# Module E:
# Distributed Scientific Computing

## Lecture E-1: Introduction to the Practice of Distributed Computing

Dr Shantenu Jha

# Module Dynamics

- Shantenu Jha [first-name at lsu dot edu]
  - Office Hours: 24x7 till end of semester!
    - But no fixed office hours
  - Office Location: 214 Johnston Hall
    - Or leave message / material with Brittany 216 JH
- Grading:
  - 2 Assignments [5% x 2]
  - 1 Project [10%]
  - Impress Me [20% without any Assignment or Project]

# Overview of Module E
# Distributed Scientific Computing

- E1: Introduction to the Practice of DC
  - WLCG + Application Examples [30mins]
  - Summarize and compare with HPC [10mins]
    - Distributed Computing is Hard. So Why?
  - Introduction to SAGA and installing it [20mins]
  - Mod E Project [10mins]
- E2: Introduction to the Practice of DC - II
  - Examples of "Production" Grid Infrastructure [20min]
    - HPC vs HTC, Research vs Production, Commercial
  - More on SAGA [40 mins]
  - Your first (?) Distributed Application [15mins]

# Overview of Module E
# Distributed Scientific Computing

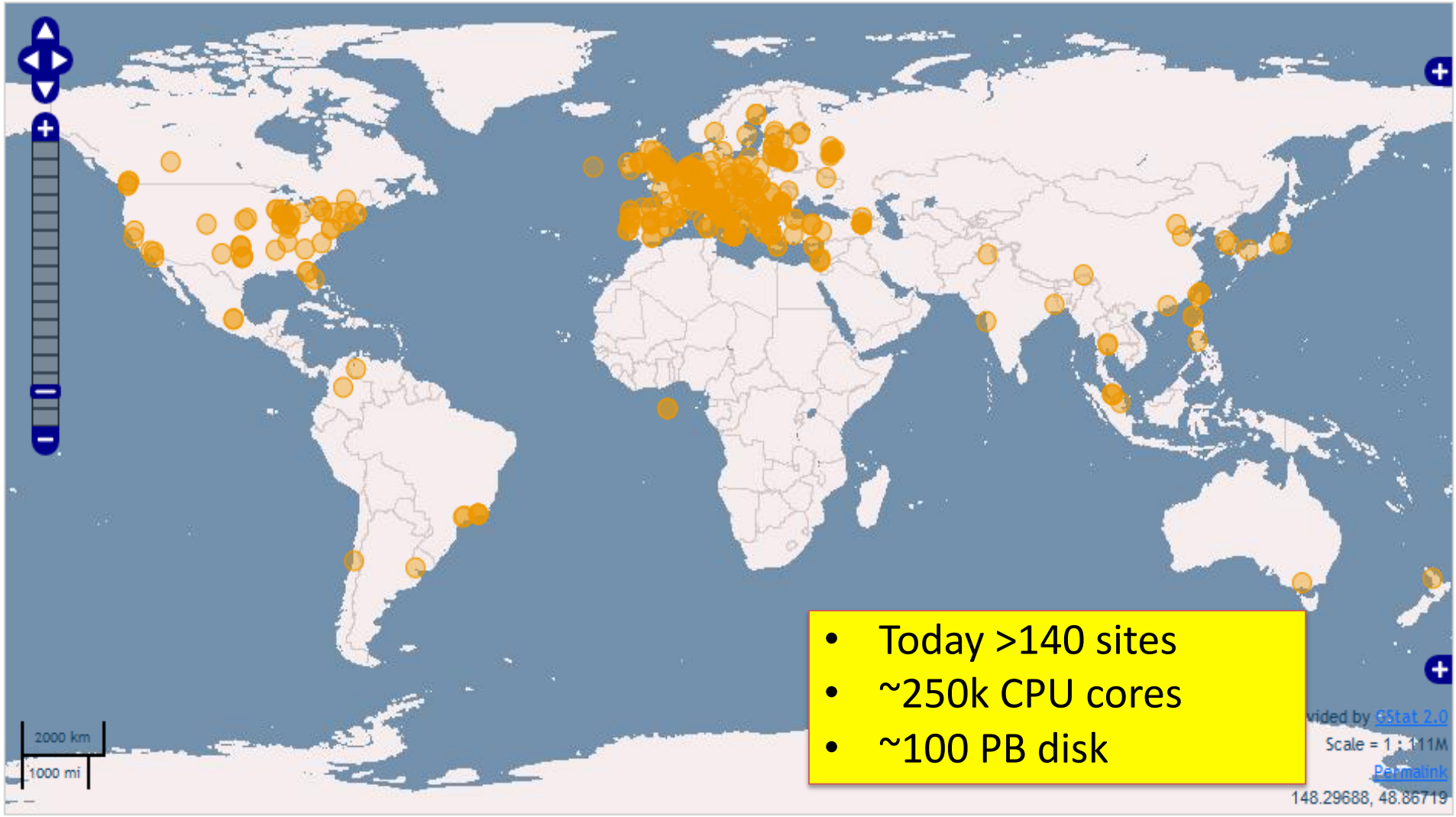
- E3: Introduction to Cloud Computing
  - Introduction to Commercial / Enterprise DC aka Cloud Computing (EC2, Azure, Google) [30mins]
  - Using FutureGrid [30mins]
  - Using the Master-Worker Pattern [15mins]
- E4: To Distribute or not to Distribute?
  - Distributed Applications Redux [20mins]
  - Principles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

# 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) ?

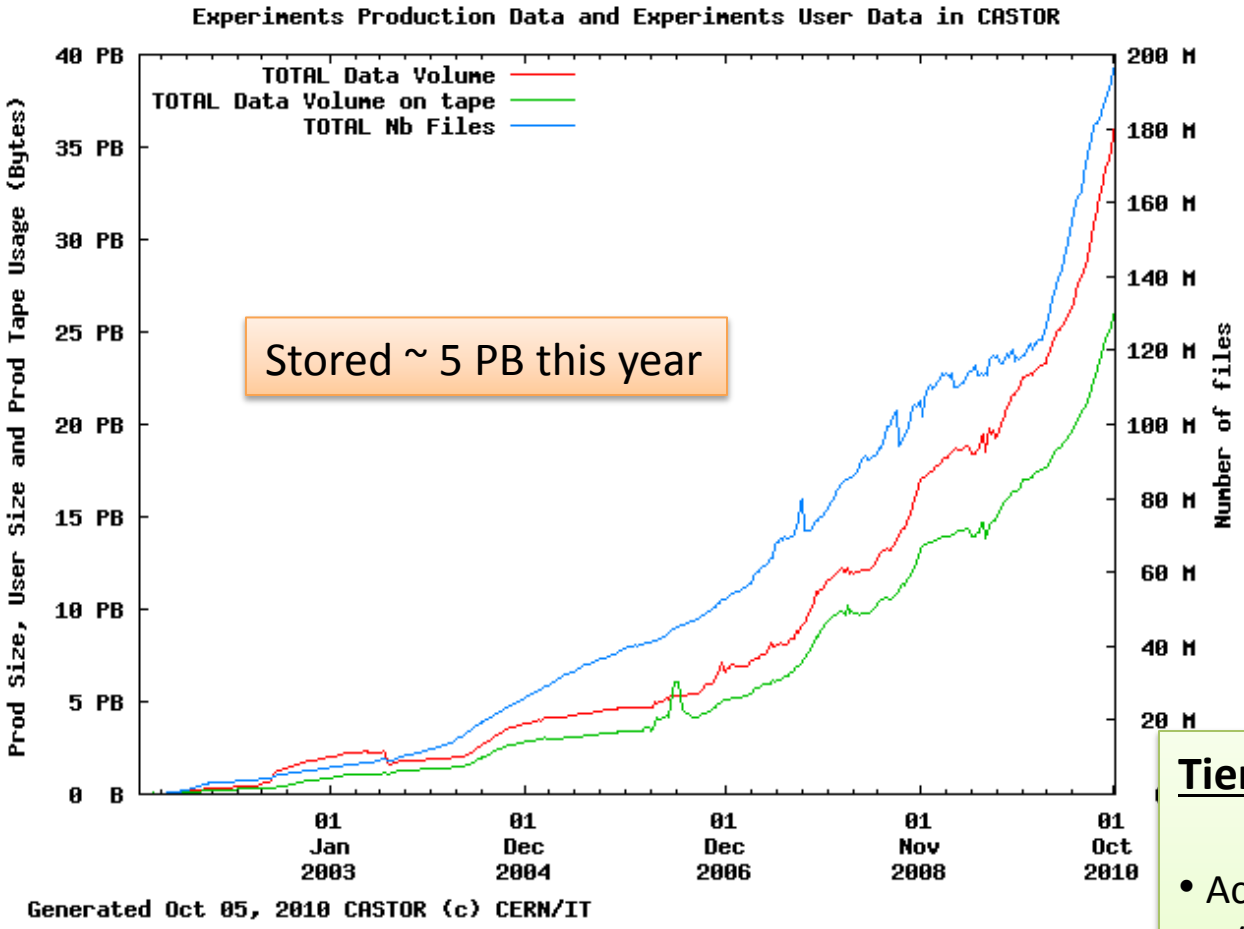
# WLCG: Worldwide LHC Computing Grid
# A Driver of Grid Comuting

