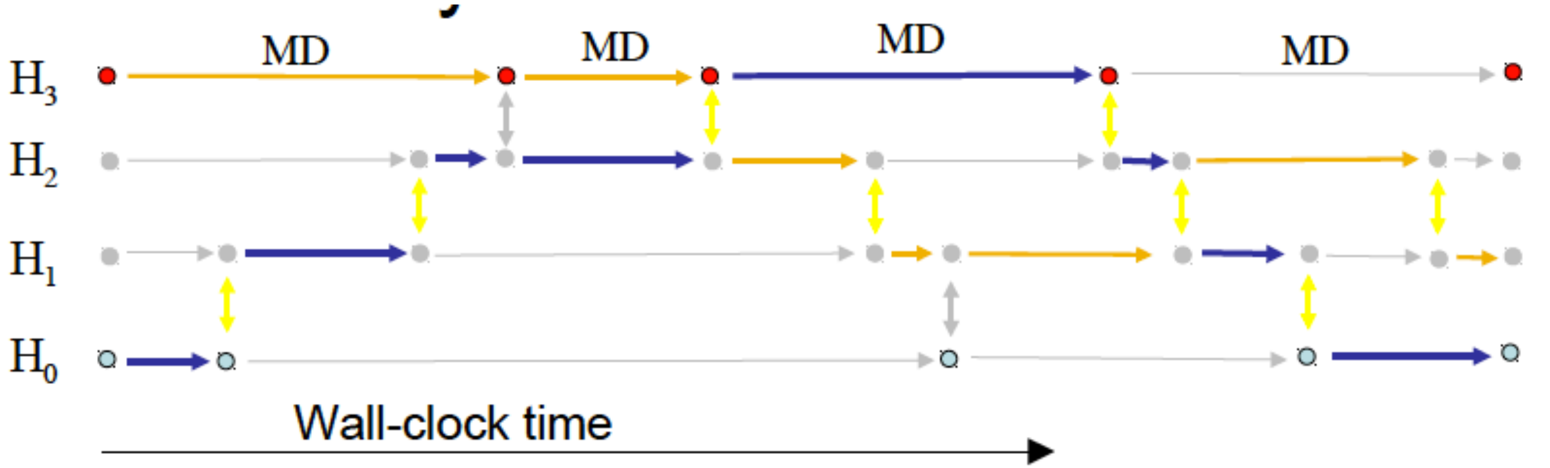


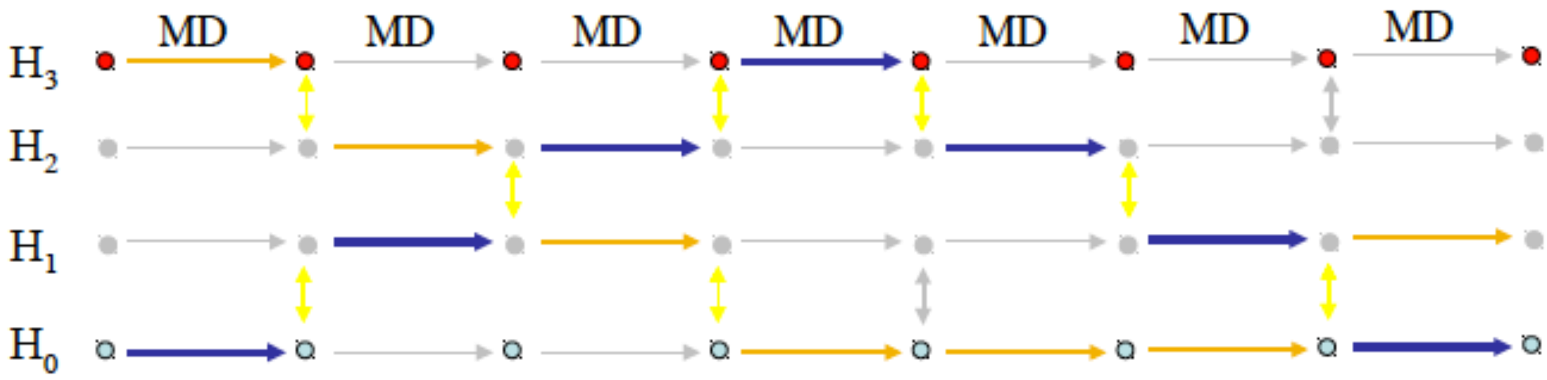






Irregular Synchronization





PART II: HANDS ON TUTORIAL

Getting Started

- These slides:

https://github.com/saga-project/tutorials/blob/master/lecture/xsede13.pdf

- URL for Tutorial:

https://github.com/saga-project/tutorials/wiki/XSEDE13

PART III: PRINCIPLES AND PRACTISE OF PILOT ABSTRACTIONS

PART III: PRINCIPLES AND PRACTISE OF PILOT ABSTRACTIONS

OR

What makes BigJob Unique?

