THE UNIVERSITY OF CHICAGO

Scientific Workflow Systems for 21st Century

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Collaborators:
Many many people, see the “More Information” slide at the end

MWGS08: MidWest Grid School 2008
September 17th, 2008

Argonne NATIONAL LABORATORY

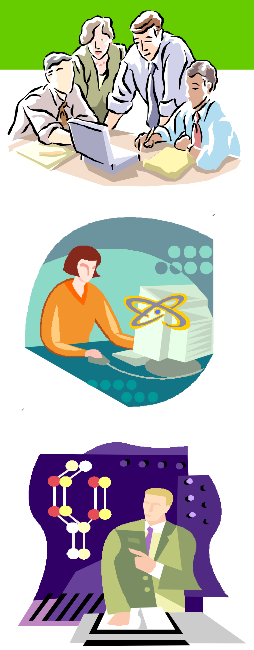
Workflow Systems

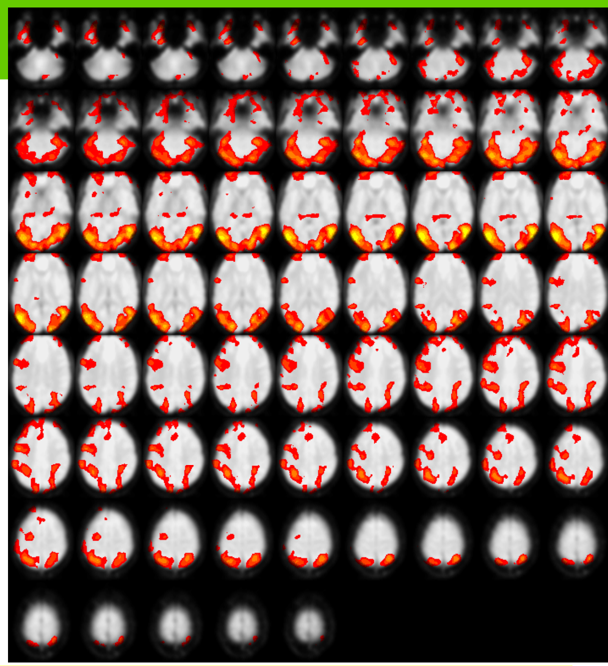
Describes computation components, ports and channels

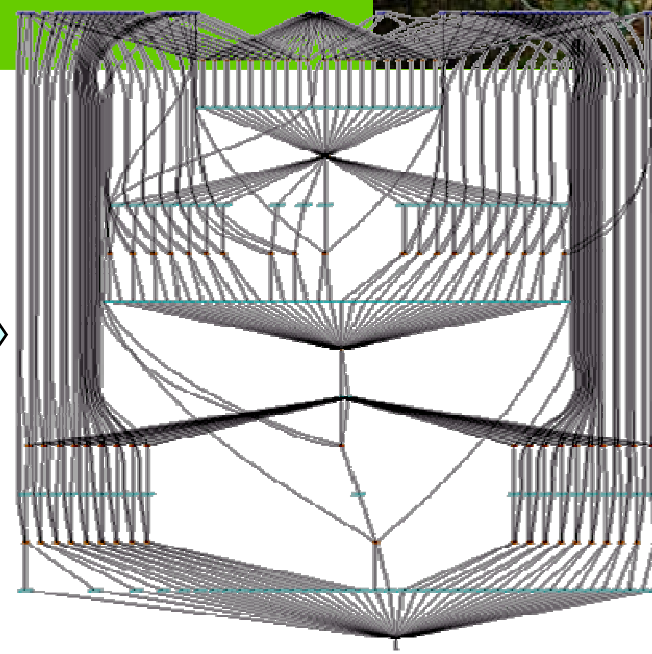
Describes data and event flow

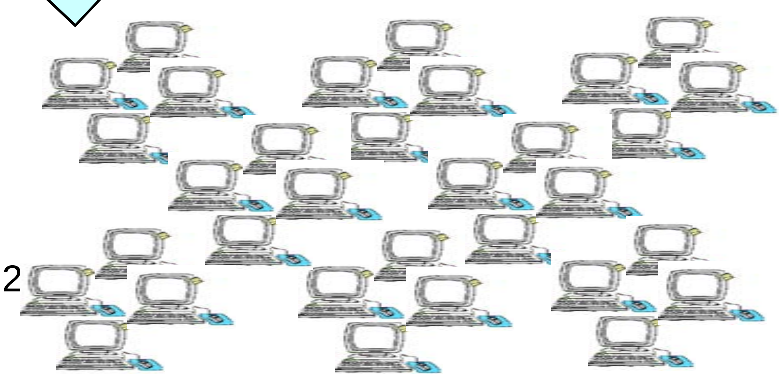
Coordinate the execution of the components

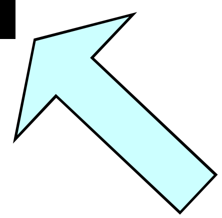
Functional MRI (fMRI)



