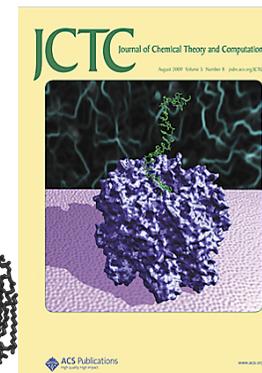
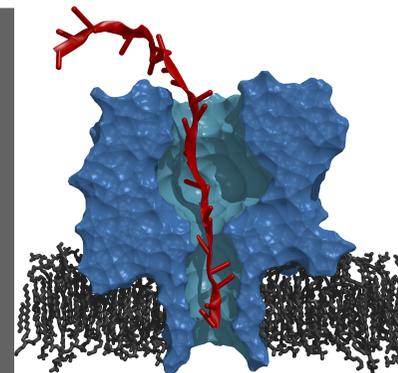


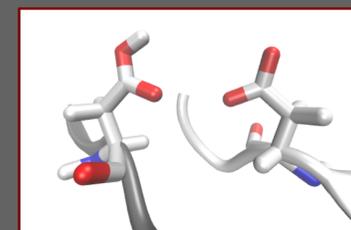
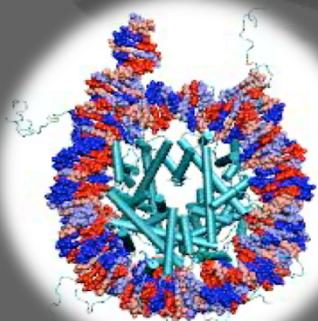
RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY



A Fresh Perspective on Distributed Applications and Infrastructure: Abstractions, Models and Implementations

Shantenu Jha

<http://radical.rutgers.edu>



Outline

- Extreme-Scale Distributed Computing (XSD)
 - Status Quo: Understanding the landscape of XDC-2013
 - Beyond HPC/HTC: Requirements for “many simulations” scenarios
- Abstractions, Models and Implementations for many-simulations
 - Five Myths associated with Pilot-Jobs
 - Address using Abstractions, Models and Implementations
 - P* Model of Pilot-Abstractions
- Future Directions: Next Generation Middleware (NGMW)
 - Many aspects and considerations, but focus on resource management for many simulations

Why Distributed Extreme Scale Computing?

- Support new science at the next scale(s)
 - Flexible Coupling X-flops of distributed compute with X-bytes of data
 - Simulations integrated with distributed data analysis
 - Real-time computing coupled with distributed data from scientific experiments (LSST, SKA)
- New execution modes for efficient and effective utilization of collective set of resources
 - Off-load workloads from leadership to less powerful machines
 - On-load workloads from distributed systems onto leadership
 - Strategically and synergistically, not competitively
- Support the “long tail of science”

XDC: ATLAS

- Observation:
 - “.. Distributed computing will persist ” for integrated HPC + HTC
Richard Mount (SLAC), c.f. <http://goo.gl/pJzljH>
- Requirement:
 - ATLAS in >2018 needs:
 - Non-monolithic extreme-scale and integrated HPC + HTC
- Challenges:
 - Mostly economic, but also how to manage workload decomposition
 - Development and deployment of future supercomputing applications
 - *Role for flexible execution strategies*
- Question:
 - “.. Are systems of the complexity of ATLAS Distributed Computing sustainable long-term?”

XDC in Relation to “Traditional” EC

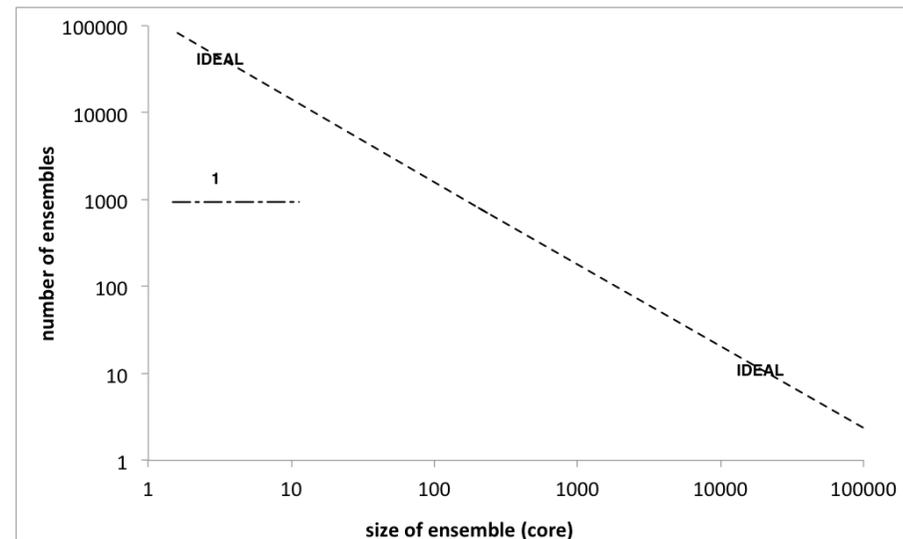
- Many applications as opposed to a set of kernels that need to be optimized
 - Metrics of performance varied, i.e. not just peak performance
- Capture different modes of extreme-scale computing:
 - Couple X-flops with X-byte: both simulation and analysis.
 - Integrate multiple large-scale resources as an aggregated capability.
- Application structure simple, but infrastructural requirements difficult
 - Task-level composition and coordination is important and varied
 - External data infrastructure, repositories
- Important role for middleware, explains why DC SW environment is complex
 - Middleware: Platform to build common and integrated services, whilst hiding heterogeneous software & system access layers
- Design to support extreme scale collectively for many scalable applications
 - Community (HEP) applications, essentially similar

Extreme Scale Distributed Computing in 2013

- First generation of DC characterized by “gluing it” together
 - Many local solutions, lack of end-to-end solutions
- Inability to reason about spatio-temporal execution properties
 - Given a *general* workload there is an inability to estimate how long a workload will take? And where (and why) it will execute?
 - Complete absence of analytical models of applications, infrastructure
 - And we do not know how wrong our estimates would be!
- We are still learning how to architect large-scale systems
 - Scaling remains difficult for *individual* scientists
 - < 1% can do $O(100)$ tasks of $O(10\text{GB})$ over $O(10)$ nodes
 - Macroscopic vs microscopic theory of distributed systems!

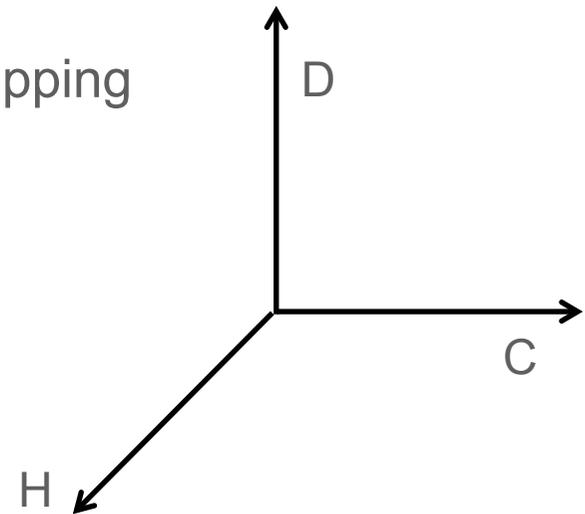
“Many Simulations” Pathway to Extreme Scale

- Problems in computational science *naturally* amenable to “many simulations” model of computing:
 - Many free energy calculations, enhanced sampling problems.
 - Many multi-physics simulations are also multi components.
- Single “application” might be broken into many smaller simulations
- This is not *just* HTC or HPC, but complex application objectives
 - Isn’t about just peak perf, nor maximal throughput
 - Given access to X cores/nodes – slice/dice or distribute as needed