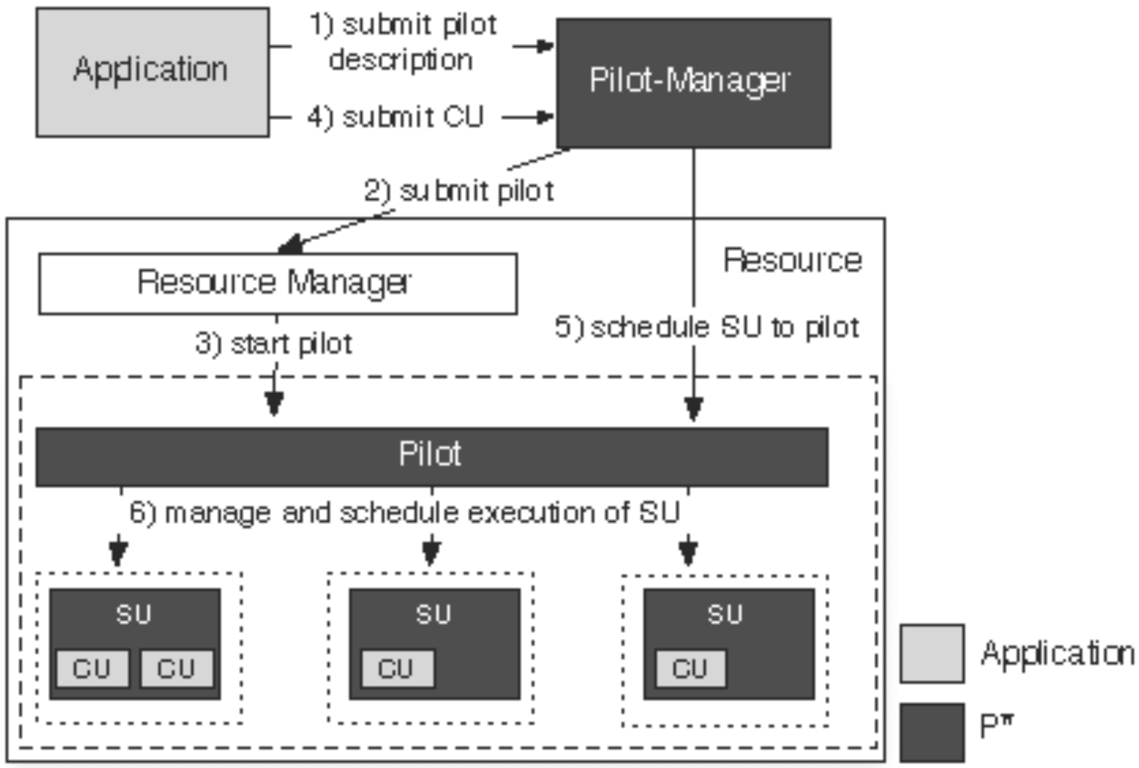
P*: A Model of Pilot-Abstractions

P*: A Conceptual Model for Pilot Abstractions

- A minimal but complete model
  - **Minimal**: Towards a common understanding of pilot-jobs
    - Provides vocabulary and model for analysis and insight across different PJ systems
    - Does not try to provide closed form answers to all issues
  - **Complete:** Can be used to design a priori an effective Pilot-Job
- Establish basic terminology, functionality and inter-relationship
  - What is the pilot? What is the agent? Push or pull?
- Provide a framework for comparison across Pilot implementations
  - Many existing pilot implementations can be understood using P*
  - Not all pilot implementations must adhere strictly to P*
- Unified view of Pilot-Abstractions
  - Provides symmetrical treatment for compute and data (and eventually network)

