Module E:
Distributed Scientific Computing

Lecture E-1 and E-2: Introduction to the Theory and Practice of Distributed Computing

Dr Shantenu Jha

http://radical.rutgers.edu

Module Dynamics

- Shantenu Jha [first-name at lsu dot edu]
  - Email/skype
  - Other matter: Coordinate with Dr Frank Loeffler
- Grading:
  - 2 Assignments [5% x 2]
  - 1 Project [10%]

Overview of Module E
Distributed Scientific Computing

- E1: Introduction to Distributed Scientific Computing
  - Examples of Distributed Applications
    - WLCG + Application Examples
    - Summarize and compare with HPC
  - Examples of "Production" Grid Infrastructure
    - HPC vs HTC, Research vs Production, Commercial
  - Distributed Computing is Hard. Why?
- E2: Introduction to the Practice of DSC
  - Introduction to SAGA
  - Introduction to Pilot-Jobs (BigJob)
- Mod E Project

Overview of Module E
Distributed Scientific Computing

- E3: Introduction to Cloud Computing
  - Introduction to Commercial / Enterprise DC aka Cloud Computing (EC2, Azure, Google) [30mins]
  - Using FutureGrid [30mins]
  - Using the Master-Worker Pattern [15mins]
- E4: To Distribute or not to Distribute?
  - Distributed Applications Redux [20mins]
  - Principles of Distributed Computing [20mins]
  - Project Discussion/Presentation [40mins]

E1: References

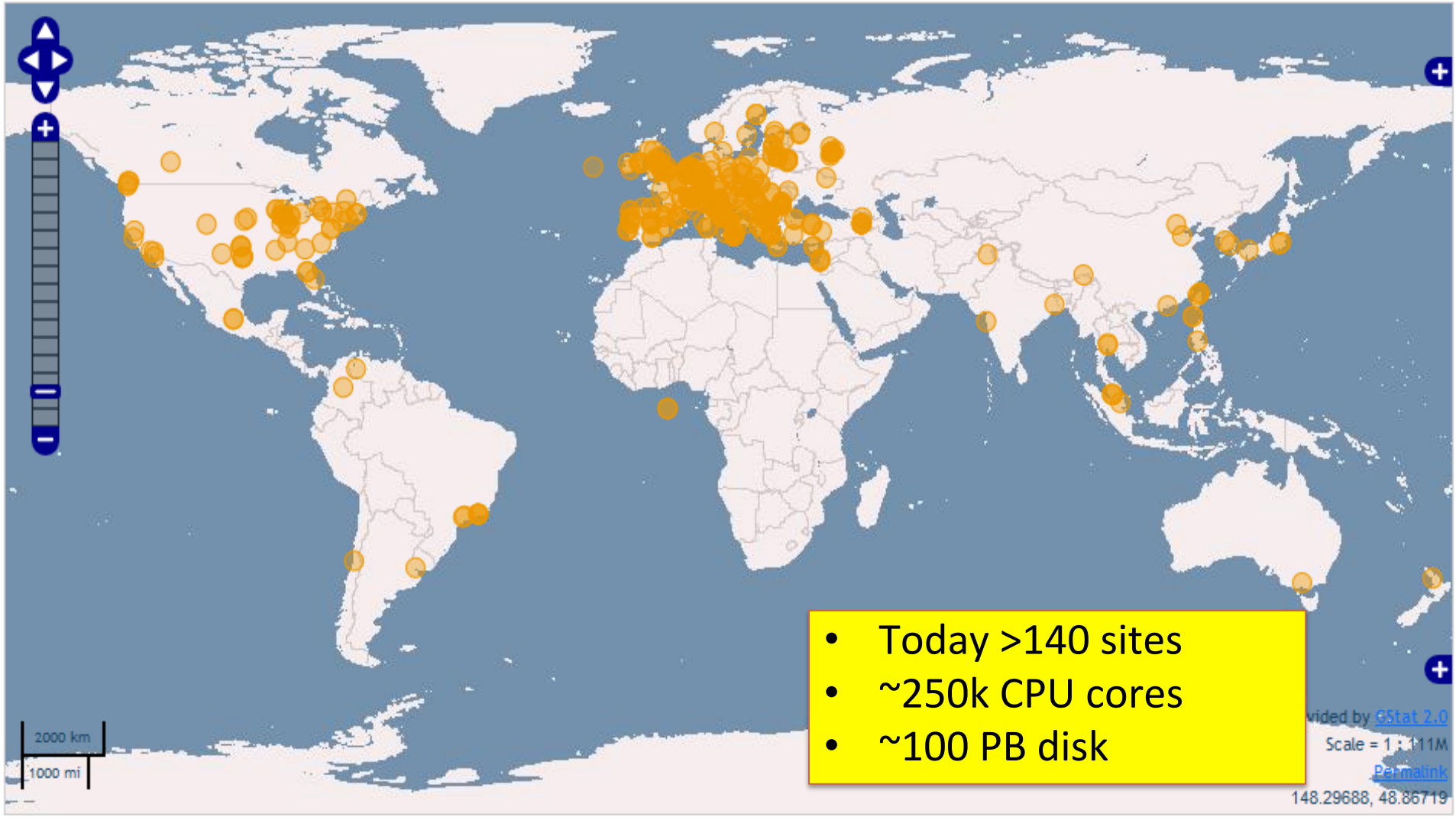
- Distributed Computing Practice for Large-Scale Science & Engineering Applications:
  - https://www.github.com/saga-project/radical.wp/raw/master/publications/pdf/dpa-surveypaper_draft.pdf
- D. S. Katz, S. Jha, M. Parashar, O. Rana, and J. Weissman. Survey and Analysis of Production Distributed Computing Infrastructures. Technical Report CI-TR-7-0811. Computation Institute, University of Chicago & Argonne National Laboratory:
  - http://www.ci.uchicago.edu/research/papers/CI-TR-7-0811

Distributed Applications

- We will analyze a few "representative" DA
  - Montage: Image Mosaicking
  - Distributed-MPI / Meta-computing Simulations
  - Ensemble-based, Replica-Exchange Simulations
  - ClimatePrediction.net
  - SCOOP [Roughly put: Hurricane Modelling]
- For each application we will understand four features:
  - A Brief Overview of the Application
  - Why it is (was) distributed?
  - How is the Application Distributed?
  - What are the challenges (or successes) ?

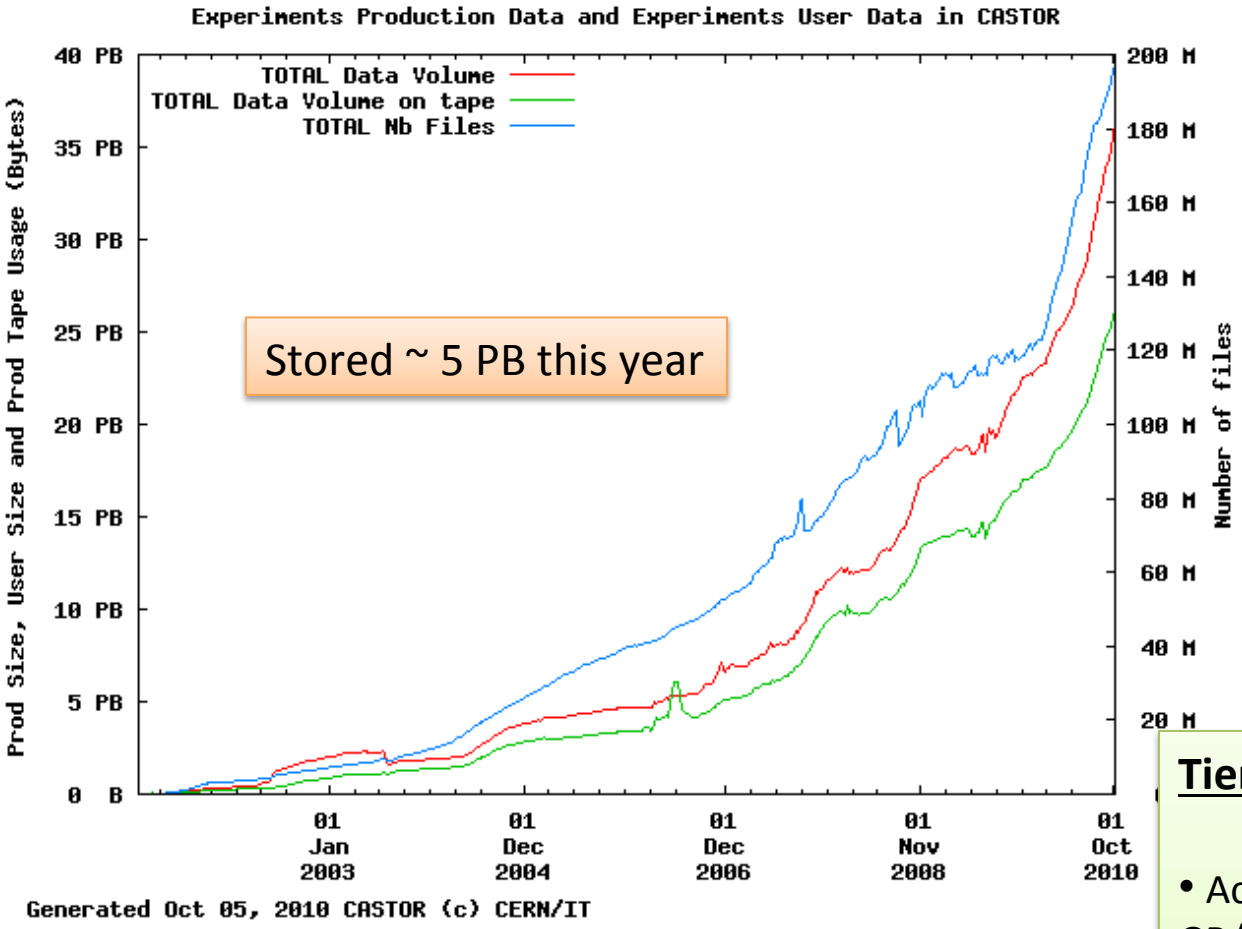
WLCG: Worldwide LHC Computing Grid
A Driver of Grid Comuting

Worldwide resources for WLCG



- Today >140 sites
- ~250k CPU cores
- ~100 PB disk

6 months of LHC data



Stored ~ 5 PB this year

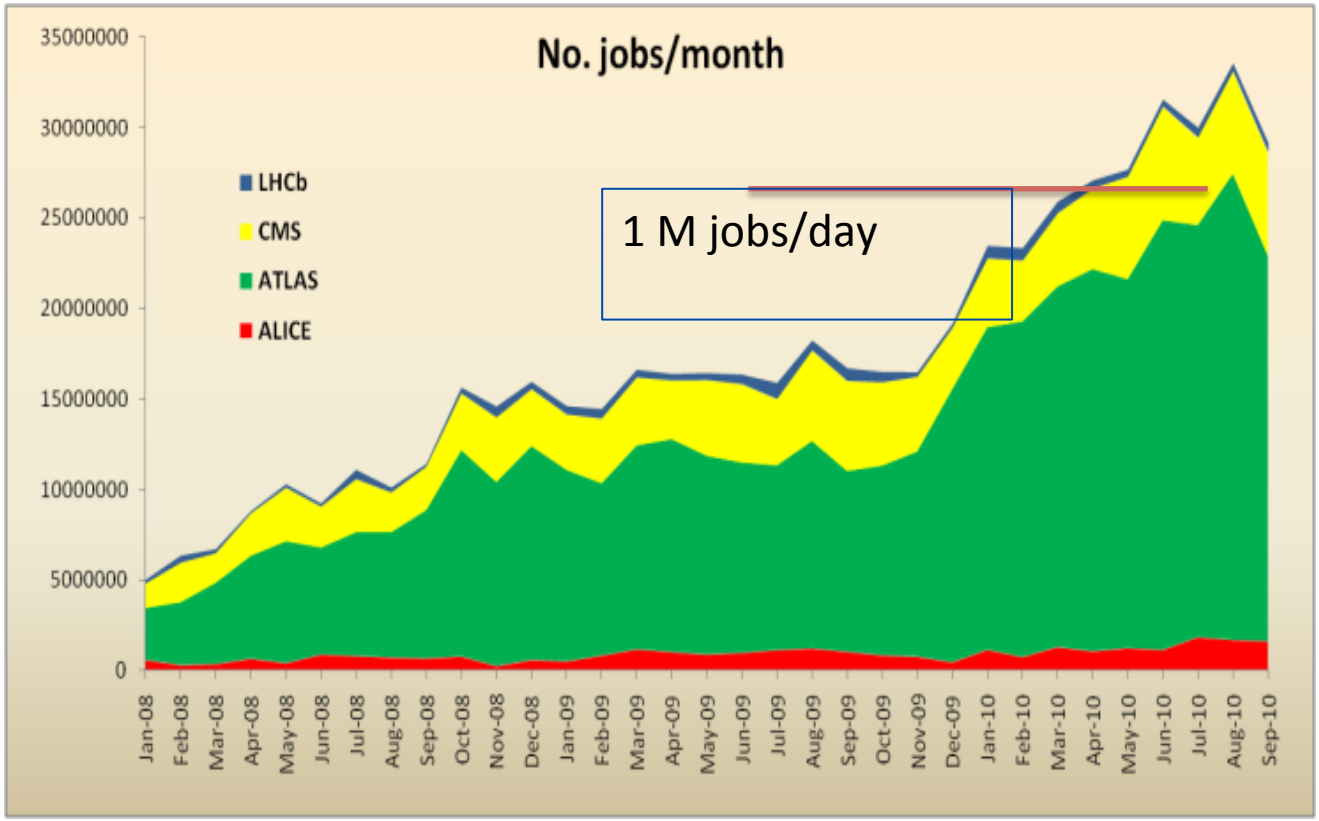
Tier 0 storage:

- Accepts data at average of 2.6 GB/s; peaks > 7 GB/s
- Serves data at average of 7 GB/s; peaks > 18 GB/s
- **CERN Tier 0 moves ~ 1 PB data per day**

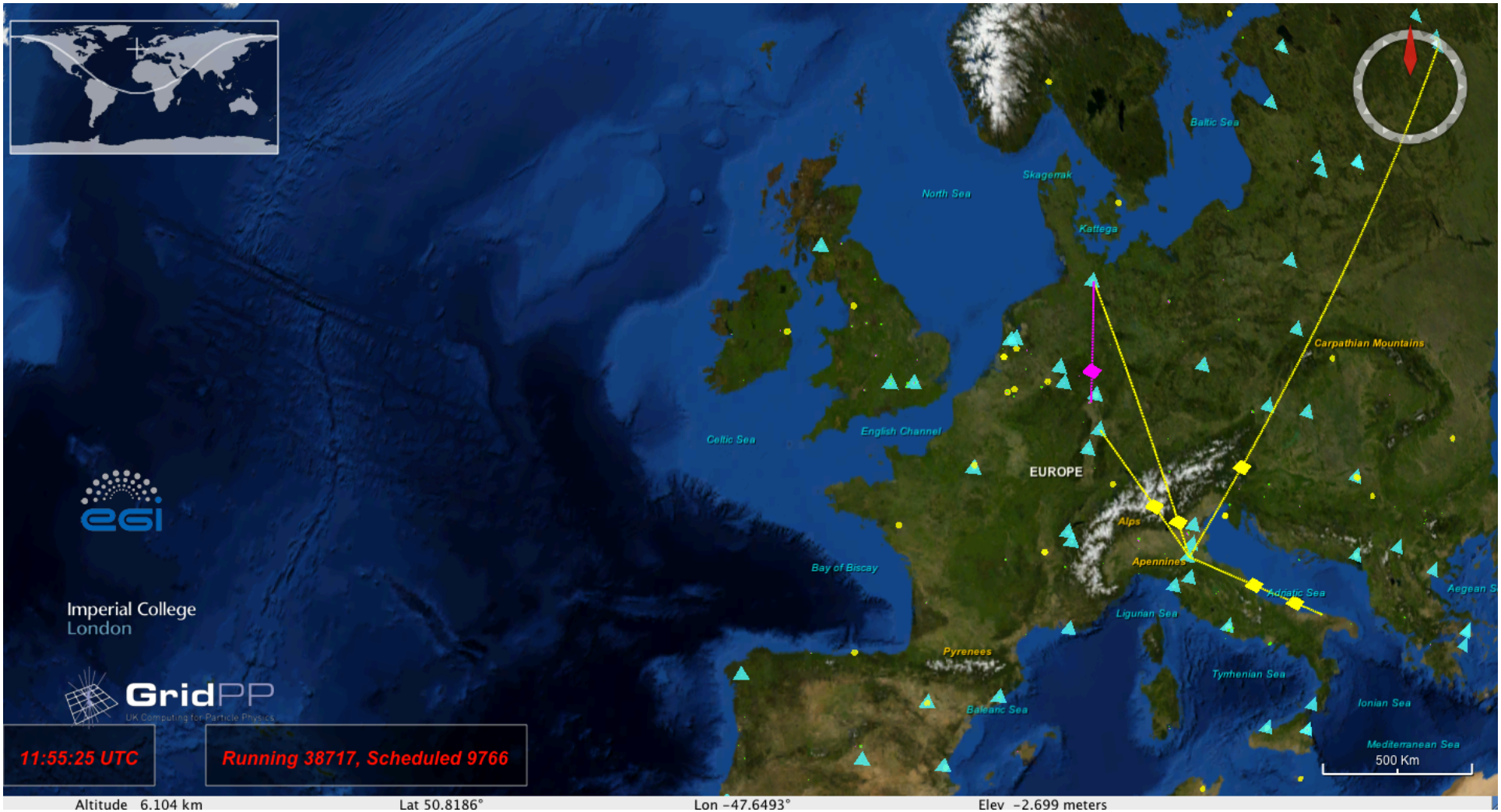
WLCG Usage

- Use remains consistently high
  - 1 M jobs/day; >>100k CPU-days/day
  - Actually much more inside pilot jobs

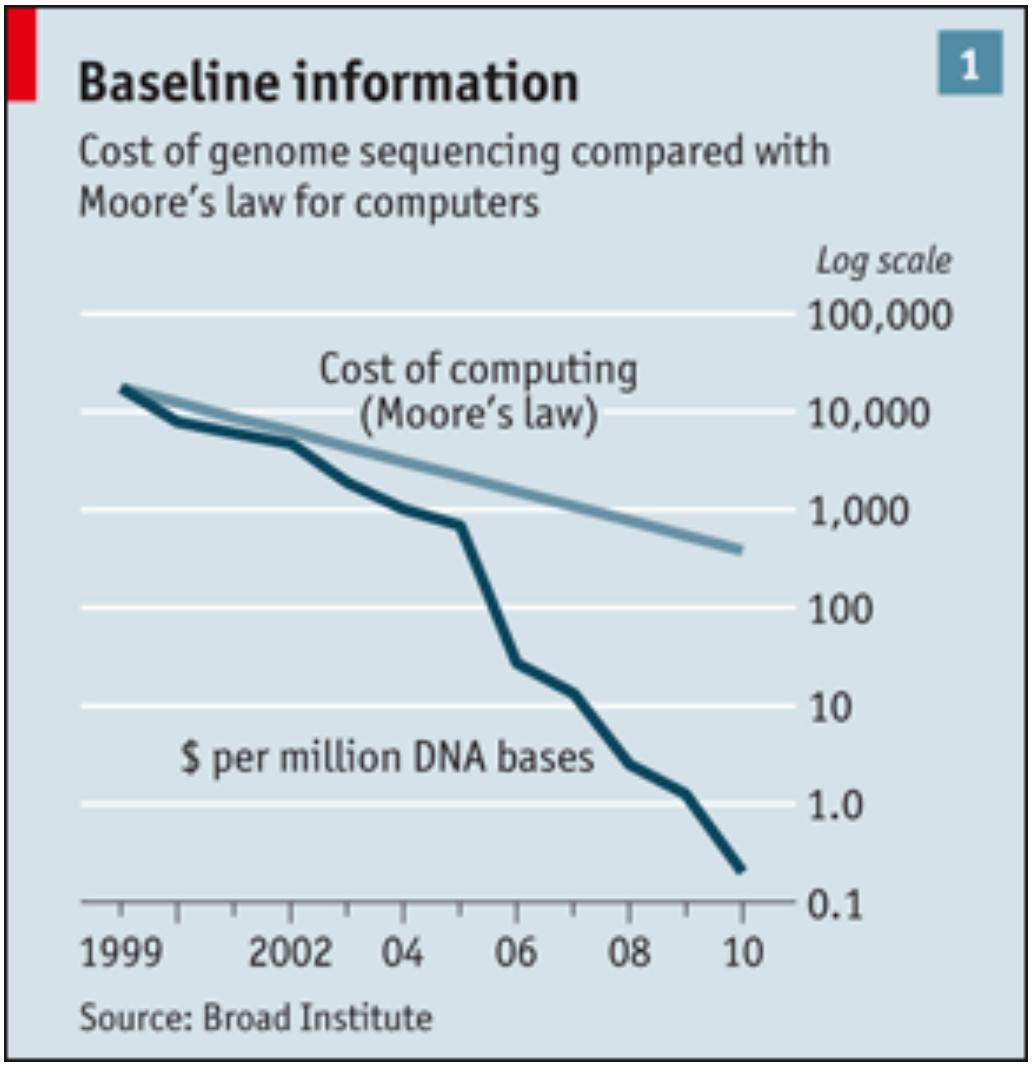
~50% utilisation



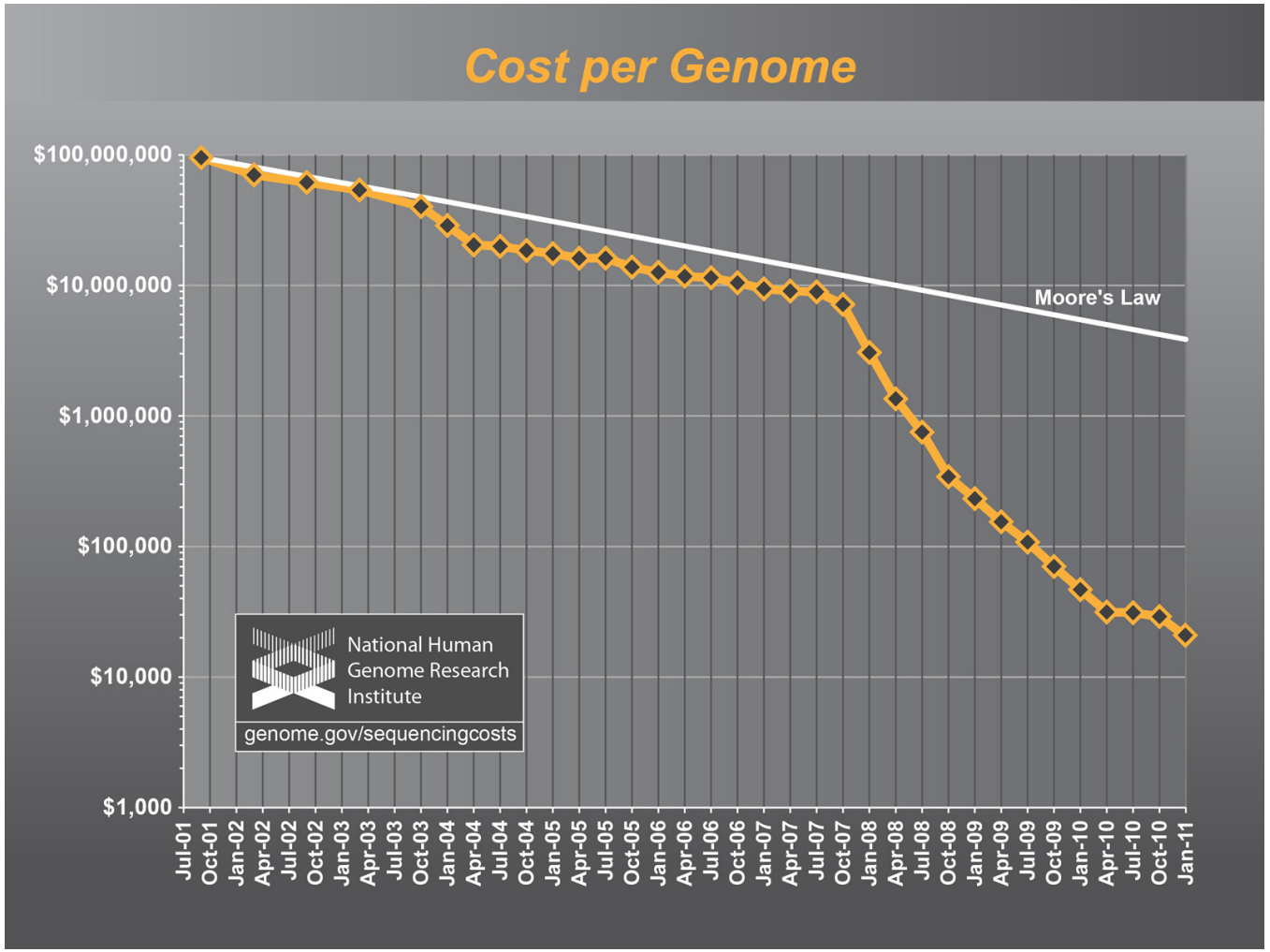
For realtime info: http://rtm.hep.ph.ic.ac.uk/webstart.php



An Interesting Observation….