From Many Simulations to Complex Applications

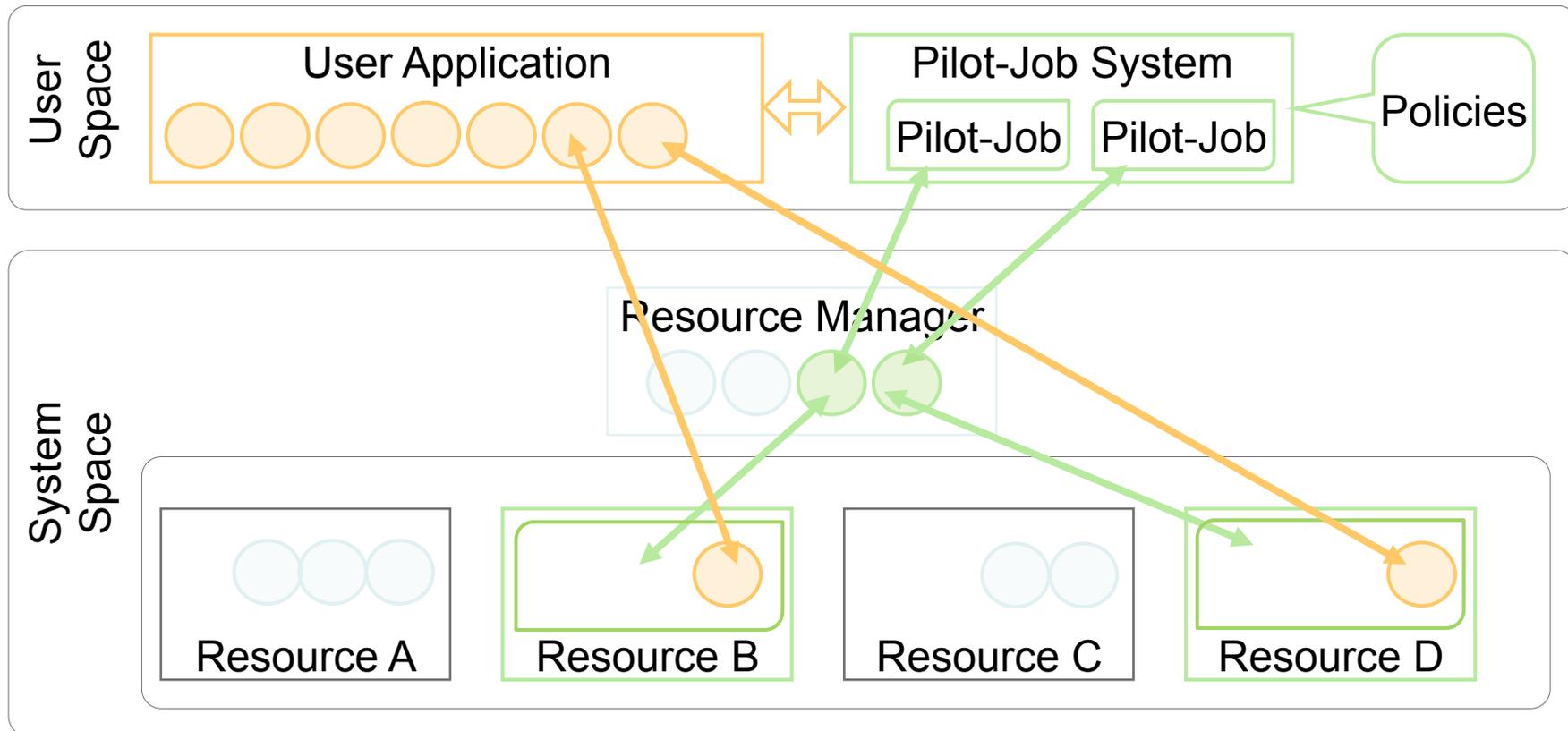
- Starting from uncoupled heterogeneous simulations, varying levels of coordination and dependency can be gradually added and “tuned”
 - Homogeneous/Heterogeneous
 - Complexity of simulation-resources mapping
 - Coupling between simulations
 - Different coordination mechanism
 - Dependencies
 - Constraints, scheduling, data transfer
- Depending upon the above properties, the importance and feasibility of distribution varies
- Need abstractions that:
 - A1: Decouple workload and resource utilization
 - A2: Dynamic Resource Utilization (Growing/shinking resource pool)



Abstractions, Models and Implementations

Pilot Abstractions

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)
 - Build higher-level capabilities without explicit resource management

Landscape of Pilot-Job Systems

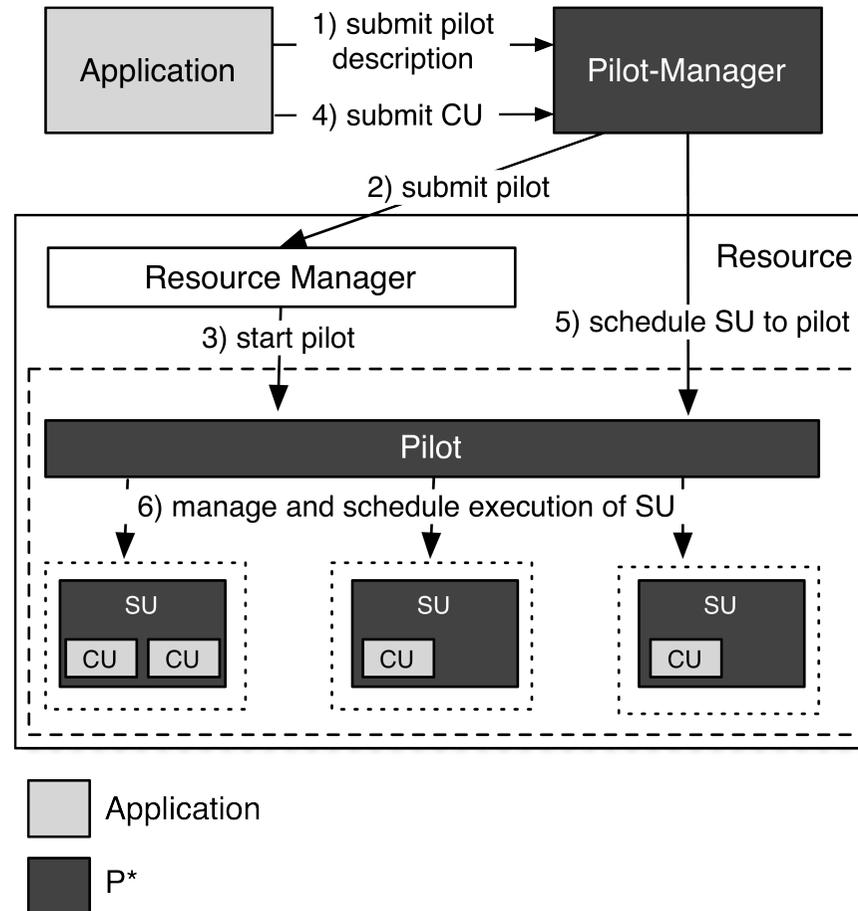
- There are many PJS offerings, often semantically distinct
 - PanDA, DIANE, DIRAC, Condor Glide-In, SWIFT, ToPoS Falkon, BigJob...
 - *Why do you think there has been a proliferation of PJs?*
- Conceptual & practical barriers to extensibility (& interoperability)
 - The landscape of PJS reflects, in addition to PJS specifics, the broader eco-system of distributed middleware & infrastructure
 - Software Engineering issues, interfaces, standardization
- 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?
- Data remains a dependent variable, not a primary variable
 - Introduce the concept of Pilot-data

Pilot-Jobs (PJ): Five Myths

- PJs do not need well defined architecture, model and semantics, or PJs are such a simple concept, it doesn't need more “attention”
 - Not to confuse “simple to use” with simple to design“
- PJ 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 are only about meta-scheduling/reducing Q delays/unfairly game HPC
 - There are interesting usage modes beyond “cycle stealing”
- PJ do not help with next-generation “data-intensive” applications
 - PJ for NGS O(10-100) GB per task on existing DCI

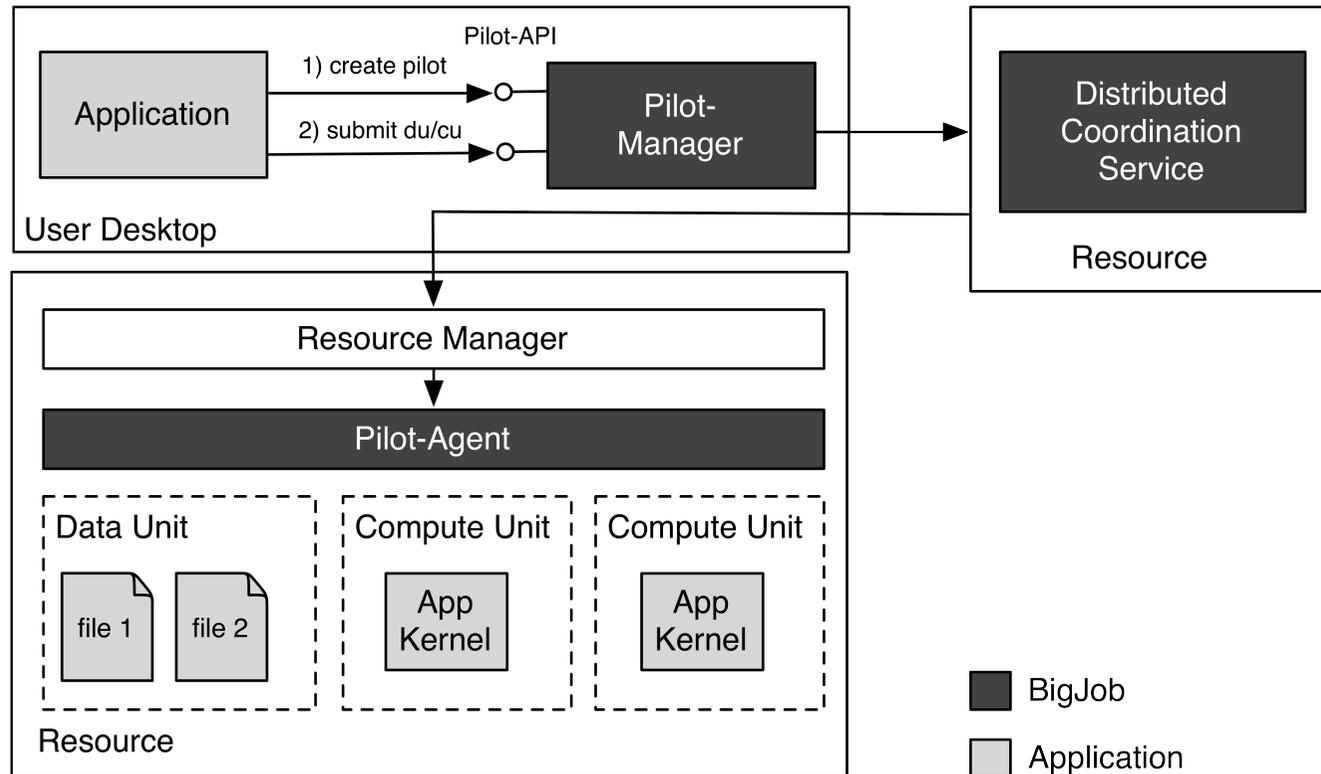
P* Model: Elements, Characteristics and API

- 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.