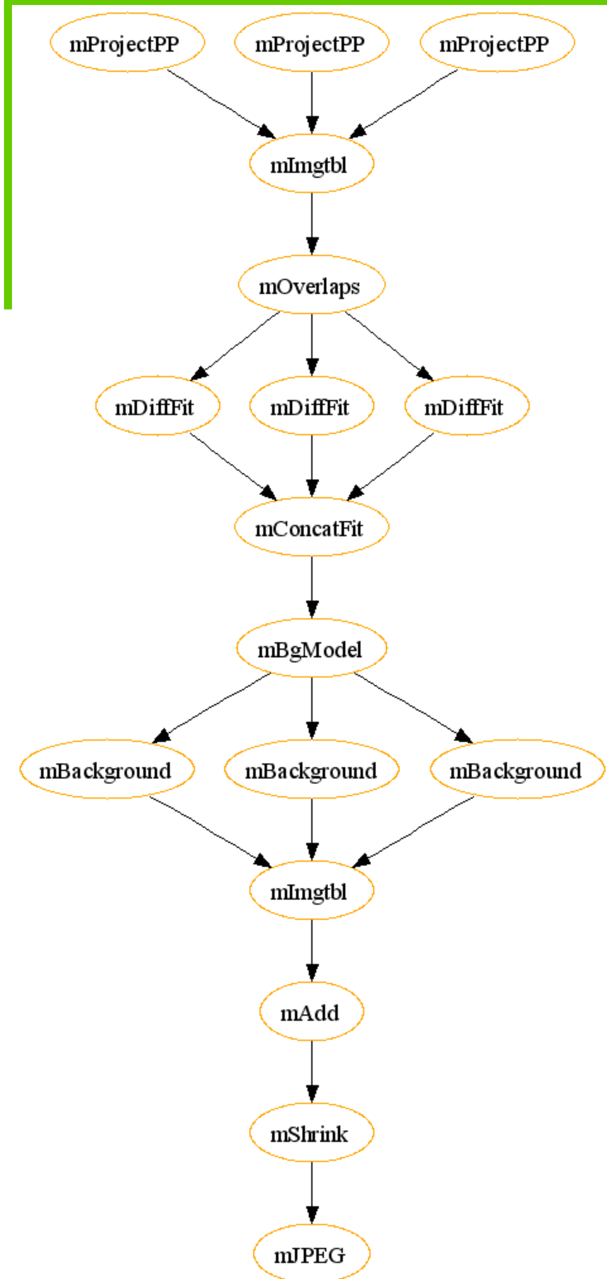








- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies

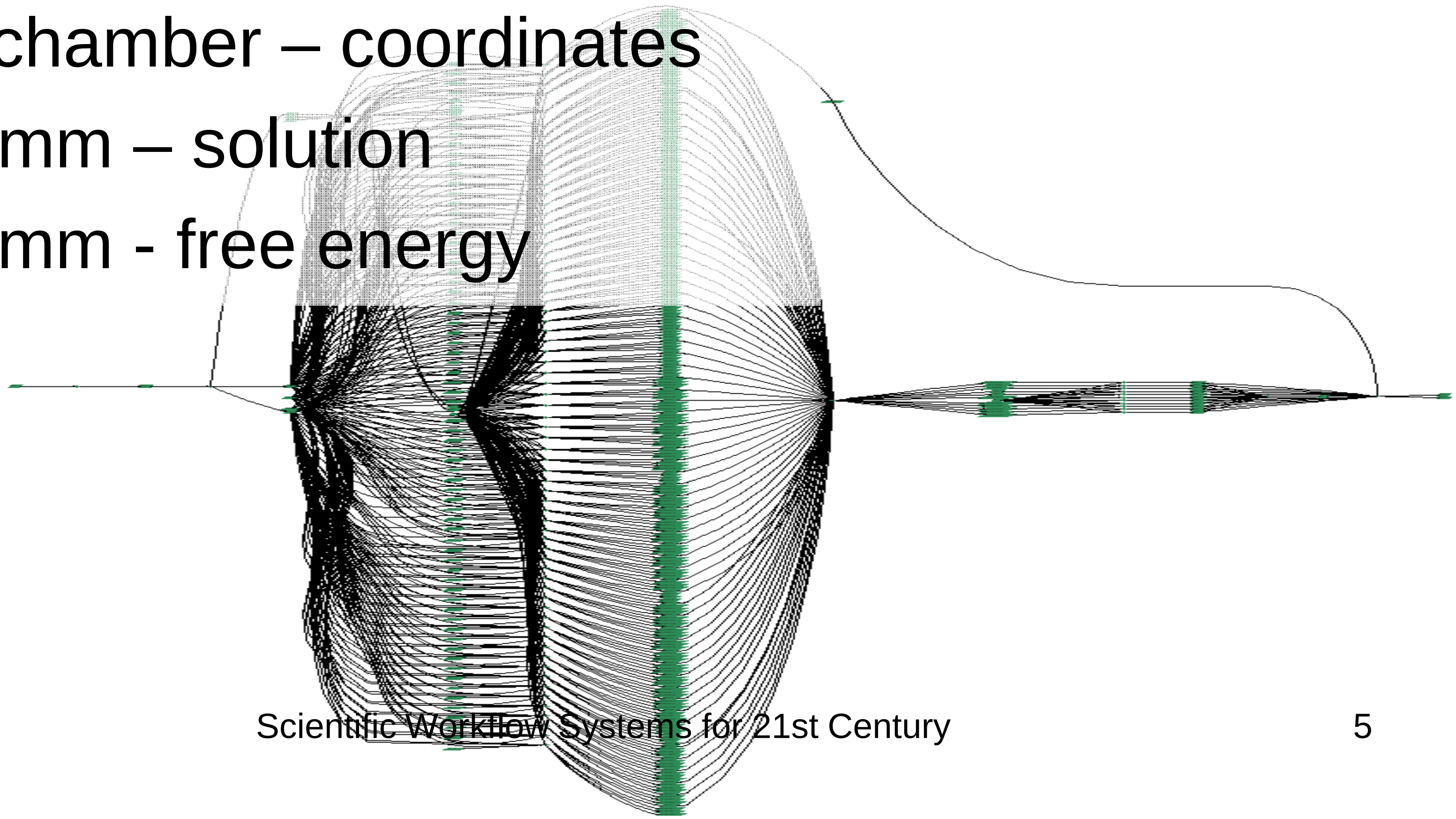


Montage

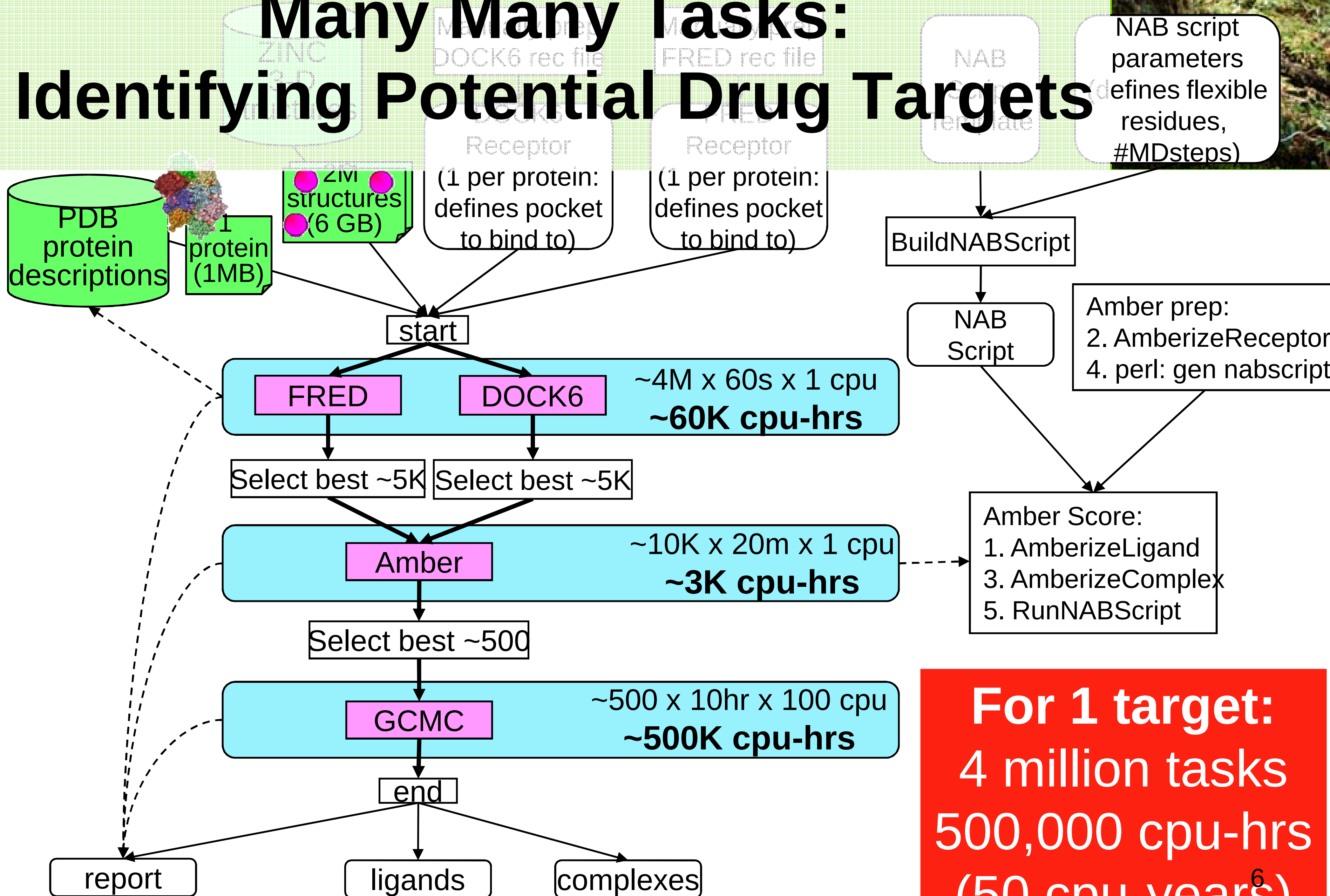
B. Berriman, J. Good (Caltech)
J. Jacob, D. Katz (JPL)

Molecular Dynamics

- Determination of free energies in aqueous solution
  - Antechamber – coordinates
  - Charmm – solution
  - Charmm - free energy



Many Many Tasks:
Identifying Potential Drug Targets



For 1 target:
4 million tasks
500,000 cpu-hrs
(50 cpu-years)