HW Assignment: Think about the implications of this graph.



"There is a growing gap between the generation of massively parallel sequencing output and the ability to process and analyze the resulting data. New users are left to navigate a bewildering maze of base calling, alignment, assembly and analysis tools with often incomplete documentation and no idea how to compare and validate their outputs. Bridging this gap is essential, or the coveted $1,000 genome will come with a $20,000 analysis price tag."

CENTER FOR COMPUTATION
& TECHNOLOGY

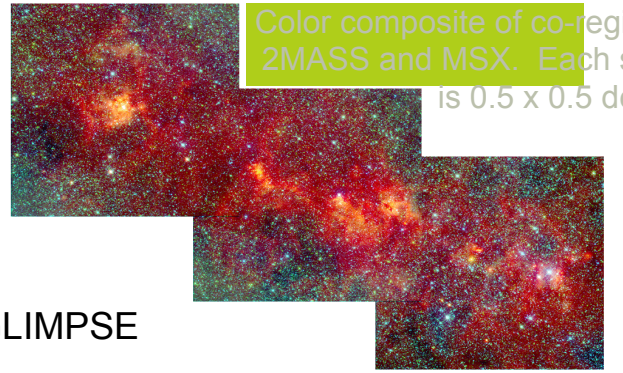
MONTAGE

What is Montage?

- Delivers custom, science grade image mosaics
  - An image mosaic is a combination of many images containing individual pixel data so that they appear to be a single image from a single telescope or spacecraft
  - User specifies projection, coordinates, spatial sampling, mosaic size, image rotation
  - Preserve astrometry (to 0.1 pixels) & flux (to 0.1%)
- Modular, portable "toolbox" design
  - Loosely-coupled engines for image reprojection, background rectification, co-addition
    - Control testing and maintenance costs
    - Flexibility; e.g custom background algorithm; use as a reprojection and co-registration engine
  - Each engine is an executable compiled from ANSI C
- Public service deployed
  - Order mosaics through web portal

David Hockney *Pearblossom Highway* 1986

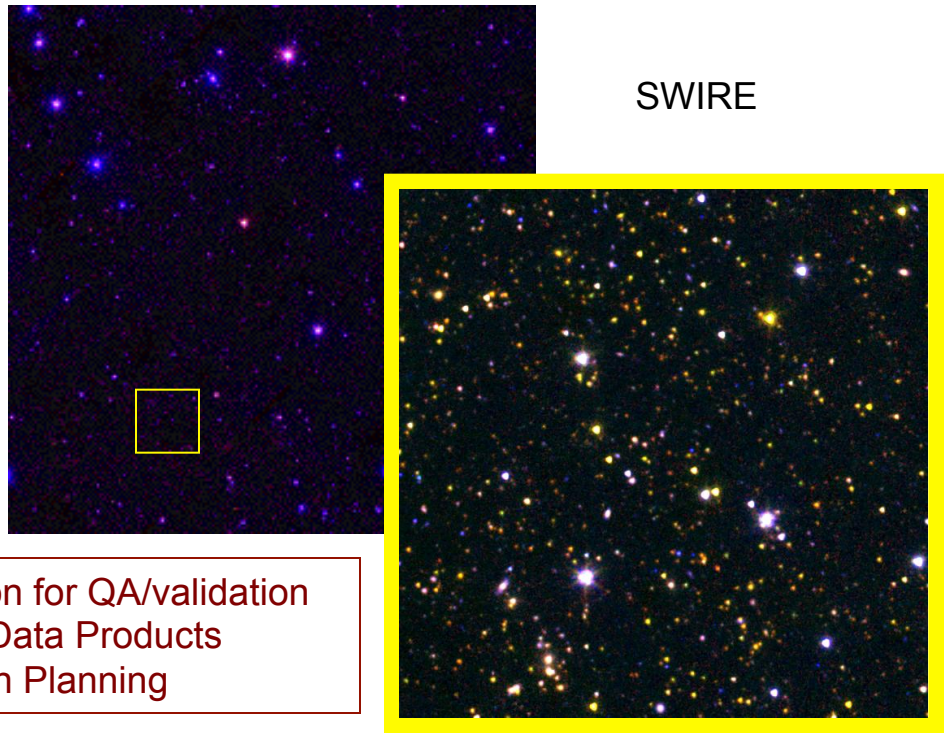
Montage use by Spitzer Legacy Teams



Color composite of co-registered 2MASS and MSX. Each square is 0.5 x 0.5 degrees

GLIMPSE

3-color GLIMPSE image mosaic over 1.1° x .8°

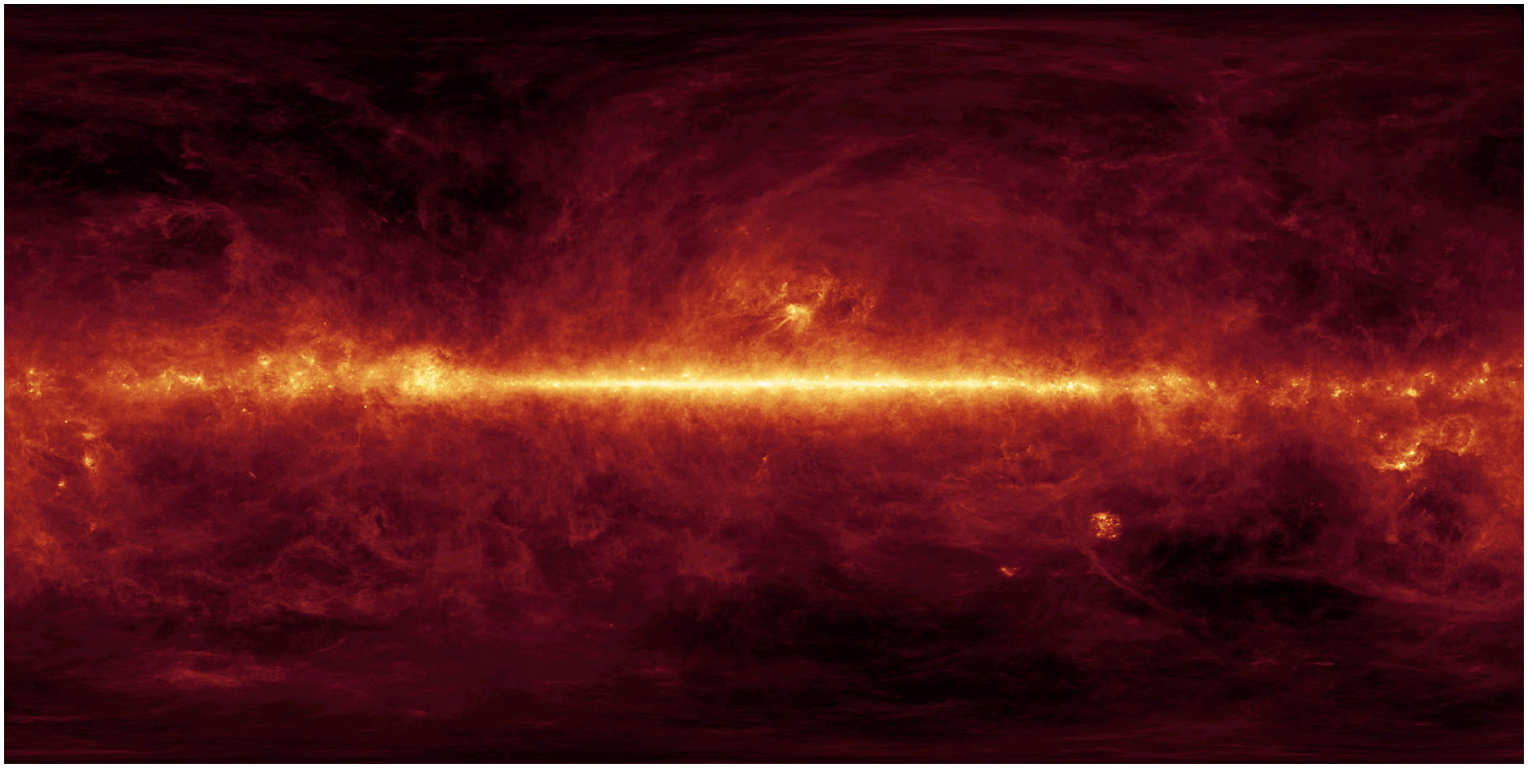


SWIRE

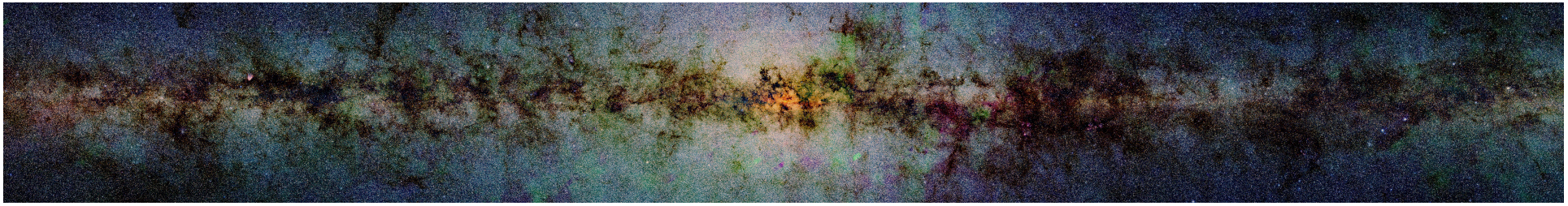
Data Federation for QA/validation
Building Data Products
Mission Planning

Right: Spitzer IRAC 3 channel mosaic (3.6μm in green, 4.5μm in red, and i-band optical in blue); high redshift non-stellar objects are visible in the full resolution view (yellow box).

Montage Use by Spitzer E/PO Group

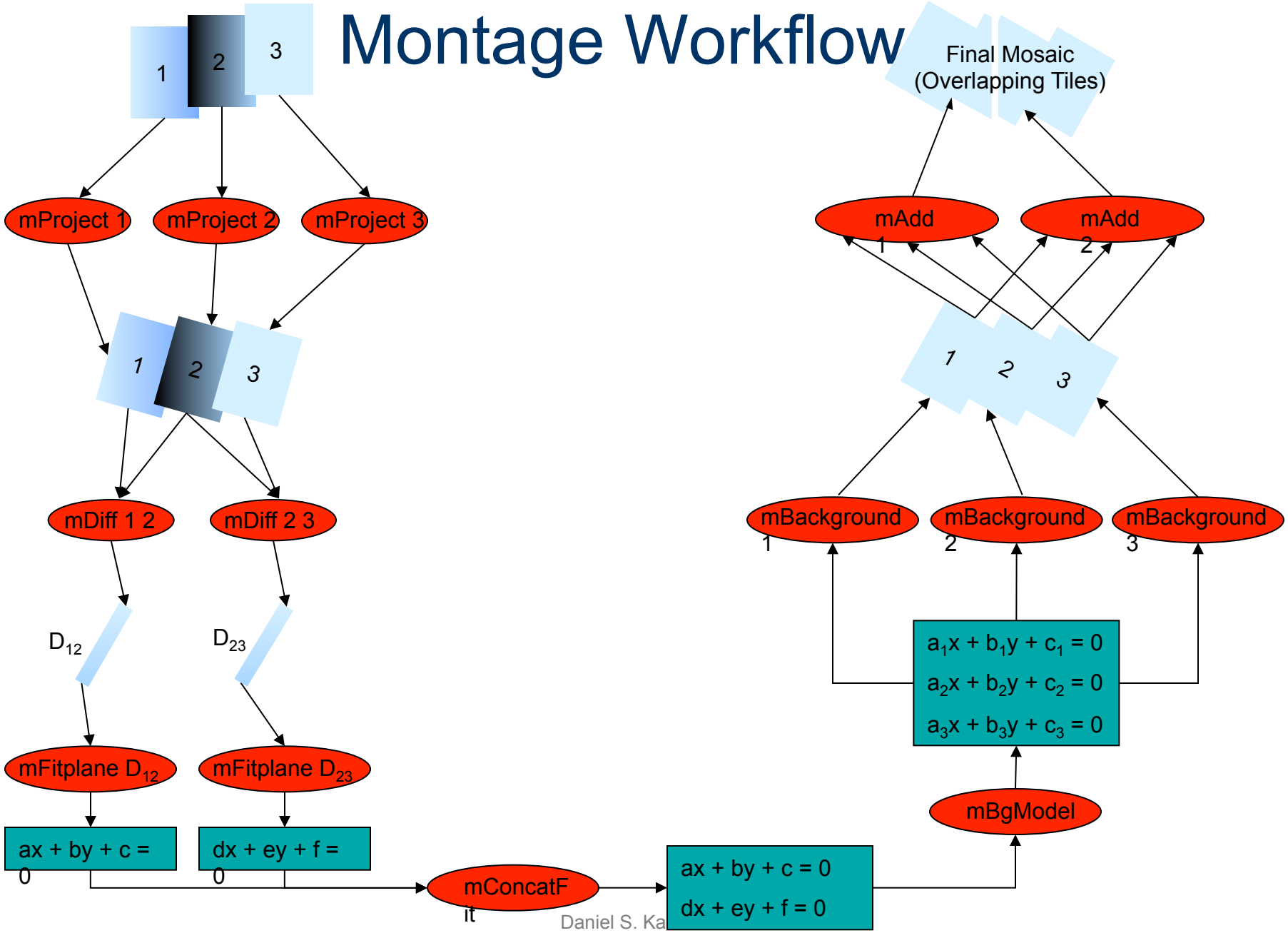


100 μm sky; aggregation of COBE and IRAS maps (Schlegel, Finkbeiner and Davis, 1998) 360° x 180°, CAR projection

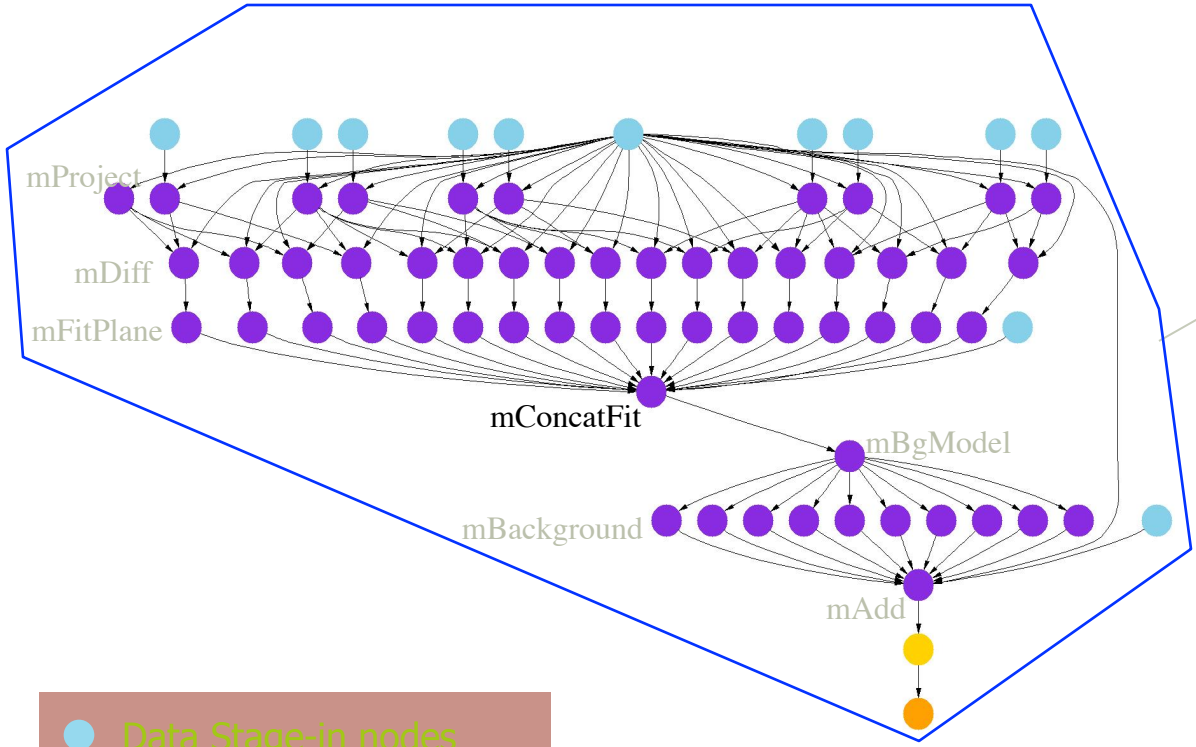


Panoramic view of galactic plane as seen by 2MASS, 44° x 8°, 158,400 x 28,800 pixels; covers 0.8% of sky

Montage Workflow



Montage on the Grid Using Pegasus



Maps an abstract workflow to an executable form

Pegasus

Grid Information Systems

Information about available resources, data location

Condor DAGMan

Executes the workflow

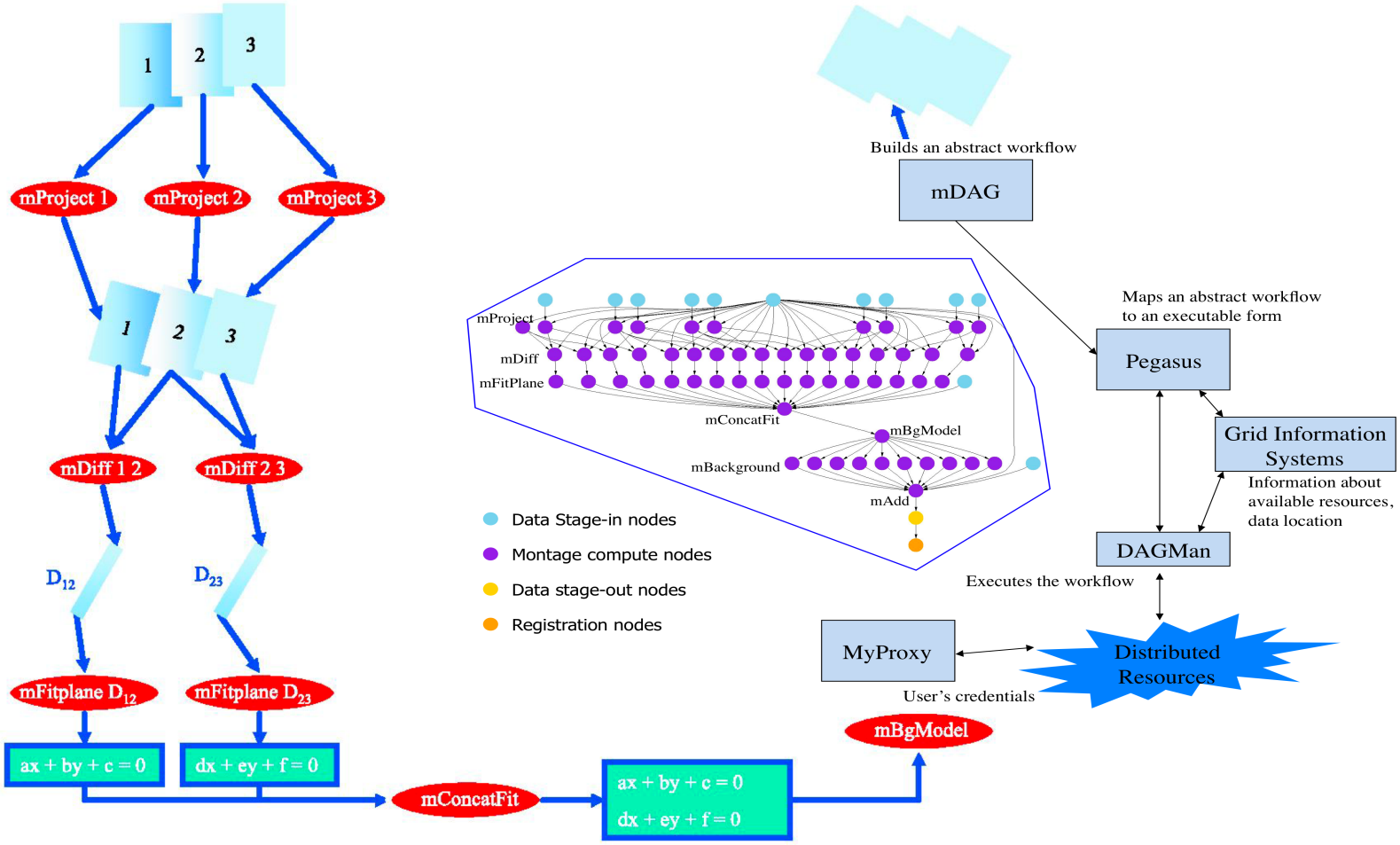
MyProxy

Grid

User's grid credentials

- Data Stage-in nodes
- Montage compute nodes
- Data stage-out nodes
- Registration nodes