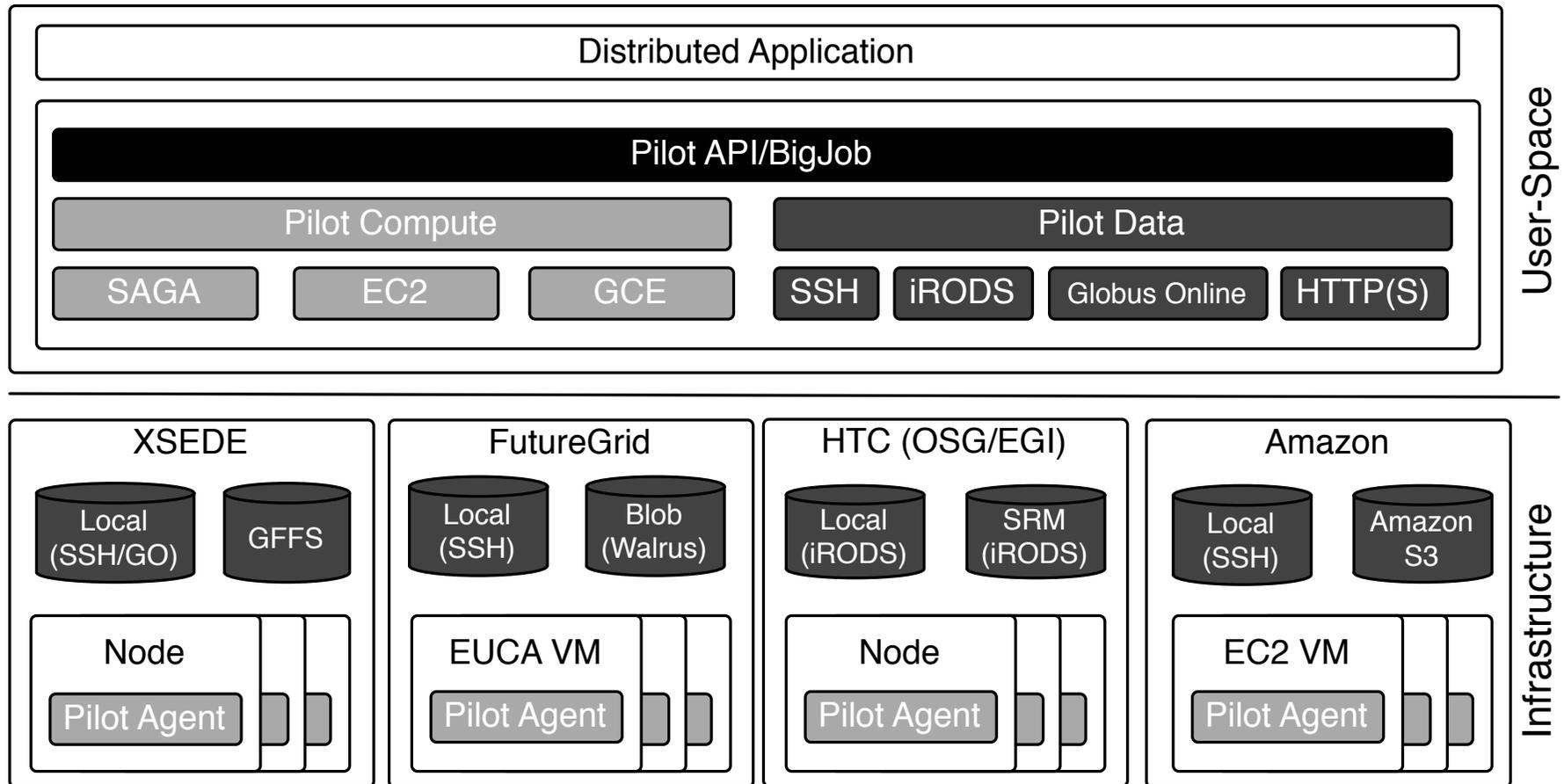


“P*: A Model of Pilot-Abstractions”, *8th IEEE International Conference on e-Science 2012*, 2012

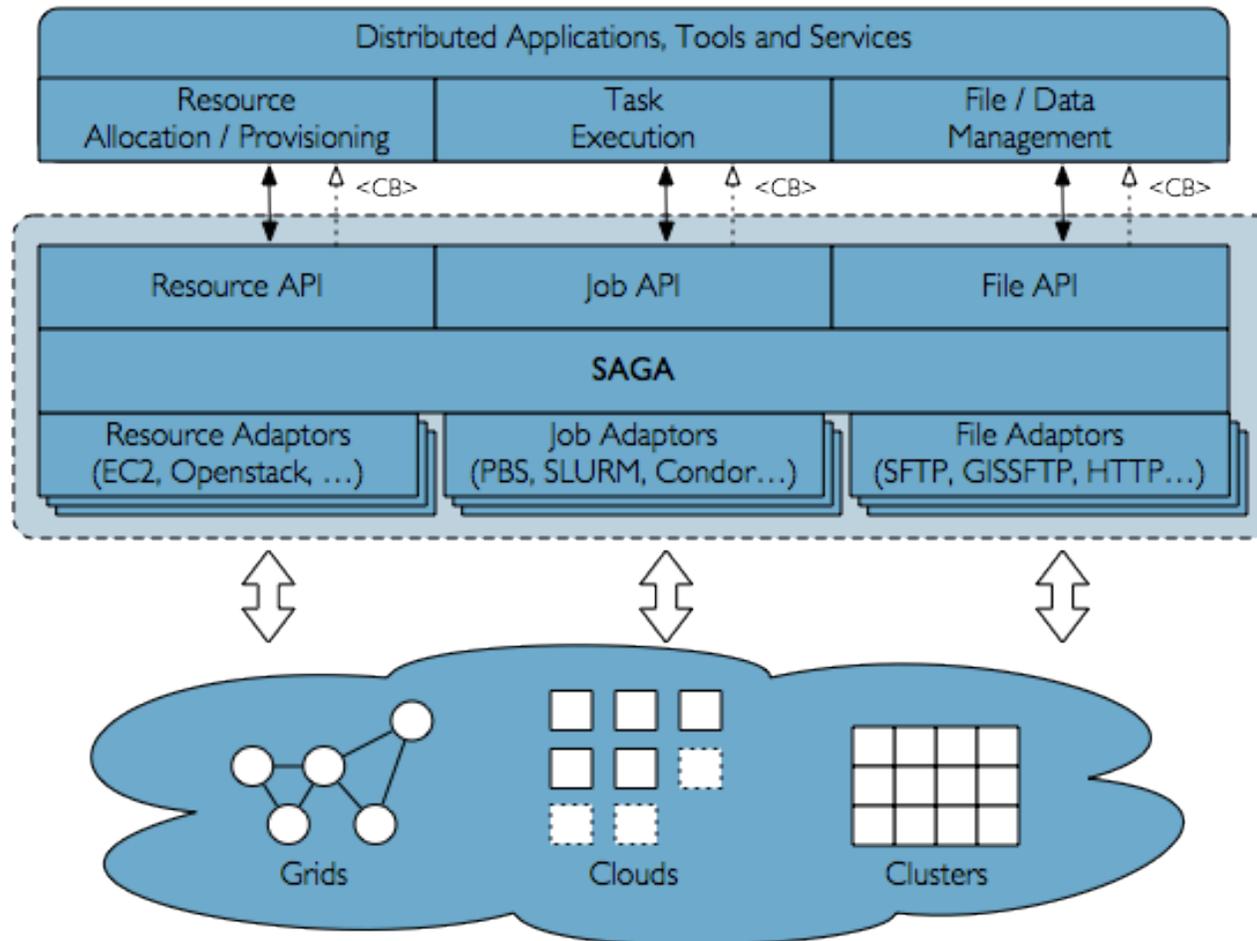
BigJob: Architecture



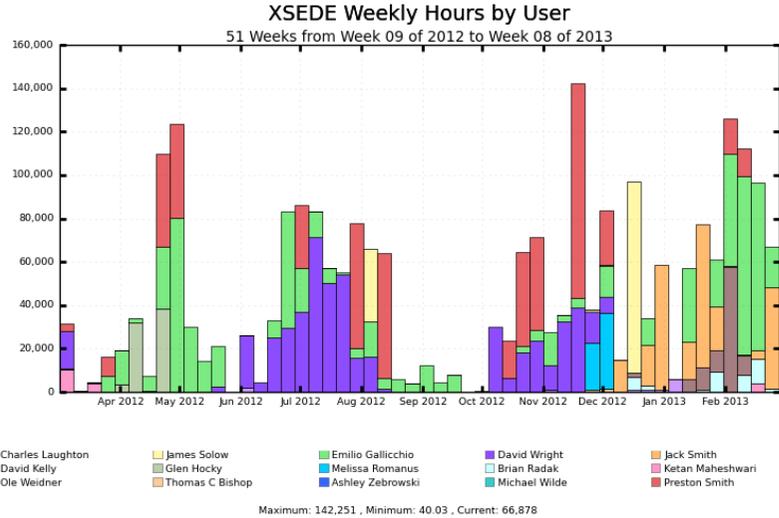
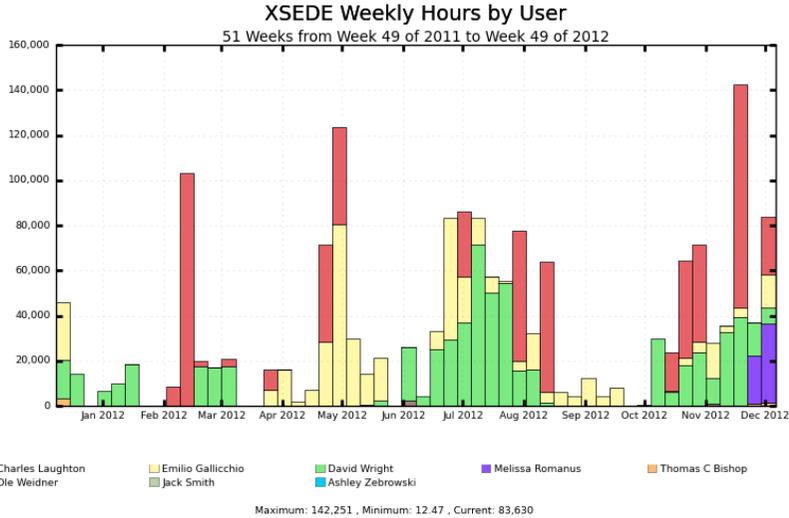
Supported Infrastructures