Worldwide resources for WLCG




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

# 6 months of LHC data




Stored ~ 5 PB this year

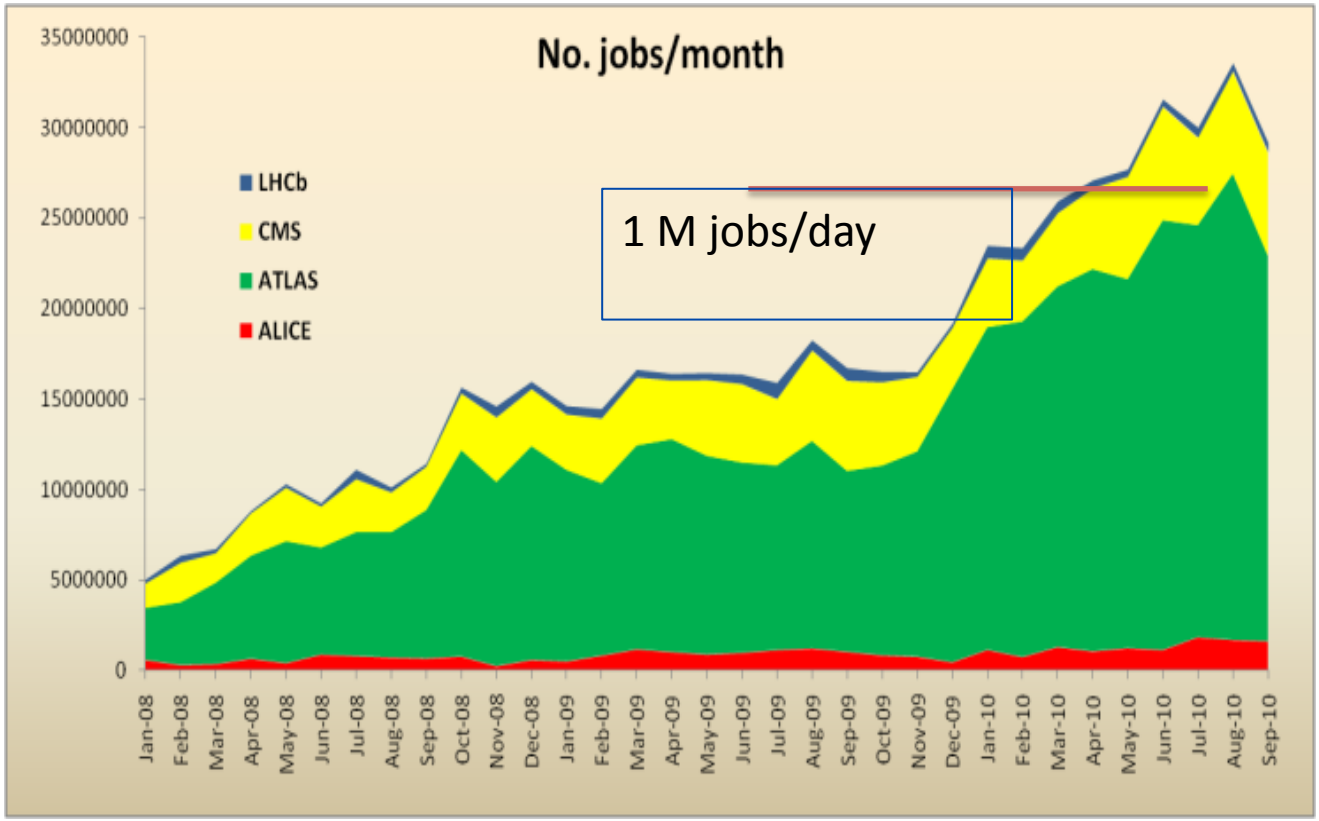
**Tier 0 storage:**

- 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**

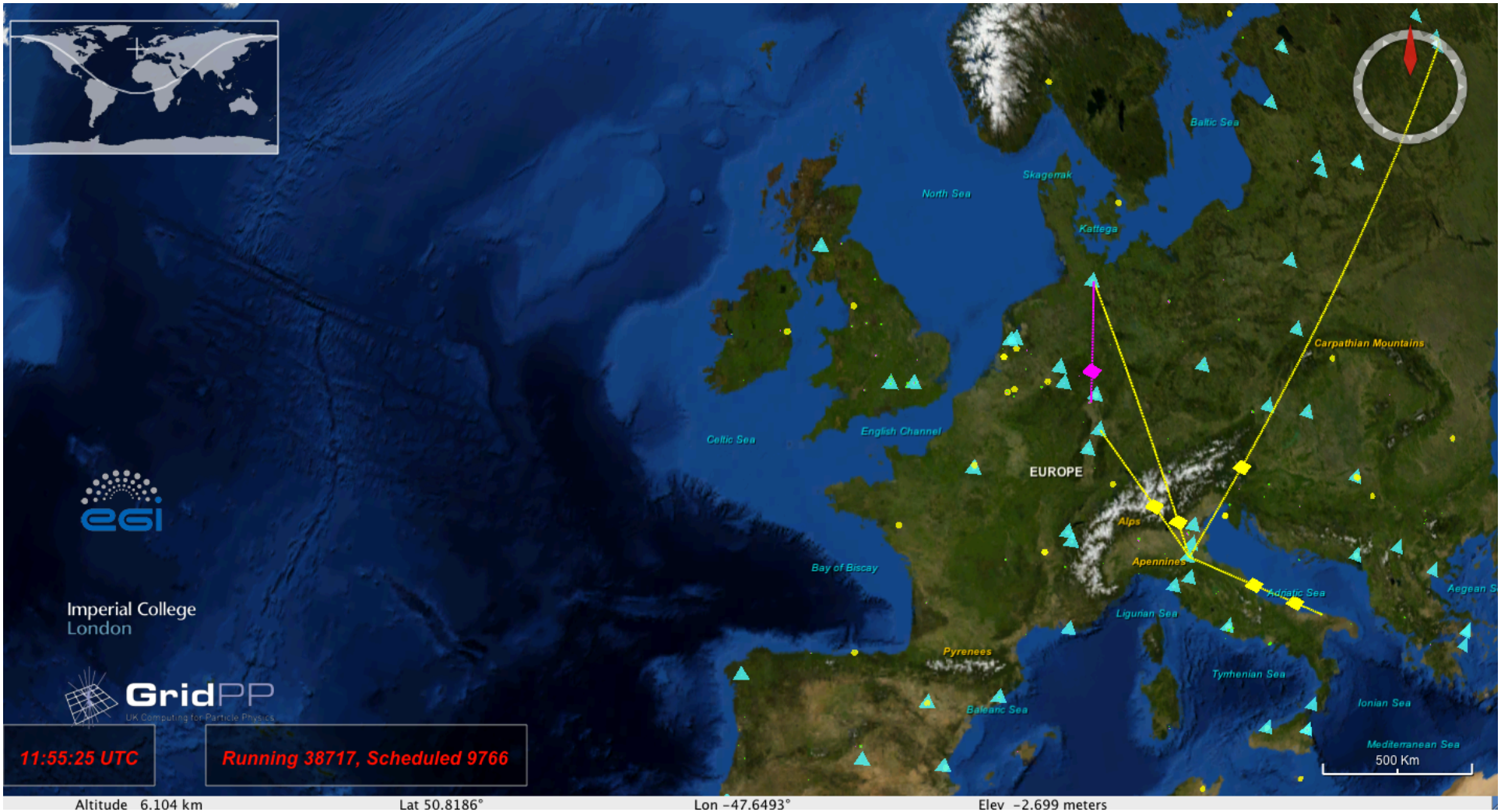
# WLCG Usage

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

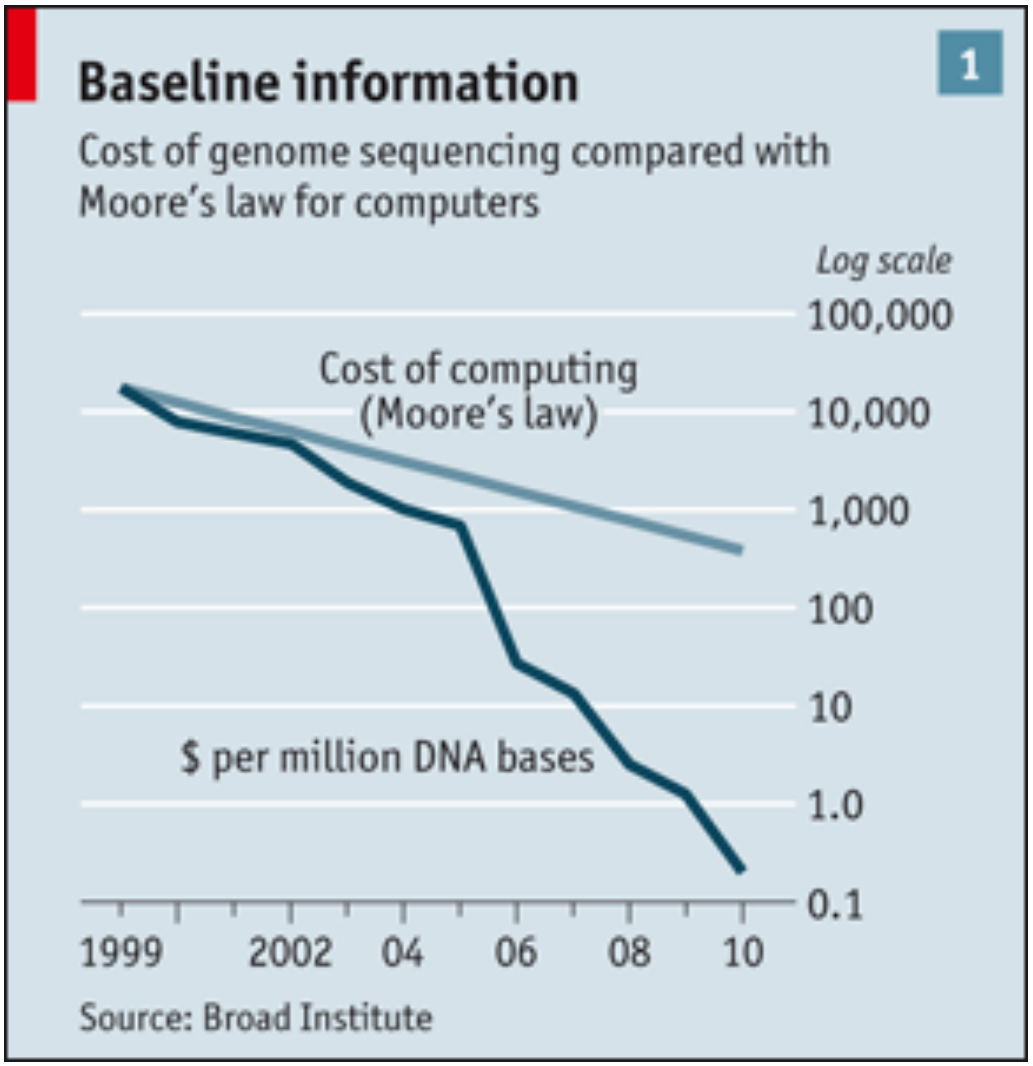
~50% utilisation



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



# An Interesting Observation….




HW Assignment:  Think about the implications of this graph.

# What is Montage?

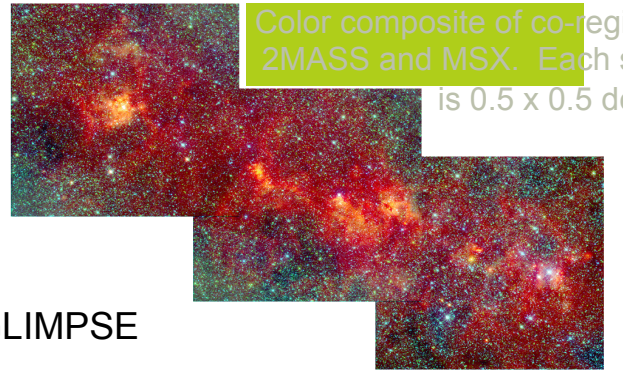
- Delivers custom, science grade image mosaics
  - An image mosaic is a combination of many images containing individual pixel data so that they appear to be a single image from a single telescope or spacecraft
  - User specifies projection, coordinates, spatial sampling, mosaic size, image rotation
  - 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