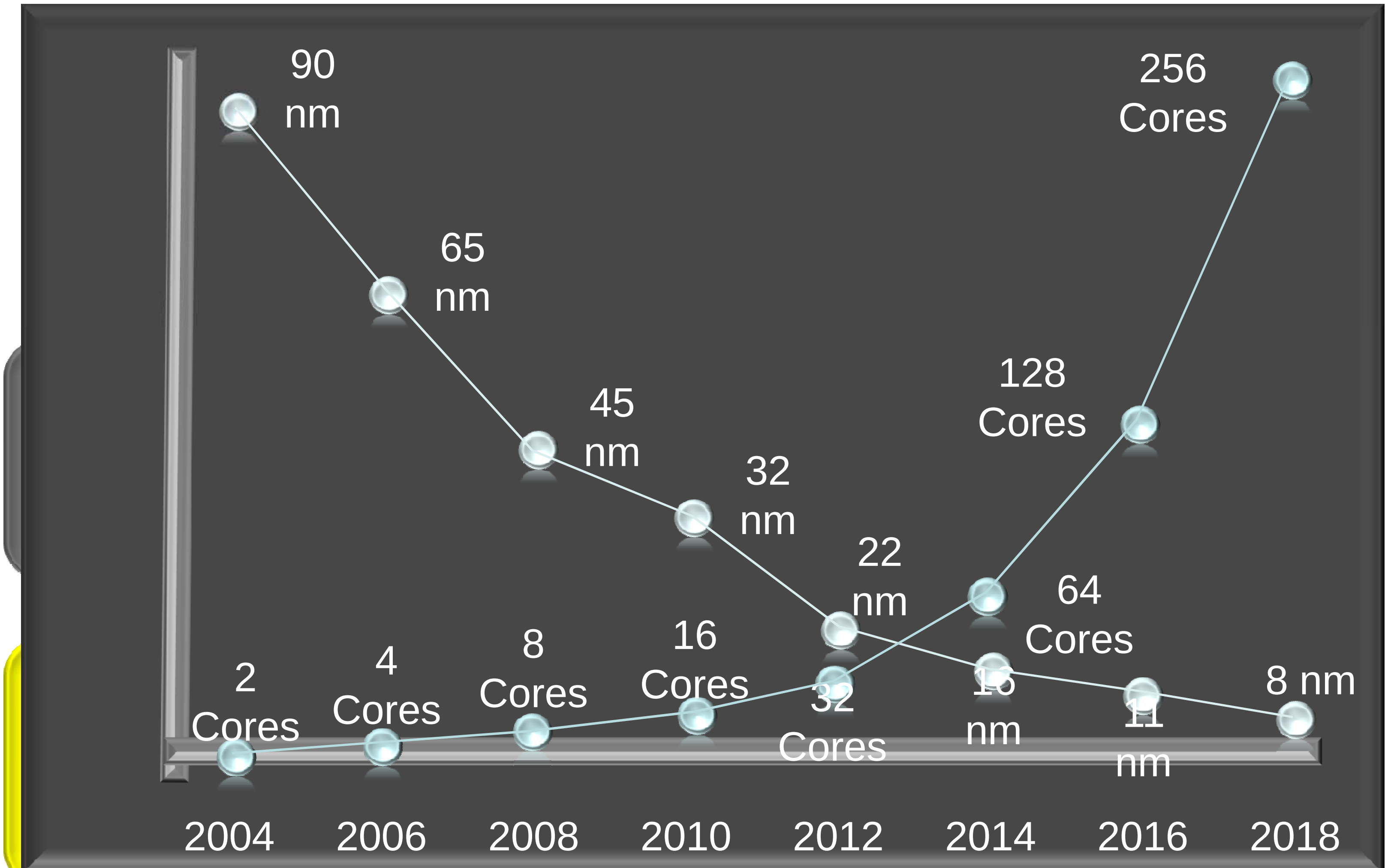
Characterizing Scientific Workflows

- Inherit *workflow system* definition +…
- Describe complex scientific procedures
- Automate data derivation processes
- High performance computing (HPC) to improve throughput and performance
- Provenance management and query

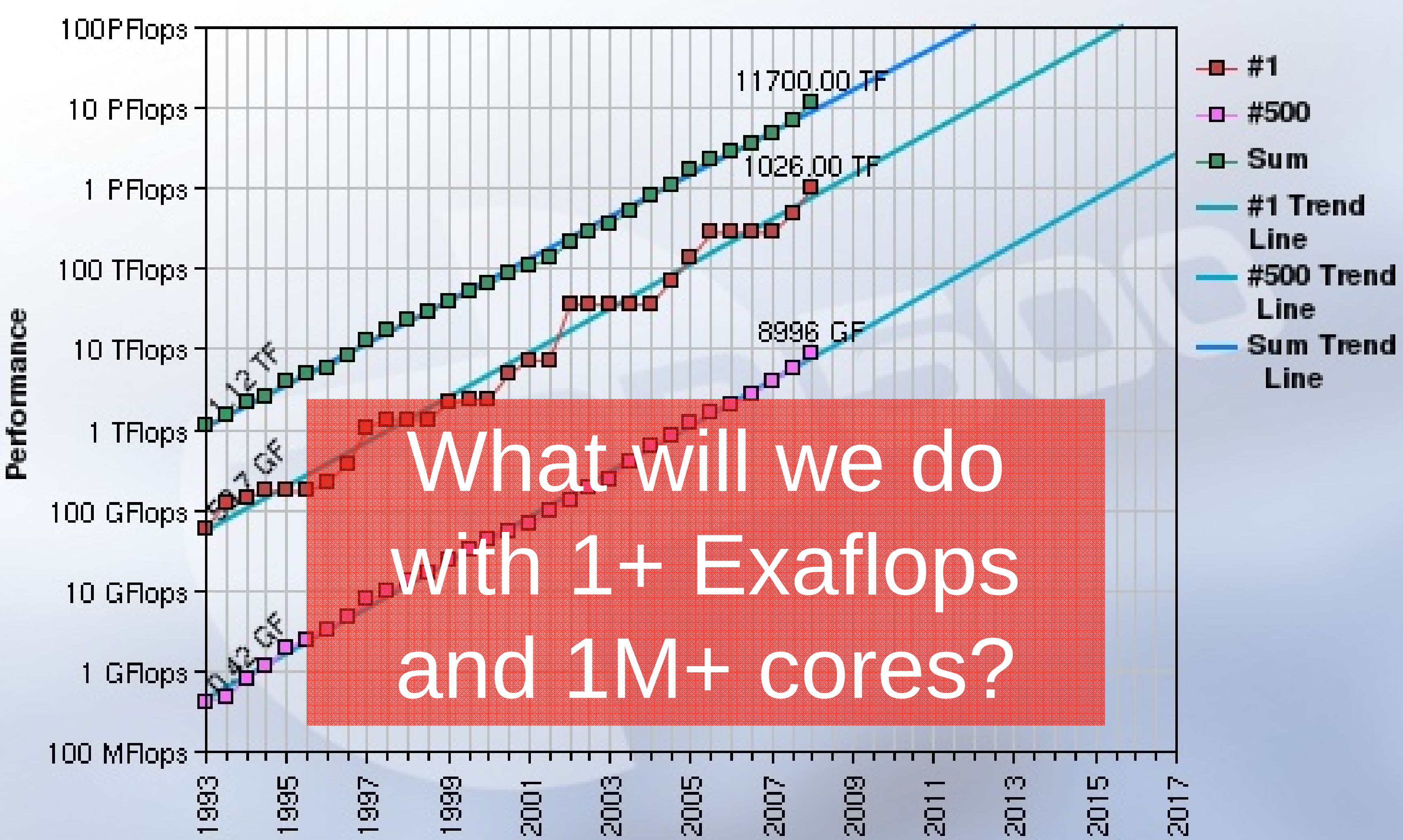
Characterizing Scientific Applications

- Increasing in scale and complexity
  - Wide inherent parallelism
  - Multiple successive stages
  - Wide range of number of tasks
    - thousands to billions
- Potentially compute intensive
  - Widely varying task execution times
    - 100s of ms to 10s of hours
- Potentially data intensive
  - I/O operation rates and aggregate I/O rates
  - Meta-data creation and modification
  - Significant data re-use
    - Produced data is consumed by later stages