SAGA Interoperability Layer for BigJob



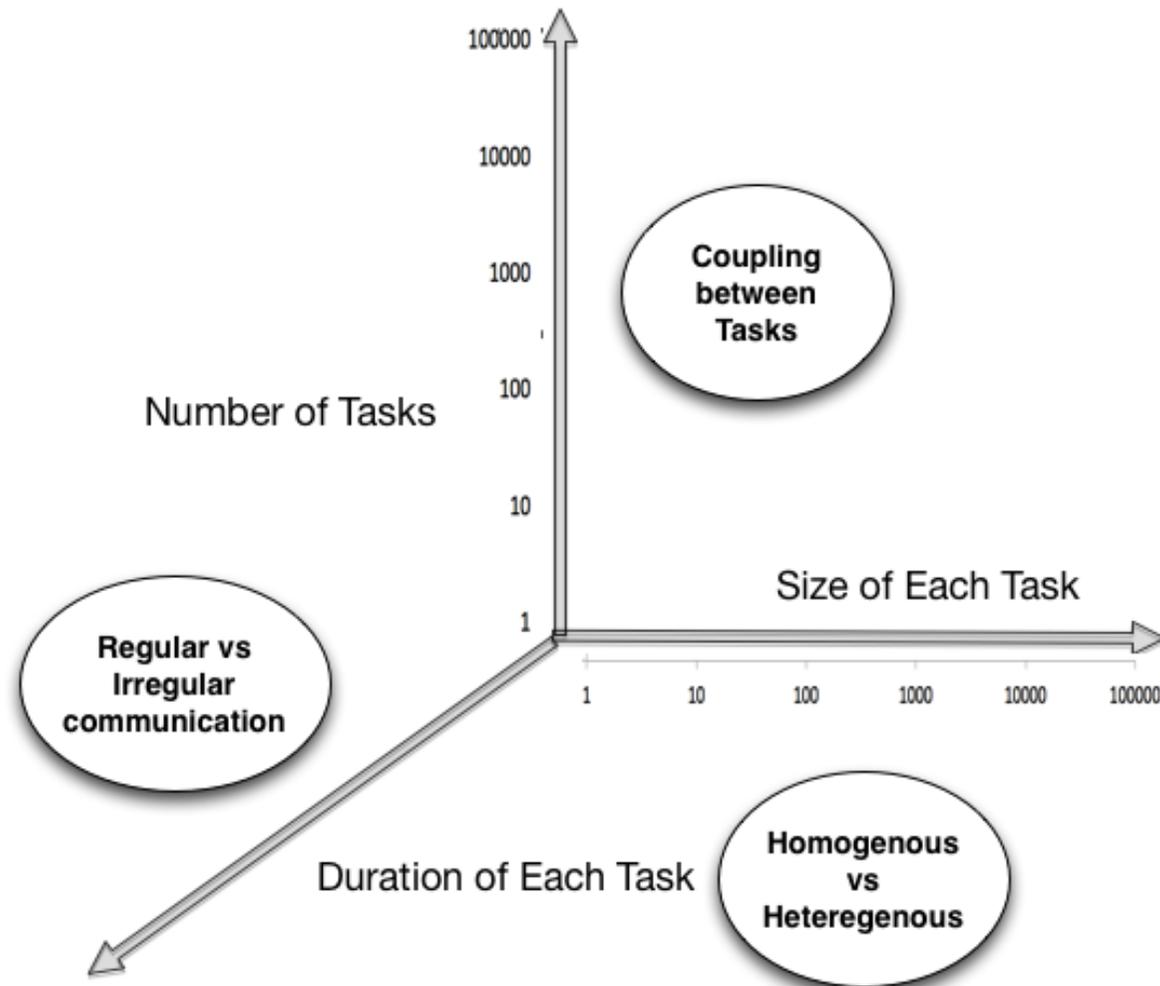
BigJob: (Partial) Usage on XSEDE Machines



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

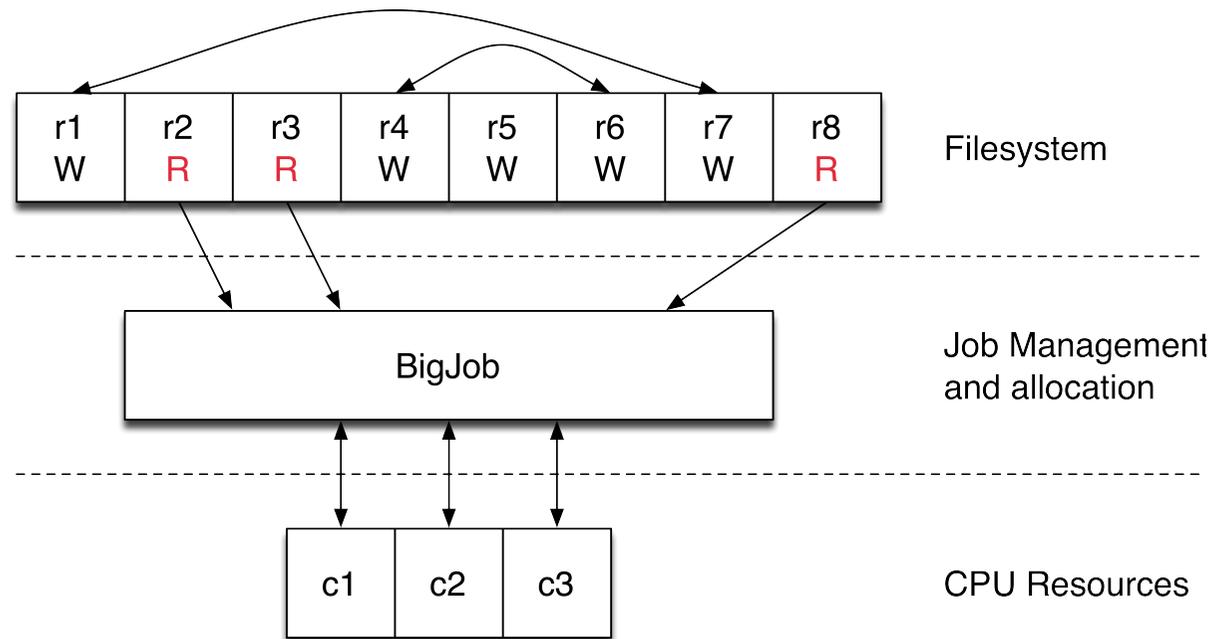


Scaling Along Many Dimensions



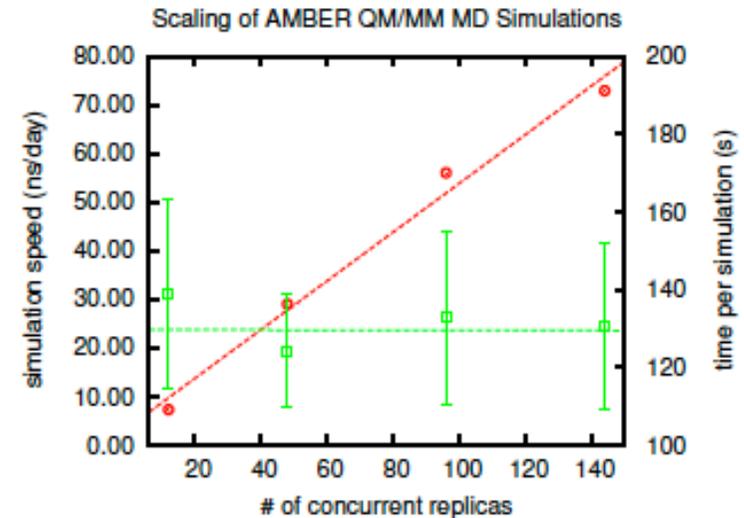
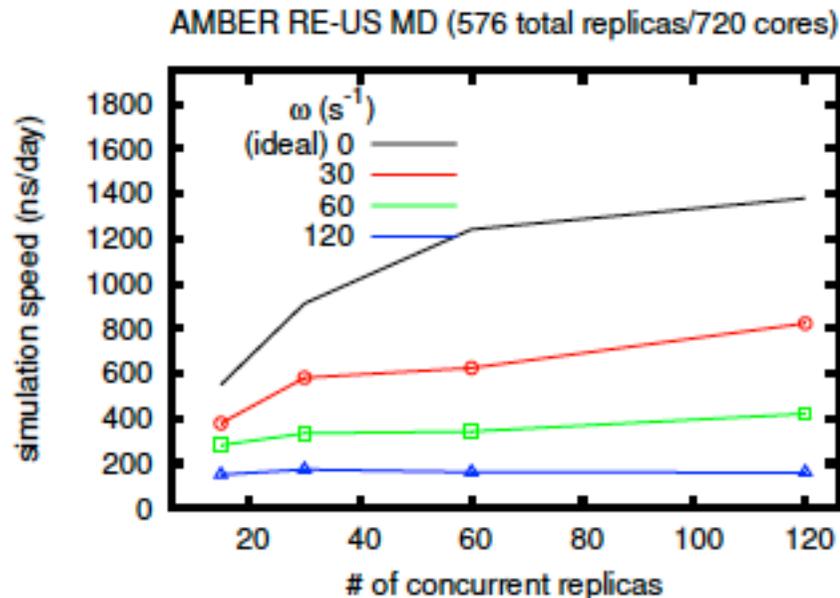
Async Replica-Exchange Library

- Built to perform file-based asynchronous parallel replica exchange.
- Example of a Platform independent library.