CENTER FOR COMPUTATION
& TECHNOLOGY

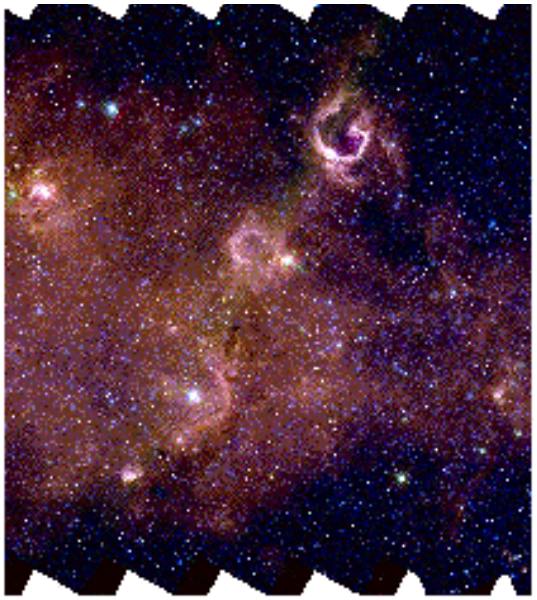
# MONTAGE

# Montage use by Spitzer Legacy Teams

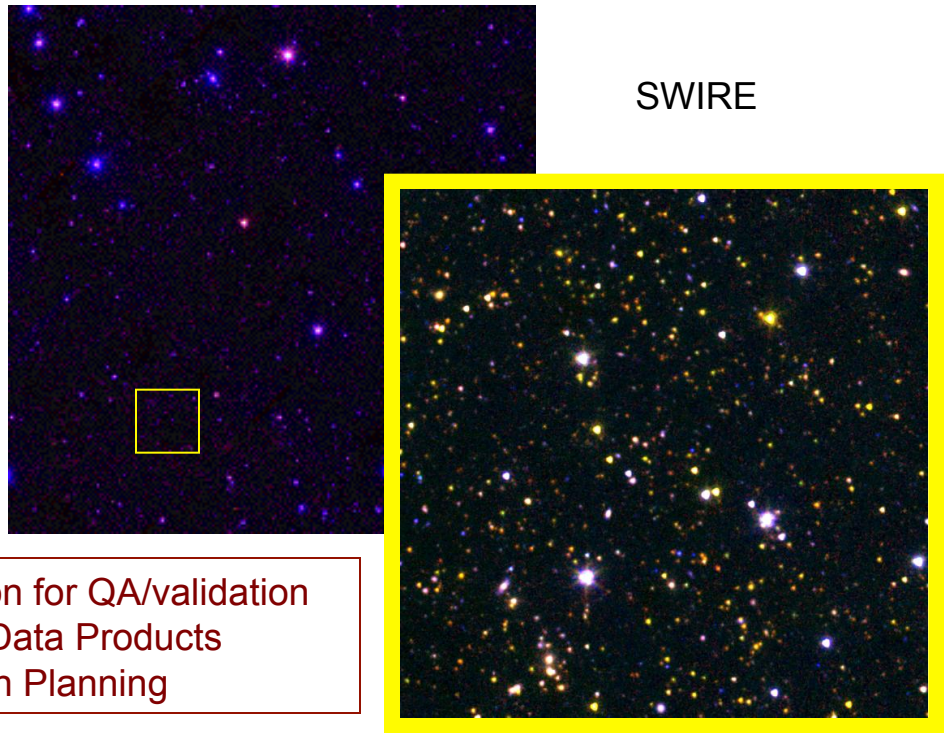


Color composite of co-registered 2MASS and MSX. Each square is 0.5 x 0.5 degrees

GLIMPSE



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

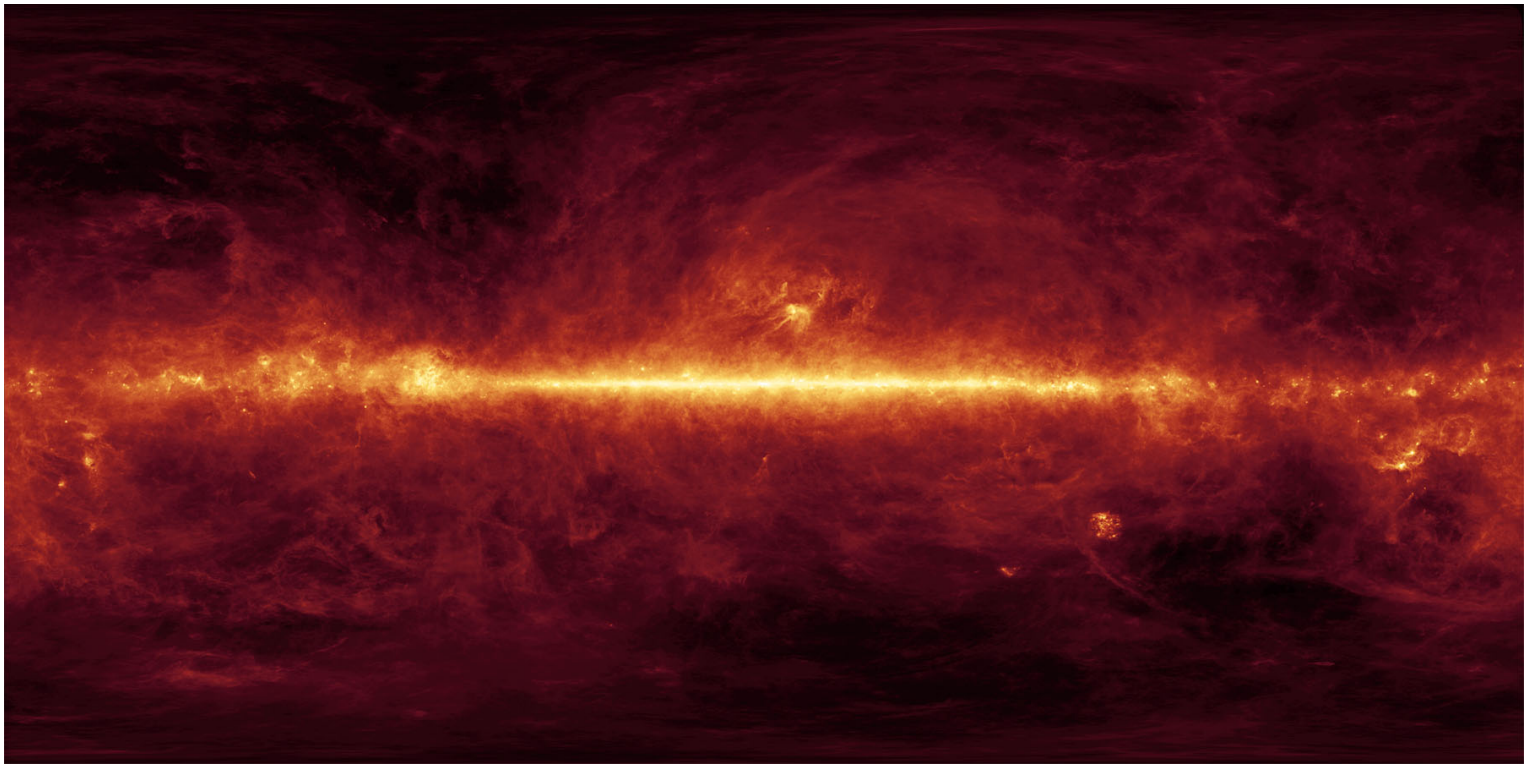


SWIRE

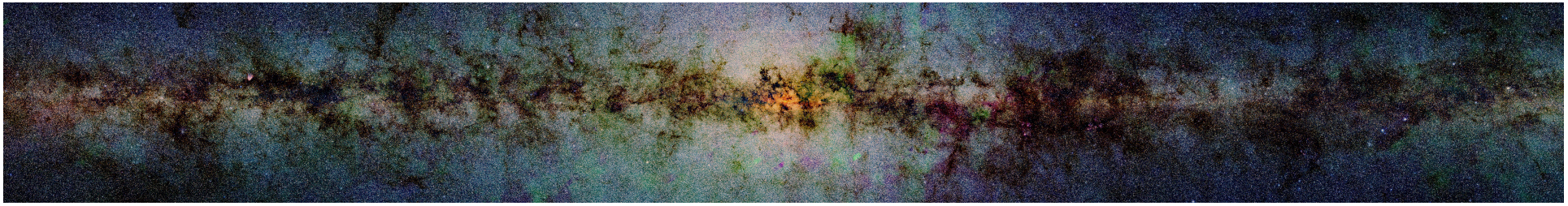
Data Federation for QA/validation
Building Data Products
Mission Planning

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

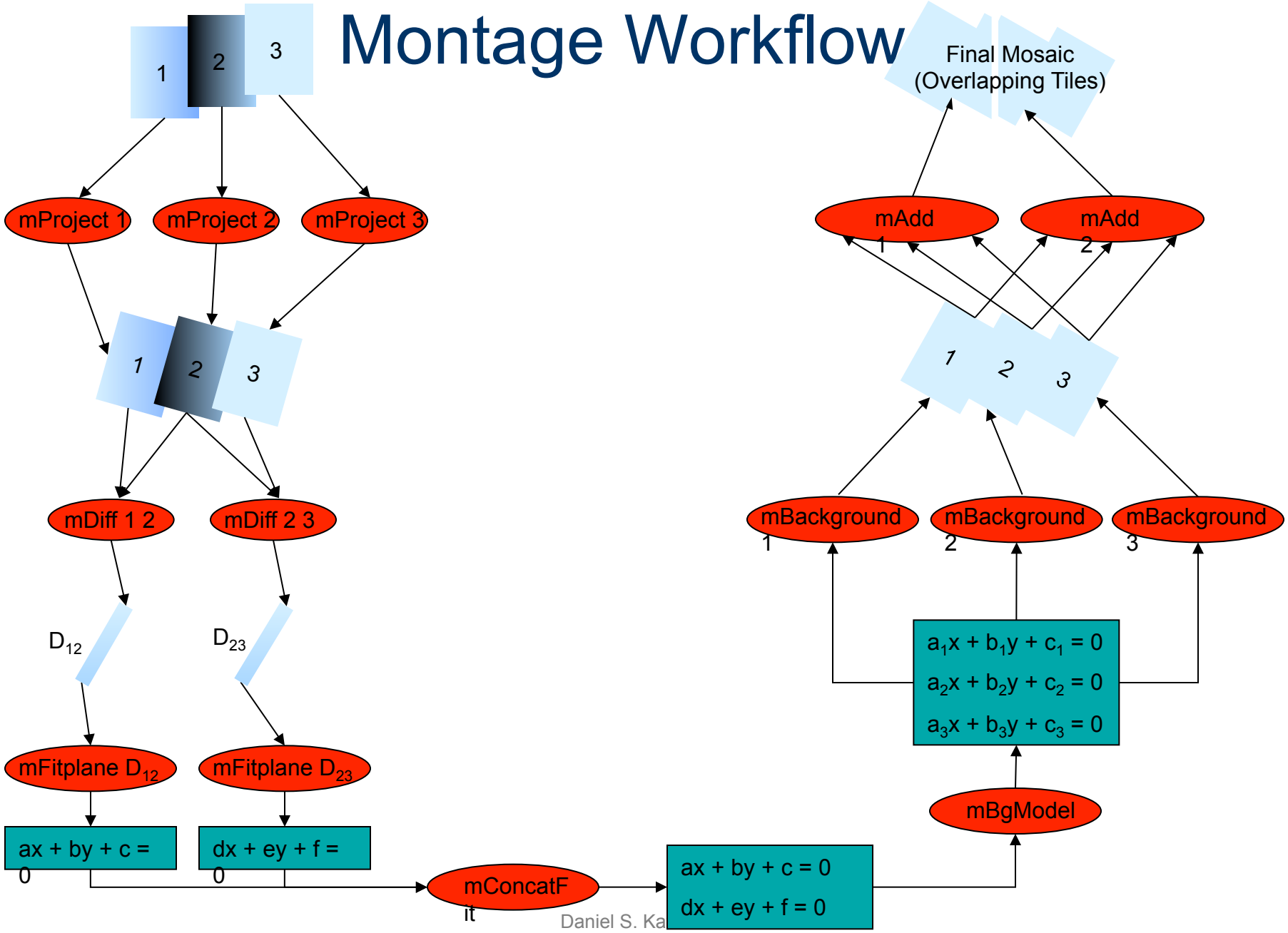


100 μm sky; aggregation of COBE and IRAS maps (Schlegel, Finkbeiner and Davis, 1998) 360° x 180°, CAR projection