Many-Core Growth Rates

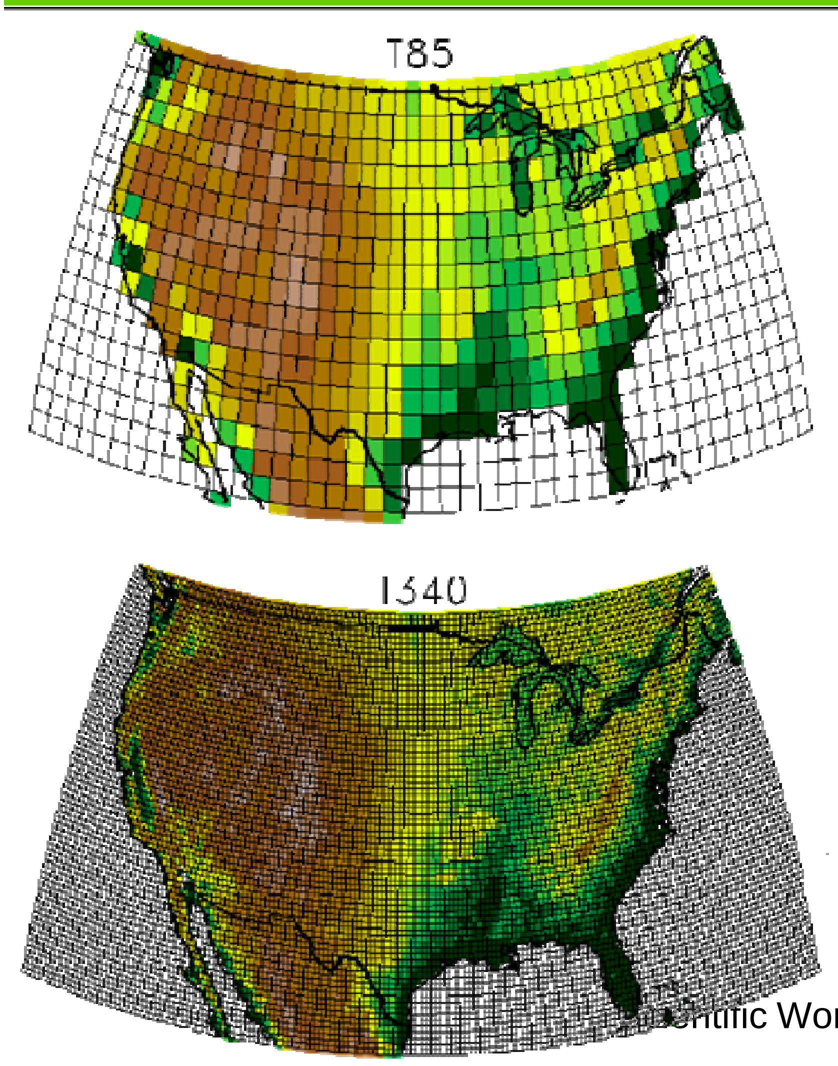


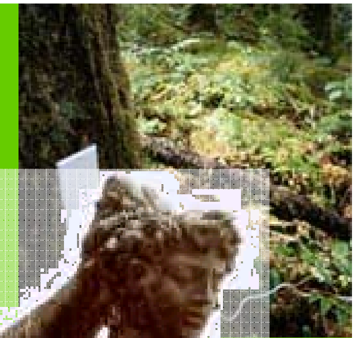
Projected Performance Development



What will we do with 1+ Exaflops and 1M+ cores?

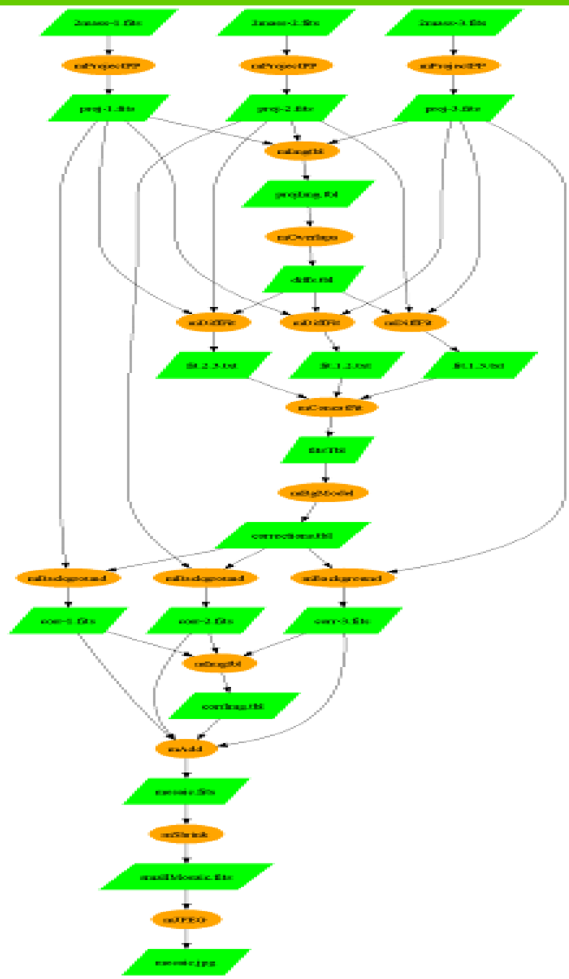
1) Tackle Bigger and Bigger Problems





Computational Scientist as **Hero**

2) Tackle Increasingly Complex Problems



Computational Scientist
as
Logistics Officer

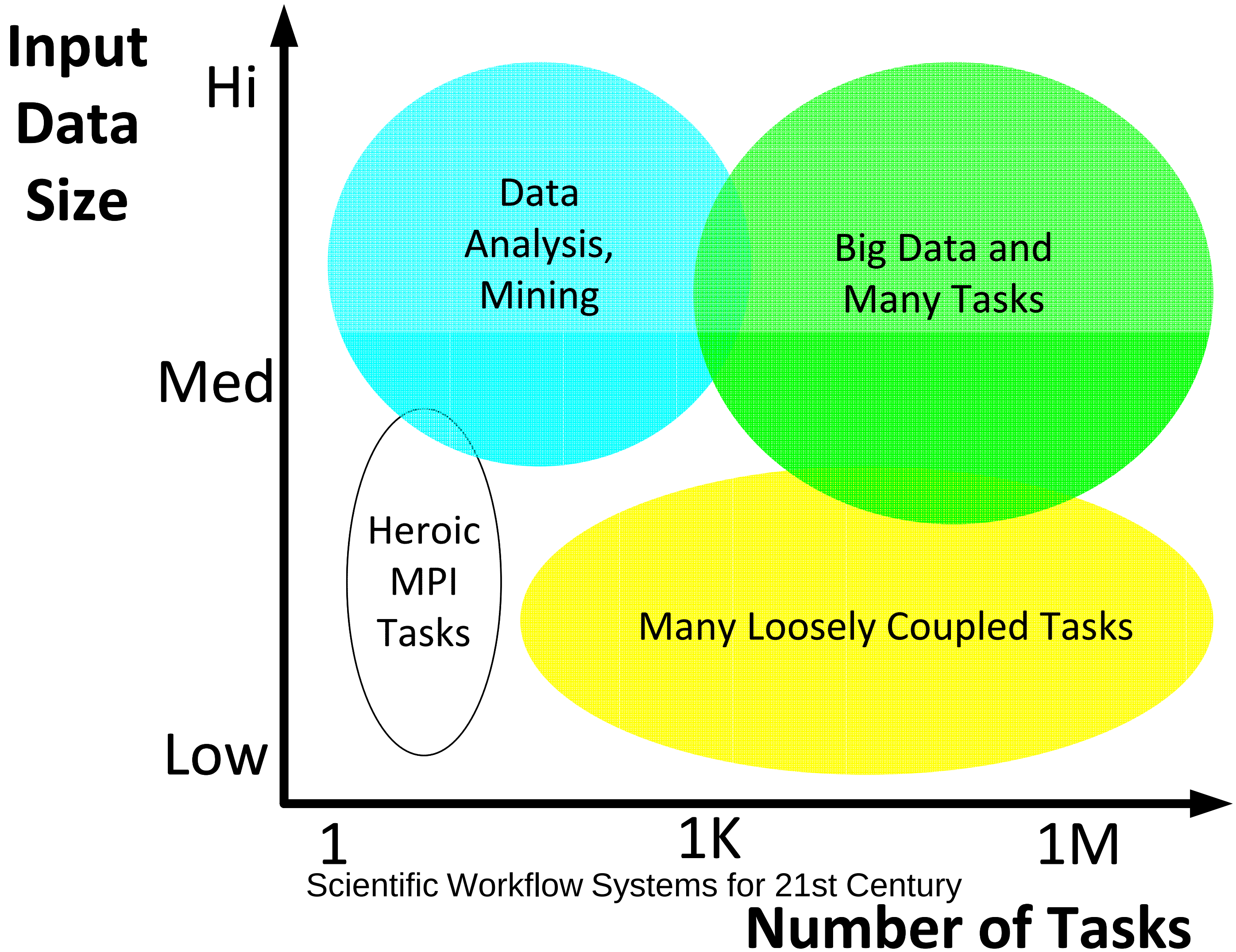
“More Complex Problems”

- Ensemble runs to quantify **climate model uncertainty**
- Identify **potential drug targets** by screening a database of ligand structures against target proteins
- Study **economic model sensitivity** to parameters
- Analyze **turbulence dataset** from many perspectives
- Perform **numerical optimization** to determine optimal resource assignment in energy problems
- Mine collection of data from **advanced light sources**
- Construct databases of computed properties of **chemical compounds**
- Analyze data from the **Large Hadron Collider**
- Analyze **log data** from 100,000-node parallel computations

Programming Model Issues

- Massive **task parallelism**
- Massive **data parallelism**
- Integrating **black box applications**
- Complex **task dependencies** (task graphs)
- **Failure**, and other execution management issues
- **Data management**: input, intermediate, output
- **Dynamic computations** (task graphs)
- **Dynamic data access** to large, diverse datasets
- **Long-running** computations
- Documenting **provenance** of data products