Montage: DAG-based Workflow Application Exemplar



Understanding Montage

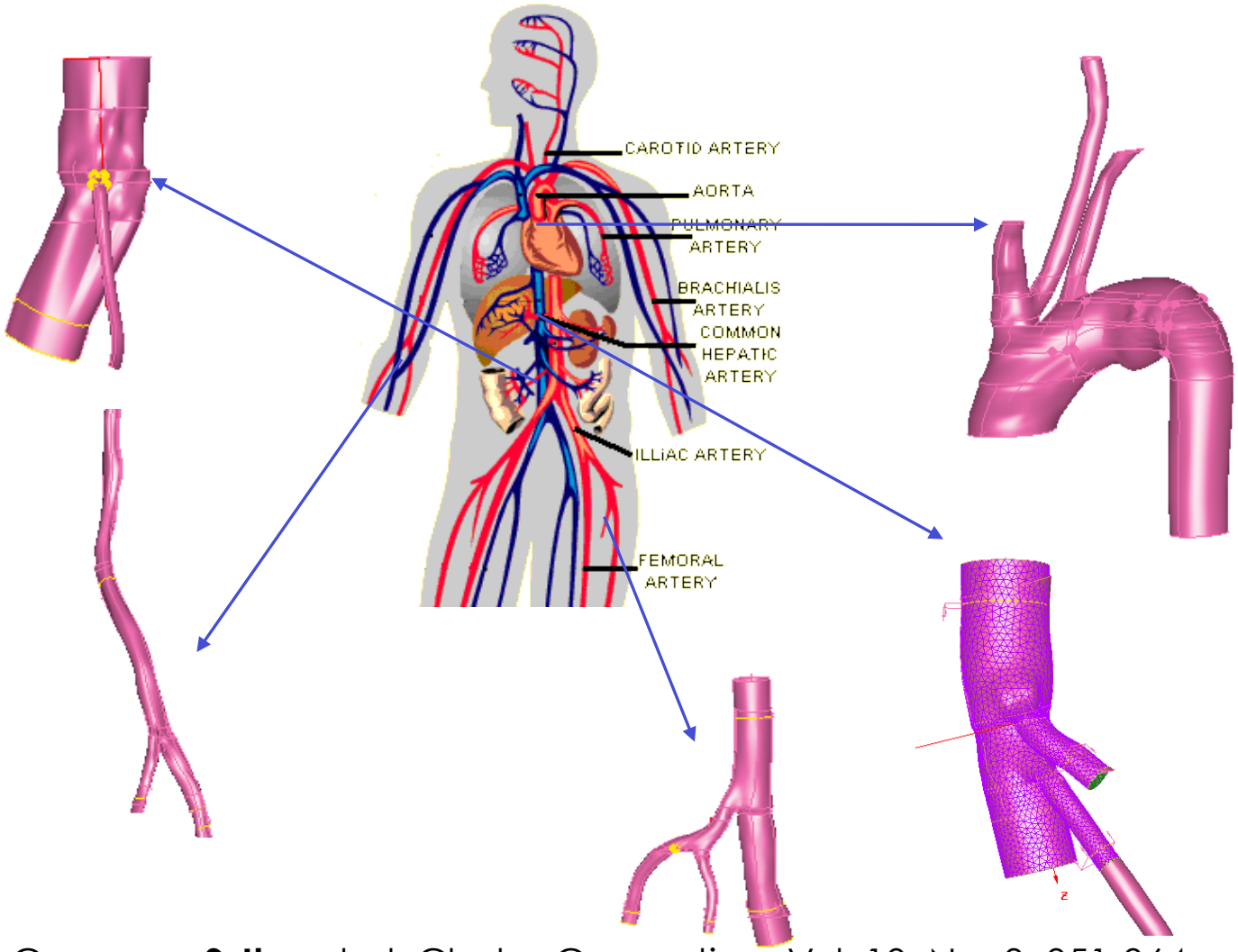
- Why distributed?
  - Scale Processing (over come size limits) above local limits
- How distributed?
  - DAG is created
  - DAG is executed via a DAG-Enactor [DAGMAN]
    - Other approaches exist
- Challenges/Issues/Success?
  - Mapping to right resources – to effectively use different resources, network (as now data transfer can become bottleneck)

CENTER FOR COMPUTATION
& TECHNOLOGY

NEKTAR

CENTER FOR COMPUTATION & TECHNOLOGY

Simulation of Blood Flow in Human Arterial Tree



B Boghosian, P Coveney, **S Jha** et al, Cluster Computing, Vol. 10, No. 3, 351-364 (2007,) DOI 10.1007/s10586-007-0029-4 Courtesy George Karniadakis (Brown)

CENTER FOR COMPUTATION
& TECHNOLOGY

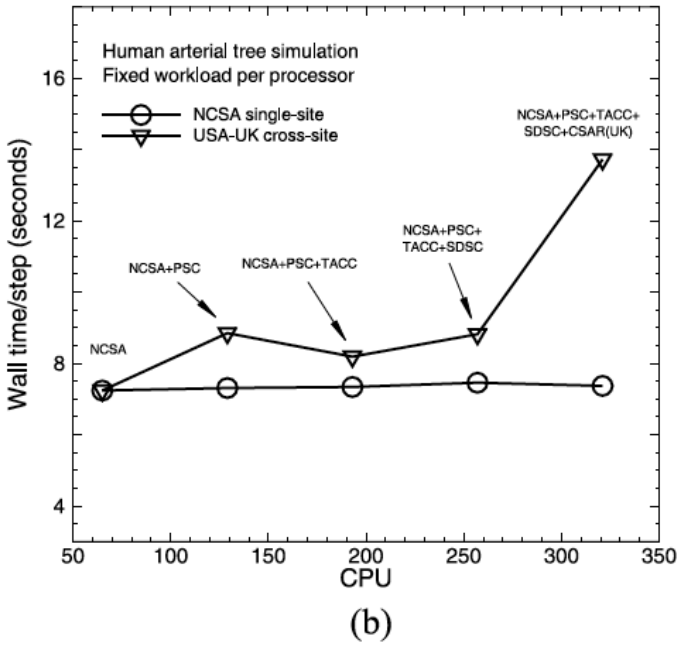


Fig. 1 USA–UK cross-site performance of arterial tree application: (**a**) wall time/step as a function of the total number of processors for a fixed problem size in cross-site computations involving NCSA-PSC-TACC-SDSC. (**b**) wall time/step as a function of the total number of processors in USA-UK cross-site computations for a fixed workload per processor; As the problem size increases, the number of processors (or sites) is increased in proportion such that the workload per processor remains unchanged

Cluster Computing, Vol. 10, No. 3, 351-364 (2007,) DOI 10.1007/s10586-007-0029-4

CENTER FOR COMPUTATION & TECHNOLOGY

Understanding NeKTAR

- Why distributed?
  - Full model Required 7 TB of RAM!! Factor of 10 greater
    - Scaled down resolution and tried to get as much as possible
  - Fundamental limitation on scalability of full 3D problem
    - Reformulated
- How distributed?
  - MPI Based Application, use "distributed" library version of MPI
  - Resources selected in advance
- Challenges/Issues/Success?
  - Co-scheduling
  - Single point of failure
    - Exponentially increasing with # of systems/resources used!

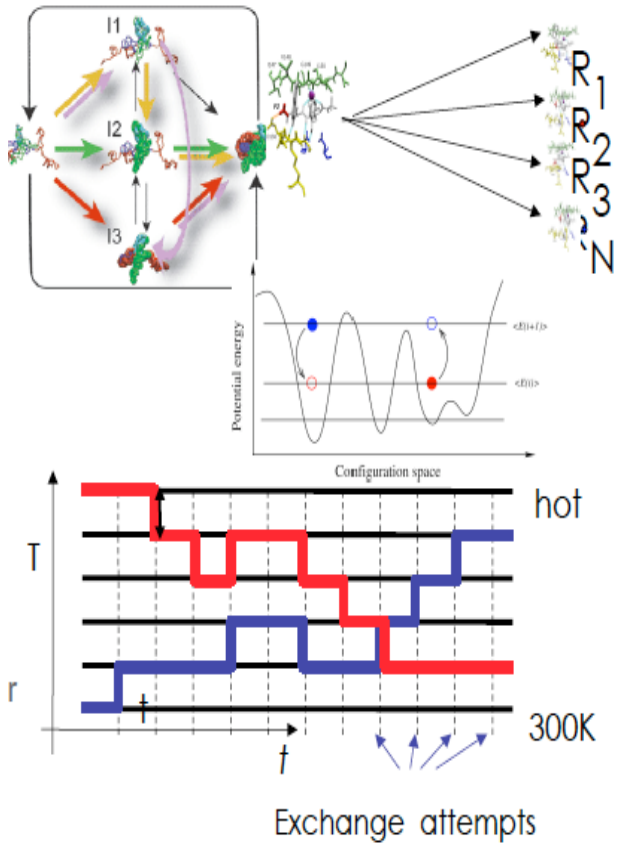
CENTER FOR COMPUTATION
& TECHNOLOGY

ENSEMBLE-BASED REPLICA-EXCHANGE

CENTER FOR COMPUTATION & TECHNOLOGY

Ensemble-based & Replica-Exchange Simulations

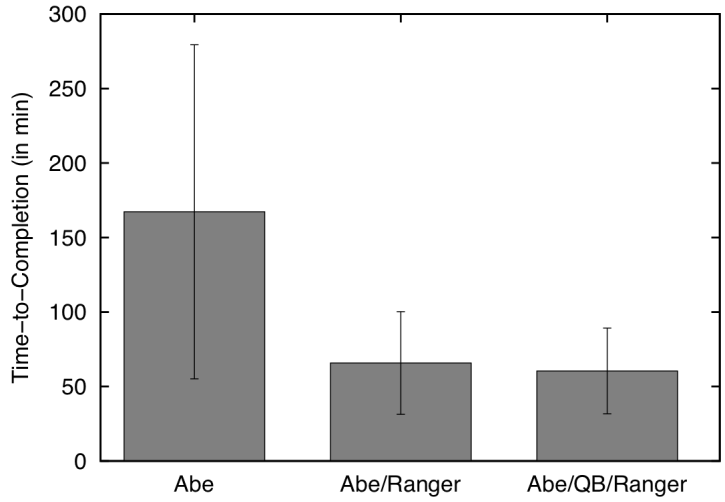
- Ensemble-based:
  - Many uncoupled simulations
  - But not necessarily uncoupled in analysis!
- Replica-Exchange (RE) methods:
  - Represent a class of algorithms that involve a large number of loosely coupled ensembles.
- RE simulations are used to understand a range of physical phenomena
  - Protein folding, unfolding etc
  - MC simulations
- Many successful implementations
  - Eg folding@home [replica based]

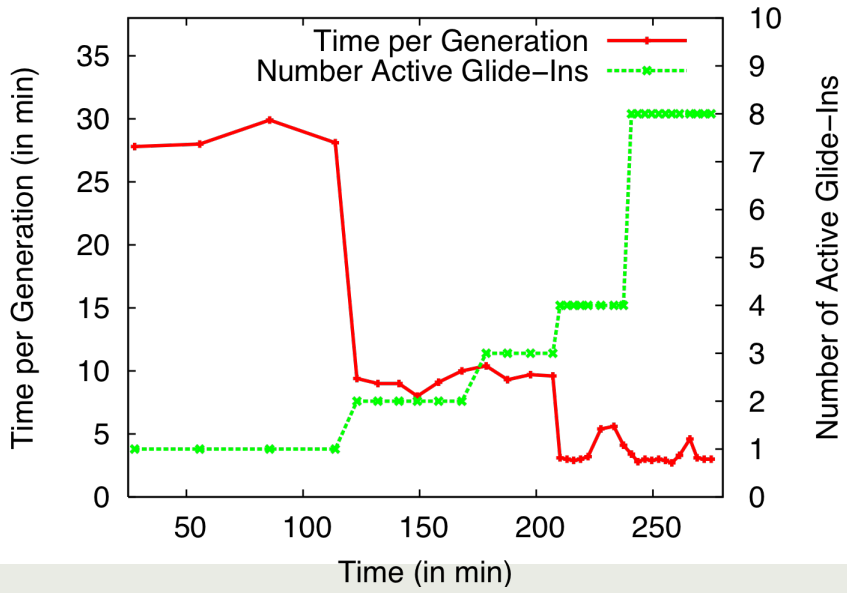


CENTER FOR COMPUTATION & TECHNOLOGY

Distributed Adaptive Replica Exchange (DARE)
Multiple Pilot-Jobs on the "Distributed" TeraGrid

- Ability to dynamically add HPC resources. On TG:
  - Each Pilot-Job 64px
  - Each NAMD 16px
- Time-to-completion improves
  - No loss of efficiency





Luckow, Kim, Schnor, Jha Adaptive Replica-Exchange, Phil. Trans of Royal Society A, 28 June 2009 vol. 367 no. 1897 2595-2606

CENTER FOR COMPUTATION & TECHNOLOGY

Understanding Replica-Exchange

- Why Distributed?
  - Many un-coupled units (ensembles/replica)
  - More resources, the merrier!!
- How Distributed?
  - Many implementations exist (eg folding@home)
  - SAGA-based "Pilot-Jobs" to use many distributed TG resources
- Limitations and Success?
  - Getting SAGA working on all machines!
  - Finding the best set of resources
  - Coordinating work across all the resources

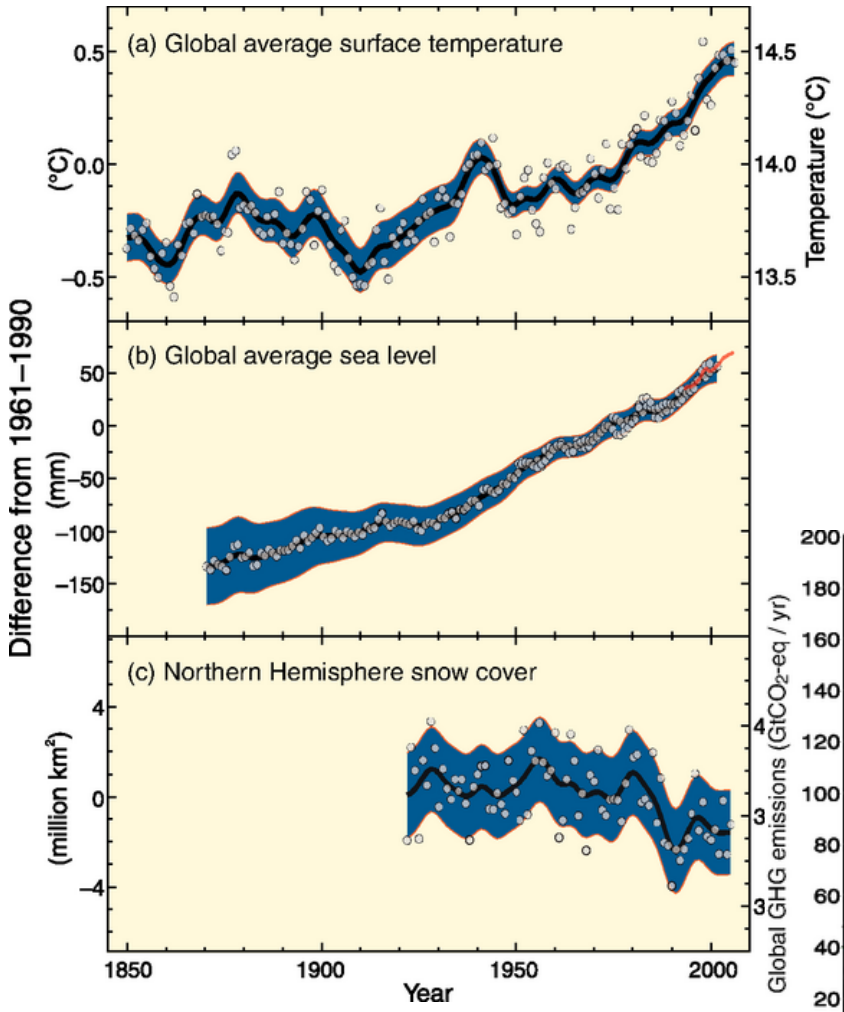
CENTER FOR COMPUTATION
& TECHNOLOGY

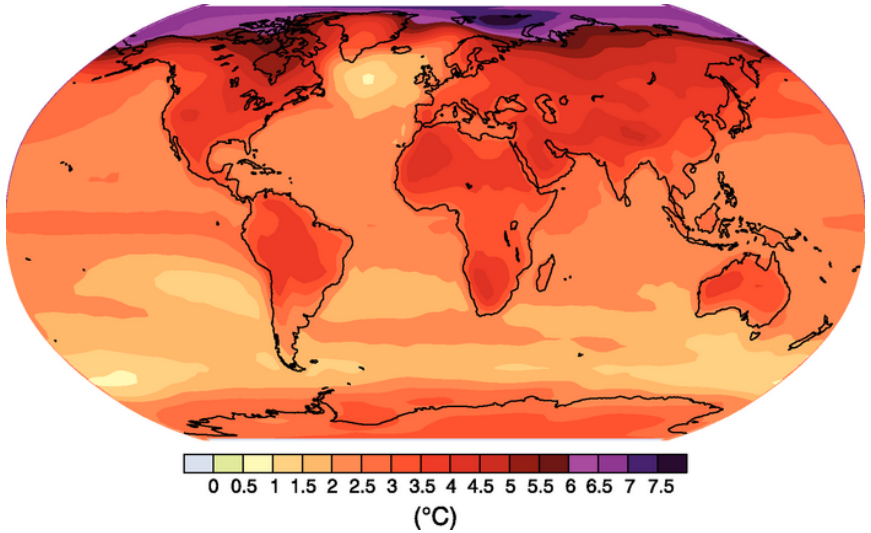
CLIMATEPREDICTION.NET

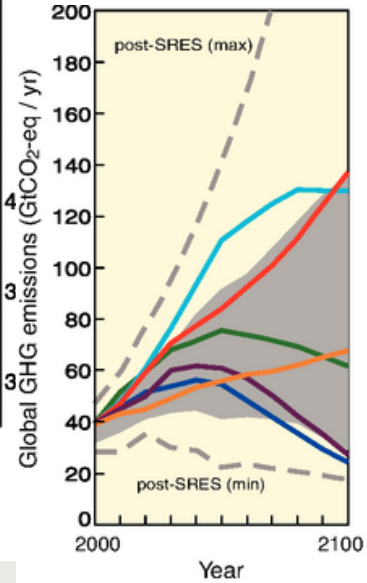
CENTER FOR COMPUTATION & TECHNOLOGY

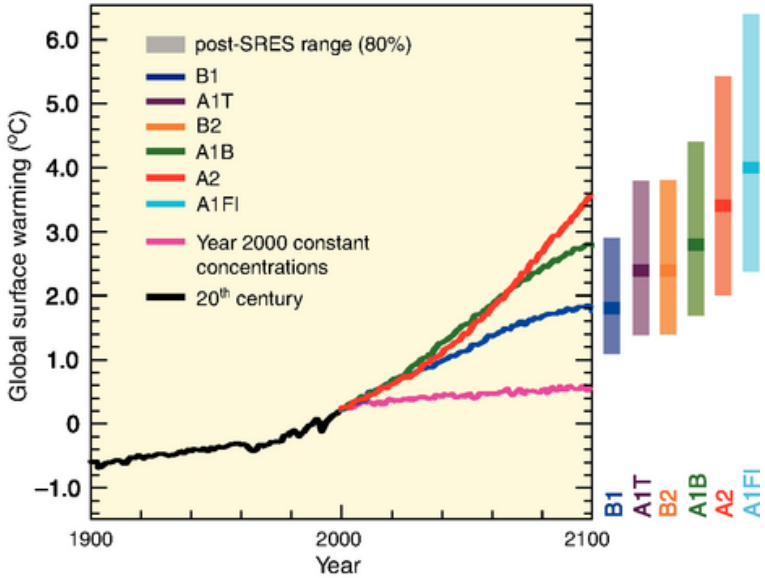
IPCC AR4:

http://www.ipcc.ch/publications_and_data/ar4/syr/en/contents.html









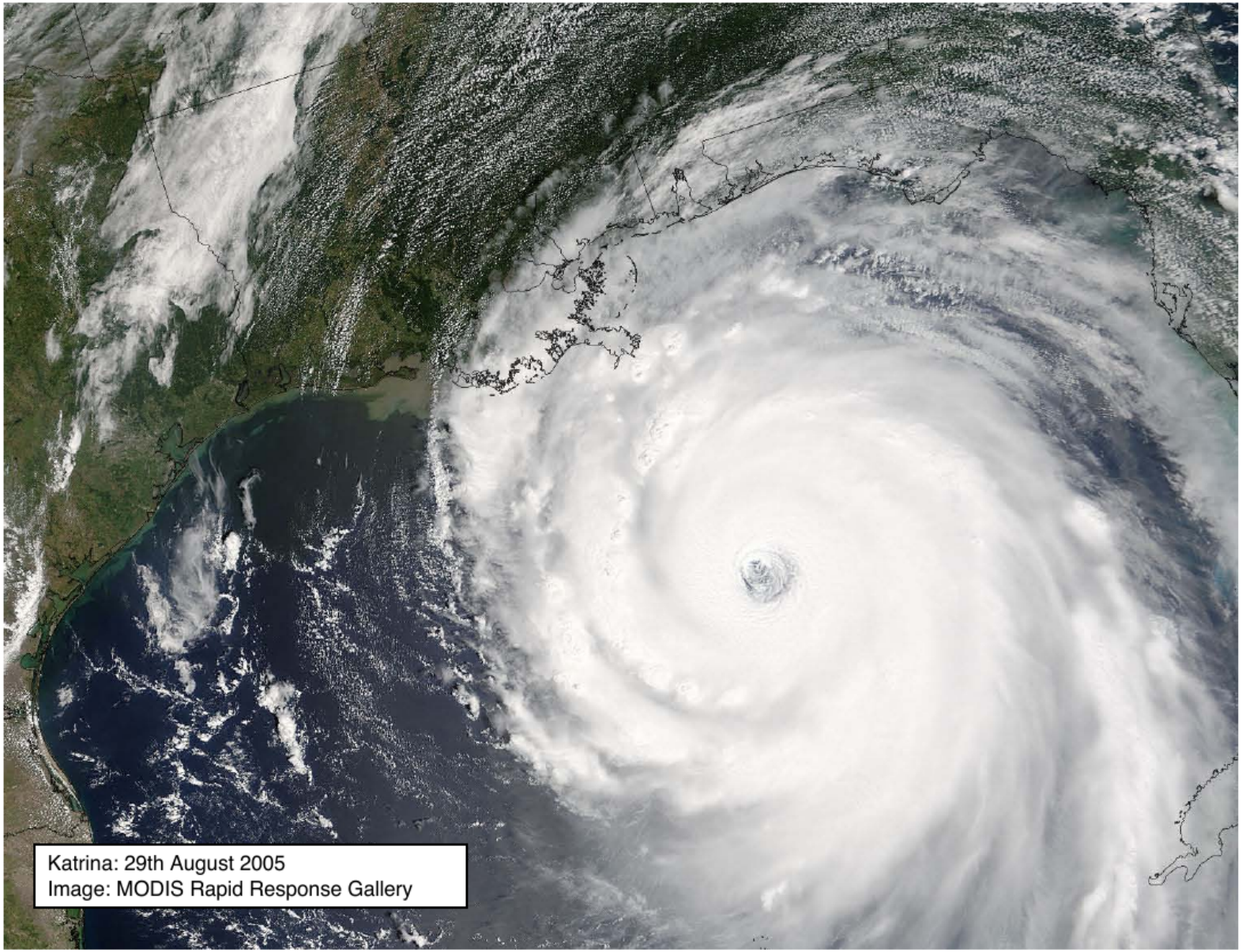
CENTER FOR COMPUTATION & TECHNOLOGY

Understanding ClimatePrediction.net

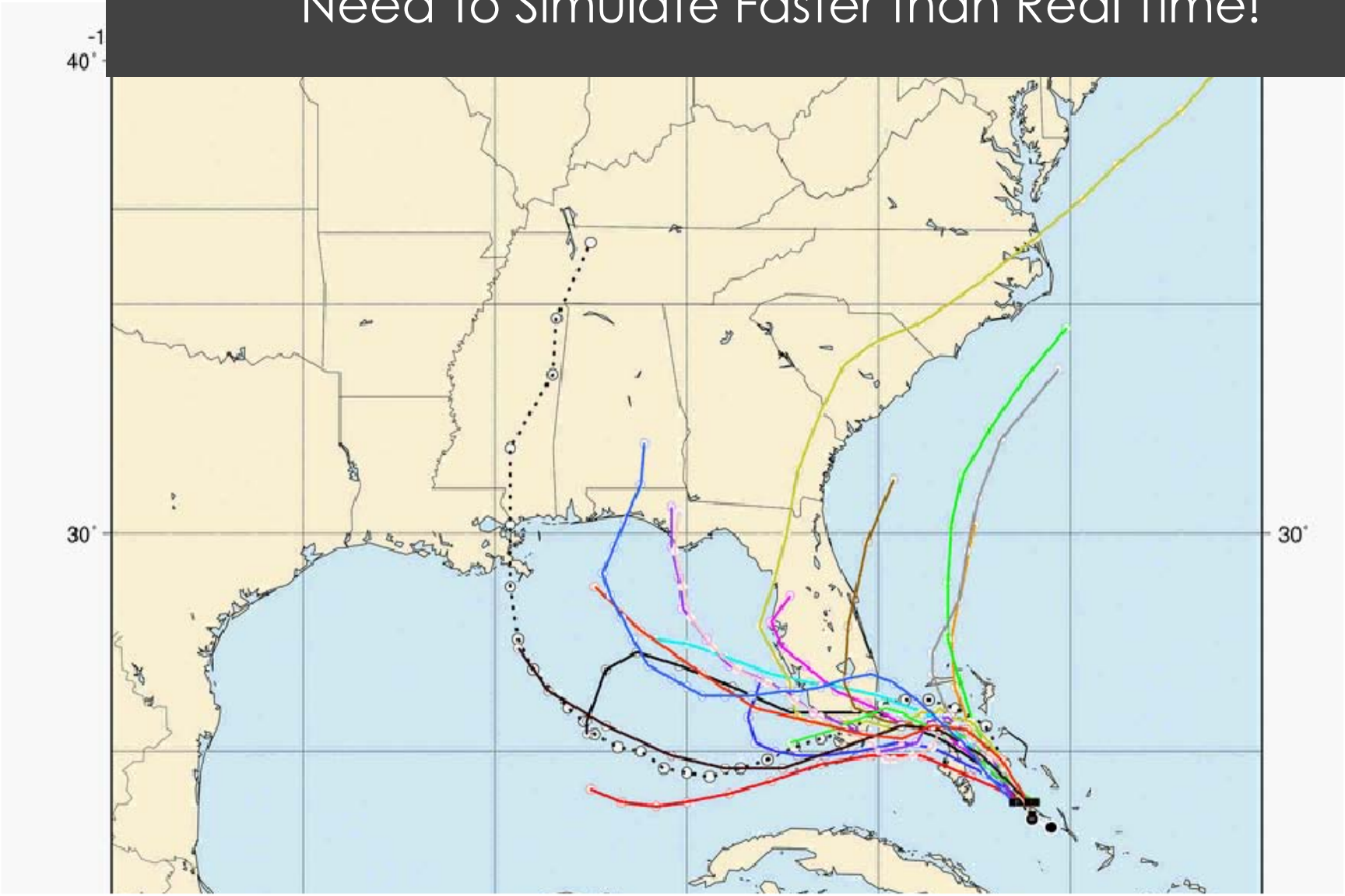
- Why Distributed?
  - Many small indep. Comp. tasks – naturally decomposable
  - Access many more resources without owning
    - Petaflop computing years before Petaflop Computing era!?
- How Distributed?
  - BOINC -- basis for @HOME projects [Volunteer Computing]
  - "Trickles" – job reporting to the Master (project server)
  - Data too large to aggregate and analyze centrally
    - Hence must operate on data in-situ
- Limitations and Success?
  - Coordinating work across all the resources
  - Managing changing number of resources and failures

CENTER FOR COMPUTATION
& TECHNOLOGY

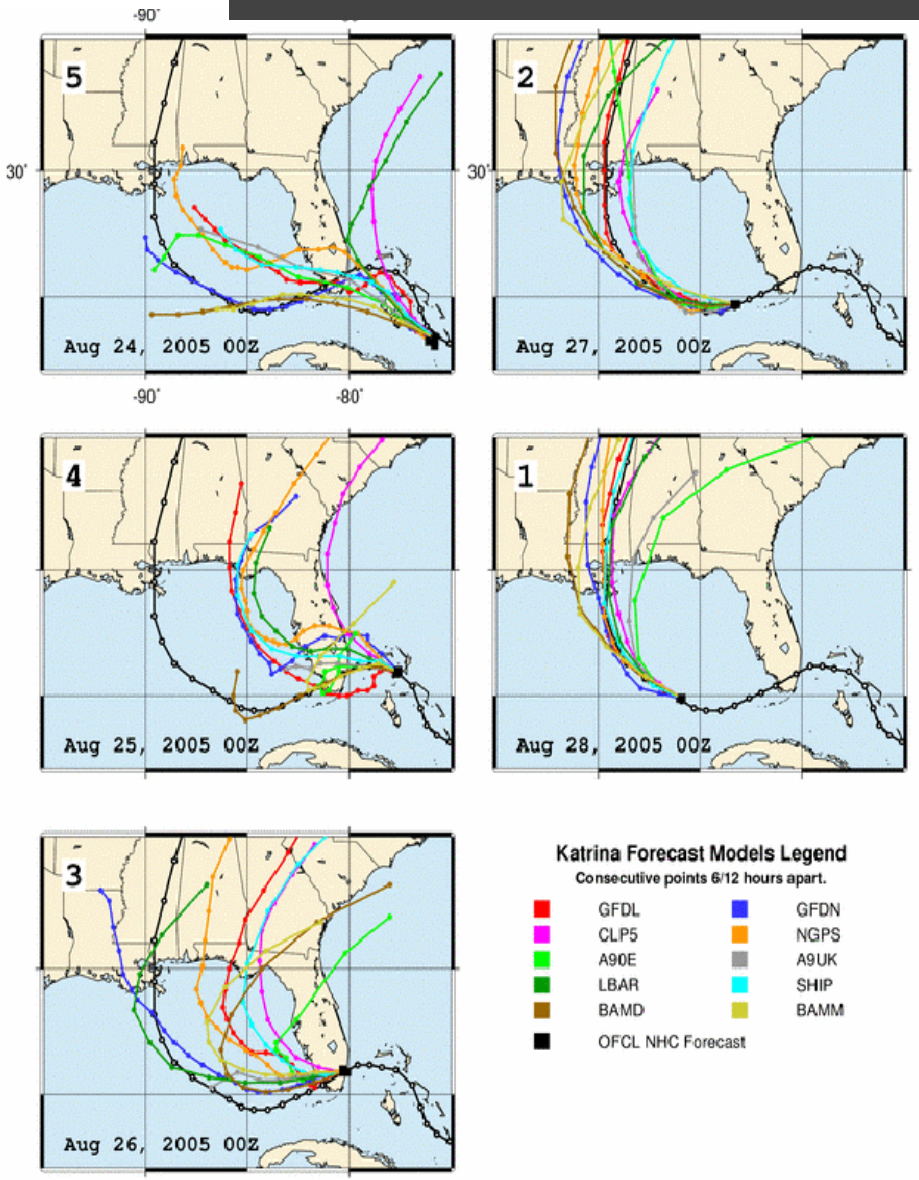
SCOOP..

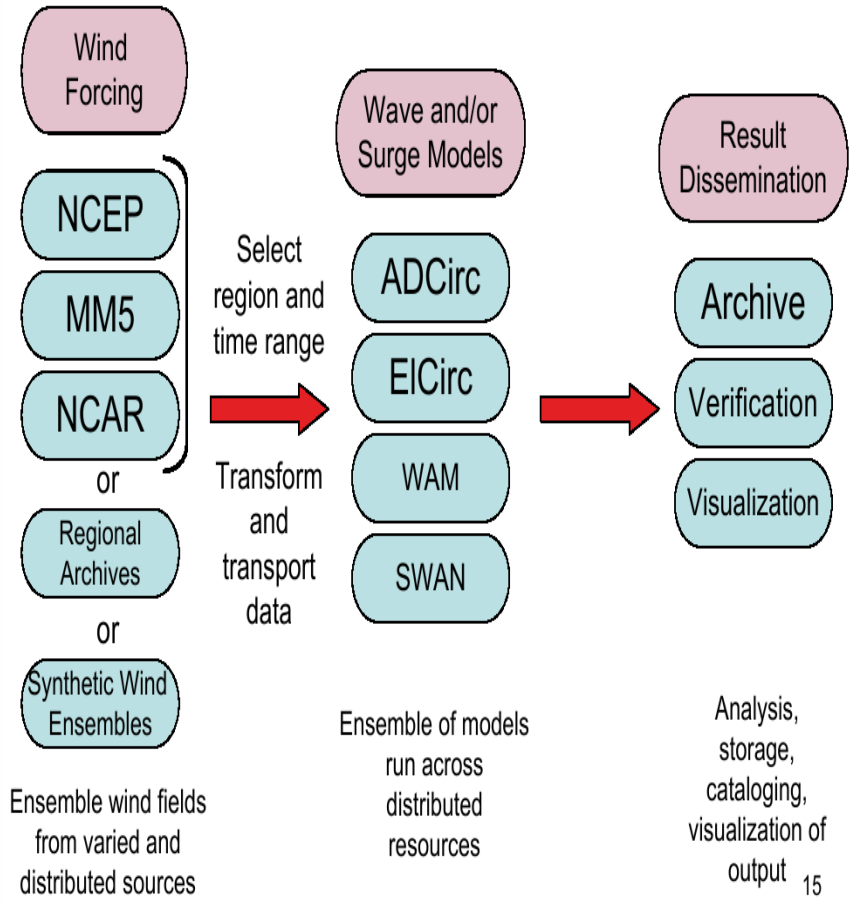

Katrina: 29th August 2005
Image: MODIS Rapid Response Gallery

Need To Simulate Faster than Real Time!



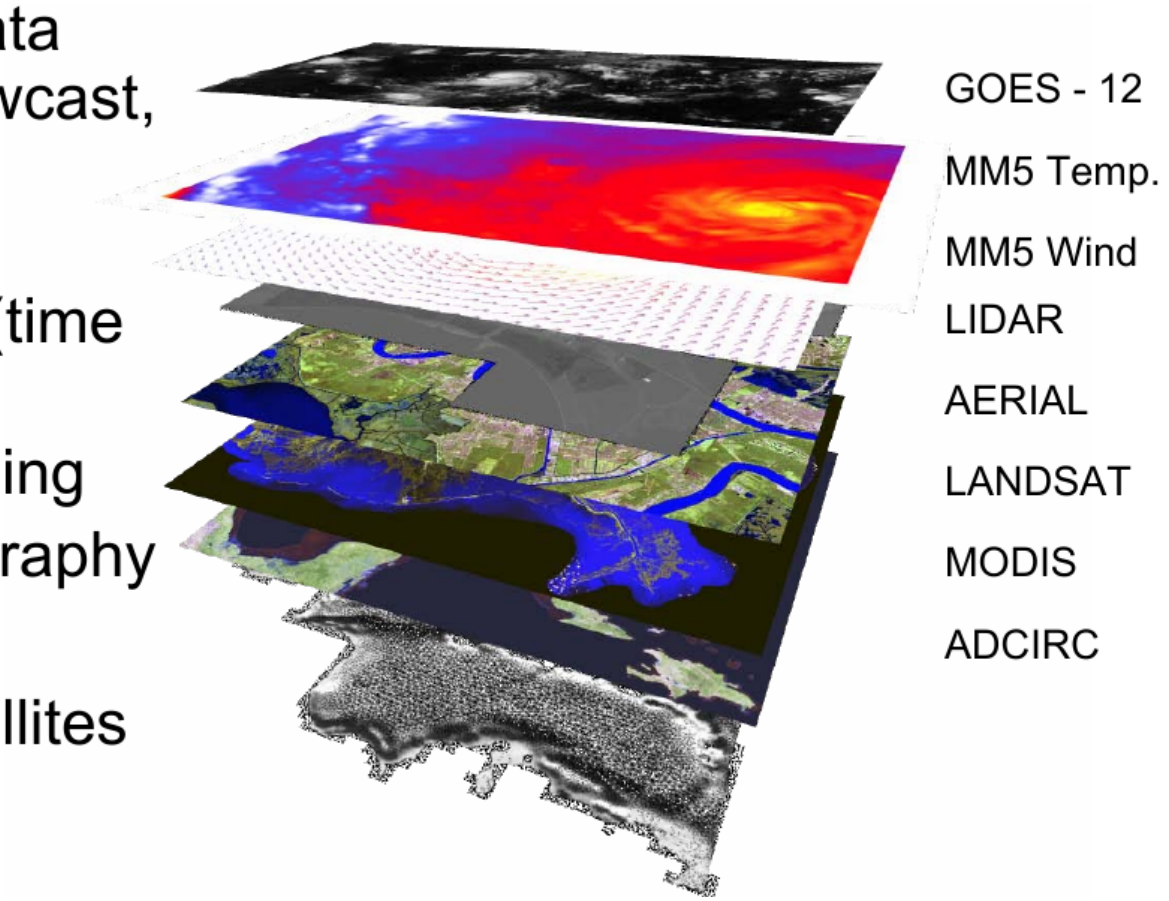
Heterogenous: Compute Models



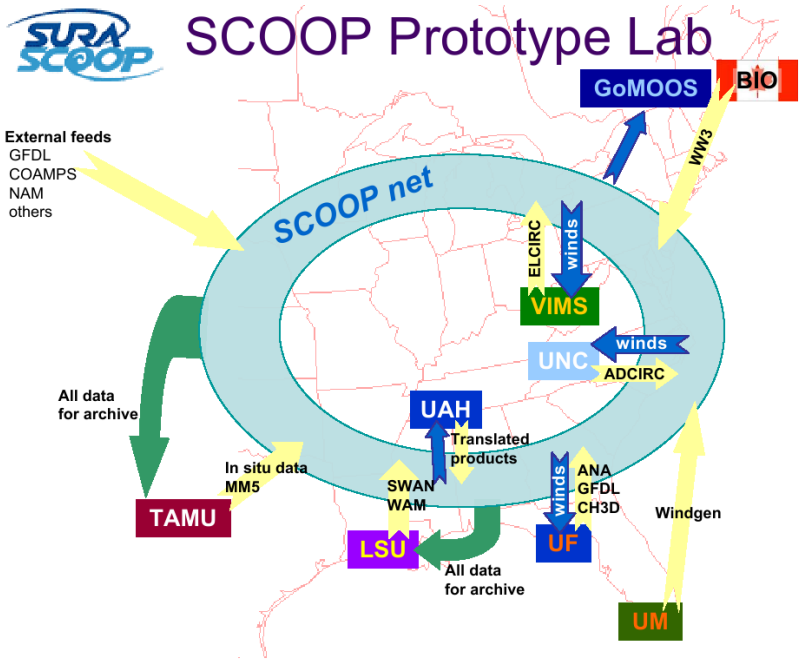


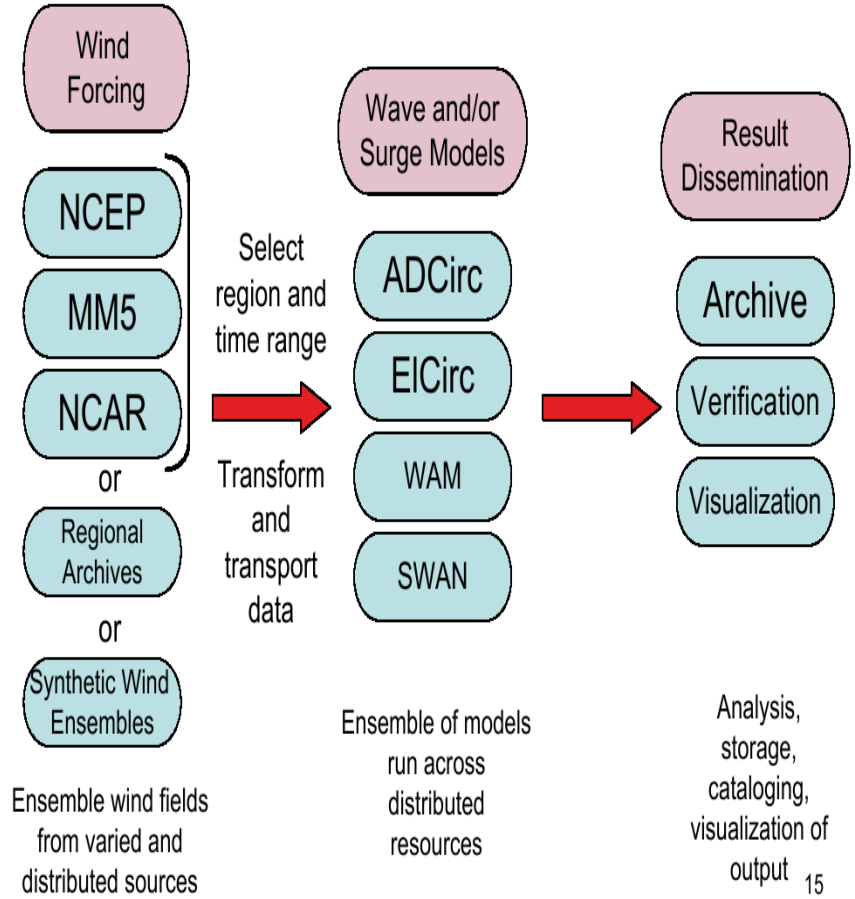
Many Heterogeneous Components

- Simulation data (forecast, nowcast, hindcast)
  - 2D, 3D
- Sensor data (time series)
- Remote imaging
- Aerial photography
- LIDAR
- Weather satellites
- GIS



Naturally Distributed..