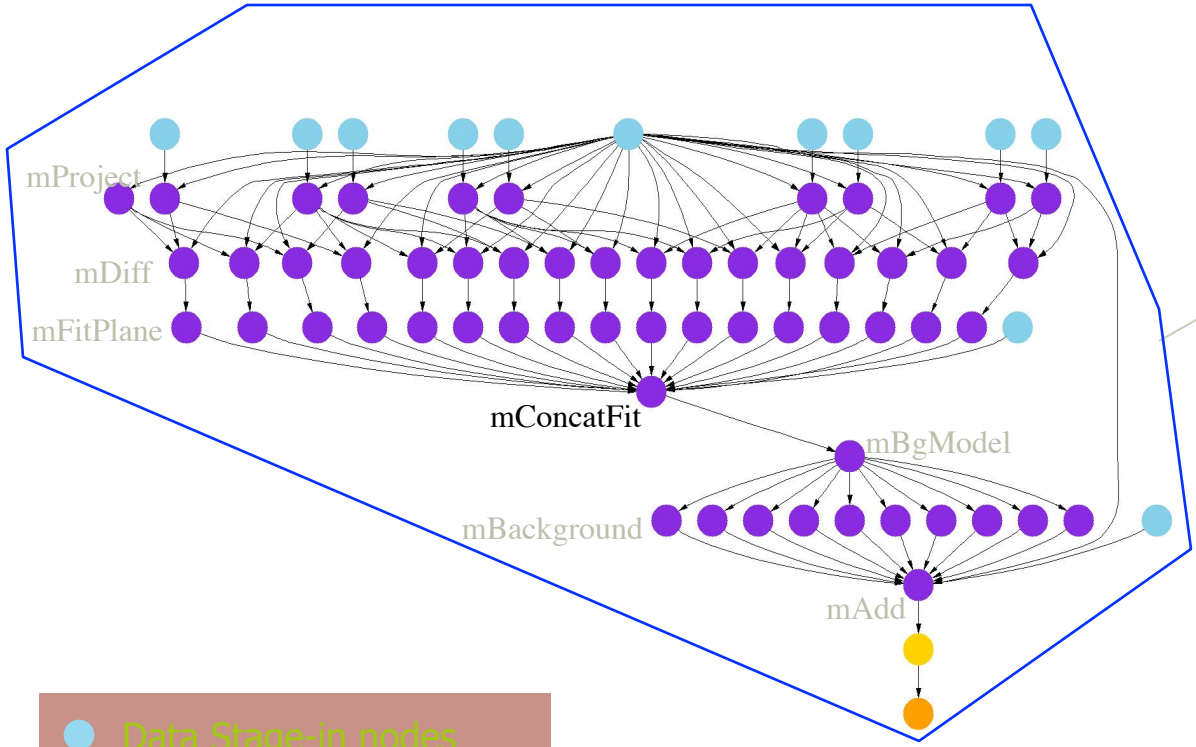


Panoramic view of galactic plane as seen by 2MASS, 44° x 8°, 158,400 x 28,800 pixels; covers 0.8% of sky

# 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

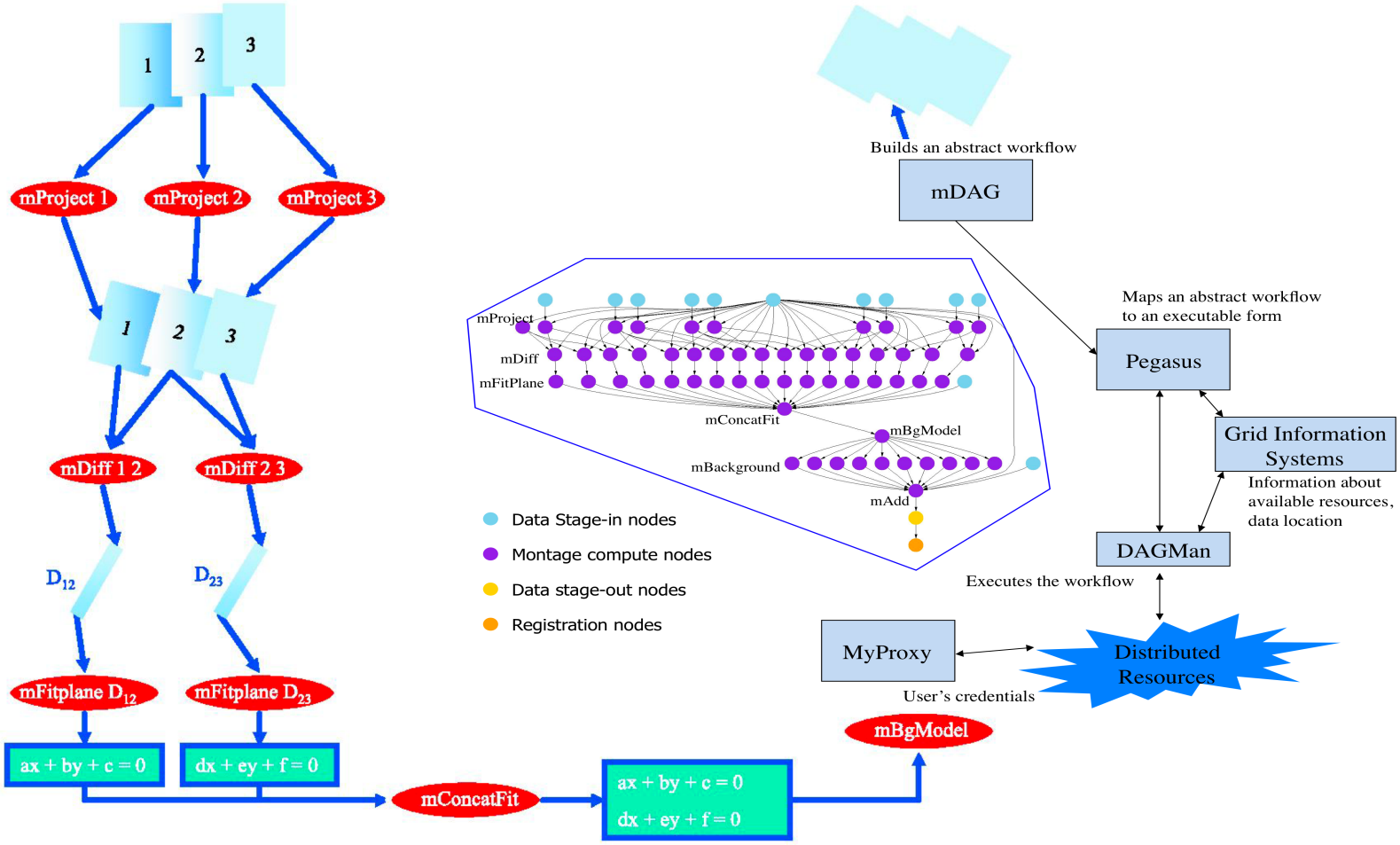
MyProxy

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

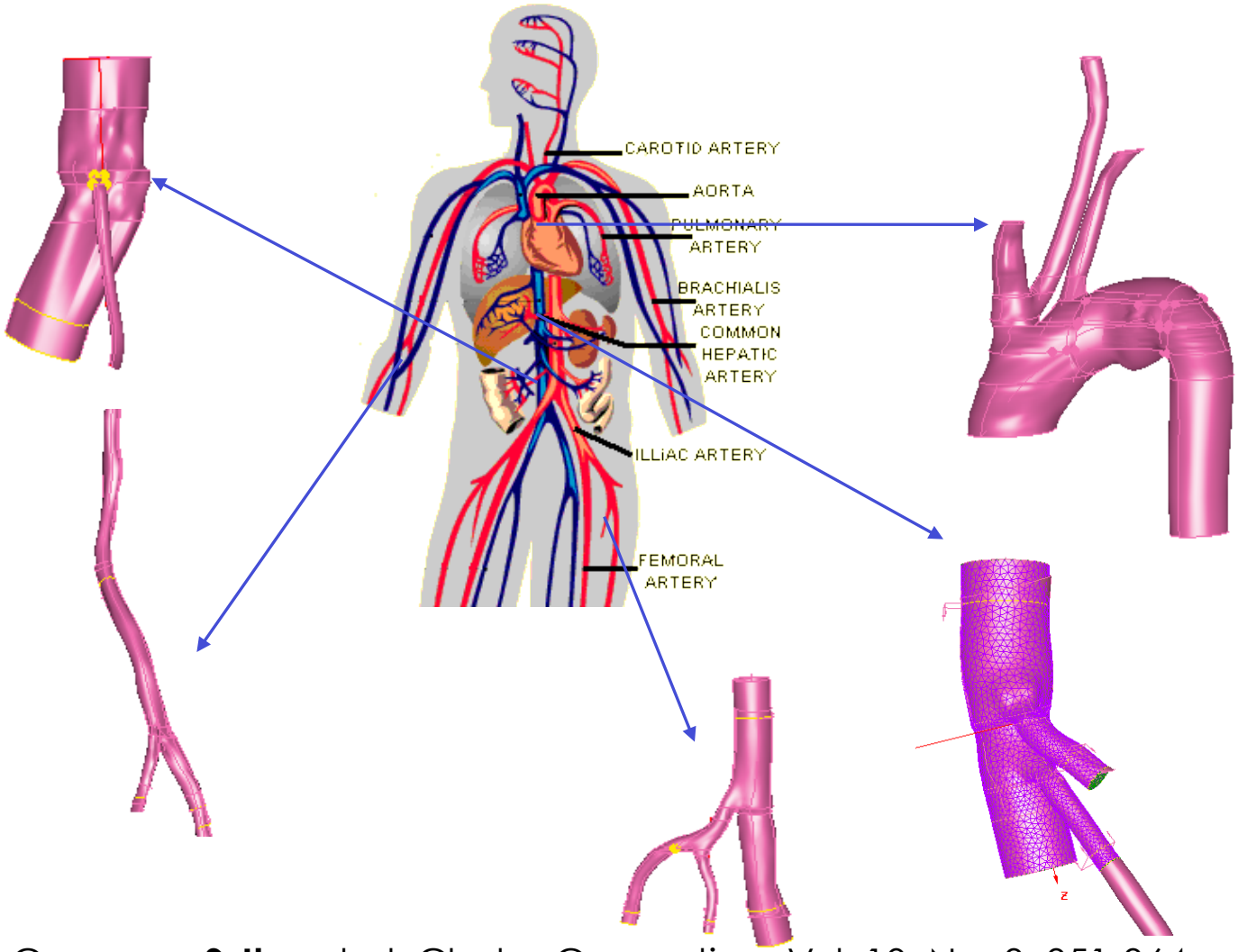
- Why distributed?
  - Scale Processing (over come size limits) above local limits
- How distributed?
  - DAG is created
  - DAG is executed via a DAG-Enactor [DAGMAN]
    - Other approaches exist
- Challenges/Issues/Success?
  - Mapping to right resources – to effectively use different resources, network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