Problem Types



An Incomplete and Simplistic View of Programming Models and Tools

Single task, modest data
MPI, etc…

Many Tasks
DAGMan+Pegasus
Karajan+Swift+Falkon

Much Data
MapReduce/Hadoop
Dryad

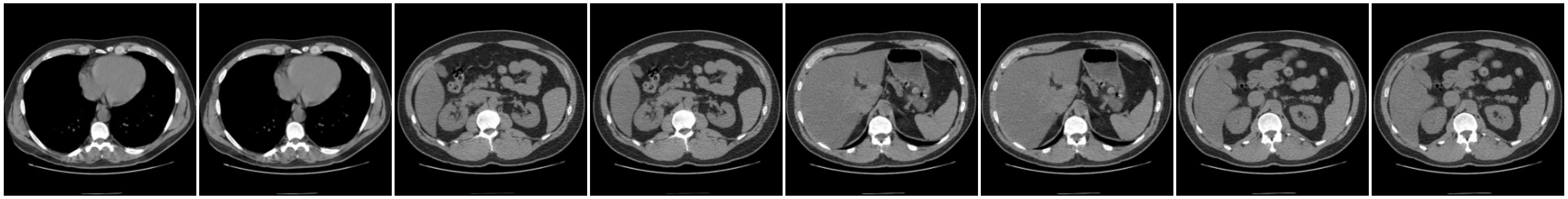
Complex Tasks, Much Data
Dryad, Pig, Sawzall
Swift+Falkon (using data diffusion)

Major Challenges to Large Scale Scientific Computation

- Managing heterogeneous scientific data
  - Idiosyncratic layouts & formats (file sys, db, spreadsheet, XML, etc…)
- Describing complex science problems
- Coordinating distributed diverse computation procedures
  - Executables, scripts, Web services
- Long wait queue times
- Scheduling & executing numerous tasks reliably and efficiently
  - Large quantity of data (Petabytes/year)
  - Large number of parallel/dependent tasks (10^3~10^6 tasks)
- Organizing, archiving and tracking
  - Datasets, procedures, workflows, provenance
- Supporting data-intensive applications

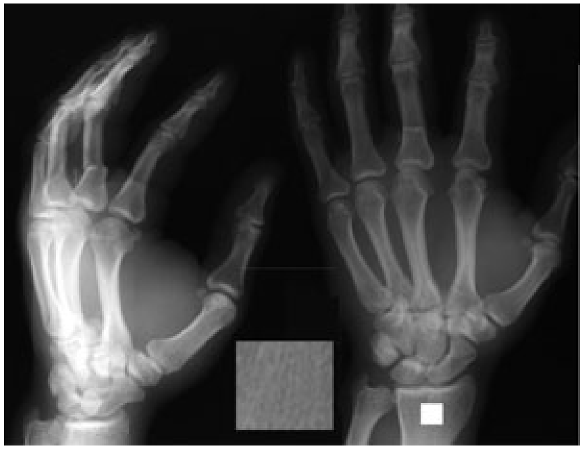
Grid Opportunities in Medical Imaging A Case Study

- Daniela S. Raicu, PhD
  - Assistant Professor
  - Email: draicu@cs.depaul.edu
  - Lab URL: http://facweb.cs.depaul.edu/research/vc/
  - Original slides: http://www.ci.uchicago.edu/wiki/bin/viewfile/VDS/DslCS/DSLWorkshop2007?rev=1;filename=draicu_DSL_workshop.pdf

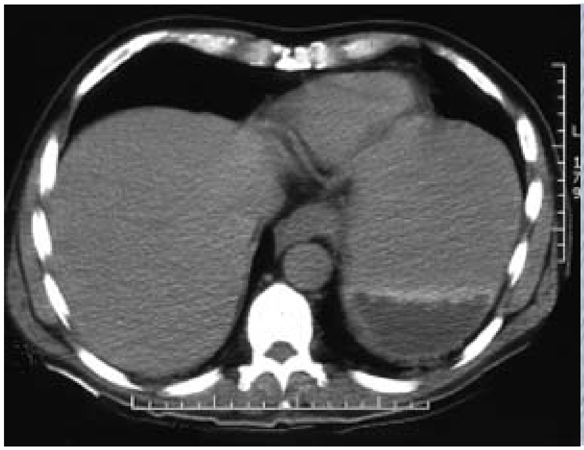


What is Medical Imaging (MI)?

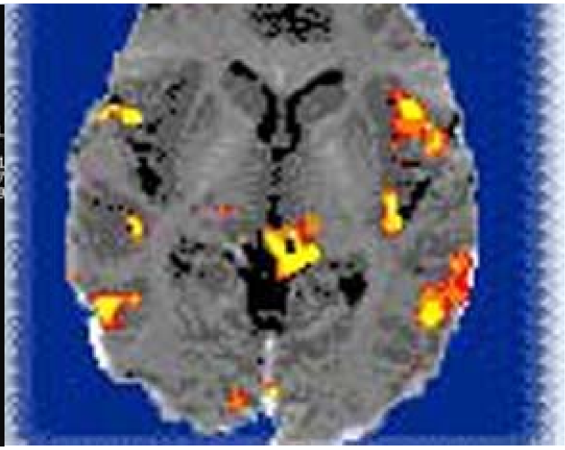
The study of *medical imaging* is concerned with the interaction of all forms of radiation with tissue and the development of appropriate technology to extract clinically useful information from observation of this technology.



X-Ray



CT



fMRI

CT Medical Imaging (MI)@ CTI

Project 1: ***Texture-based soft-tissue segmentation***

Segmentation

Analysis

Project 2: ***Content-based medical image retrieval***

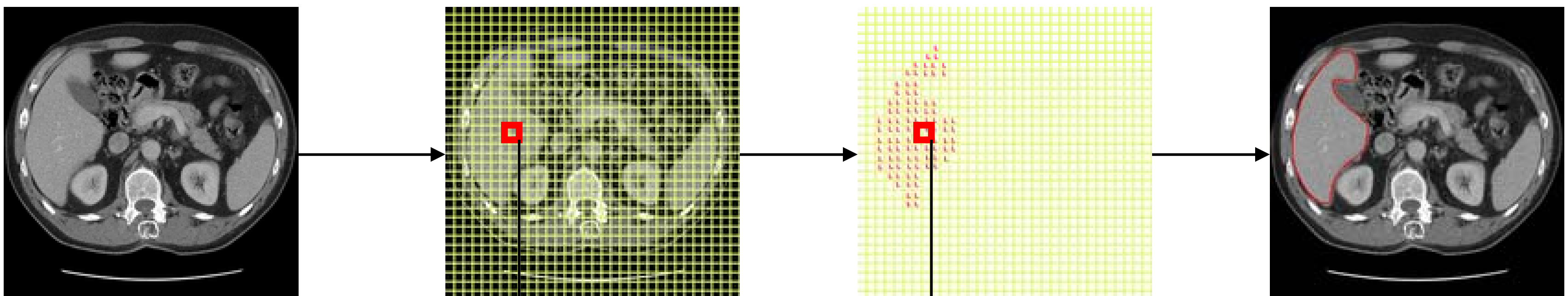
Visualization

Classification

Retrieval

Soft-tissue Segmentation in Computed Tomography

Goal: context-sensitive tools for radiology reporting
Approach: pixel-based texture classification



Pixel Level Texture Extraction

$\left[d_1, d_2, \ldots d_k\right]$

Pixel Level Classification

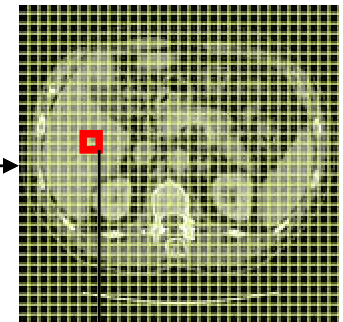
$\left[tissue\_label\right]$

Organ Segmentation

Soft-tissue Segmentation in Computed Tomography

Pixel-based texture extraction:

Input Patient Data Characteristics:
- hundreds of images per patient
- image spatial resolution: 512 x512
- image gray-level resolution: 2^{12}