Understanding SCOOP

- Why Distributed?
  - Naturally Distributed: Geographically distributed producers and end-users
  - High Peak Demand and quick response time
    - But with very low duty cycle --- Economic Argument !!
- How Distributed?
  - Customized workflows
- Limitations and Success?
  - Not Robust – Many components that need to come together
  - Coordinating work across all the resources
  - Co-scheduling / Advanced Scheduling / Prediciton a challenge

Distributed Applications
How do they differ from traditional HPC applications?

- Performance Models:
  - Not just "peak utilization"; e.g., HPC & HTC (# of jobs)
- Skillful Decomposition vs Aggregation
  - Primacy of Coordination across distributed resources
- Usage Modes:
  - The same application has multiple usage modes
  - How applications are developed, deployed and executed is often determined by the infrastructure
- Execution Environment
  - Typically not as well controlled or defined
  - Varying resource conditions, application requirements
- HW Assignment: Think about applications we've studied and see if the above is true?

Overview of Module E
Distributed Scientific Computing

- E1: Introduction to Distributed Scientific Computing
  - Examples of Distributed Applications : [30 minutes]
    - WLCG + Application Examples
    - Summarize and compare with HPC
  - Examples of "Production" Grid Infrastructure [30 min]
    - HPC vs HTC, Research vs Production, Commercial
  - Distributed Computing is Hard. Why? [15]
- E2: Introduction to the Practice of DSC [45 mins]
  - Introduction to SAGA and installing it
  - Writing your first distributed application
- Mod E Project [10mins]

Production Distributed Infrastruc

- HPC Grids
  - XSEDE/TeraGrid [DEISA]
- HTC Grids
  - OSG, EGI
- Research Grid Infrastructure
  - FutureGrid, [Grid5000, DAS]
- Commercial Infrastructure
  - AWS (EC2, S3), Azure

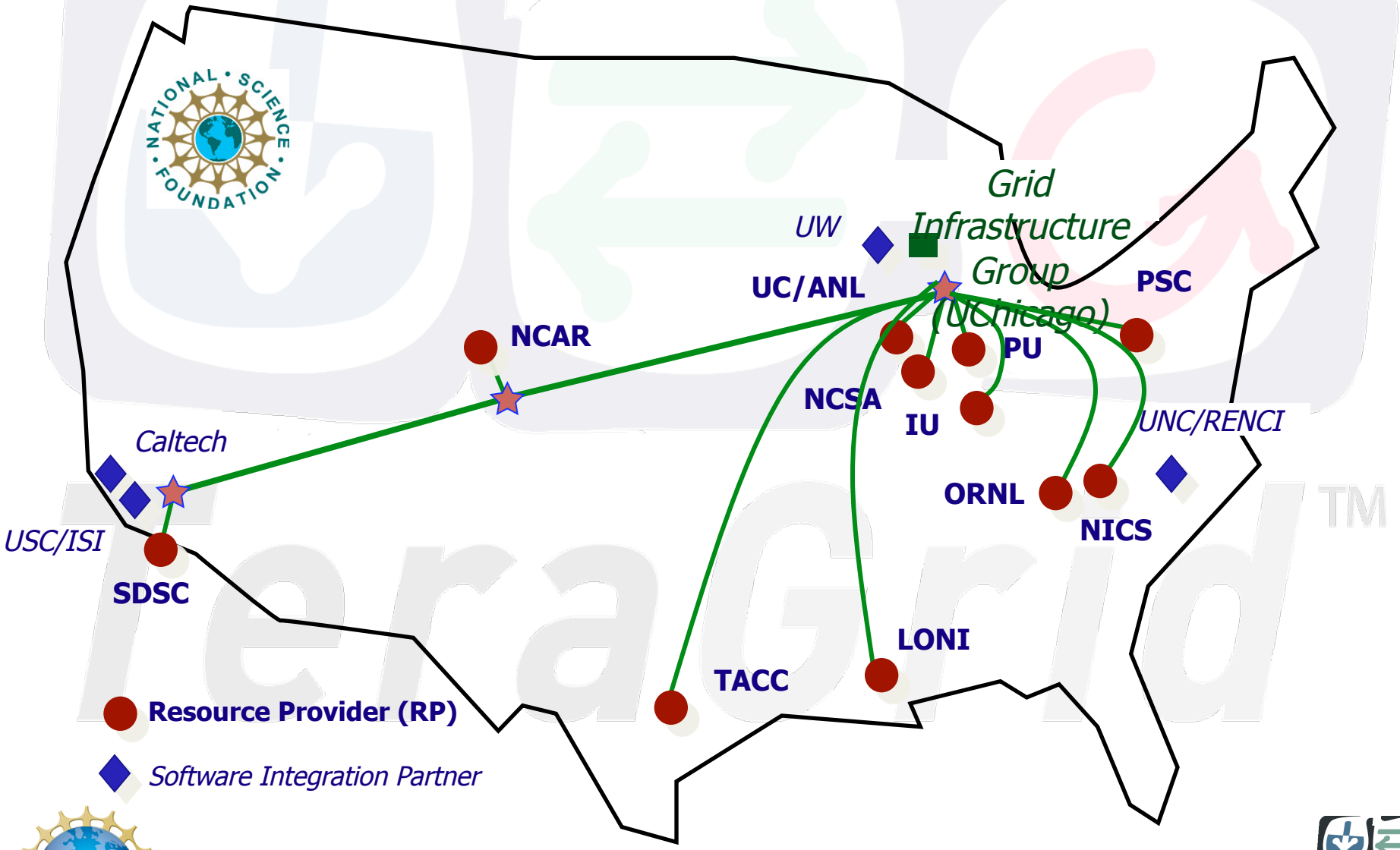
Why are we examining DI?

- Sapir-Whorf Hypothesis:
  - Language Influences the Habitual Thought
  - Scientific Computing Analog:
    - Infrastructure shapes the practice and formulation of research
    - Think historically how CI has shaped research
- "I have a research Q. Which CI should I use?"
  - Go to TG? Go to OSG? Or should I pay for resource?
  - Might the policies of these DI influence your decision?
- This lecture (part) aims to provide a quick overview and understanding of the infrastructures available
  - For each, (i) Description, (ii) Management, (iii) Typical Usage Modes and applications supported, (iv) Policies

TeraGrid

US National Cyberinfrasctructure for Scientific Research

11 Resource Providers



TeraGrid™

TeraGrid Resources and Services

- **Computing**
  - More than one petaflop of computing power today and growing
  - 500 Tflop Ranger system at Texas Advanced Computing Center (TACC)
  - 466 TFlop Kraken (Cray Xt4) at National Institute for Computational Sciences (NICS), University of Tennessee
- **Remote visualization servers and software**
  - 60 TFlop condor-based viz resource at Purdue University
- **Data**
  - Allocation of data storage facilities
  - Over 100 Scientific Data Collections
- **Central allocations process**
- **Technical Support**
  - Central point of contact for support of all systems
  - Advanced Support for TeraGrid Applications (ASTA)
  - Education and training events and resources
  - Over 30 Science Gateways

TeraGrid Usage Modes in CY2008

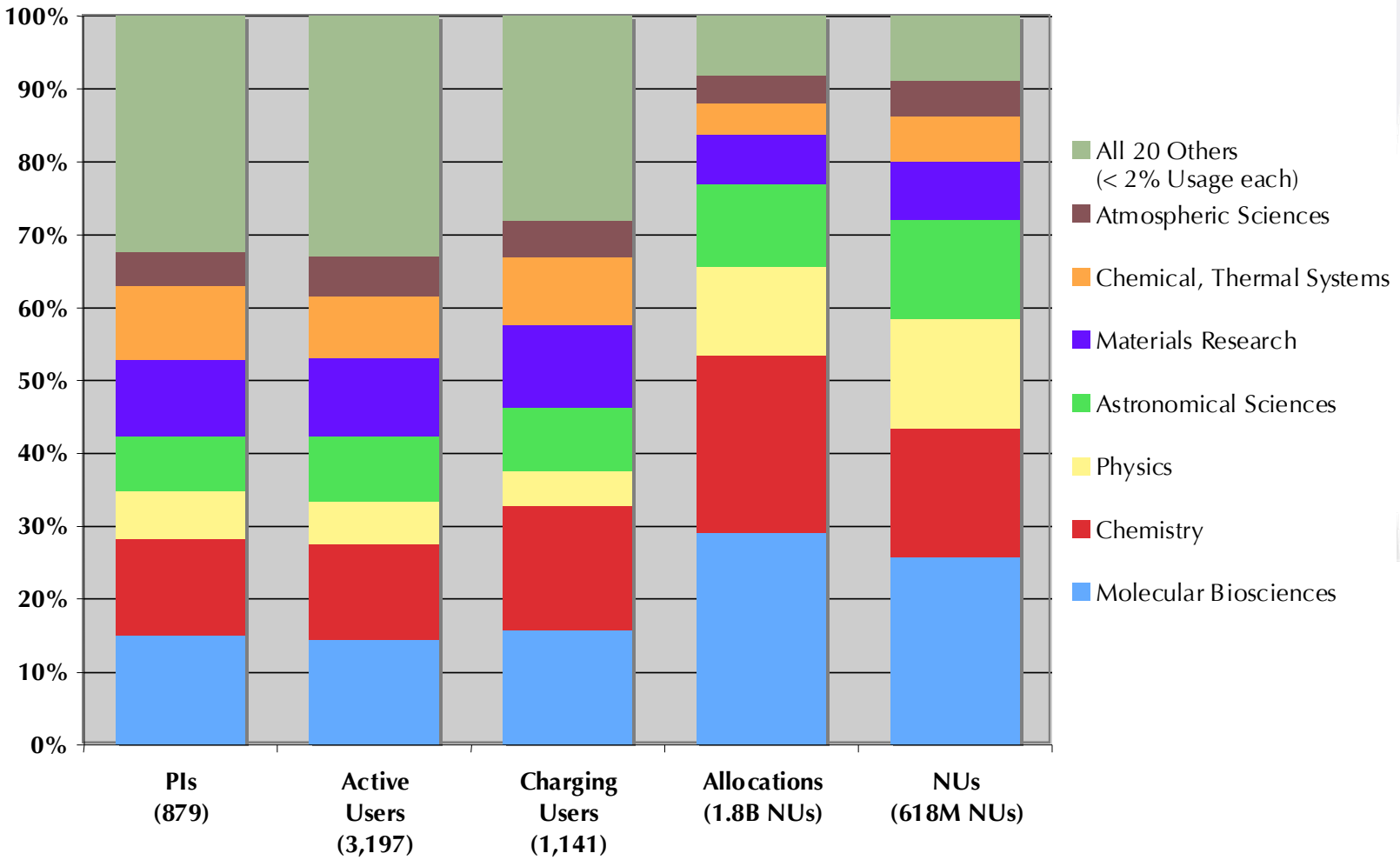
Use of TeraGrid Resources	# of projects
Batch Computing on Individual Resources	850
Exploratory and Application Porting	650
Workflow, Ensemble, and Parameter Sweep	160
Science Gateway Access	100
Remote Interactive Steering and Visualization	35
Tightly-Coupled Distributed Computation	10

Policies and Operational Details

- Management: GIG – central point of coordination
- Top-down process of determining who the resource providers (RP) are
  - Eg LSU/LONI is an RP via a NSF HPCOPS award
- User needs allocation to access resource
  - Eg TRAC, PRAC, Educational
  - Allocation typically bound to a resource
- We talked about co-scheduling
  - Is not provided for all intents and purposes
- Users typically use a single resource at a time
  - See Usage Mode table

TeraGrid™

TeraGrid Usage



Open Science Grid

http://www.openscience.org

- Bottom-Up Organization: OSG brings together computing and storage resources from campuses and research communities into a common, shared grid infrastructure over research networks via a common set of middleware

- Philosophy: OSG offers participating research communities low-threshold access to more resources than they could afford individually, via a combination of dedicated, scheduled and opportunistic alternatives

- Management: OSG is a consortium of software, service and resource providers and researchers, who together build and operate the OSG project

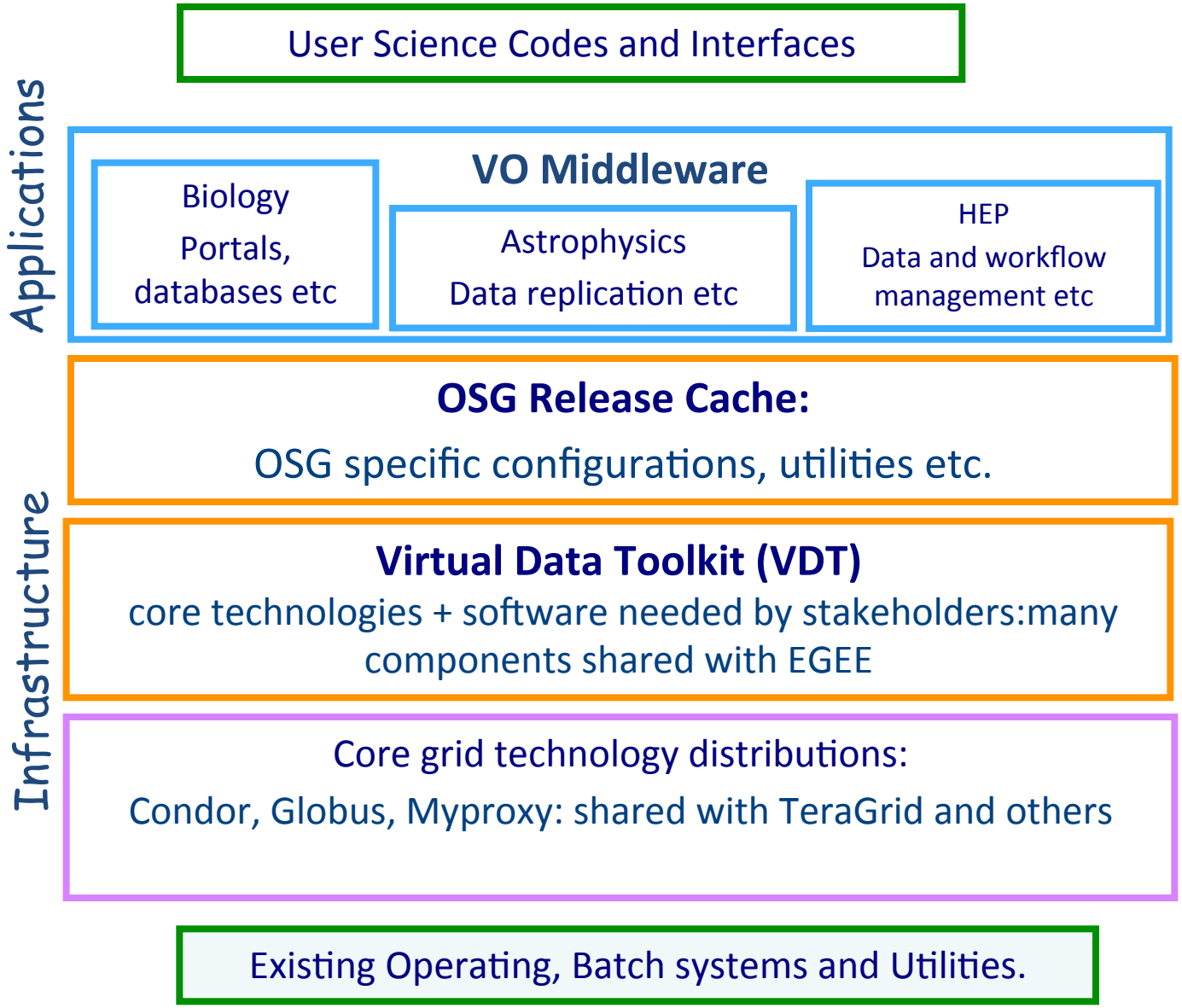
Open Science Grid

http://www.openscience.org

- OSG Consortium members' independently owned and managed resources make up the distributed facility, agreements between them provide the glue for it
  - Organized around Virtual Organizations

- Software: Virtual Data Toolkit provides packaged, tested and supported collections of software for installation on participating compute and storage nodes and a client package for end-user researchers.

OSG Middleware



It takes VOs to make OSG work!

```
        cdf Collider Detector at Fermilab
        cms Compact Muon Solenoid
compbiogrid CompBioGrid
        des Dark Energy Survey
      dosar Distributed Organization for Scientific and Academic Research
      dzero D0 Experiment at Fermilab
     engage Engagement
   fermilab Fermi National Accelerator Center
       fmri Functional Magnetic Resonance Imaging
       gadu Genome Analysis and Database Update
       glow Grid Laboratory of Wisconsin
        gpn Great Plains Network
      grase Group Researching Advances in Software Engineering
     gridex Grid Exerciser (GEx)
       grow Grid Research and Education Group at Iowa
     gugrid Georgetown University Grid
       i2u2 Interactions in Understanding the Universe Initiative
       ligo Laser Interferometer Gravitational-Wave Observatory
   mariachi Mixed Apparatus for Radar Investigation . . .
    nanohub nanoHUB Network for Computational Nanotechnology (NCN)
      nwicg Northwest Indiana Computational Grid
        osg Open Science Grid
     osgedu OSG Education Activity
     sbgrid Structural Biology Grid
       sdss Sloan Digital Sky Survey
       star Solenoidal Tracker at RHIC
    usatlas United States ATLAS Collaboration
```

OSG Usage Modes

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 & long sequences of similar jobs passing through data sets; 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

European Grid Initiative
http://www.egi.eu/

- The objective of EGI.eu (a foundation established under Dutch law) is to create and maintain a pan-European Grid Infrastructure in collaboration with National Grid Initiatives (NGIs) in order to guarantee the long-term availability of a generic e-infrastructure for all European research communities and their international collaborators
- Coordinating activities between European NGIs EGI.eu will
  - Operate a secure integrated production grid infrastructure that federates resources from providers around Europe
  - Work with software providers within Europe and worldwide to provide high-quality innovative software solutions that deliver the capability required by our user communities

European Grid Initiative
http://www.egi.eu/

- Management Model: EGI Council with representatives from all National Grid Projects
- EGI: Follow-on project to EGEE, EGEE-II and EGEE-III
- Usage Modes:
  - Mostly HTC but not confined to HTC
  - All of OSG Usage Modes and more
  - Many diverse research areas and not just particle-physics

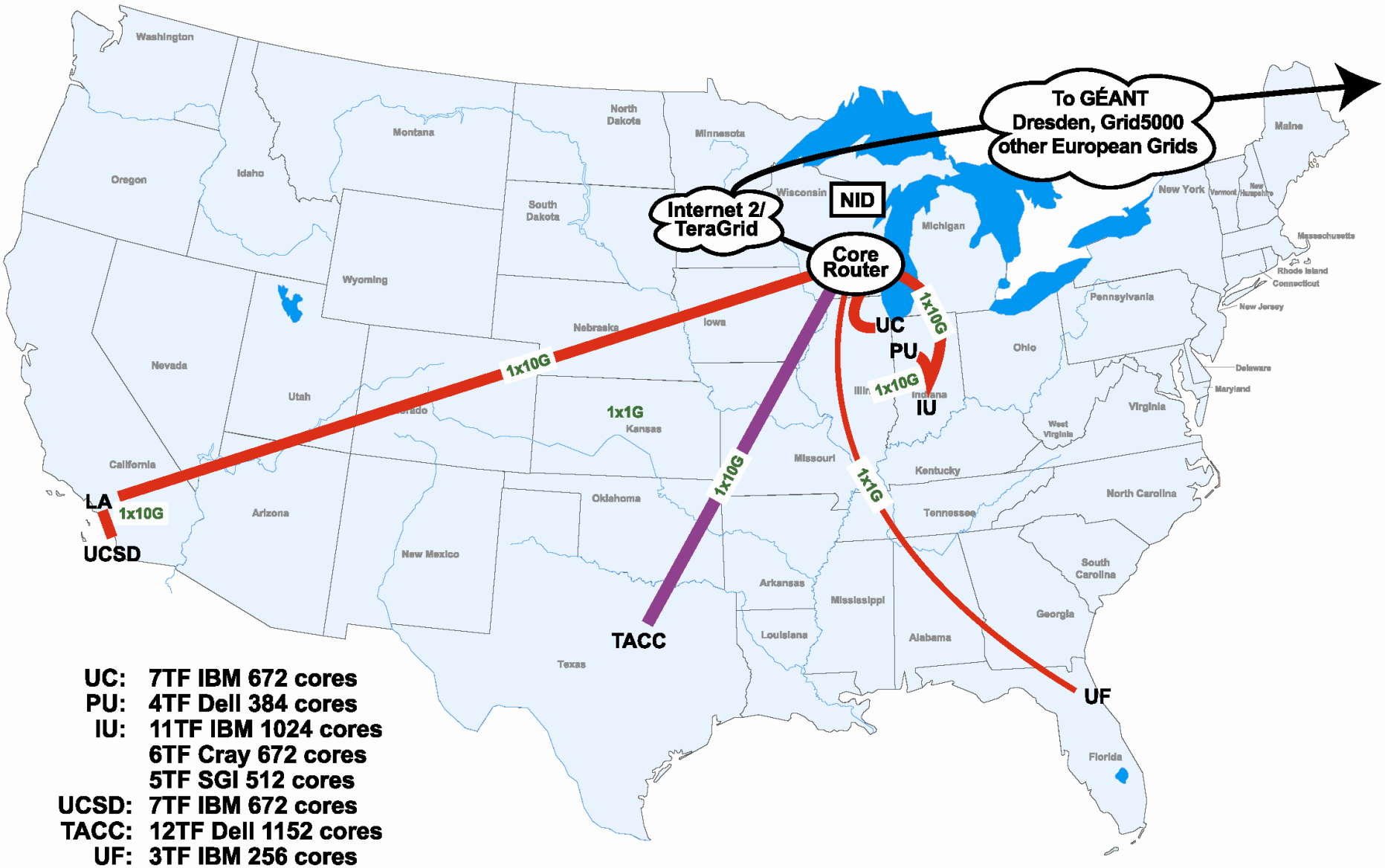
FutureGrid Goals

- Support the research on the future of distributed, grid, and cloud computing.
- Build a robustly managed simulation environment or testbed to support the development and early use in science of new technologies at all levels of the software stack: from networking to middleware to scientific applications.
- Mimic TeraGrid and/or general parallel and distributed systems – FutureGrid is part of TeraGrid and one of two experimental TeraGrid systems (the other is based on GPU)
- Enable major advances in science and engineering through collaborative development of science applications and related software using FG
- FutureGrid is a (small 5600 core) Science Cloud but it is more accurately a virtual machine based simulation environment