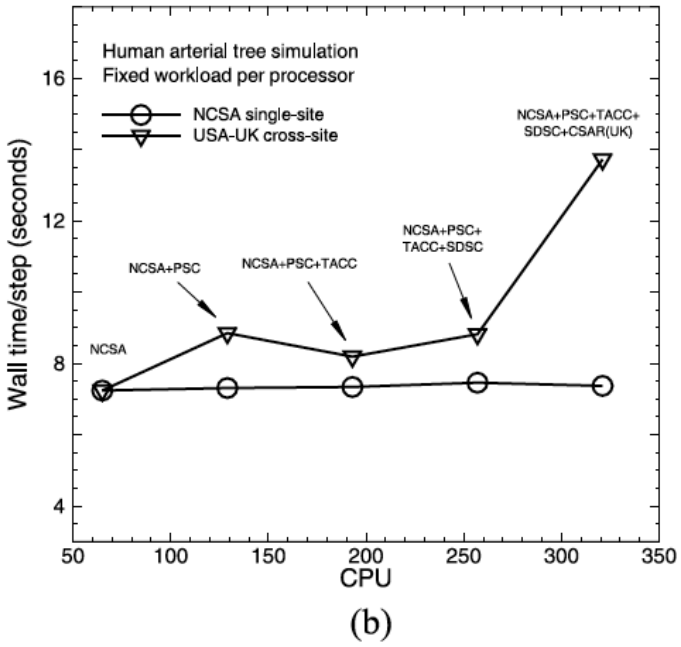



**Fig. 1** USA–UK cross-site performance of arterial tree application: (**a**) wall time/step as a function of the total number of processors for a fixed problem size in cross-site computations involving NCSA-PSC-TACC-SDSC. (**b**) wall time/step as a function of the total number of processors in USA-UK 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!

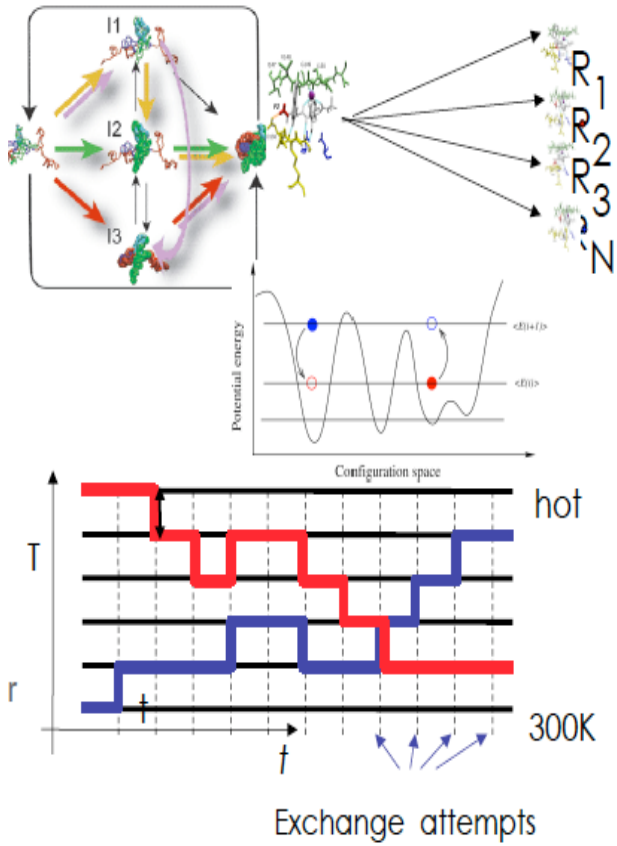

CENTER FOR COMPUTATION
& TECHNOLOGY

# ENSEMBLE-BASED REPLICA-EXCHANGE

CENTER FOR COMPUTATION & TECHNOLOGY

# Ensemble-based & Replica-Exchange Simulations

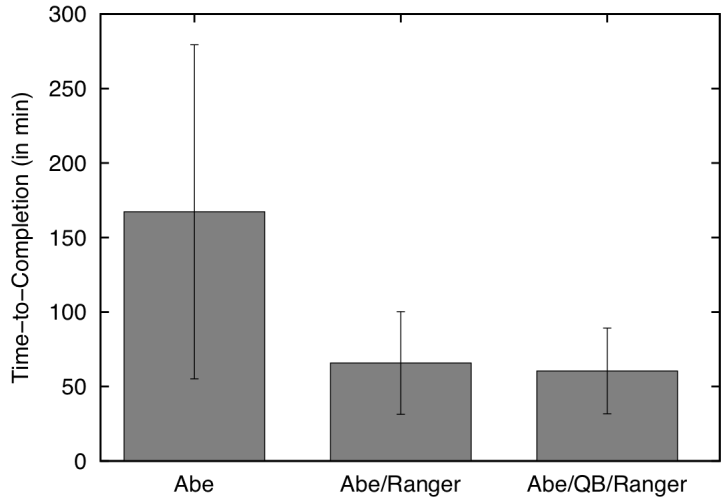
- Ensemble-based:
  - Many uncoupled simulations
  - But not necessarily uncoupled in analysis!
- Replica-Exchange (RE) methods:
  - Represent a class of algorithms that involve a large number of loosely coupled ensembles.
- RE simulations are used to understand a range of physical phenomena
  - Protein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