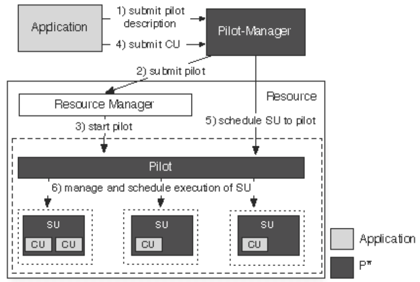
P* Model: Elements, Characteristics and API

- **Elements:**
  - Pilot-Compute (PC)
  - Pilot-Data (PD)
  - Compute Unit (CU)
  - Data Unit (DU)
  - Scheduling Unit (SU)
  - Pilot-Manager (PM)
- **Characteristics:**
  - Coordination
  - Communication
  - Scheduling
- **Pilot-API**



RUTGERS

P* Elements



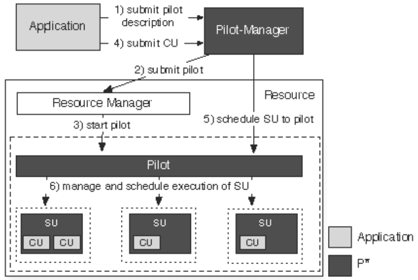
Pilot-Compute

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

Pilot-Data

- 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

P* Elements



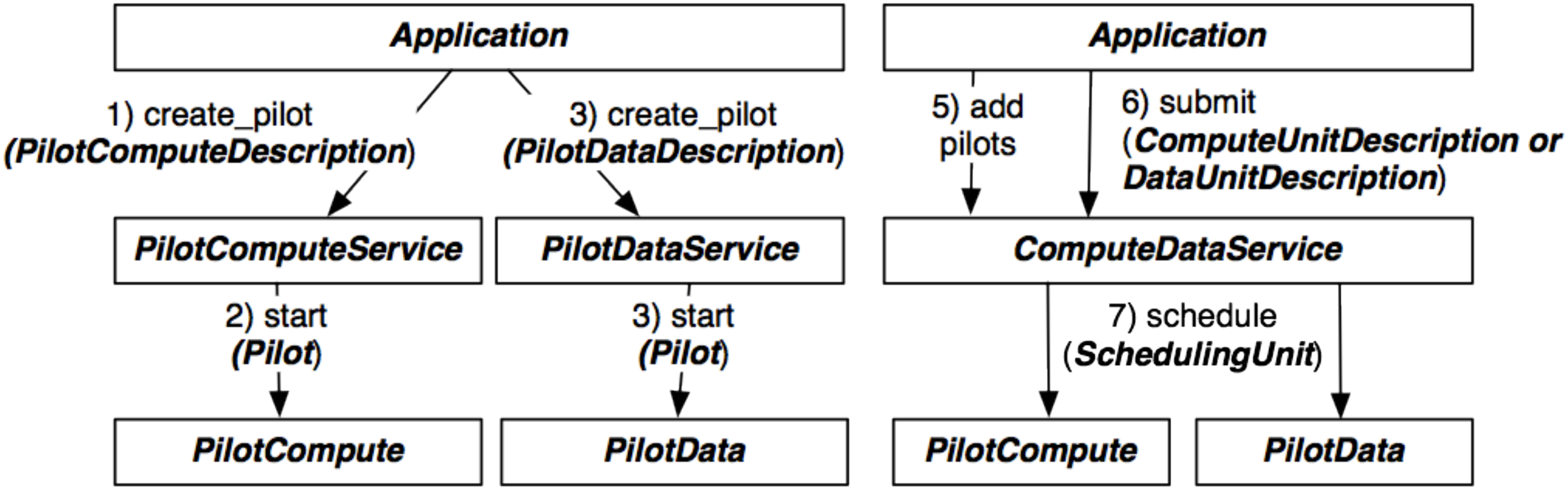
Compute Unit (CU)

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

Data Unit (DU)