Pixel Level Texture Extraction

$[d_1, d_2, \ldots d_k]$

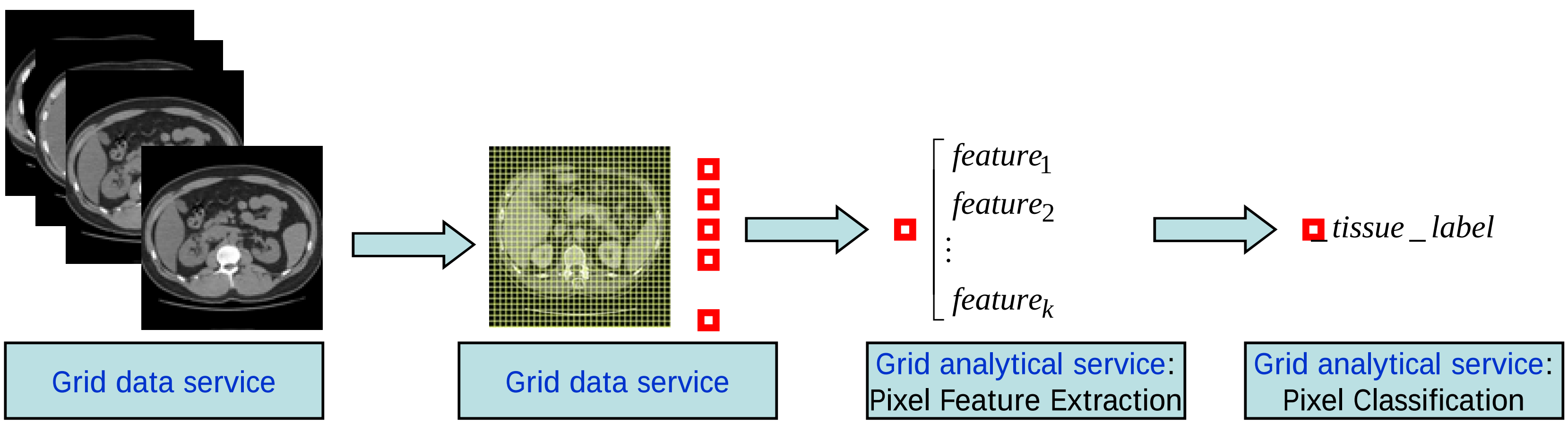
Challenges:
- Storage:
  - Input: 0.5+ terabyte of raw data dispersed over about 100K+ images
  - Output: 90+ terabytes of low-level features in a 180 dimensional feature space
- Compute:
  - 24 hours of compute time = 180 features for a single image on a modern 3GHz workstation

Output Data Characteristics:
- low-level image features (numerical descriptors)
- k=180 Haralick texture features per pixel (9 descriptors x4 directions x5 displacements)

Grid Computing Opportunities

<u>Challenges and Grid Opportunities:</u>

- Storage
  - define the logical and physical organization of the medical image datasets along with the relevant information extracted from them
- Compute time
  - coordinate the *parallel execution* of the texture feature extraction and classification algorithms such that each *image, pixel and feature* could be processed independently of the other images, pixels and features, respectively.



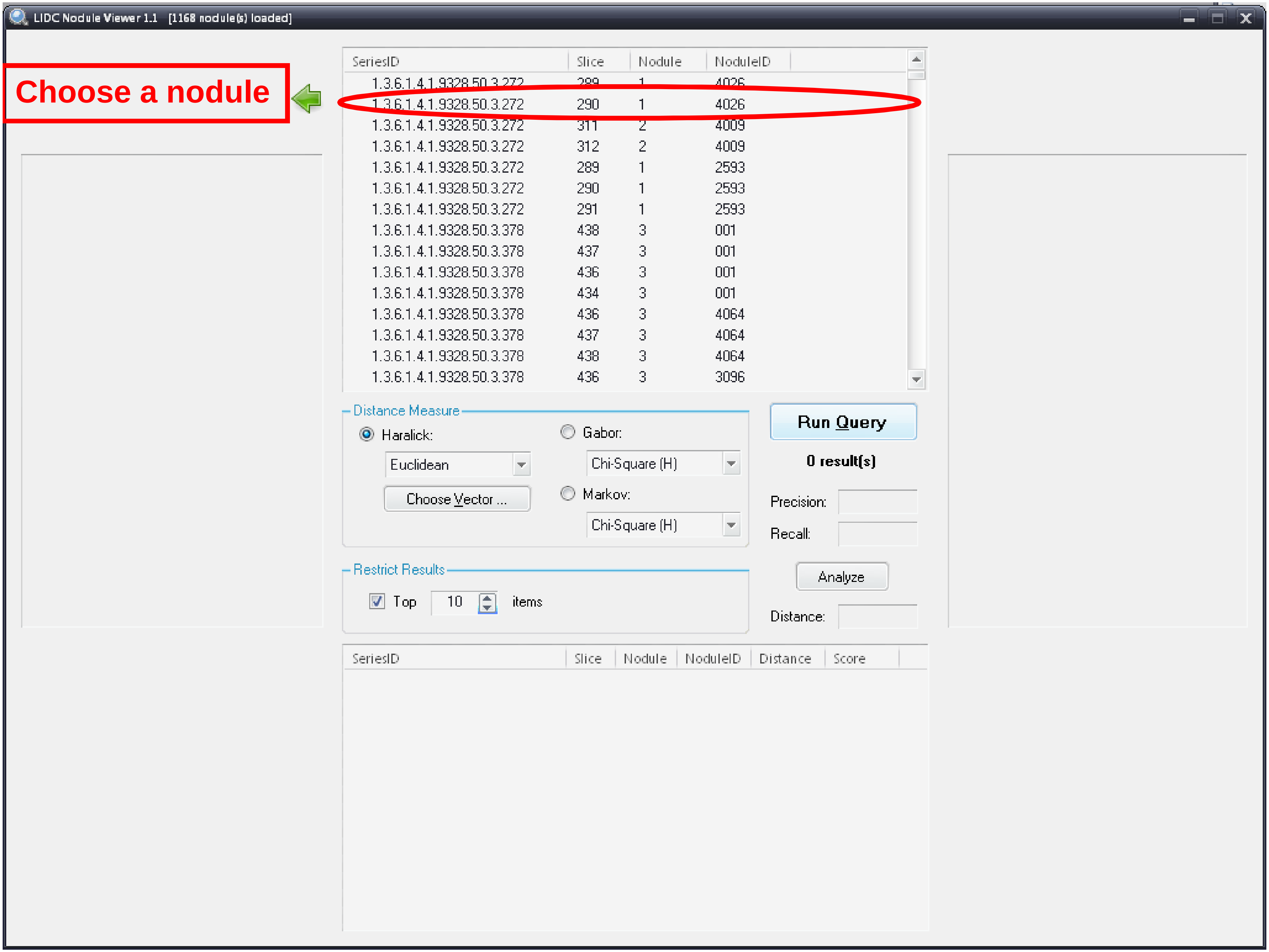
Content-based medical image retrieval (CBMS) systems