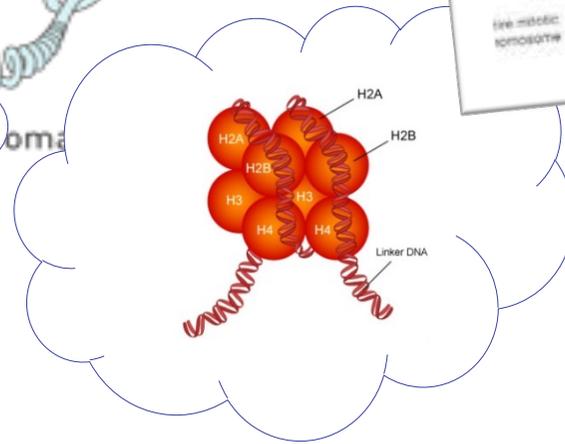
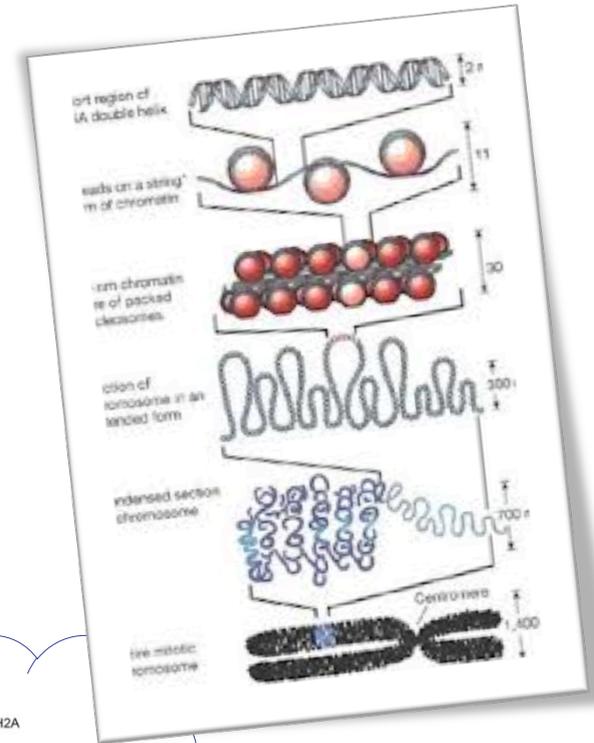
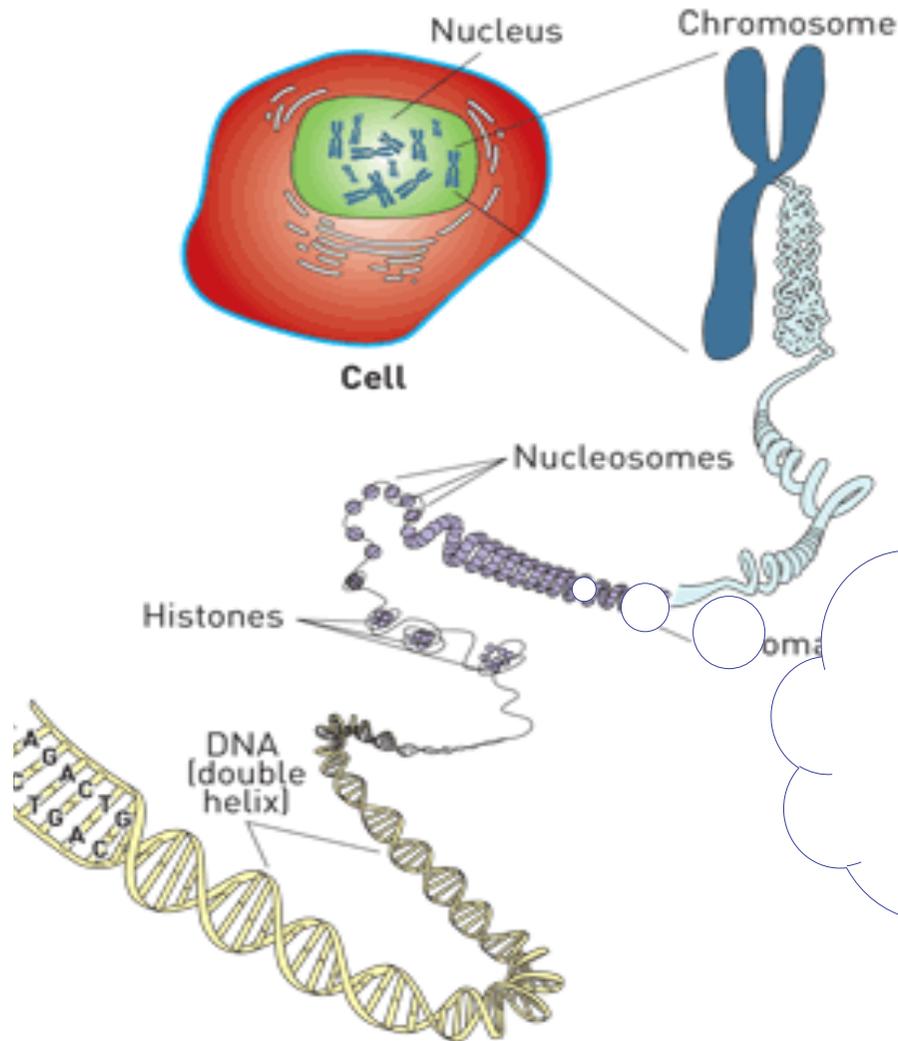
<https://github.com/saga-project/asyncre-bigjob>

Async Replica-Exchange Library



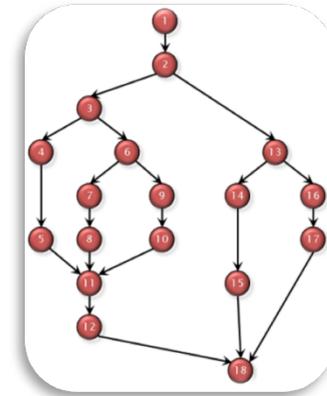
- Ideal performance considered to be zero coupling in this case.
- Diminished results due to coordination overheads.
- Scientists are free to choose the best tradeoff between simulation speed and number of concurrent replicas.
- BigJob-based Repex: Amongst the earliest QM/MM.

Scalable online Genomics



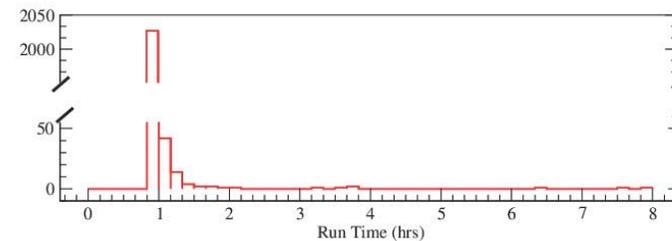
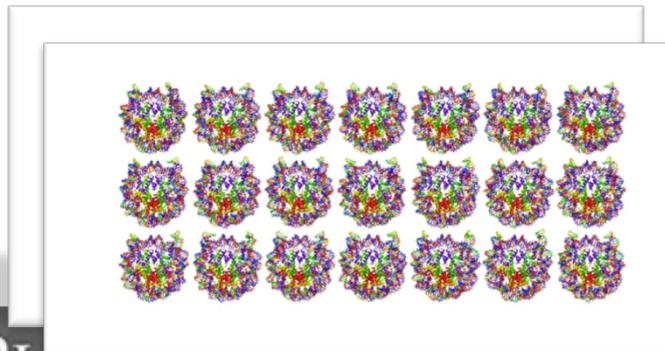
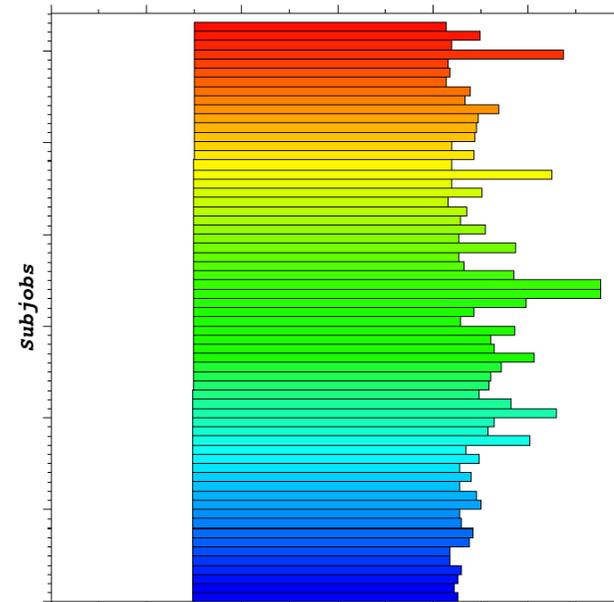
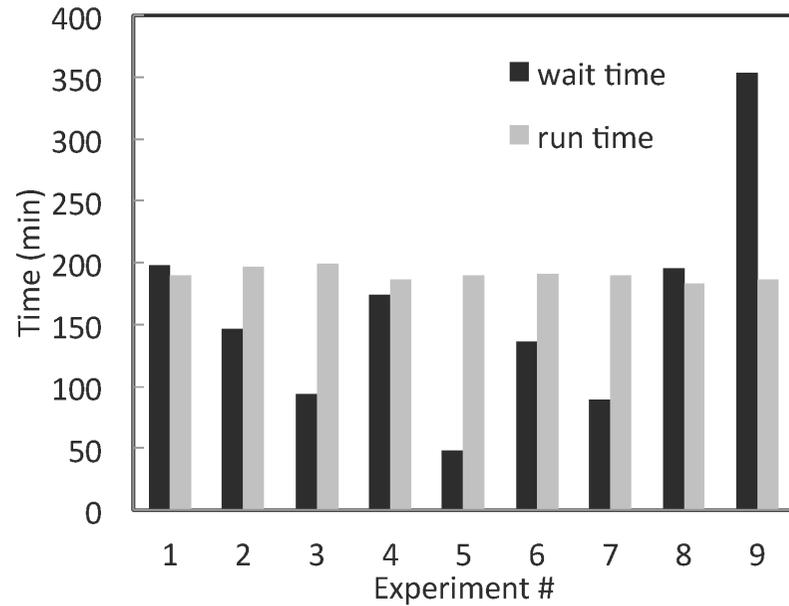
Computational Workflow