FutureGrid Hardware

System type	# CPUs	# Cores	TFLOPS	RAM (GB)	Secondary storage (TB)	Default local file system	Site
Dynamically configurable systems							
IBM iDataPlex	256	1024	11	3072	335*	Lustre	IU
Dell PowerEdge	192	1152	12	1152	15	NFS	TACC
IBM iDataPlex	168	672	7	2016	120	GPFS	UC
IBM iDataPlex	168	672	7	2688	72	Lustre/PVFS	UCSD
Subtotal	784	3520	37	8928	542		
Systems not dynamically configurable							
Cray XT5m	168	672	6	1344	335*	Lustre	IU
Shared memory system TBD	40**	480**	4**	640**	335*	Lustre	IU
Cell BE Cluster	4						
IBM iDataPlex	64	256	2	768	5	NFS	UF
High Throughput Cluster	192	384	4	192			PU
Subtotal	552	2080	21	3328	10		
Total	**1336**	**5600**	**58**	**10560**	**552**		

- FutureGrid has dedicated network (except to TACC) and a network fault and delay generator
- Can isolate experiments on request; IU runs Network for NLR/Internet2
- (Many) additional partner machines will run FutureGrid software and be supported (but allocated in specialized ways)



UC: 7TF IBM 672 cores
PU: 4TF Dell 384 cores
IU: 11TF IBM 1024 cores
6TF Cray 672 cores
5TF SGI 512 cores
UCSD: 7TF IBM 672 cores
TACC: 12TF Dell 1152 cores
UF: 3TF IBM 256 cores

Add Network Fault Generator and other systems running FutureGr

FutureGrid Usage Scenarios

- Developers of end-user applications who want to develop new applications in cloud or grid environments, including analogs of commercial cloud environments such as Amazon or Google.
  - Is a Science Cloud for me?
- Developers of end-user applications who want to experiment with multiple hardware environments.
- Grid/Cloud middleware developers who want to evaluate new versions of middleware or new systems.
- Networking researchers who want to test and compare different networking solutions in support of grid and cloud applications and middleware. (Some types of networking research will likely best be done via through the GENI program.)
- Education as well as research
- Interest in performance requires close to OS support

FutureGrid Architecture

- Open Architecture allows to configure resources based on images
- Managed images allows to create similar experiment environments
- Experiment management allows reproducible activities
- Through our modular design we allow different clouds and images to be "rained" upon hardware.
- Note will be supported 24x7 at "TeraGrid Production Quality"
- Will support deployment of "important" middleware including TeraGrid stack, Condor, BOINC, gLite, Unicore, Genesis II

Amazon AWS

http://aws.amazon.com

- Story goes: Build capacity for X-mas. What do with spare capacity year around?
- "Utility Computing"
  - Around long before Amazon EC2
  - $0.10 per CPU-hour, plus bandwidth cost
- *aaS Model:
  - * = Infrastructure, Software, almost anything
- AWS: A set of APIs which give users access to Amazon technology and content
  - IaaS, but also "people as a service" – Mechanical Turk

Amazon Simple Storage Service (S3)

- Data Storage in Amazon Data Center
- Web Service interface
- No set-up fee, No monthly minimum
- Storage: $0.15 per GB/Month
- Data Transfer: $0.20/GB to transfer data
- Private and public storage
- Each object up to 5GB in size

Amazon Elastic Compute Cloud

- A Web service that provides resizable compute capacity in the cloud. Designed to make Web-scale computing easier
- A simple Web service interface that provides complete control of your computing resources
- Quickly scales capacity, both up and down, as your computing requirements change
- Changes the economics of computing:
  - Pay only for capacity that used; no cost of ownership
    - *a + bc* becomes just *bc*

Amazon Elastic Compute Cloud

- No start-up, monthly, or fixed costs
  - $0.10 per CPU hour
  - $0.20 per GB transferred across Net
- No cost to transfer data between Amazon S3 and Amazon EC2
- More when we do Cloud Computing next week

Azure

- Description: Microsoft's "Platform as a Service" (Paas) offering
  - Platform that is "Available" and "Scalable"
  - Cloud Based around virtualization
- Explicit Cost to Use
  - No cost to transfer data, only to use/store
- "Democratization of Infrastructure"
- Rich Data Abstractions
  - Large user data items: blobs
  - Service state: tables
  - Service workflow: queues
  - Simple and Familiar Programming Interfaces
    - REST: HTTP and HTTPS

Each VM Has…

At Minimum

- CPU: 1.5-1.7 GHz x64
- Memory: 1.7GB
- Network: 100+ Mbps
- Local Storage: 500GB

Up to

- CPU: 8 Cores
- Memory: 14.2 GB
- Local Storage: 2+ TB

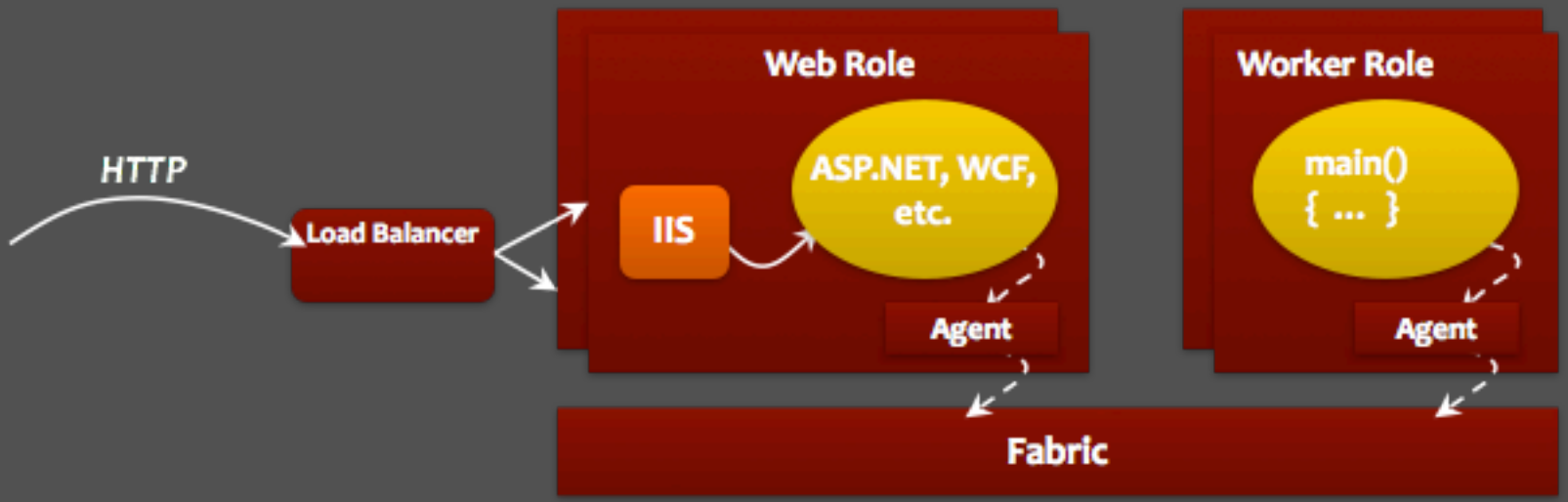
Microsoft .NET Framework

Windows Server Internet Information Services 7.0

Windows Server 2008 Enterprise

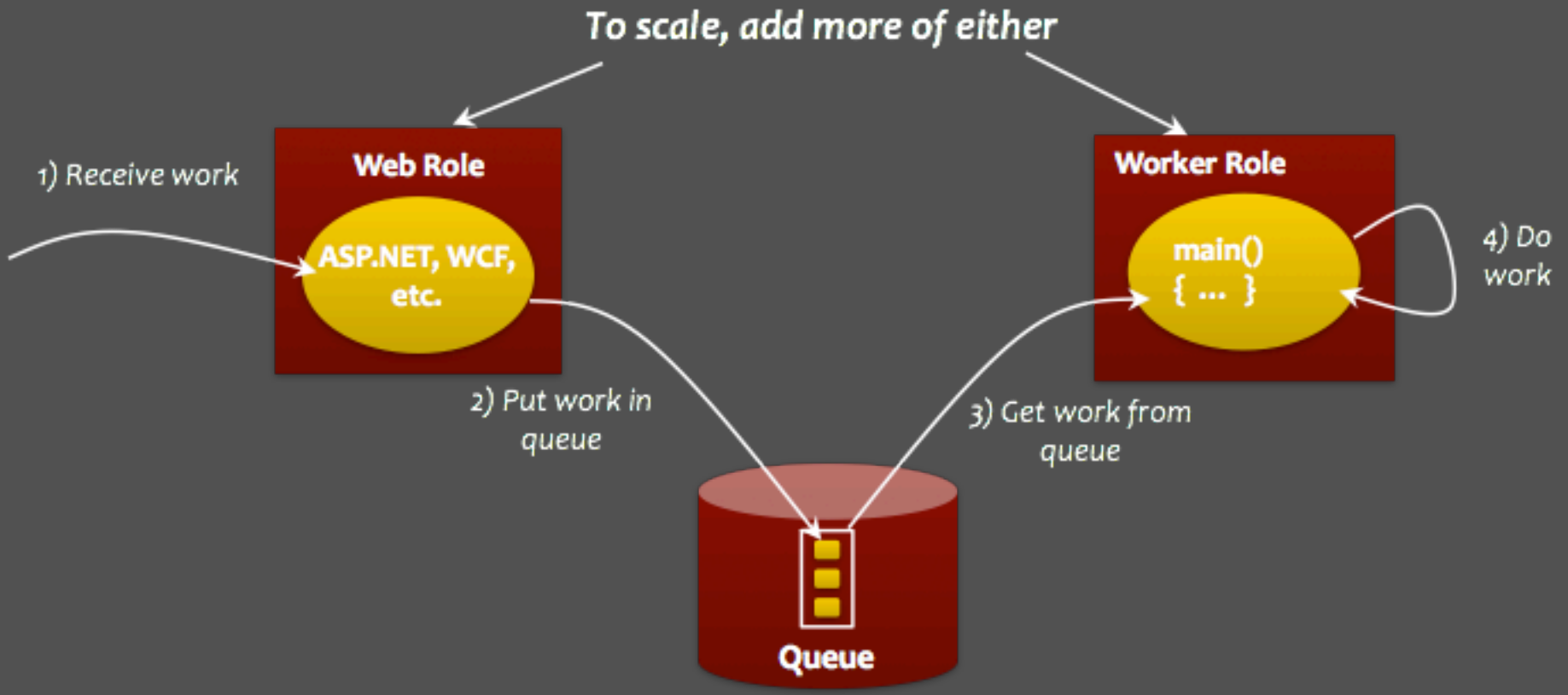
Windows Azure Compute Service

A closer look

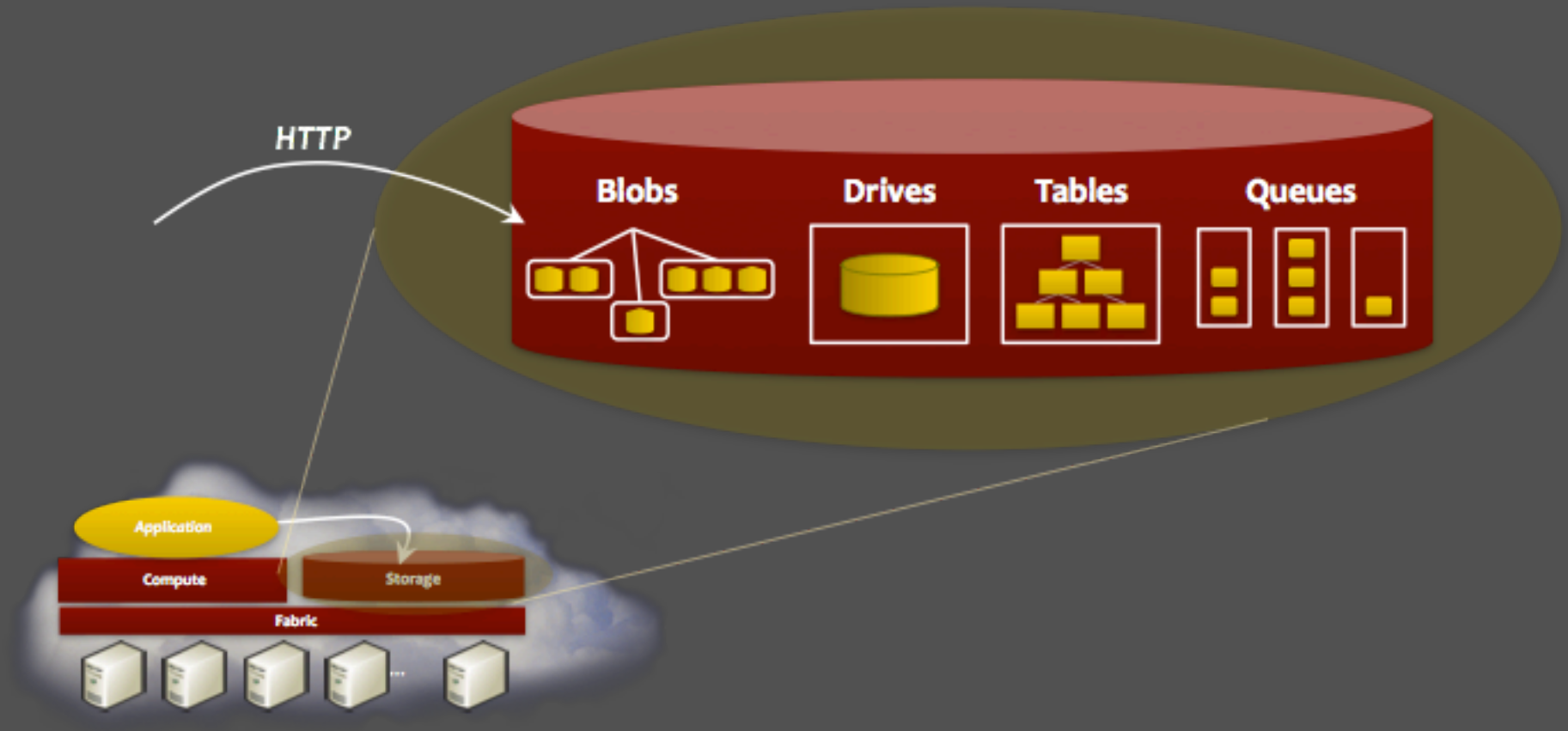


Suggested Application Model

Using queues for reliable messaging



Azure Storage Service
A closer look



DI - Summary

- A rich space of infrastructure & capabilities
- Not all DI have delivered on "their Grid" role
  - Why is difficult to answer, as many co-dependent factors, e.g. policy, social/technical/software ecosystem, external pressures..
  - But in general, "Narrow grid" (e.g OSG) have done better versus "Broad grids" (eg Teragrid)
- Interesting developments in the commercial sector: Google, Microsoft and Amazon
  - First time ever, Academic Research being done on commerical infrastructure!
  - Due to rise of data-intensive computing but also well designed infrastructure (Azure) and effective abstractions for applications (MapReduce)

Overview of Module E
Distributed Scientific Computing

- E1: Introduction to Distributed Scientific Computing
  - Examples of Distributed Applications : [30 minutes]
    - WLCG + Application Examples
    - Summarize and compare with HPC
  - Examples of "Production" Grid Infrastructure [30 min]
    - HPC vs HTC, Research vs Production, Commercial
  - Distributed Computing is Hard. Why?
- E2: Introduction to the Practice of DSC [45 mins]
  - Introduction to SAGA and installing it
  - Writing your first distributed application
- Mod E Project [10mins]

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Introduction to SAGA and SAGA-BigJob

Piloting Many Simulations on Many Supercomputers

RADICAL TEAM
http://radical.rutgers.edu

“Many Simulations” Scenarios: Context

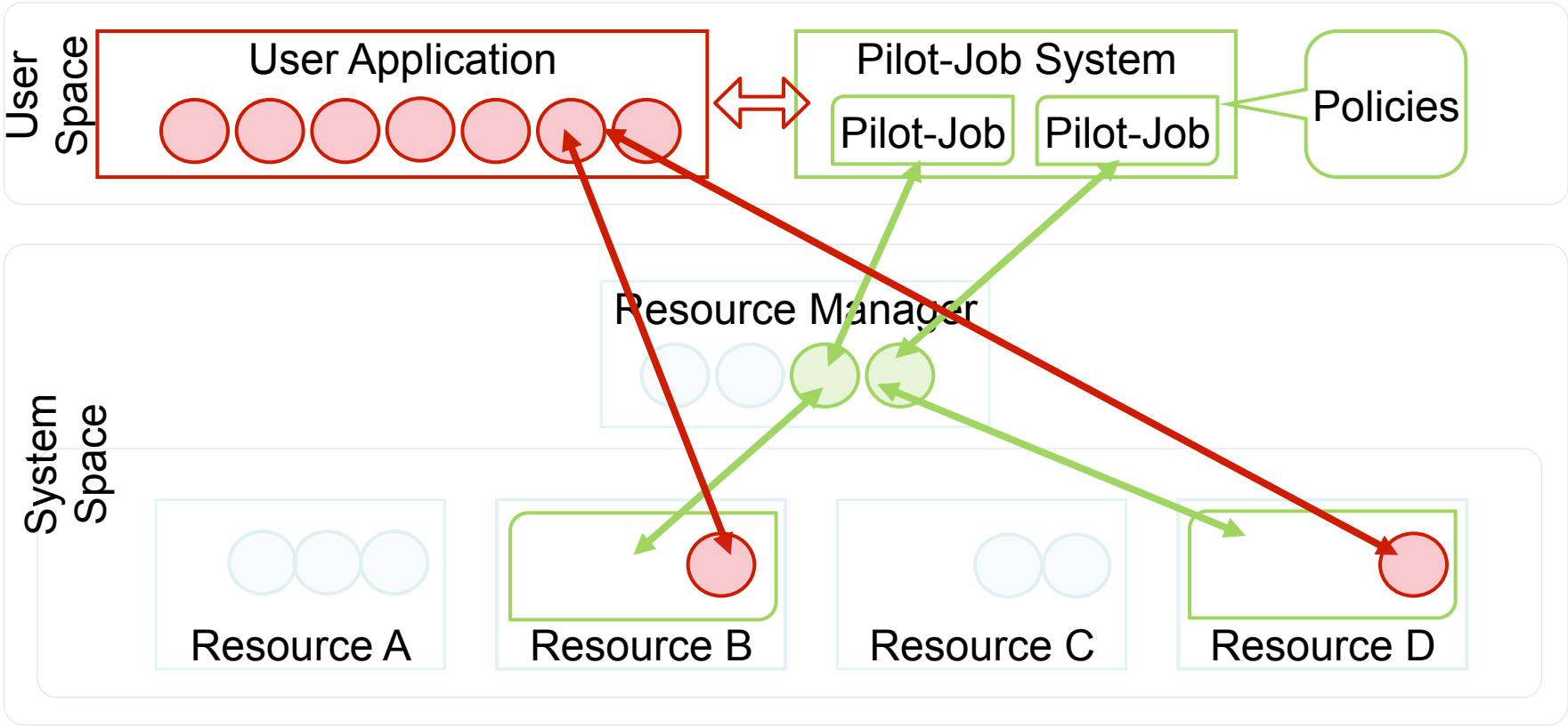
- A Single “Application” is broken into many smaller tasks
  - Naturally decomposed or “by design” (algorithmic and infrastructural)
  - Varying task duration: hours to days to weeks
  - Often these tasks are “coupled”:
    - General concept for data sharing, synchronization, other dependency
- Decouple workload and resource management
- 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

- 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-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

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

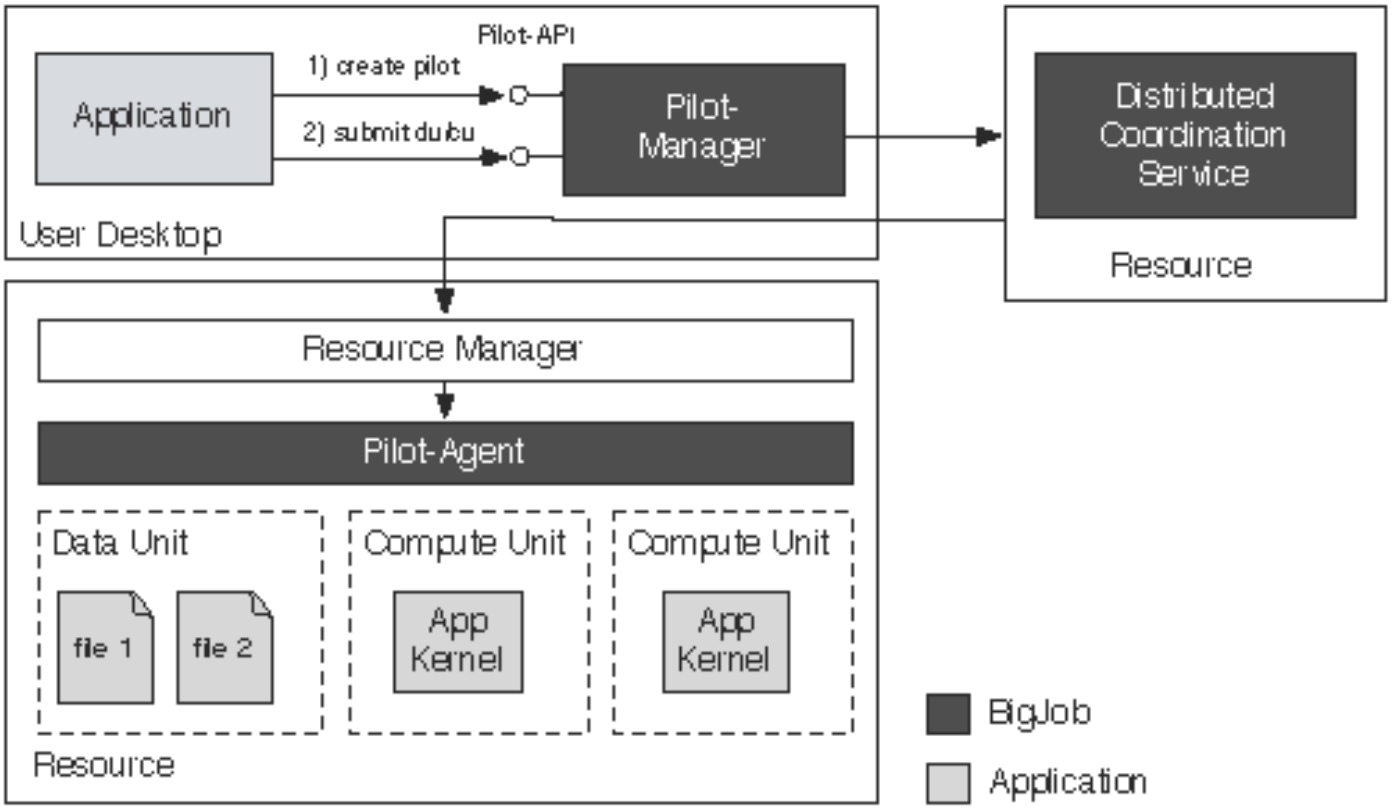
- **Multi-level scheduling**. Tasks of a workload are scheduled on one or more pilots and the pilots are then scheduled on a given resource