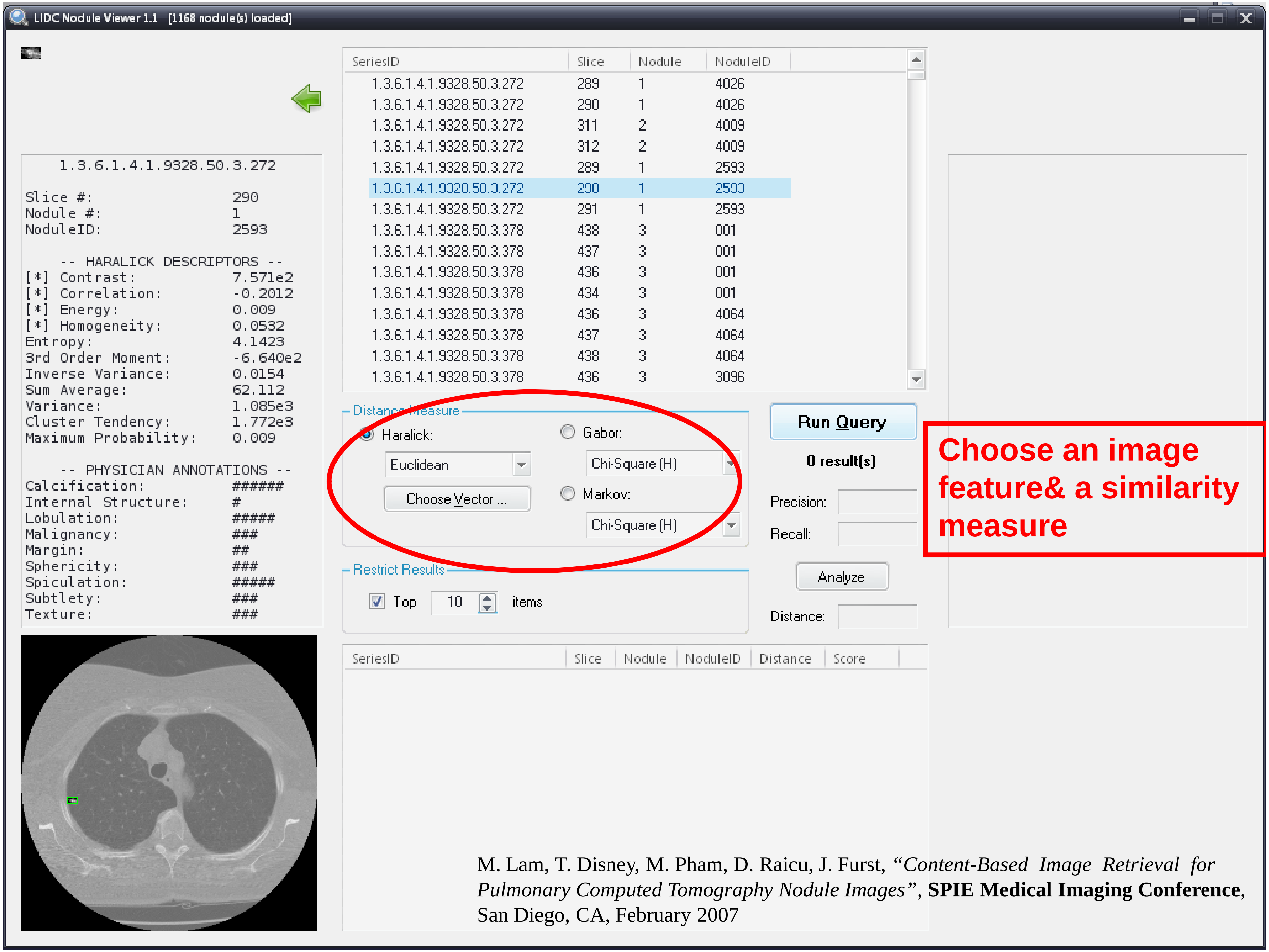
Definition of Content-based Image Retrieval:

Content-based image retrieval is a technique for retrieving images on the basis of automatically derived image features such as texture and shape.

Applications of Content-based Image Retrieval:

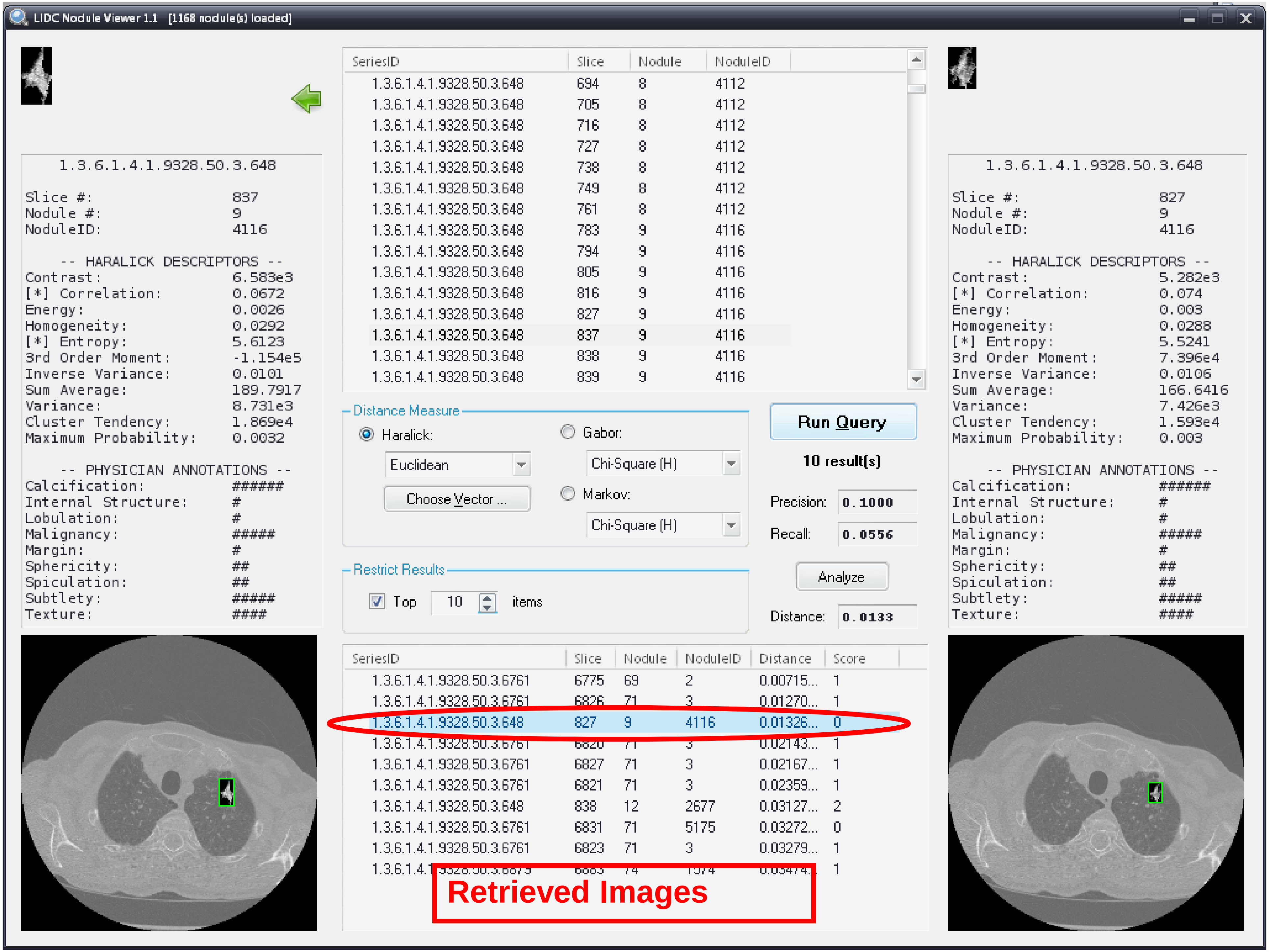
- Teaching
- Case-base reasoning
- Evidence-based medicine


LIDC Nodule Viewer 1.1 [1168 nodule(s) loaded]
Choose a nodule
SeriesID Slice Nodule NoduleID
1.3.6.1.4.1.9328.50.3.272 290 1 4026
1.3.6.1.4.1.9328.50.3.272 311 2 4009
1.3.6.1.4.1.9328.50.3.272 312 2 4009
1.3.6.1.4.1.9328.50.3.272 289 1 2593
1.3.6.1.4.1.9328.50.3.272 290 1 2593
1.3.6.1.4.1.9328.50.3.272 291 1 2593
1.3.6.1.4.1.9328.50.3.378 438 3 001
1.3.6.1.4.1.9328.50.3.378 437 3 001
1.3.6.1.4.1.9328.50.3.378 436 3 001
1.3.6.1.4.1.9328.50.3.378 434 3 001
1.3.6.1.4.1.9328.50.3.378 436 3 4064
1.3.6.1.4.1.9328.50.3.378 437 3 4064
1.3.6.1.4.1.9328.50.3.378 438 3 4064
1.3.6.1.4.1.9328.50.3.378 436 3 3096
Distance Measure
Haralick:
Euclidean
Choose Vector ...
Gabor:
Chi-Square (H)
Markov:
Chi-Square (H)
Run Query
0 result(s)
Precision:
Recall:
Analyze
Distance:
Restrict Results
Top 10 items
SeriesID Slice Nodule NoduleID Distance Score



Choose an image feature& a similarity measure

M. Lam, T. Disney, M. Pham, D. Raicu, J. Furst, "*Content-Based Image Retrieval for Pulmonary Computed Tomography Nodule Images*", **SPIE Medical Imaging Conference**, San Diego, CA, February 2007