- **20ns simulation broken into chained sequence of 20 1ns simulations**
 - Output of each required as input to next
- **Hierarchical directory structure**
 - | -> Common Config files (one for each sequence)
 - | -> 5 nucleosome-free regions of chromosome
 - | - - Common Param files for each system
 - | - -> 21 threading position
 - | - - - 20 chained sequences
- **Data flow**
 - COOR, VEL, XSC → COOR', VEL', XSC' + **DCD**, **DVD**, XST, OUT, ERR
 - \ _____ /
- **Determining successful completion**
 - “WallClock” at end of OUT, and no “FATAL”
 - Size of DCD

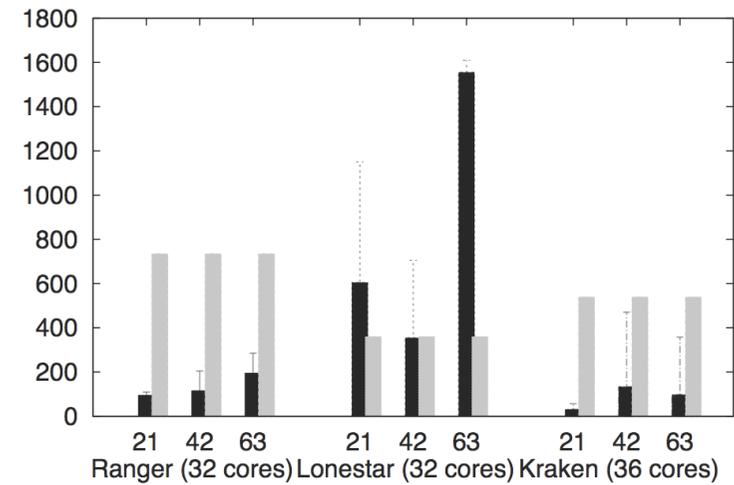
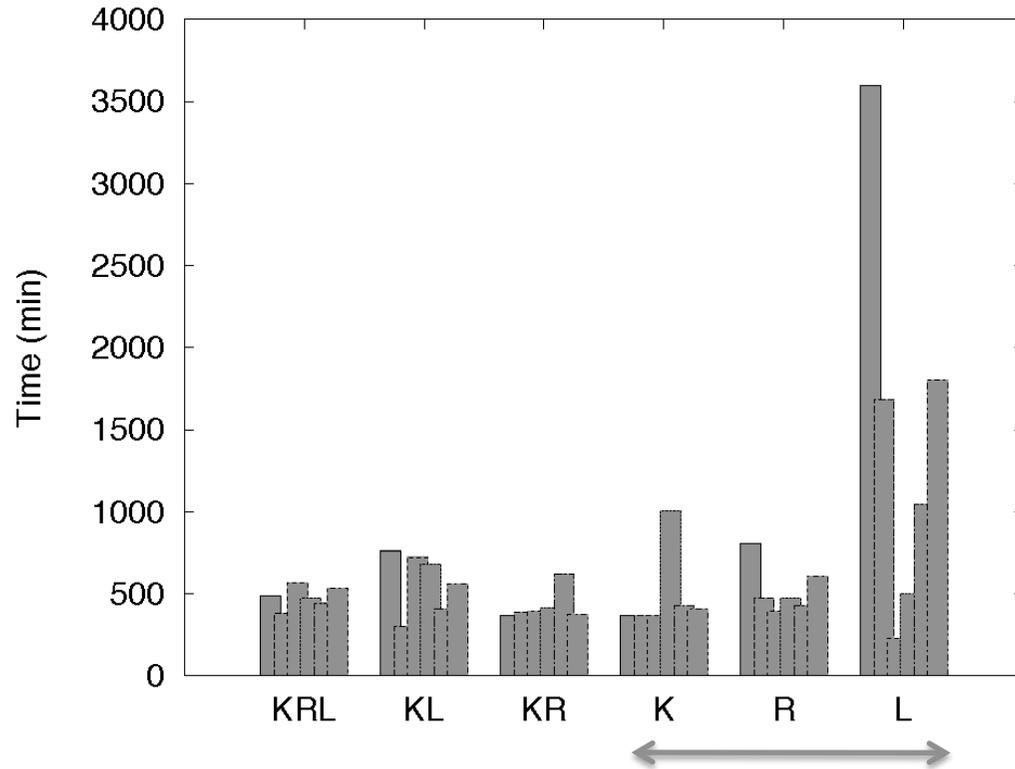


HT-HPC on Kraken

126 ensembles, each of 192 cores = 24192 cores



Scale-Out

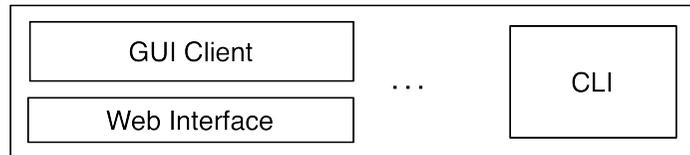


X-axis: number of tasks (size)

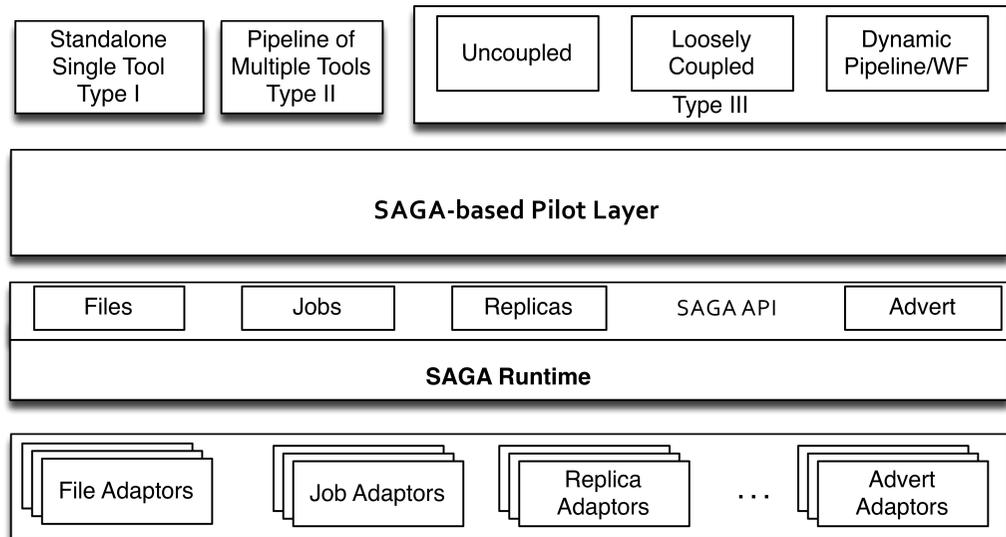
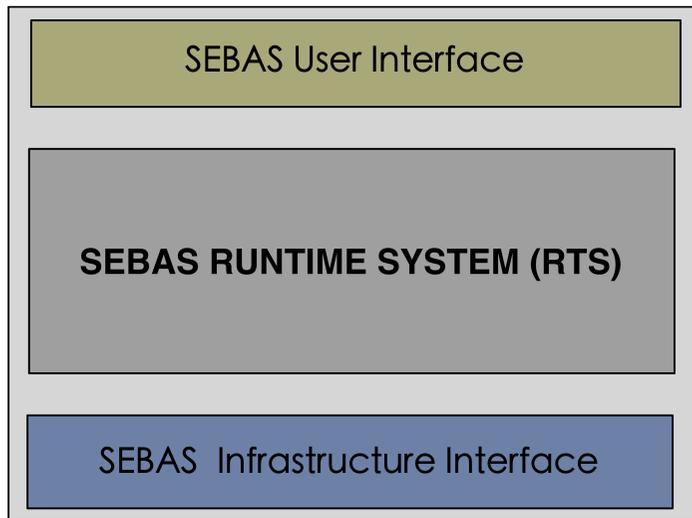
“Coarse-Grained” BigJob Performance

- Number of zero-payload tasks that BJ can dispatch per second:
 - Distributed: $O(1)$
 - 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 :
 - $O(100) \times O(1000)$
- (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

Scalable, Extensible HT Binding Energy Calculation



- Platform independent library.
- Suggestions for other libraries are welcome!



Compute and Data Infrastructure
e.g. UK-NGS/Hector, Campus resources, US-XSEDE, Clouds

In consultation with Peter Coveney and Charlie Laughton.

Pilot-Data: Design Objectives

- Abstraction for managing the computing requirements of distributed dynamic data
 - Dynamic Data: spatio-temporal variations, source/destination
- Enable reasoning about **distributed** and **dynamic** resources compute, storage and network
- Remove lower-level details:
 - Access to heterogeneous backend infrastructures
 - file I/O and networking
 - synchronization between compute and data
- Enable data-aware **decision making**:
 - Exploit data locality whenever possibly.
 - Enable “applications” to control typical trade-offs:
 - data movement, anticipated compute-time

What is Pilot-Data?