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

RUTGERS

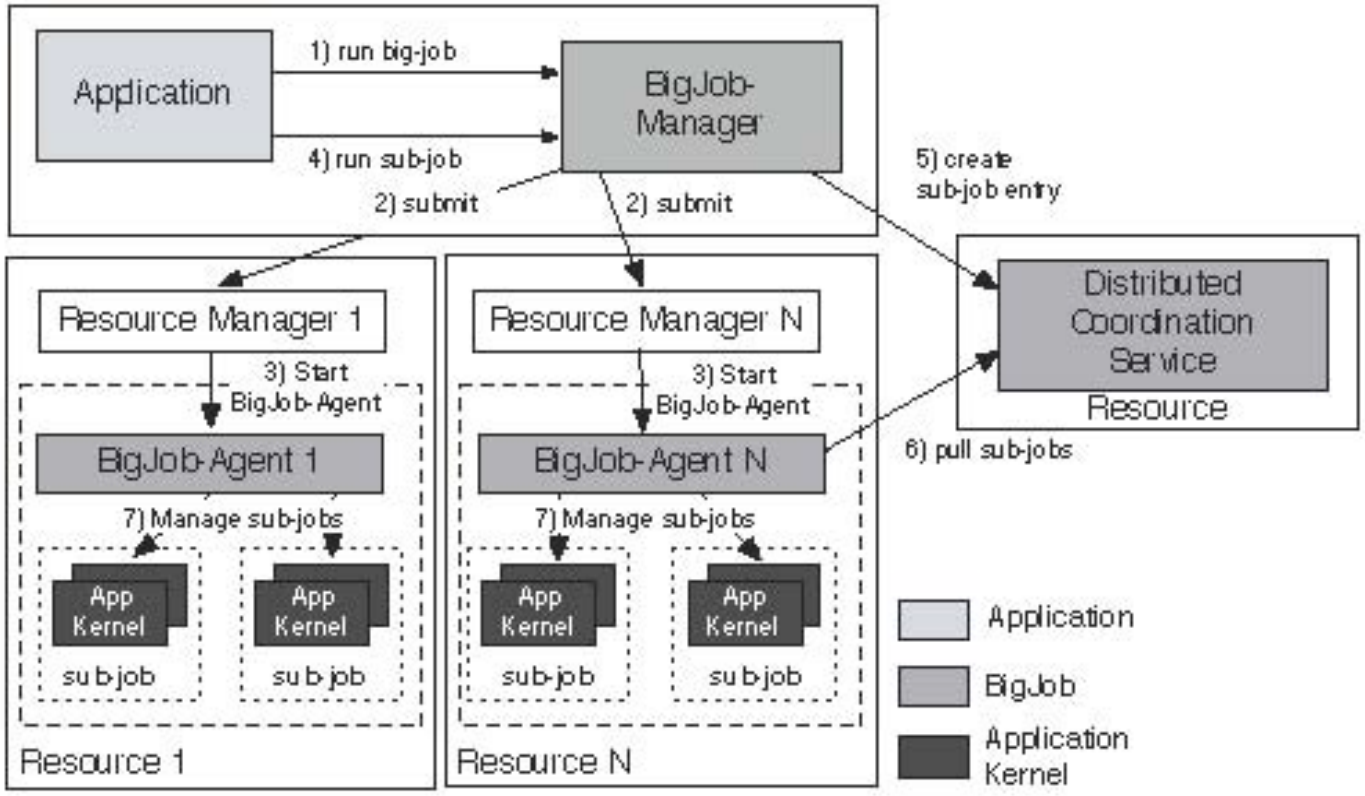
BigJob: A simple, extensible and interoperable Pilot-Job System

BigJob: Architecture

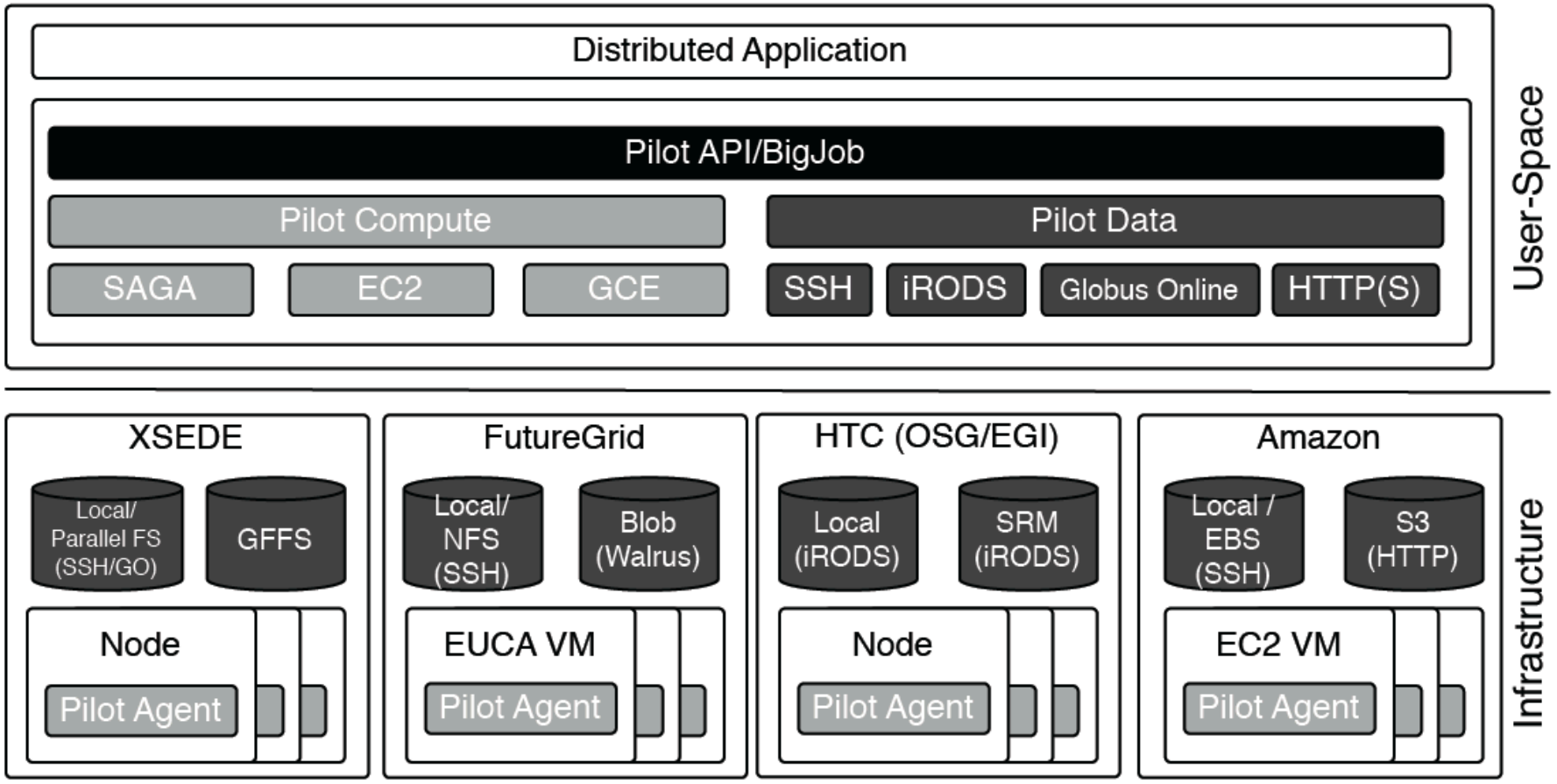


BigJob: Distributed View

(Strongly encourage you to try after the tutorial)

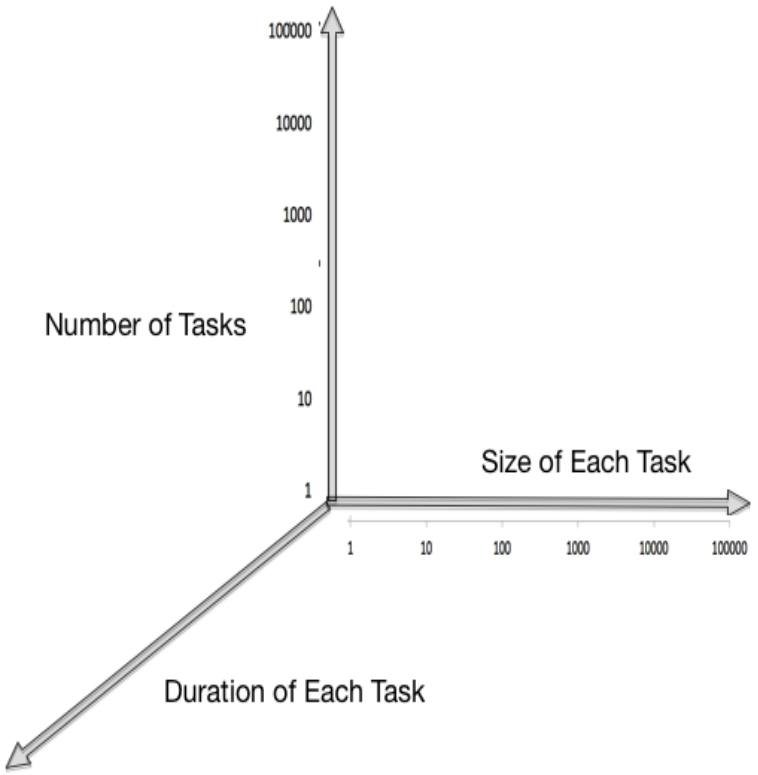


BigJob: Resource Interoperability



Design Objective: Scaling Along Many Dimensions

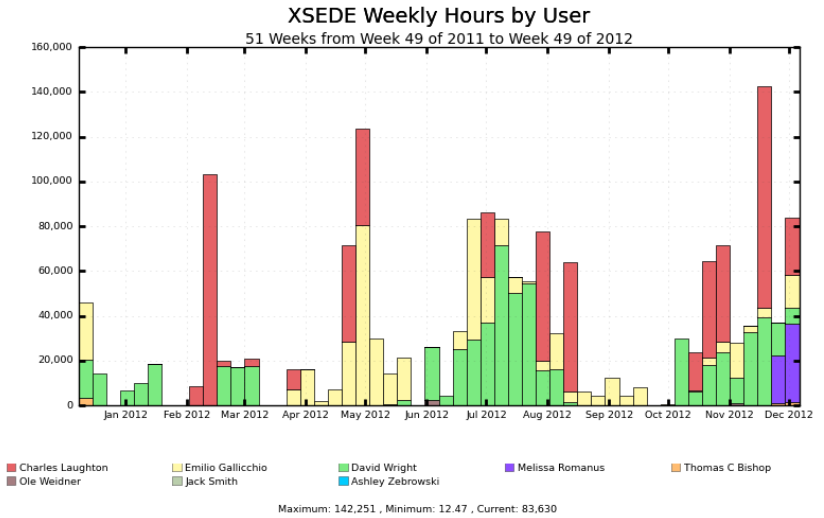
- Scaling Dimension #1: Size of each task
  - *Scale-up*
  - 1 core per task,128/512…..
- Scaling Dimension #2: Total Number of Concurrent tasks
  - *Scale-Out*
  - Enhanced (MD) sampling O(1000), statistical errors O(10^6) tasks,
- Scaling Dimension #3: Total Number of Tasks/per unit time
  - Real time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

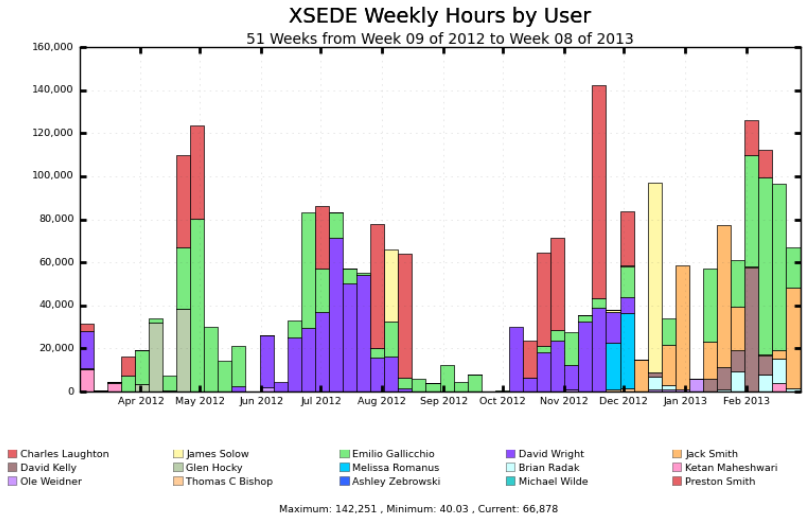


"Coarse-Grained" BigJob Performance

- 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

RUTGERS

SAGA: Interoperability Layer for BigJob

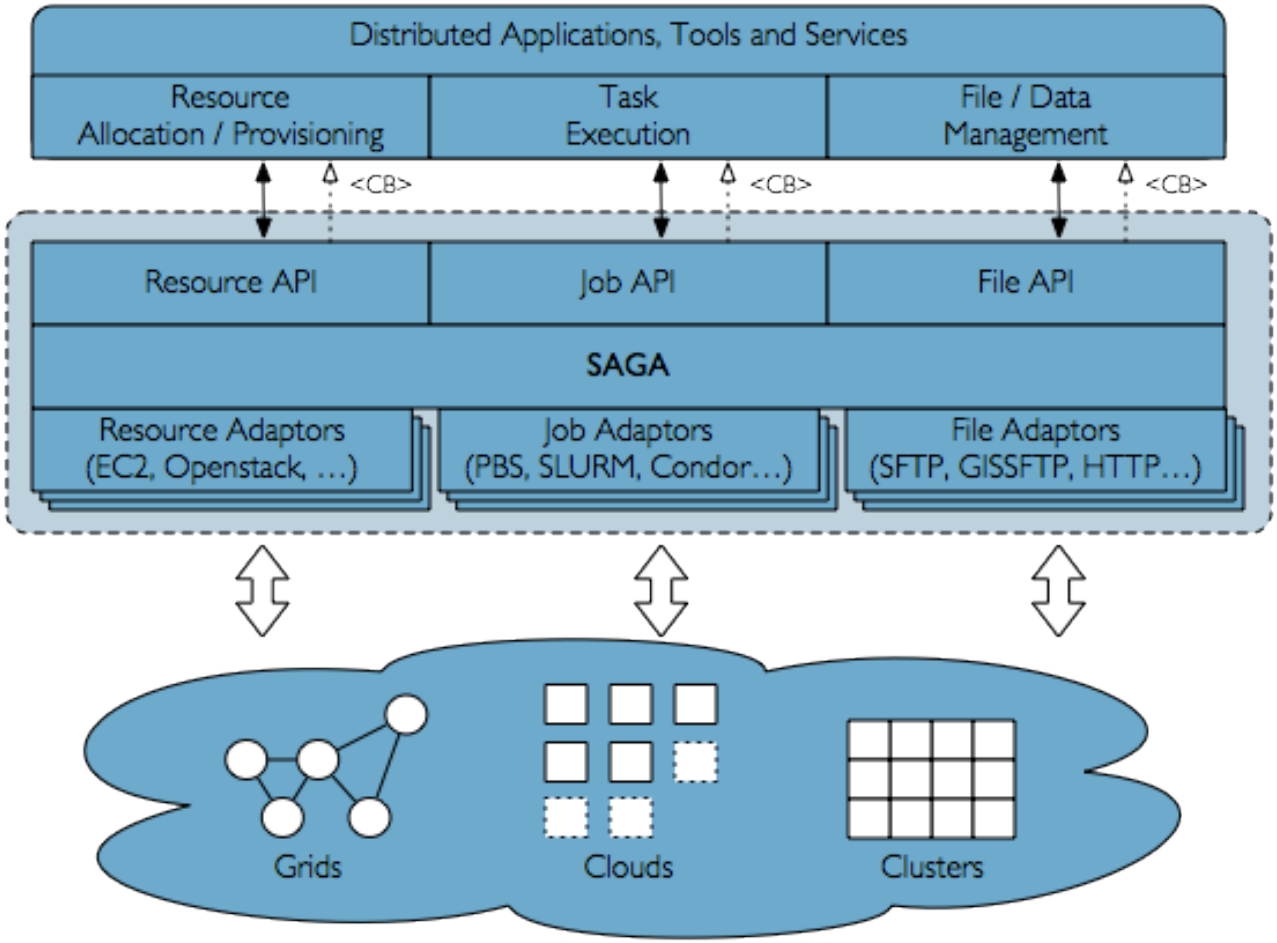
http://saga-project.org
http://saga-project.github.io/saga-python

A Light-Weight Python Access Layer for Distributed Computing Infrastructure

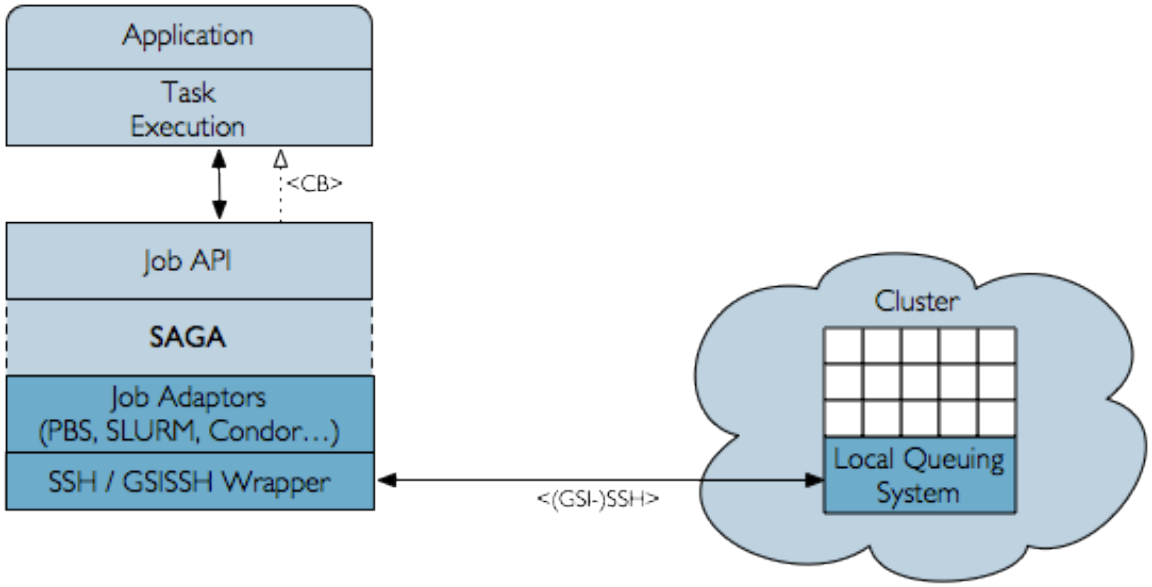
Overview

- 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

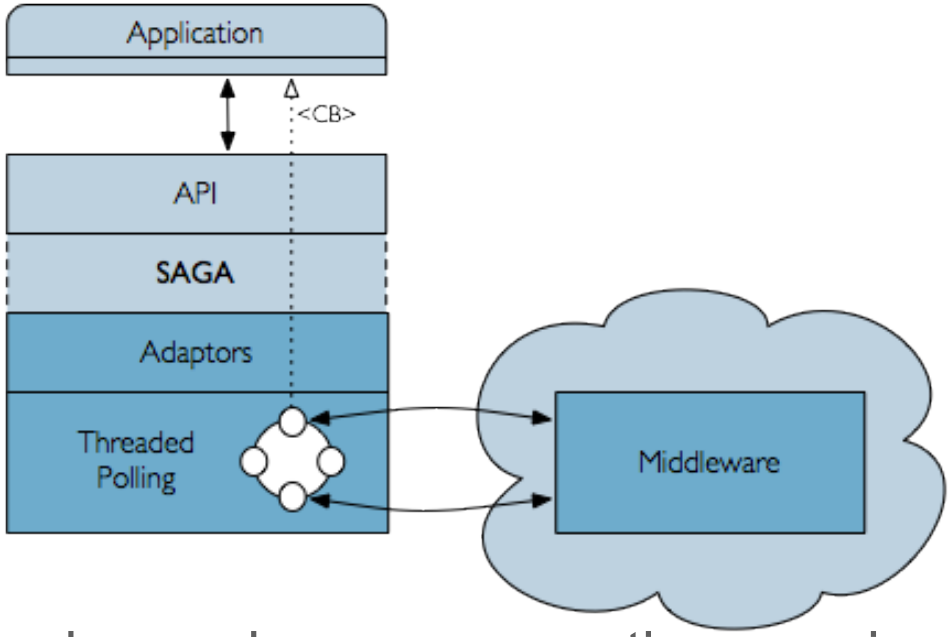

Distributed Applications, Tools and Services
Resource Allocation / Provisioning
Task Execution
File / Data Management
<CB>
<CB>
<CB>
Resource API
Job API
File API
SAGA
Resource Adaptors (EC2, Openstack, …)
Job Adaptors (PBS, SLURM, Condor…)
File Adaptors (SFTP, GISSFTP, HTTP…)
Grids
Clouds
Clusters