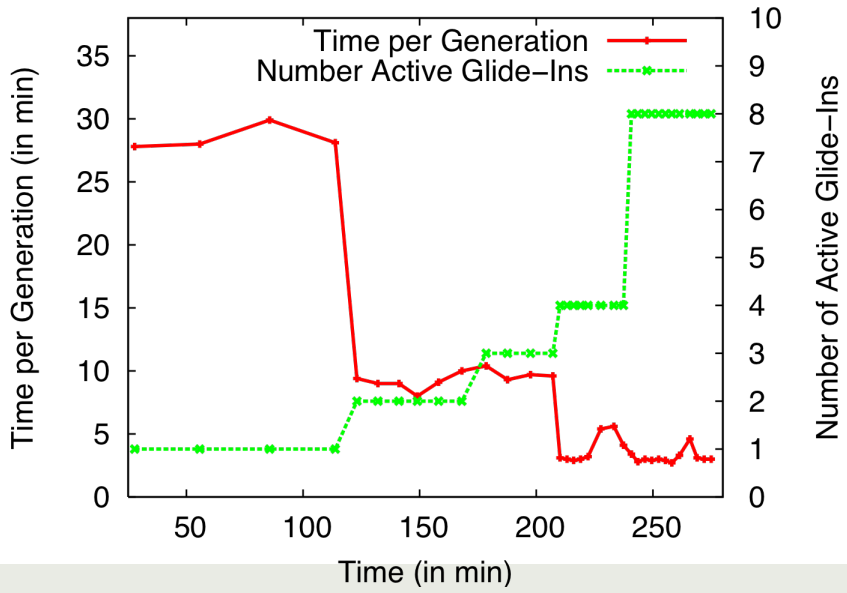


CENTER FOR COMPUTATION & TECHNOLOGY

# 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

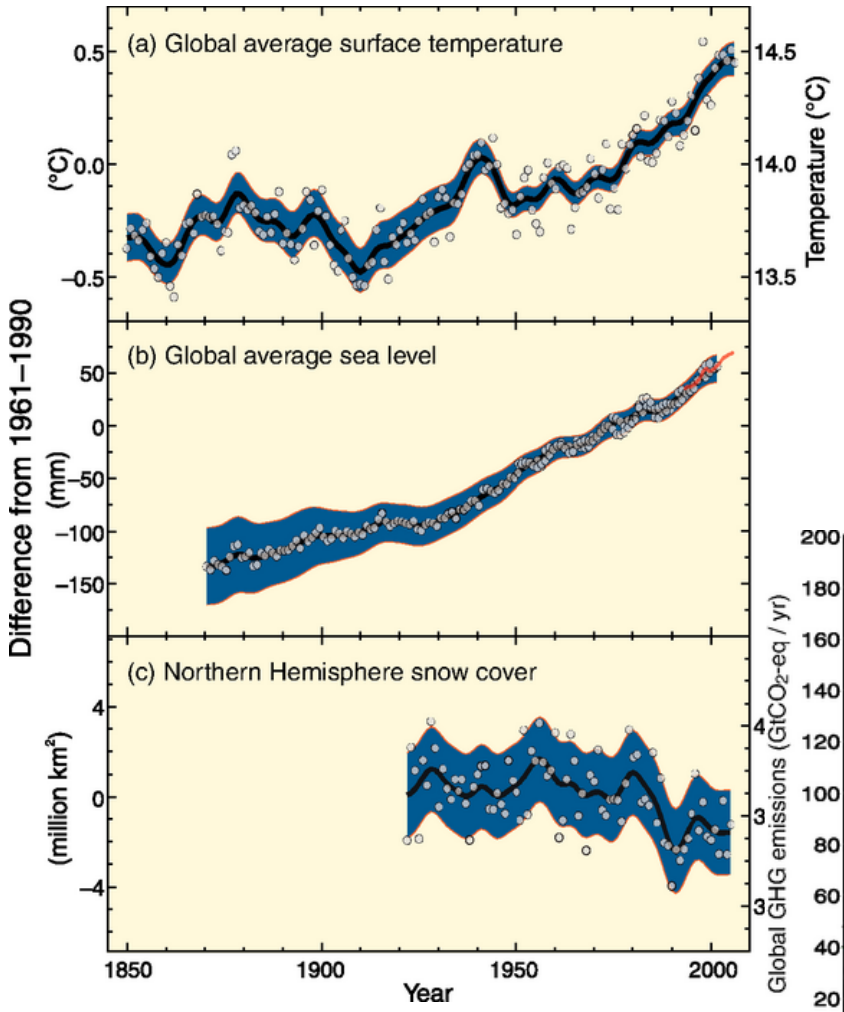

CENTER FOR COMPUTATION
& TECHNOLOGY

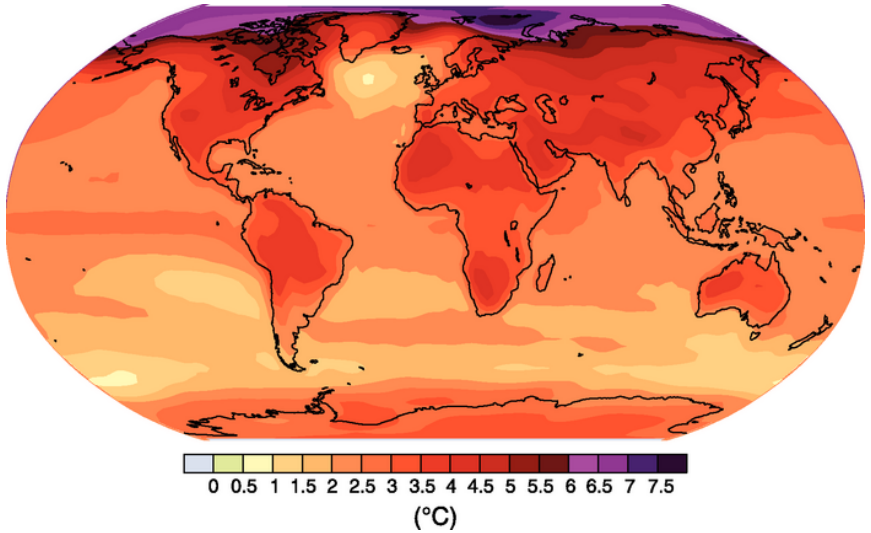
# CLIMATEPREDICTION.NET

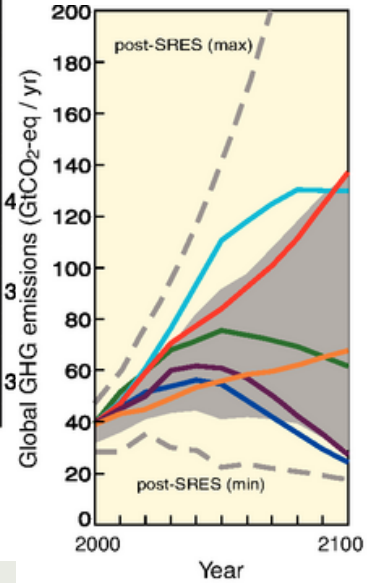
CENTER FOR COMPUTATION & TECHNOLOGY

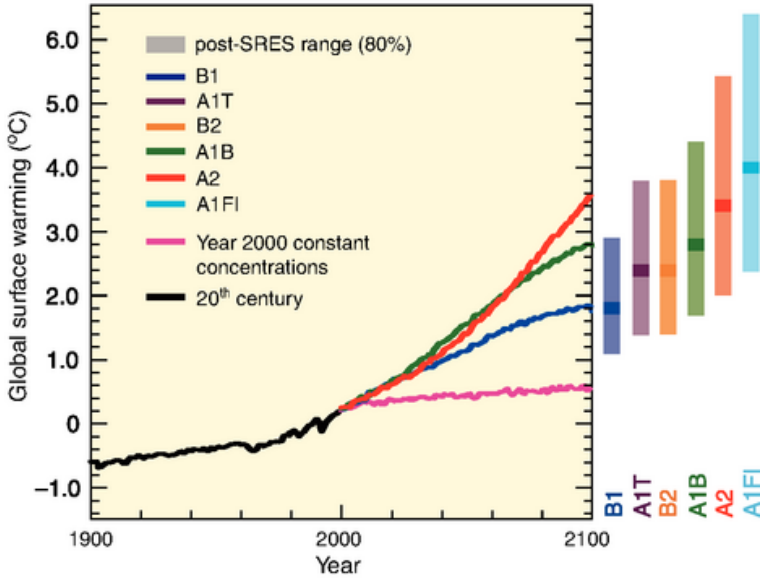
# IPCC AR4:

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



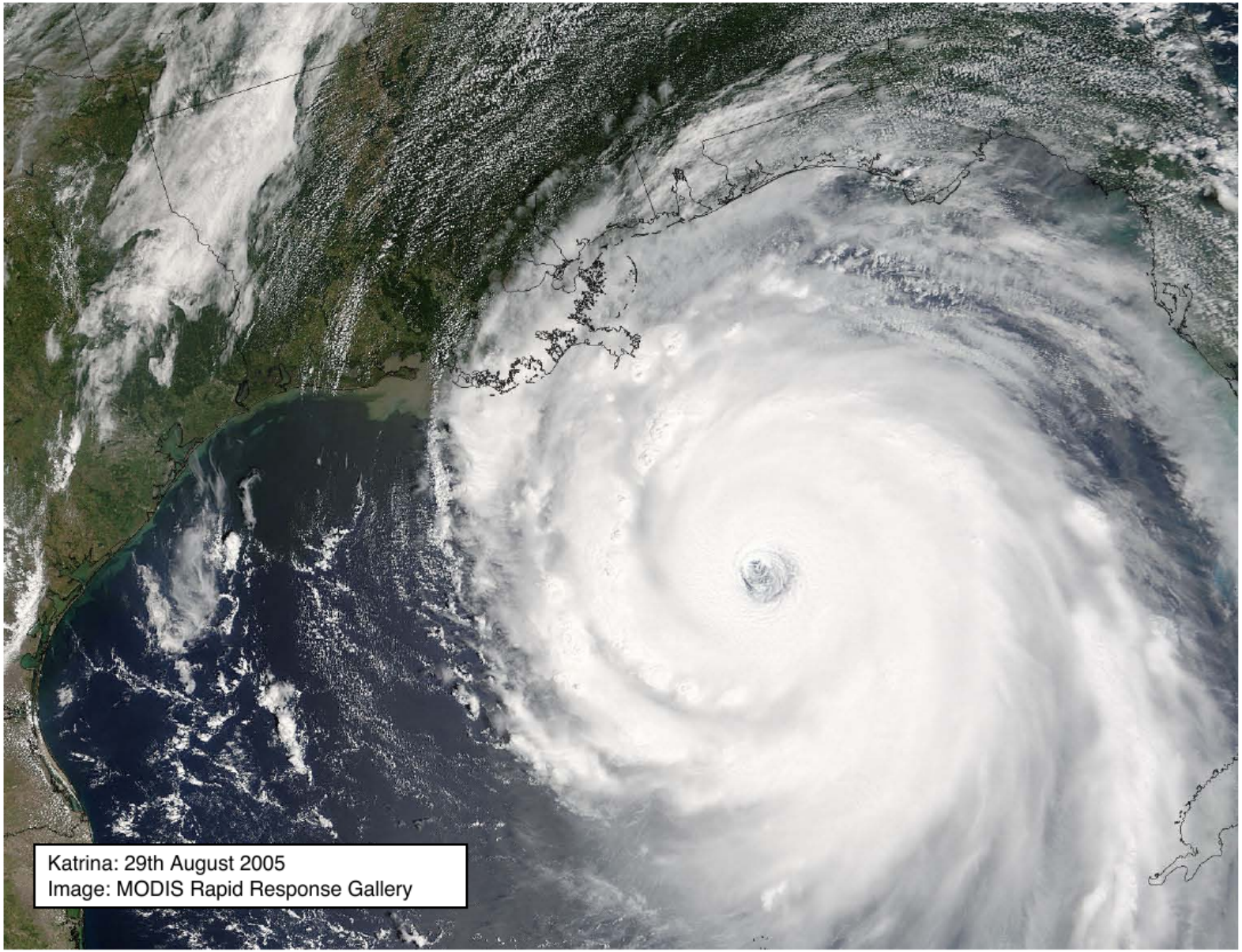
CENTER FOR COMPUTATION & TECHNOLOGY

# Understanding ClimatePrediction.net

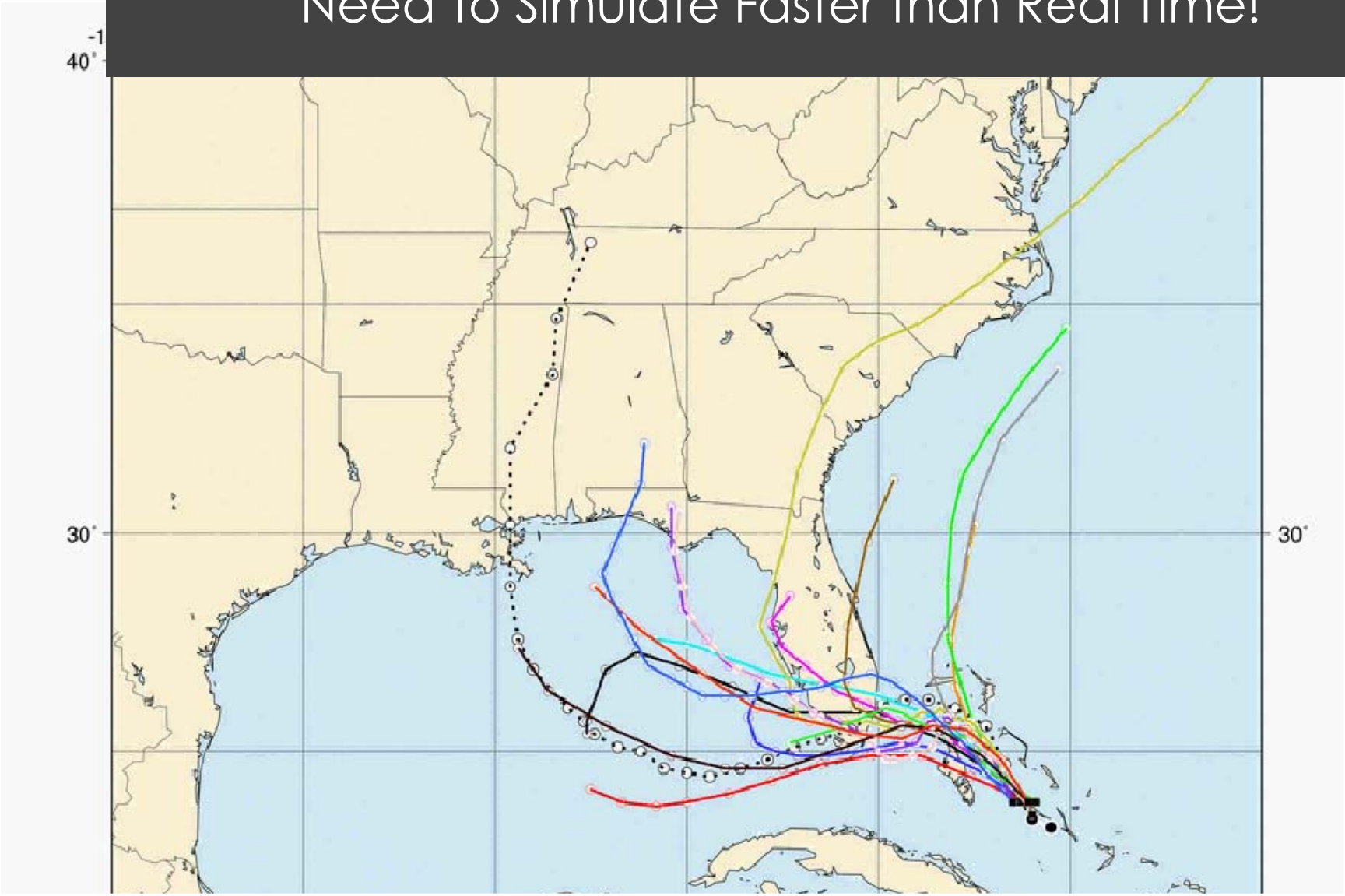
- Why Distributed?
  - Many small indep. Comp. tasks – naturally decomposable
  - Access many more resources without owning
    - Petaflop computing years before Petaflop Computing era!?
- How Distributed?
  - BOINC -- basis for @HOME projects [Volunteer Computing]
  - "Trickles" – job reporting to the 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