- Manage (dynamic) storage resources in conjunction with computational task placement
- Unified access layer to different heterogeneous storage backend and access layers: SRM, iRODS, Globus Online, S3
- Higher-level abstraction to manage distributed and dynamic data/compute in (geographically) distributed systems:
 - Data Unit: Grouping of files that are accessed together
 - Manage complex data flows consisting of multiple steps of compute across (geographically) distributed resources
- Co-location and co-scheduling of compute and data

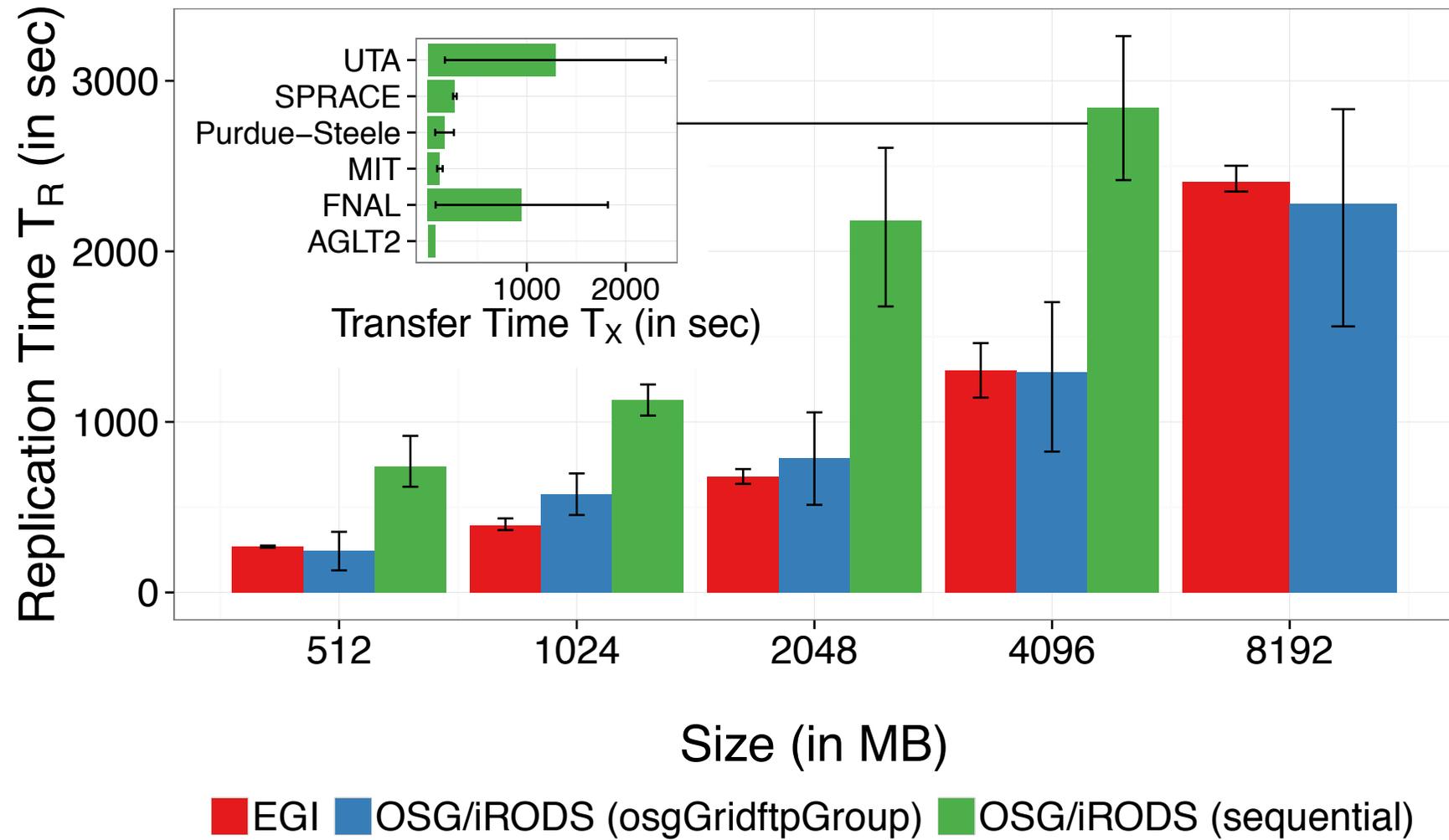
Pilot-Data on OSG

- OSG proposes three usage modes for data-intensive applications:
 - Condor-based file staging
 - SRM
 - iRODS
- Complex Decision Matrix:

	Condor Filestaging	SRM	iRODS	Pilot-Data
Data Volume	Low	High	High	Backend-specific
Complexity	Low	High	Medium	Backend-specific
Data Distribution	Local	Local	Local, Geographic	Local, Geographic
Data Replication	No	No	Yes	PD and System- specific replication
Flexibility (Multi-stage applications, data reuse, multiple infrastructure)	Low	Low	Low	High

Pilot-Data enables the user/application to tradeoff the different characteristics of data cyberinfrastructures

Pilot-Data: Controlled, Coordinated Replication

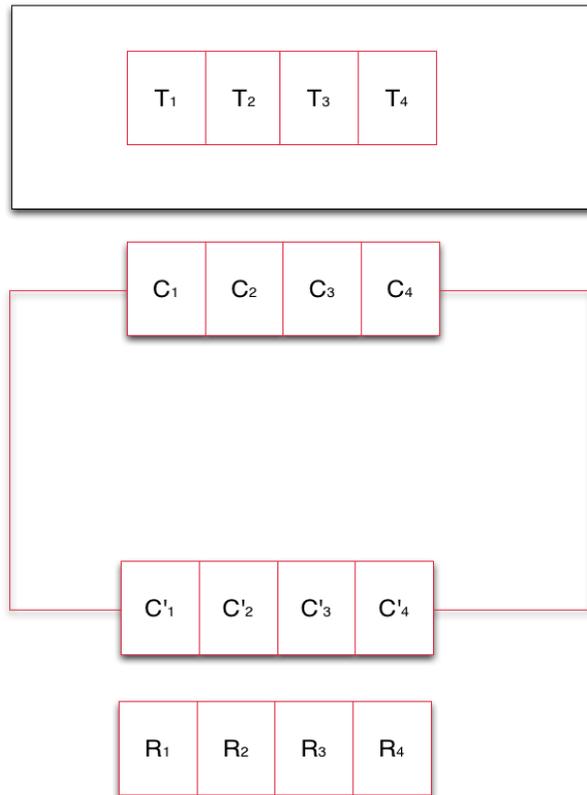


Towards NGMW: A RADICAL Perspective

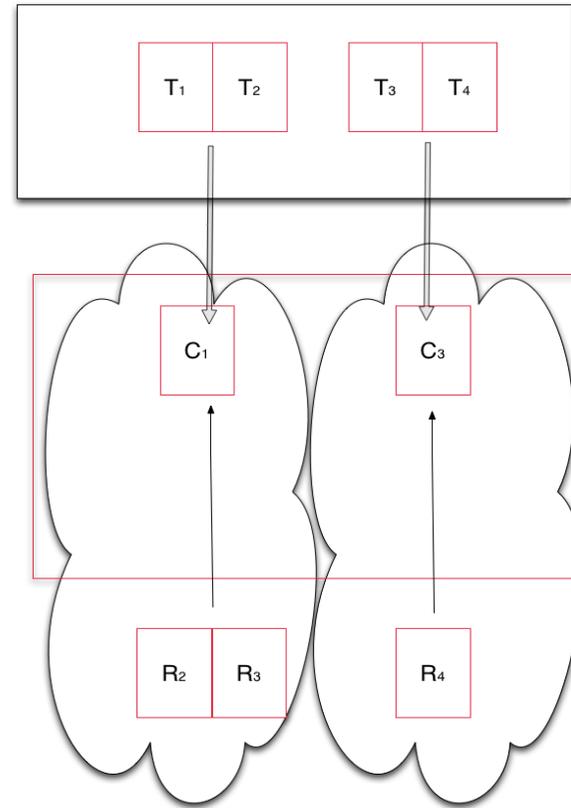
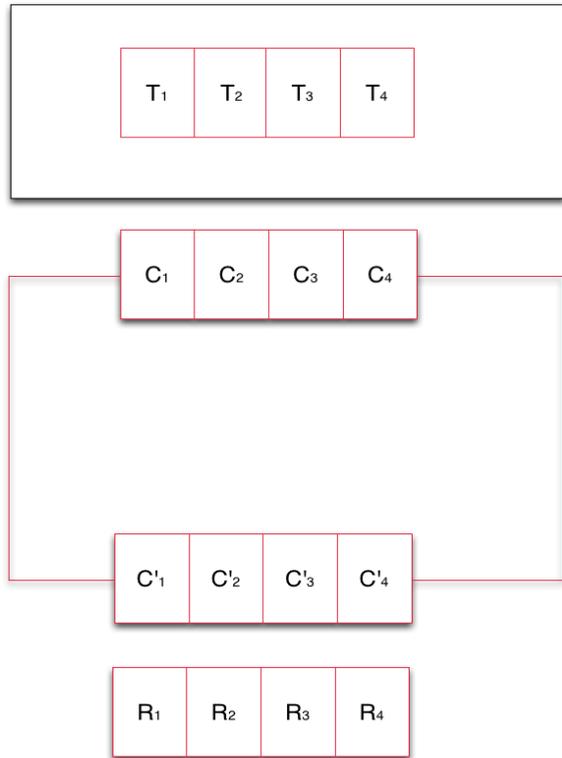
NGMW Functional Aims and Requirements

- Functional Aims
 - “Beyond glue” to support spatio-temporal execution reasoning
 - What to distribute? Where/how to distribute? When to distribute?
 - Estimate time to completion?
 - Exposes well-defined **capabilities** rather than technology
 - **Capability**: Well-defined and aggregated functionality, without regard to how, or the specific approach used, e.g., num. of tasks, throughput, probabilistic bounds on time-to-completion
- Functional Requirement
 - Support adaptive applications in conjunction with dynamic resources
 - Federate heterogeneous infrastructure to provide well-defined capabilities