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

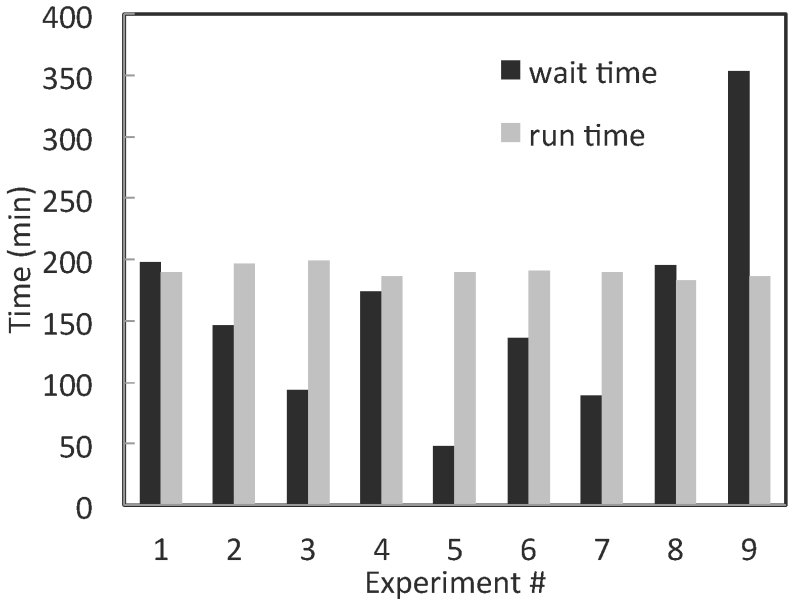
SAGA: Interoperability layer upon which other tools and applications are built

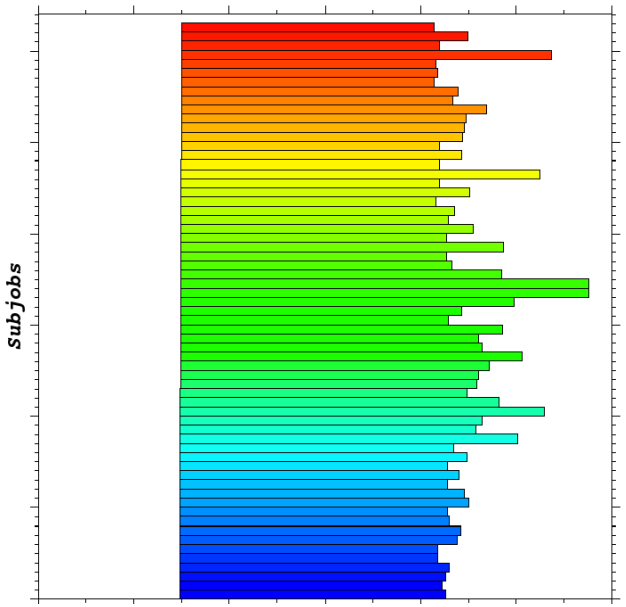
- HOW SAGA is Used?
  - Uniform Access-layer to DCI, e.g, EGI, XSEDE, etc
  - Application "Scripting Layer" to DCI
  - Build tools, middleware services and capabilities e.g. Gateways,
- WHAT is SAGA Used for?
  - Production-grade science and engineering and research tool to design, implement and reason 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

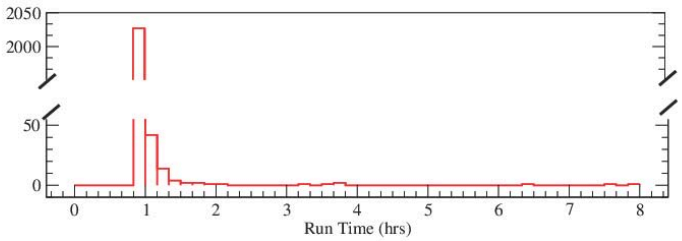
Case Studies: Varying the “coupling” between many (MD) Simulations

HT-HPC on Kraken

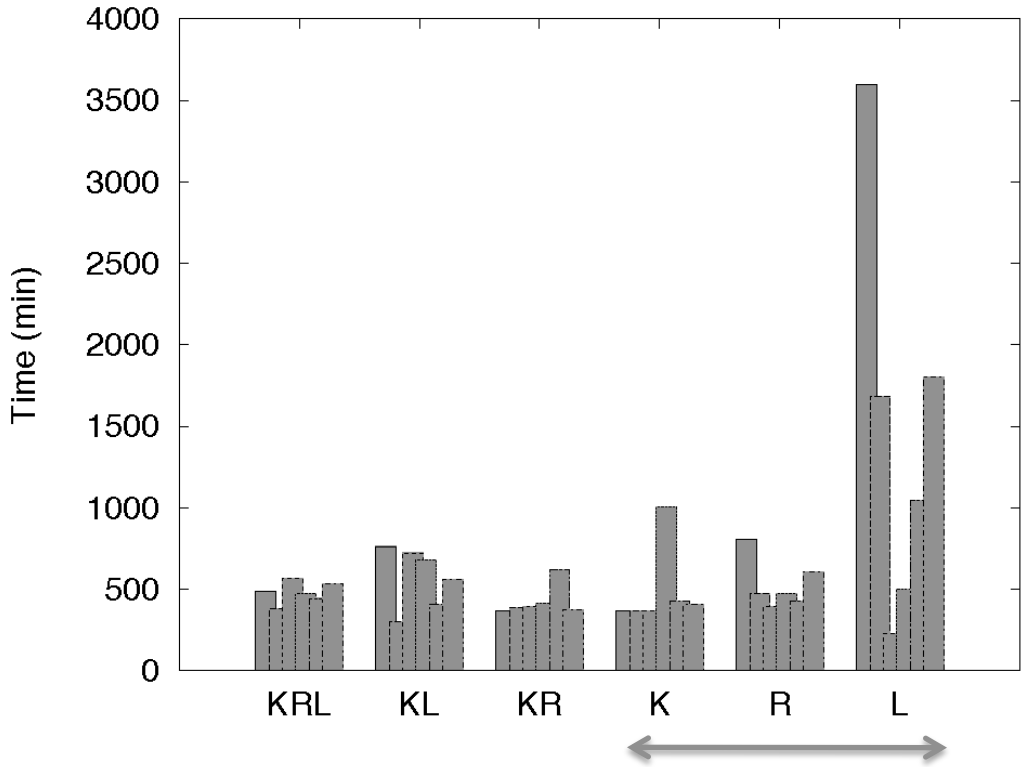
126 ensembles, each of 192 cores = 24192 cores

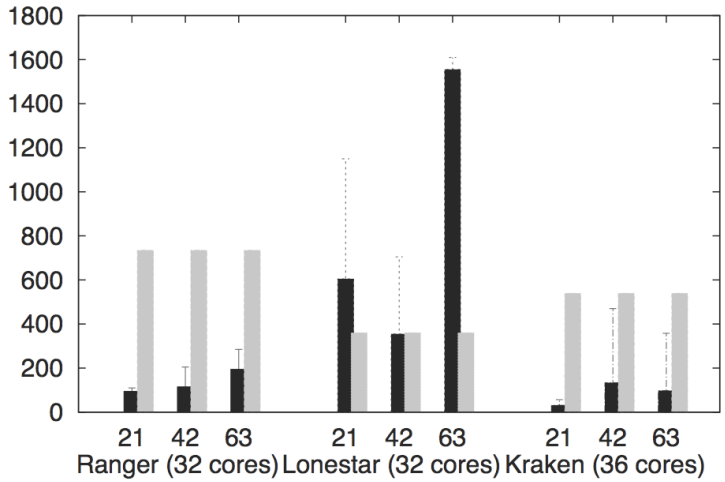






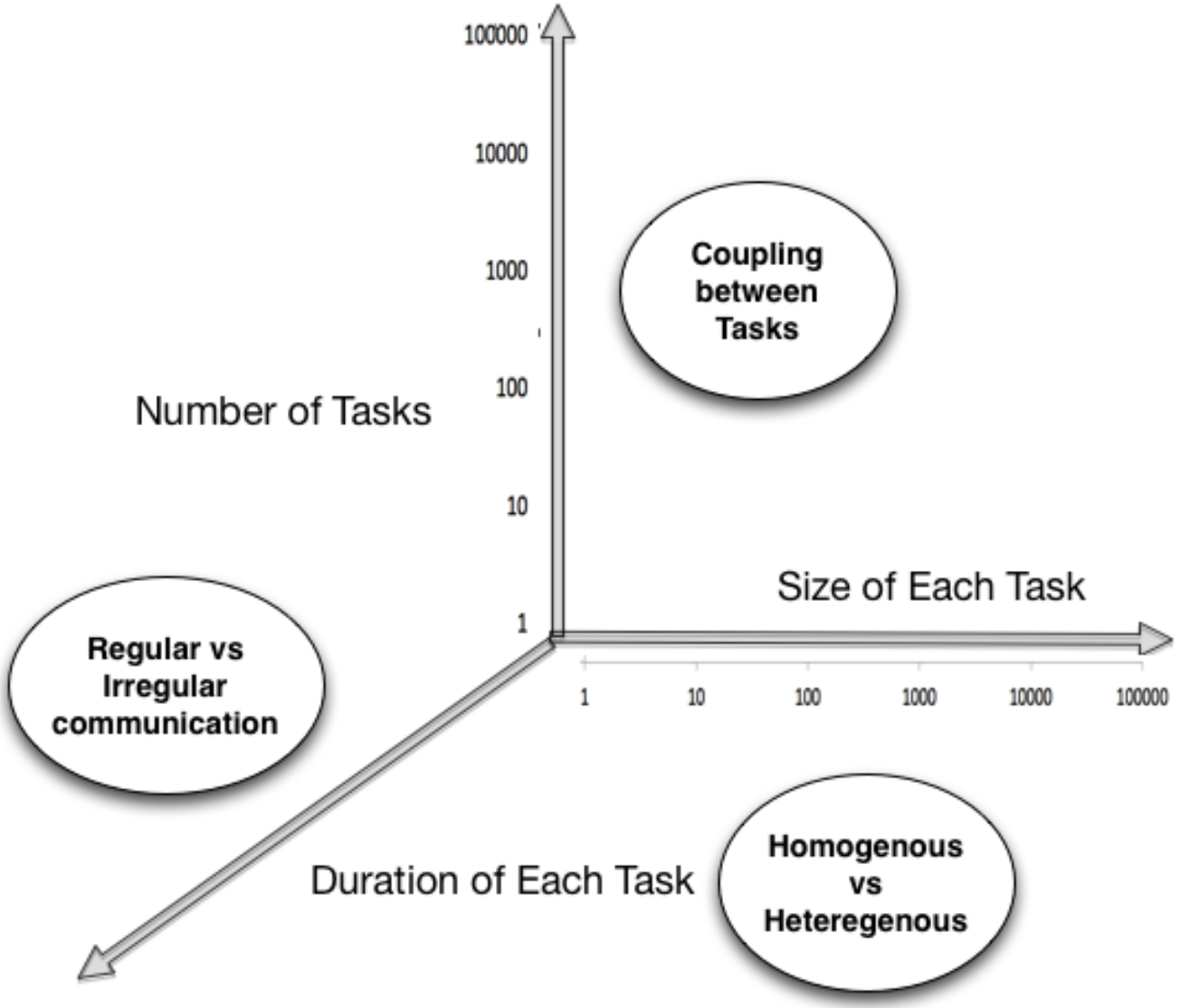
Scale-Out/Across





X-axis: number of tasks (size)

Scaling Along Many Dimensions

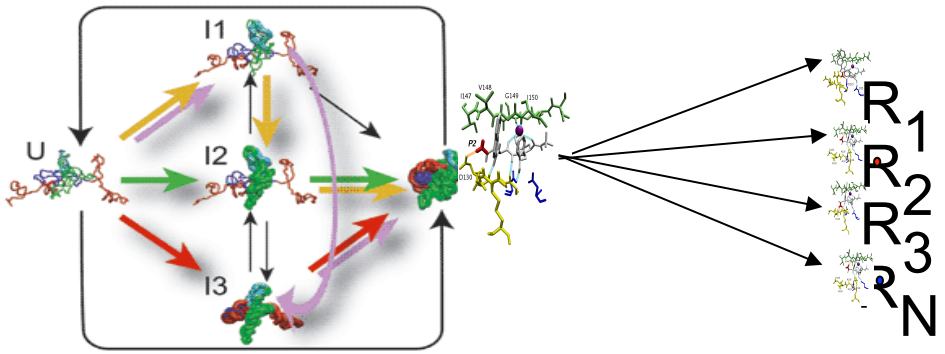


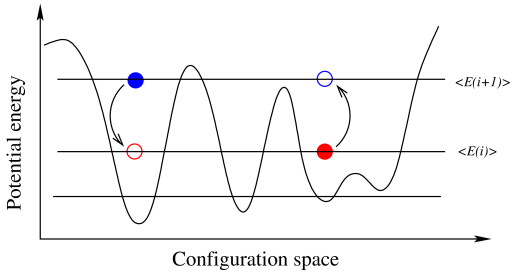
Varying the Coupling between Simulations

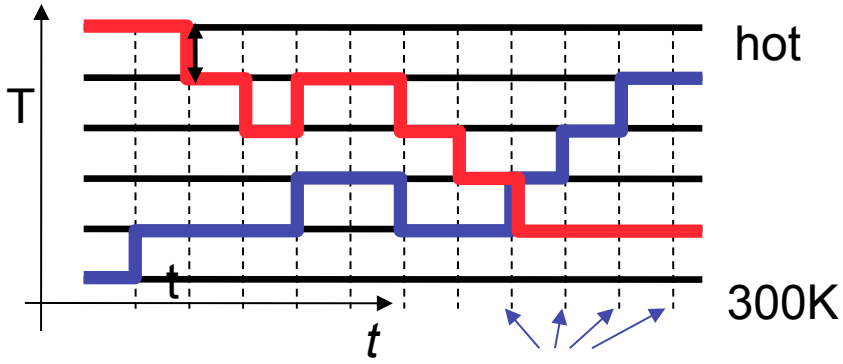
- Large Ensemble of Uncoupled (Molecular) Simulations:
  - Enhanced Sampling of bio-molecular systems
- Multiple-Chaining of Simulations
  - Long time-scale simulations
- Ensemble of Loosely Coupled Simulations
  - Replica-Exchange Class of Algorithms
  - Understand protein-ligand recognition via (multidimensional)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

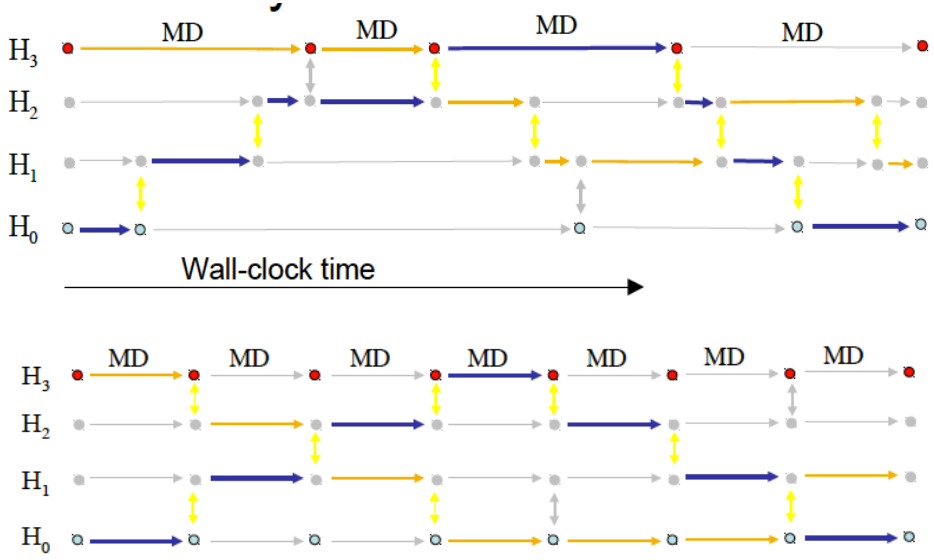
Replica Exchange Molecular Simulations



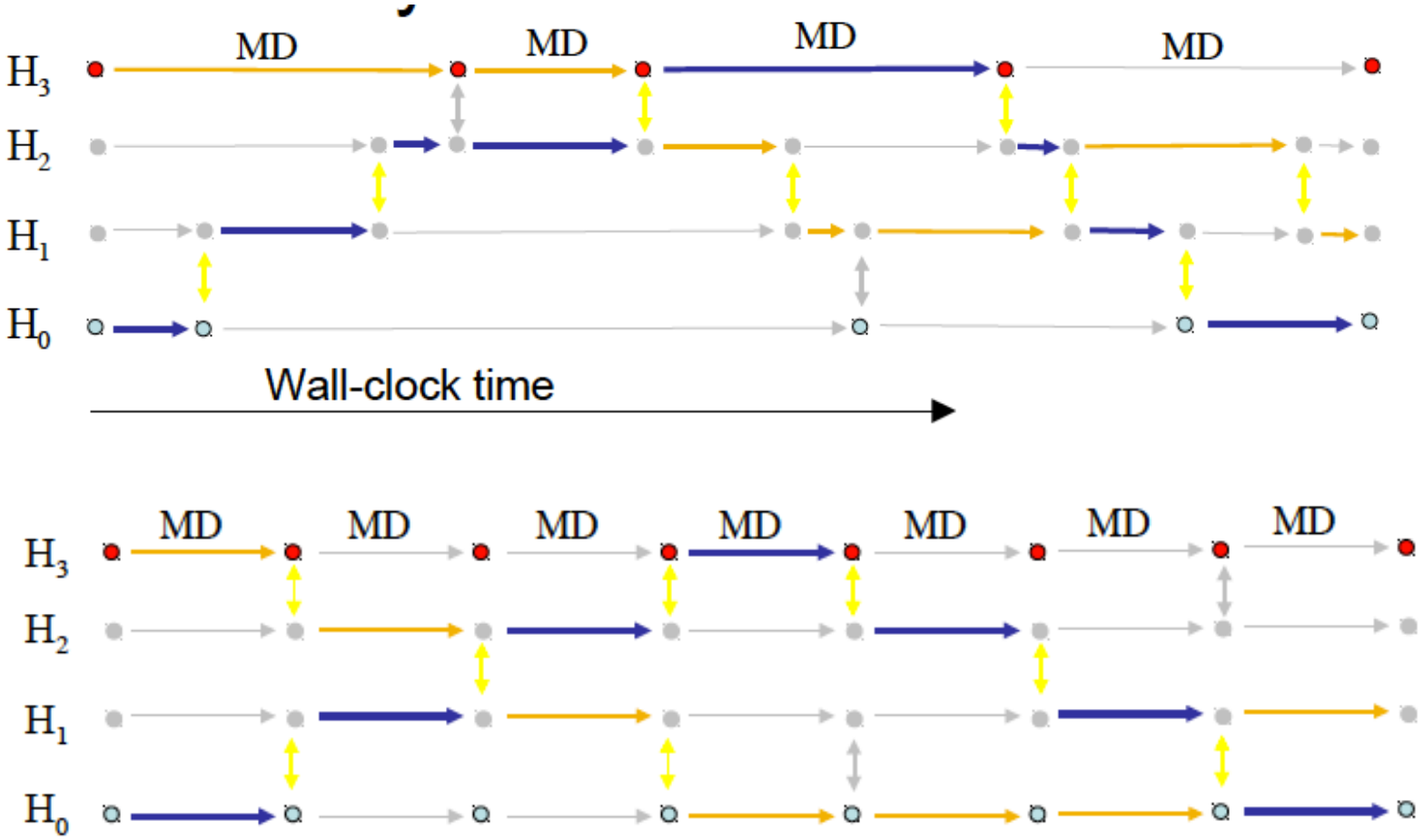




Exchange attempts



Irregular Synchronization



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

Homework #1

- Go to http://saga-project.org
- Focus on SAGA-Python
  - http://saga-project.github.io/saga-python/
- Find link to Tutorial:
  - http://saga-project.github.io/saga-python/doc/tutorial/
- Do tutorial. Send me output of every subsection, describe the output (why?) of each sub-section.
  - Use Stamped, Trestles and/or Lonestar for remote tests
- Use mailing list:
  - saga-users@googlegroups.com

Homework #2

- BigJob Homepage:
  - http://saga-project.github.io/BigJob/
- Find link to Tutorial
  - http://saga-project.github.io/BigJob/sphinxdoc/tutorial/index.html
- Do tutorial. Do tutorial. Send me output of every subsection, describe the output (why?) of each sub-section.
  - Use Stampede, Trestles and/or Lonestar for remote tests
- Use Mailing lists (bigjob-users@googlegroups.com)

If you have access to Stampede, URL for Tutorial:
https://github.com/saga-project/tutorials/wiki/XSEDE13

Module E: Project Suggestions

- Gain sufficient proficiency with SAGA to write a M-W application that uses > 1 FutureGrid resource?
- Extend Mandelbrot Set? Use > 1 FutureGrid Resource?
- PySAGA M-W? Implement another pattern?
- See FG Tutorial on Nimbus and Eucalyptus
  - Can use SAGA to submit jobs to Clouds
  - M-W to submit to FG-Bare Metal & Clouds?
- Written Project. Ideas?
- *Teamwork is acceptable provided: (i) effort is acknowledged, (ii) clear intellectual contribution from each*