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

Pilot-API: Unified API to Pilot-Compute and Pilot-Data

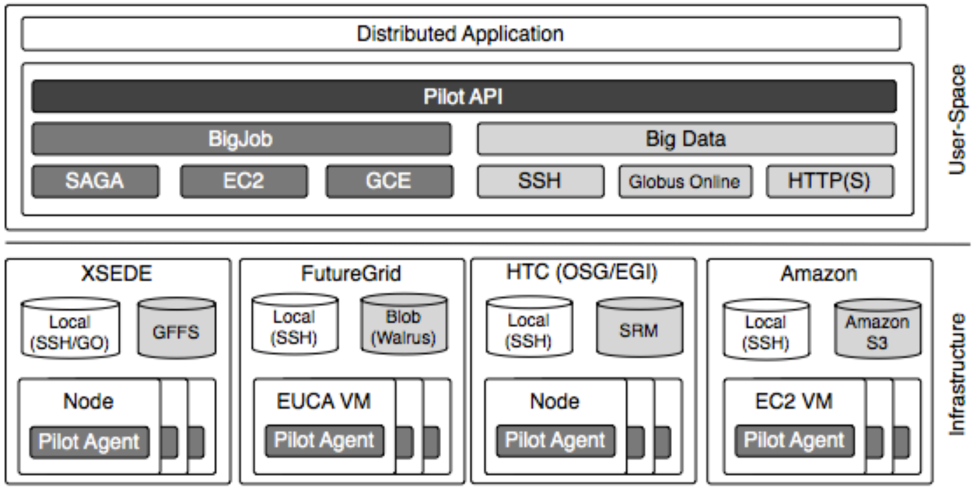


Managing Pilots

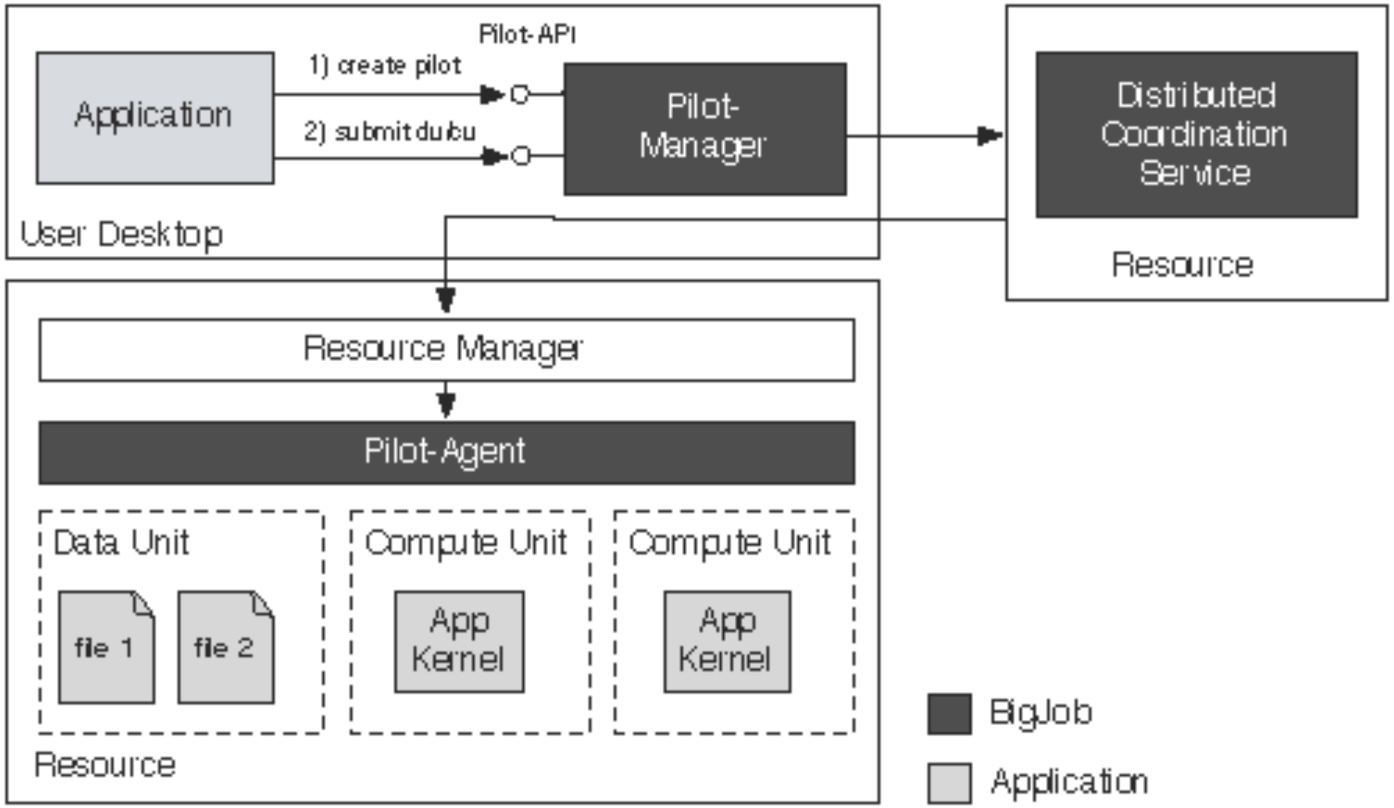
Managing Application Workload

P*: Mapping and Interoperability

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



BigJob: Architecture



BigJob @ XSEDE'13

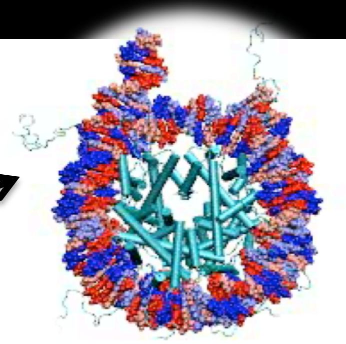
Talk by Jack Smith @ 3:45pm Tuesday