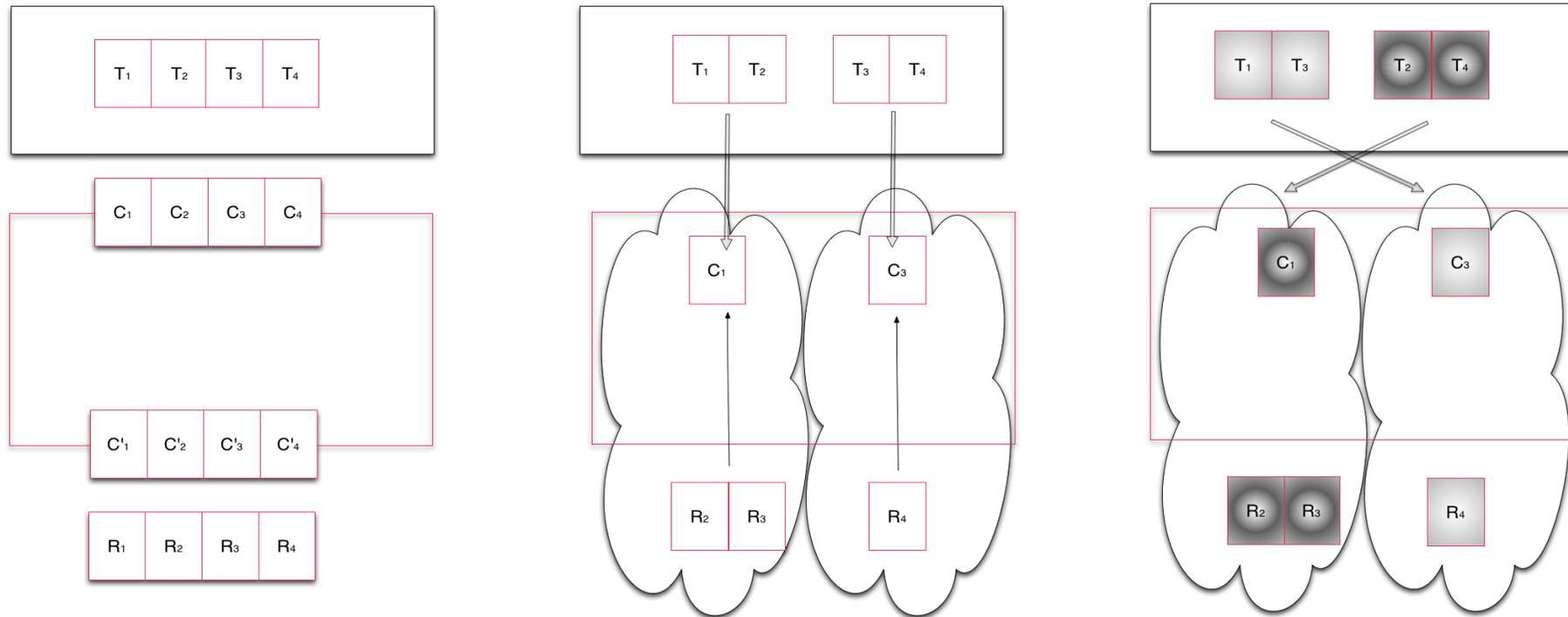
NGMW Schematic



NGMW Schematic

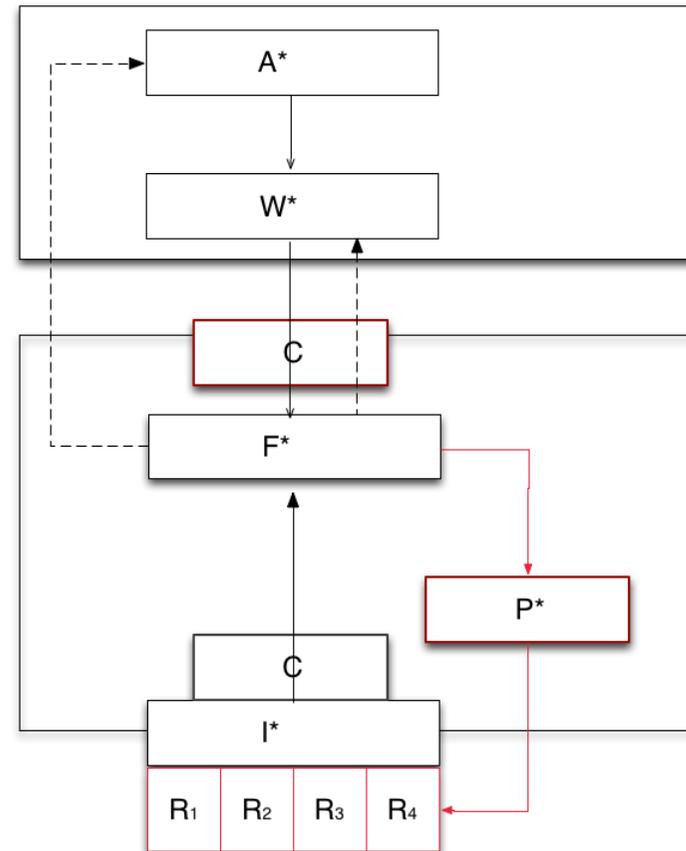


NGMW Schematic



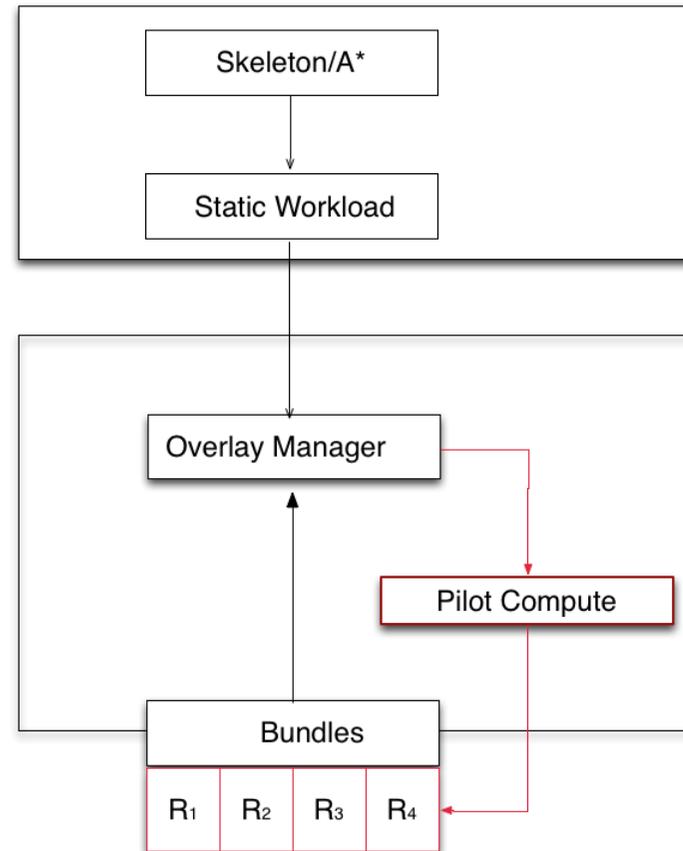
Design Objective: Multi-level Integrated Reasoning

- Transformation of application workload via system workload to infrastructure capability
- Application requests $R(100, T, 10)$, say 100 tasks, of type T , complete within 10 units of time
- Federation Layer/Manager responds with collective capability of $C(50, T, 10)$ or $C(100, T, 20)$
- Adaptive Application
 - Adaptivity can be either at A, W level
 - Application may self-throttle number of tasks, or type of task generated
 - Or workload description can be changed to meet the capability



AIMES: Demonstration of Flexible Federation (SC'13)

- Application say Bag-of-Tasks
 - Say BoT(100, H, 10)
- Generate similar workload description from different application representations
- Bundles currently support federation
 - Info on resource availability
 - Eventually resource properties
- Ultimately bundles (and I*) should be consistent with C*
- Formalize the advantages of dynamic and flexible federation
 - Performance improvements



Conclusion

- Extreme-Scale Distributed Computing (XSD)
 - Status Quo: Understanding the landscape of XDC-2013
 - Beyond HPC/HTC: Requirements for “many simulations” scenarios
- Abstractions, Models and Implementations for many-simulations
 - Five Myths associated with Pilot-Jobs
 - Address using Abstractions, Models and Implementations
 - P* Model of Pilot-Abstractions
- Future Directions: Next Generation Middleware (NGMW)
 - Many aspects and considerations, but focus on resource management for many simulations

References

- RADICAL:
 - <http://radical.rutgers.edu/>
- Publications:
 - <http://radical.rutgers.edu/publications>
 - (i) P*, (ii) Pilot-Data, and (iii) Pilot-Jobs Review Paper
- SAGA-Python:
 - <http://saga-project.github.io/saga-python/>
- BigJob: An implementation of P*
 - <http://saga-project.github.io/BigJob/>
- Tutorials:
 - <https://github.com/saga-project/tutorials/wiki/XSEDE13>

Acknowledgements

Graduate Students:

- Ashley Zebrowski
- Melissa Romanus
- Mark Santcroos
- Antons Trekalis

Undergraduate Students:

- Vishal Shah

Research Scientists:

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

Acknowledgements/Funding Sources

Active:

- NSF CAREER Award 2012 (OCI-1253644)
- CDI NSF-CDI (NSF CHE 1125332)
- ExTENCI (NSF OCI)
- SCIHM NSF-OCI (OCI-1235085)
- AIMES DoE-ASCR (DE-FG02-12ER26115)
- ExTASY CHE-1265788

Compute Time:

- NSF TeraGrid TRAC award TG-MCB090174
- NSF FutureGrid Award (No. 42)

Recent Past:

- NSF/LEQSF (2007-10)-CyberRII-01
- NSF HPCOPS NSF- OCI 0710874 award
- UK EPSRC (GR/D0766171/1) and e-Science Institute, UK
- NSF OCI 1059635
- NIH Grant Number P20RR016456