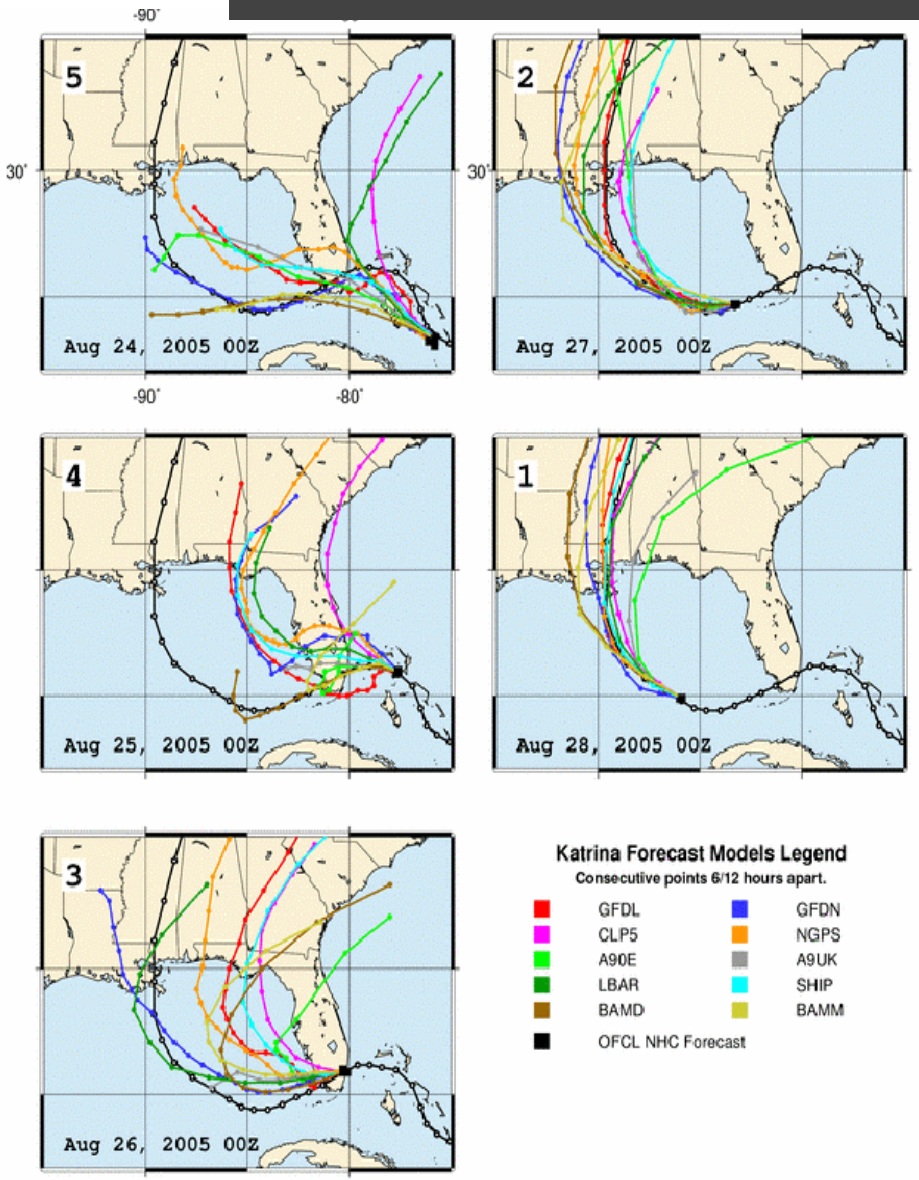
# SCOOP..