Talk by Melissa Romanus: 11am Wednesday

Poster Session: Jacob Swadling

RUTGERS

Scalable Online Comparative Genomics of Mononucleosomes



Collaboration with Tom Bishop, Jack Smith
and Work supported by XSEDE Fellows Grant (Jack Smith)
See Talk by Jack Smith @ 3:45pm Tuesday

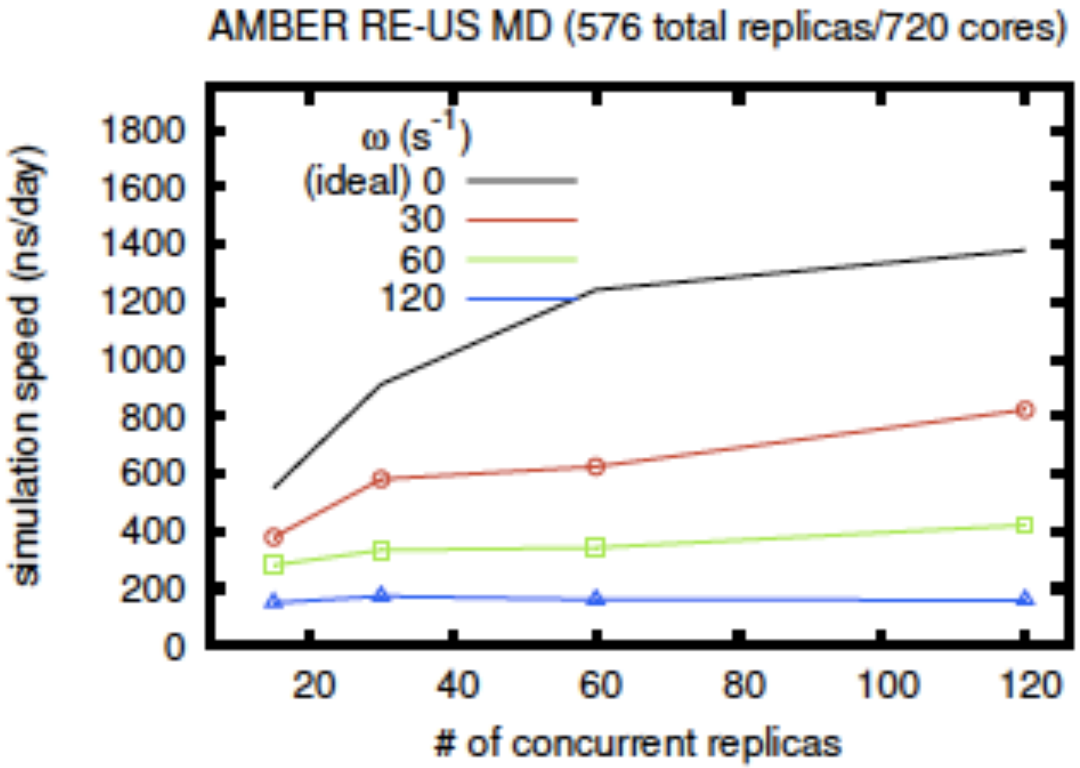
RUTGERS

Mapping Complex Biomolecular Reactions using Large-Scale Replica-Exchange Simulations on National Production Cyberinfrastructure