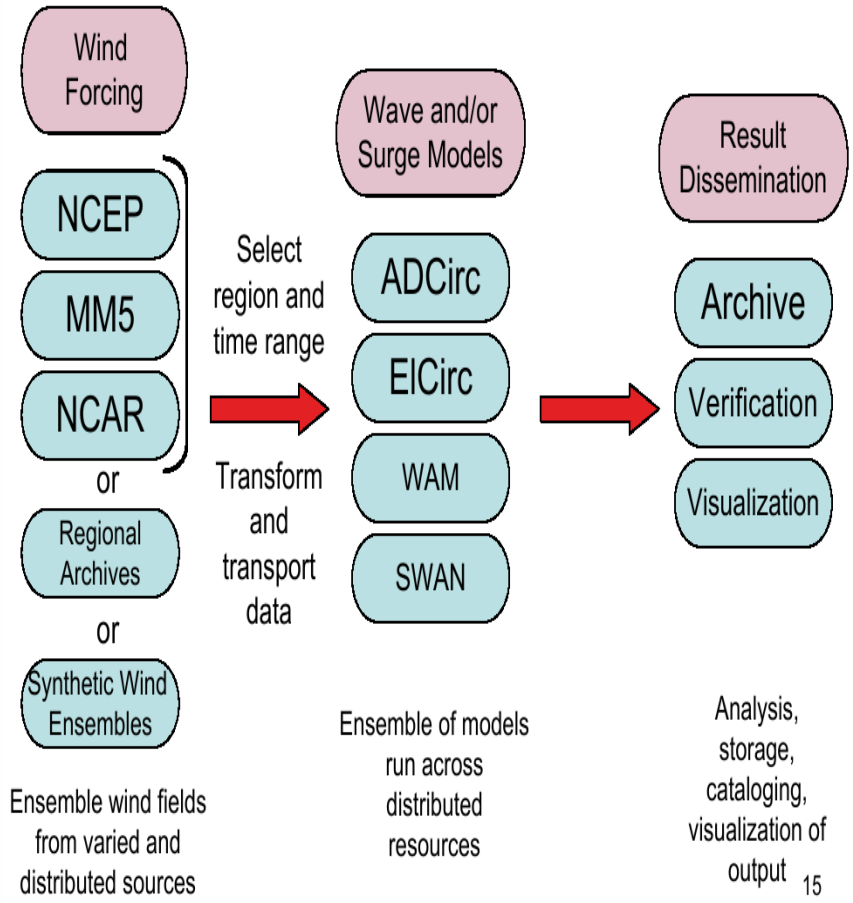

Katrina: 29th August 2005
Image: MODIS Rapid Response Gallery

# Need To Simulate Faster than Real Time!



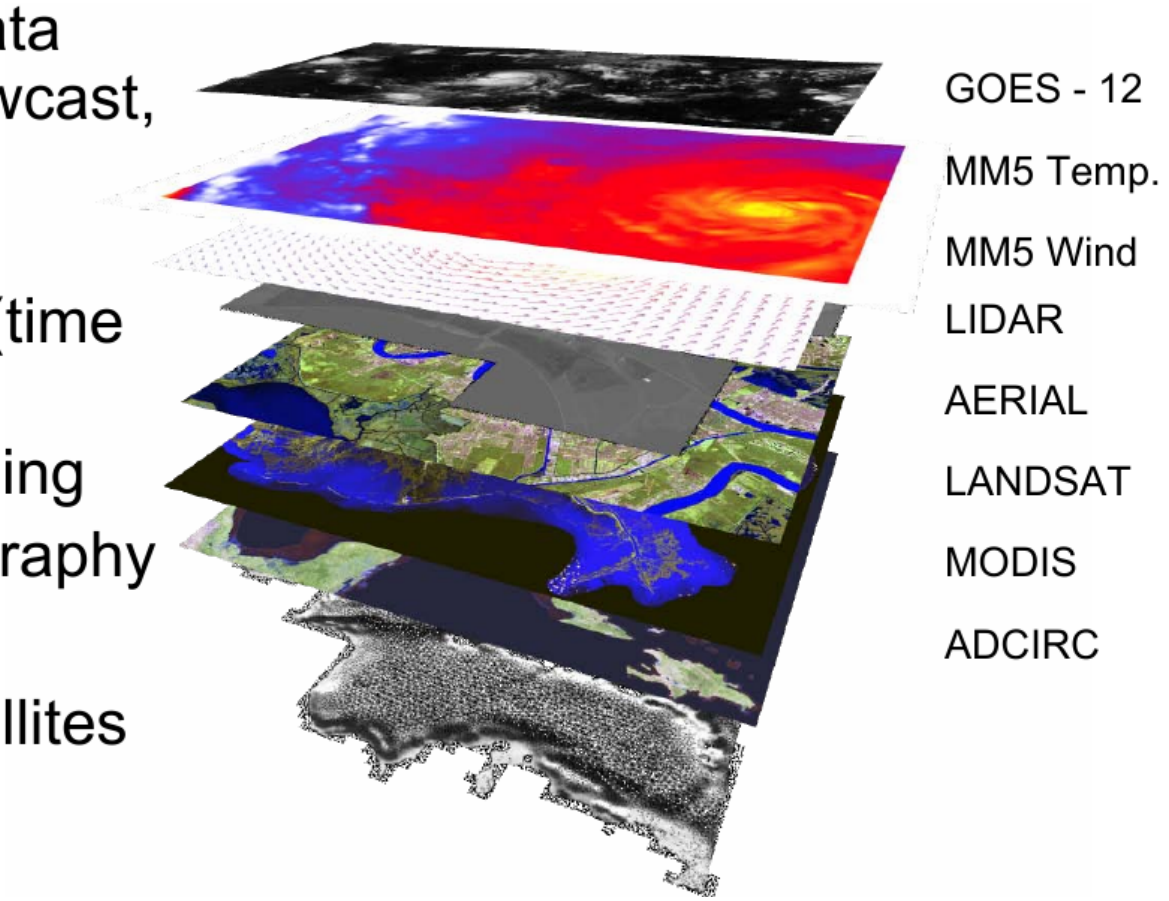
# Heterogenous: Compute Models



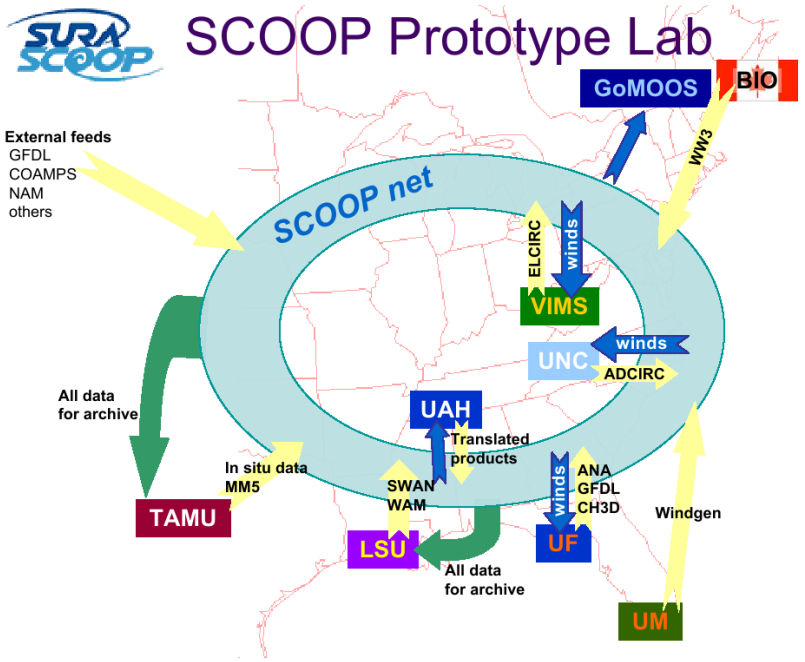



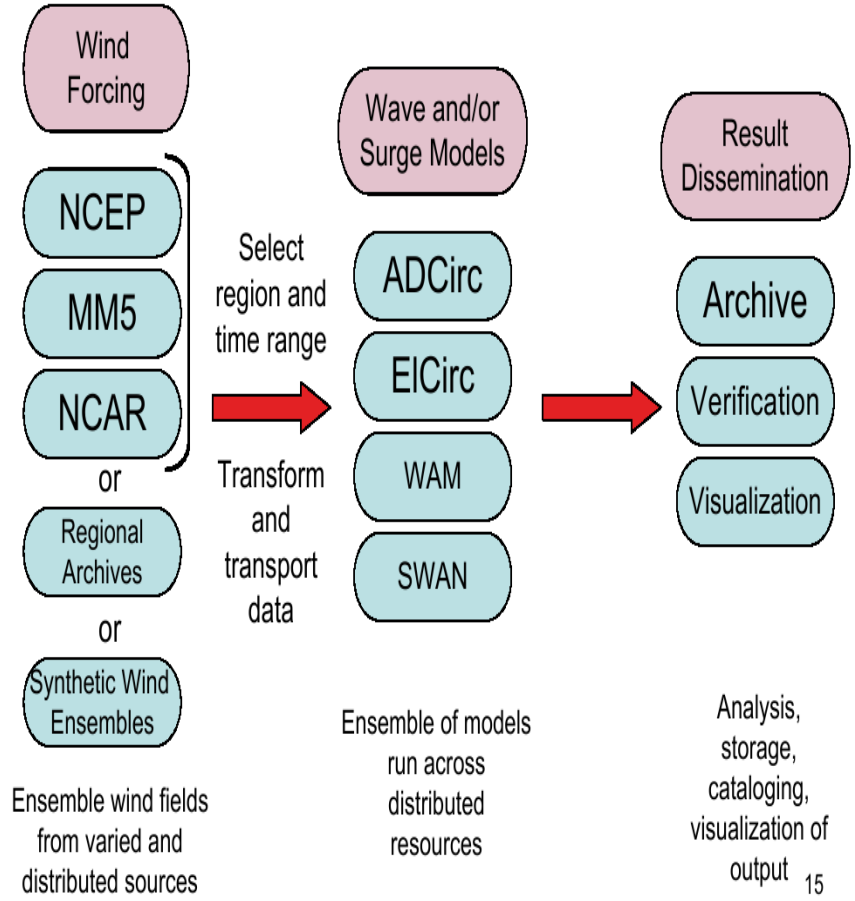
# Many Heterogeneous Components

- 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?

# SAGA: In a nutshell

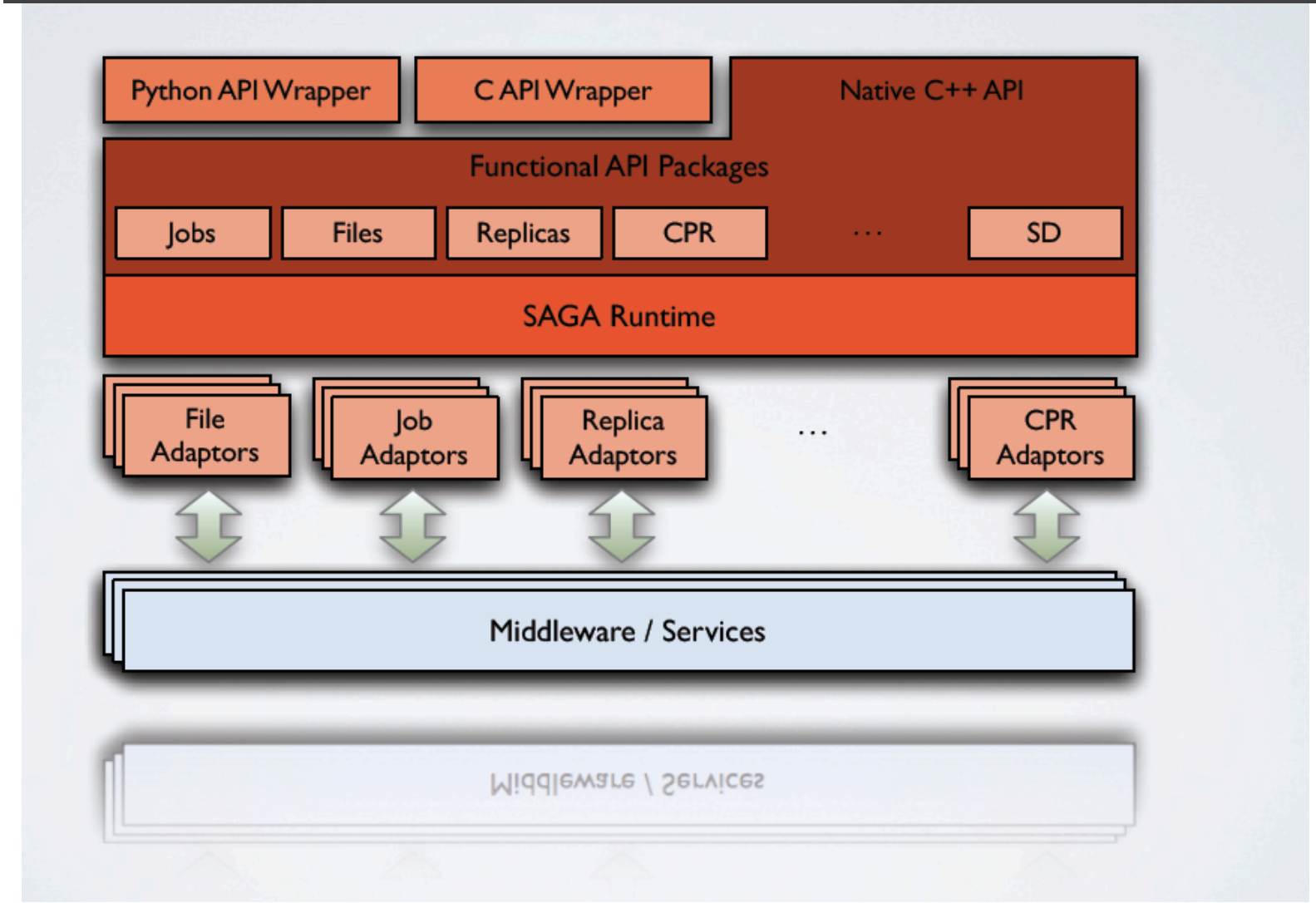
- There exists a lack of Programmatic approaches that:
  - Provide general-purpose, basic &common grid functionality for applications and thus hide underlying complexity, varying semantics..
  - The building blocks upon which to construct "consistent" higher-levels of functionality and abstractions
  - Meets the need for a Broad Spectrum of Application:
    - Simple scripts, Gateways, Smart Applications and Production Grade Tooling, Workflow…

- Simple, integrated, stable, uniform and high-level interface
  - Simple and Stable: 80:20 restricted scope and **Standard**
  - Integrated: Similar semantics & style across
  - Uniform: Same interface for different distributed systems

- SAGA: Provides Application* developers with units required to compose high-level functionality across (distinct) distributed systems

  (*) One Person's Application is another Person's Tool

# SAGA: In a thousand words..




CENTER FOR COMPUTATION
& TECHNOLOGY

# SAGA

A Simple API for Grid Applications

## Installation and Configuration


# Installation

- SAGA is written in C++ and a little bit of Python. To build and install it, you'll need at least:
  - C++ compiler and library
  - make tools
  - Python (optional)
- All SAGA components **require** the Boost C++ libraries (>= 1.33.1). They are available as binary packages on many (Linux) systems. The source installer can be downloaded at http://www.boost.org
- Adaptors require additional libraries / tools to be installed (e.g. Globus libs, PostgreSQL client libs, etc…)

# Installation (Core Components)

- Prerequisites: Boost C++ libraries and the PostgreSQL client libraries if you want to use the default advert and replica adaptors
- Download and unpack the Core Components. Decide where you want to install SAGA (local/global).
  Run **configure** and **make**:

```
$> export SAGA_LOCATION=/install/location/dir/

$> ./configure --prefix=SAGA_LOCATION --with-boost= --with-postgresql=

$> make

$> make install
```

# Installation (Python Bindings)

- Prerequisites: *SAGA Core Components* and Python (>= 2.3 with shared libraries installed)
- Download and unpack the Python Bindings
  Run **configure** and **make**:

```
$> export SAGA_LOCATION=/install/location/dir/

$> ./configure --with-python=

$> make

$> make install
```

# Installation (Globus Adaptors)

- Prerequisites: ***SAGA Core Components*** and the Globus Toolkit (available at http://www.globus.org)
- Download and unpack the Globus Adaptors
  Run **configure** and **make**:

```
$> export SAGA_LOCATION=/install/location/dir/
$> export GLOBUS_LOCATION=/path/to/your/globus/installation

$> ./configure --with-globus-location= --with-globus-flavor=

$> make

$> make install
```

- This works similar for all other SAGA Adaptors

# Installation Recap

- SAGA_LOCATION must point to your Core Components installation
- Different Adaptors may have different configure options. **./configure --help** might help ;-)
- Each adaptor comes with a file called:

**INSTALL**

Read it!

# Configure Environment

- **SAGA_LOCATION** is the only required environment variable. It makes sense to put it e.g. in your .bashrc

```
export SAGA_LOCATION=/install/location/dir/
```

- You will also have to add SAGA to your loader and Python paths if it is not installed in /usr or /usr/local

```
export LD_LIBRARY_PATH=${SAGA_LOCATION}/lib:$LD_LIBRARY_PATH
export DYLD_LIBRARY_PATH=${SAGA_LOCATION}/lib:$DYLD_LIBRARY_PATH  # On MacOS

export PYTHONPATH=${SAGA_LOCATION/lib/pythonX.Y/site-packages/:${PYTHONPATH}
```

# Test The Installation

- Use the SAGA file tool to print the contents of a file

```
$ SAGA_LOCATION/bin/saga-file cat file://localhost/etc/passwd
..
..
..
```

- Import the SAGA module into Python

```
$ python
Python 2.6.1 (r261:67515, Feb 11 2010, 00:51:29)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import saga
>>> saga
<module 'saga' from '/opt/saga-svn/lib/python2.6.1/site-packages/saga/
__init__.pyc'>
```

SAGA
A Simple API for Grid Applications

# Mepisto

- To simplify installation we have Mephisto:
  - http://faust.cct.lsu.edu/trac/mephisto
  - **But read warning at (bottom of):**
    - http://saga.cct.lsu.edu/software/cpp/download
- svn co https://svn.cct.lsu.edu/repos/saga-projects/applications/mephisto/
  - cd trunk
  - perl mephisto.pl install –target-dir=/usr/local
    - On Futuregrid might use $HOME/saga [just a suggestion]