Collaboration with Darrin York, Ron Levy

Talk by Melissa Romanus: 11am Wednesday

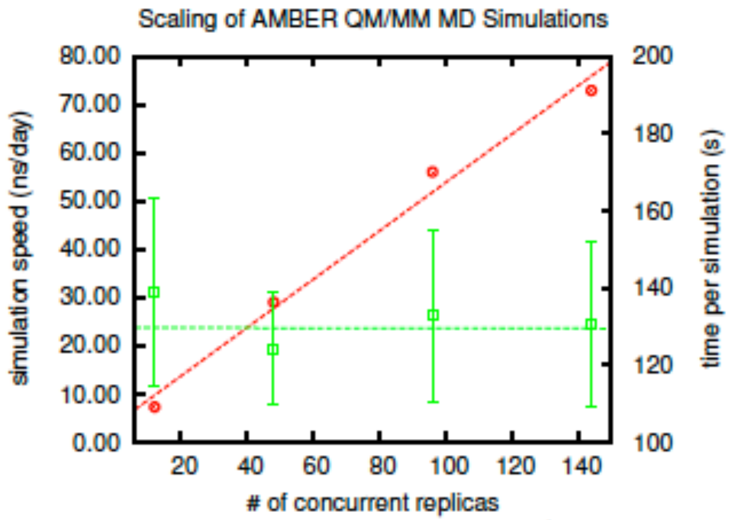
Replica-Exchange Performance in AMBER



Umbrella sampling of the backbone conformational space of alanine dipeptide. The exchange parameters are all permutations of harmonic biasing potentials on each torsion.

RUTGERS

QM/MM Replica-Exchange



- QM/MM is not generally considered parallelizable
  - RE provides efficient sampling irrespective
- Repex FW: BigJob-based Repex
  - Framework to handle O(100)--- O(1000) concurrent simulations
  - Amongst the earliest QM/MM
  - Framework similar to classical QM/MM
- Performance increases linearly with core count and number of QM/ MM simulations
  - No apparent coordination cost for additional simulations

Pilot-Mapreduce: Library for MR using Pilot-API for Dynamic Resource Management

BigJob Summary

- BigJob: Available Simple, Extensible and Interoperable PJS
- Simple: Adheres to a simple conceptual model of pilot-abstractions
  - Well defined sub-systems, execution model, semantics
  - We now know both “what” the PJS does, but also the “how”!
    - How to map tasks to pilots?
    - How to “slice and dice” resources?
- Extensible:
  - User controlled
  - Programmable (via Pilot-API)
  - Different application types, execution modes, coupling schemes
- Interoperable: Across XSEDE resources

(More general) Conclusions

- "Pilot Abstraction Works!"
  - Abstraction for separation of workload and resource management,
  - Dynamic resource utilization

- SAGA-BigJob: A Pilot-Job System that is more than a Pilot-Job System!
  - Libraries for Replica-Exchange, PMR, Binding Affinity Calculator builds upon Pilots for (dynamic) resource management

Acknowledgements

Graduate Students:

- Ashley Zebrowski
- Melissa Romanus
- Mark Santcroos
- Anton Trekalis
- Anjani Raghotham
- Dinesh Prasad

Undergraduate Students:

- Vishal Shah

Research Scientists:

- Andre Luckow
- Andre Merzky
- Matteo Turilli
- Ole Weidner

TACC Staff

- Yaakoub El-Khamra

Users of BigJob

National Science Foundation

References

- **``P*: A Model of Pilot-Abstractions"**, 8th IEEE International Conference on e-Science 2012 (DOI: 10.1109/eScience.2012.6404423)

- ***"*Distributed Computing Practice for Large-Scale Science & Engineering Applications"** Computing and Concurrency: Practice and Experience, 2012 (DOI: 10.1002/cpe.2897)

- **"Pilot-Data: An Abstraction for Distributed Data"**, arXiv:1301.6228

- **"Pilot MapReduce",** 3rd Workshop on MapReduce and its applications (2012), in conjunction with HPDC (Delft)

References

- SAGA-Python:
  - http://saga-project.github.io/saga-python/
- BigJob: An implementation of P*
  - http://saga-project.github.io/BigJob/
- RADICAL:
  - http://radical.rutgers.edu/
- Publications:
  - http://radical.rutgers/edu/publications
- Tutorials:
  - https://github.com/saga-project/tutorials/wiki/XSEDE13