LIDC Nodule Viewer 1.1 [1168 nodule(s) loaded]
1.3.6.1.4.1.9328.50.3.648
Slice #: 837
Nodule #: 9
NoduleID: 4116
-- HARALICK DESCRIPTORS --
Contrast: 6.583e3
[*] Correlation: 0.0672
Energy: 0.0026
Homogeneity: 0.0292
[*] Entropy: 5.6123
3rd Order Moment: -1.154e5
Inverse Variance: 0.0101
Sum Average: 189.7917
Variance: 8.731e3
Cluster Tendency: 1.869e4
Maximum Probability: 0.0032
-- PHYSICIAN ANNOTATIONS --
Calcification: ######
Internal Structure: #
Lobulation: #
Malignancy: #####
Margin: #
Sphericity: ##
Spiculation: ##
Subtlety: #####
Texture: ####
SeriesID Slice Nodule NoduleID
1.3.6.1.4.1.9328.50.3.648 694 8 4112
1.3.6.1.4.1.9328.50.3.648 705 8 4112
1.3.6.1.4.1.9328.50.3.648 716 8 4112
1.3.6.1.4.1.9328.50.3.648 727 8 4112
1.3.6.1.4.1.9328.50.3.648 738 8 4112
1.3.6.1.4.1.9328.50.3.648 749 8 4112
1.3.6.1.4.1.9328.50.3.648 761 8 4112
1.3.6.1.4.1.9328.50.3.648 783 9 4116
1.3.6.1.4.1.9328.50.3.648 794 9 4116
1.3.6.1.4.1.9328.50.3.648 805 9 4116
1.3.6.1.4.1.9328.50.3.648 816 9 4116
1.3.6.1.4.1.9328.50.3.648 827 9 4116
1.3.6.1.4.1.9328.50.3.648 837 9 4116
1.3.6.1.4.1.9328.50.3.648 838 9 4116
1.3.6.1.4.1.9328.50.3.648 839 9 4116
Distance Measure
Haralick:
Euclidean
Choose Vector ...
Gabor:
Chi-Square (H)
Markov:
Chi-Square (H)
Run Query
10 result(s)
Precision: 0.1000
Recall: 0.0556
Analyze
Distance: 0.0133
Restrict Results
Top 10 items
SeriesID Slice Nodule NoduleID Distance Score
1.3.6.1.4.1.9328.50.3.6761 6775 69 2 0.00715... 1
1.3.6.1.4.1.9328.50.3.6761 6826 71 3 0.01270... 1
1.3.6.1.4.1.9328.50.3.648 827 9 4116 0.01326... 0
1.3.6.1.4.1.9328.50.3.6761 6827 71 3 0.02167... 1
1.3.6.1.4.1.9328.50.3.6761 6821 71 3 0.02359... 1
1.3.6.1.4.1.9328.50.3.648 838 12 2677 0.03127... 2
1.3.6.1.4.1.9328.50.3.6761 6831 71 5175 0.03272... 0
1.3.6.1.4.1.9328.50.3.6761 6823 71 3 0.03279... 1
Retrieved Images
1.3.6.1.4.1.9328.50.3.648
Slice #: 827
Nodule #: 9
NoduleID: 4116
-- HARALICK DESCRIPTORS --
Contrast: 5.282e3
[*] Correlation: 0.074
Energy: 0.003
Homogeneity: 0.0288
[*] Entropy: 5.5241
3rd Order Moment: 7.396e4
Inverse Variance: 0.0106
Sum Average: 166.6416
Variance: 7.426e3
Cluster Tendency: 1.593e4
Maximum Probability: 0.003
-- PHYSICIAN ANNOTATIONS --
Calcification: ######
Internal Structure: #
Lobulation: #
Malignancy: #####
Margin: #
Sphericity: ##
Spiculation: ##
Subtlety: #####
Texture: ####

Grid Computing Opportunities

Challenges and Grid Opportunities:

- Compute time: Given the image retrieval system, four different layers can be identified that offer potential for parallelization:
  - Queries tend to be mutually independent. Thus, several queries can be processed in parallel. This is of interest, if several users access the system at the same time or if several queries are run in batch mode.
  - The distances from the queries to the database images can be calculated in parallel as the database images are independent from each other.
  - Parallelization is possible on the feature level, because the distances for the individual features can be calculated in parallel.
  - Multiple combinations of feature spaces and similarity metrics can be run in parallel to determine the best retrieval results
    - 2047 vectors * 3 similarity measures * 5 number of retrieved images = 30,705 combinations

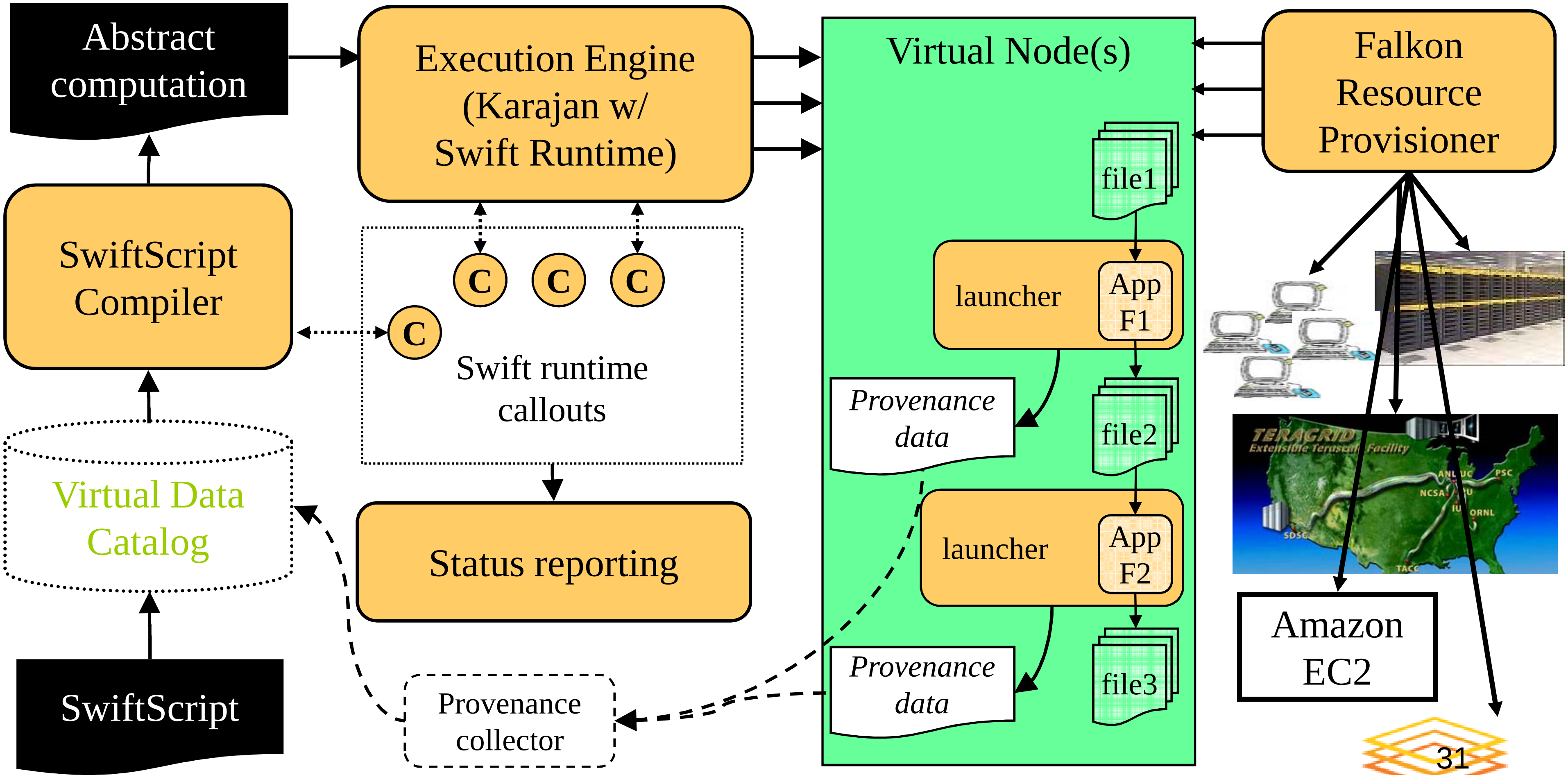
Solutions?

Workflows?

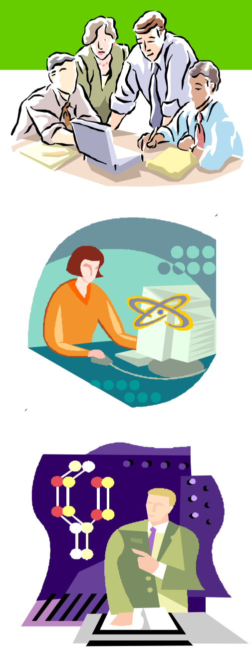
Existing and emerging workflow technologies

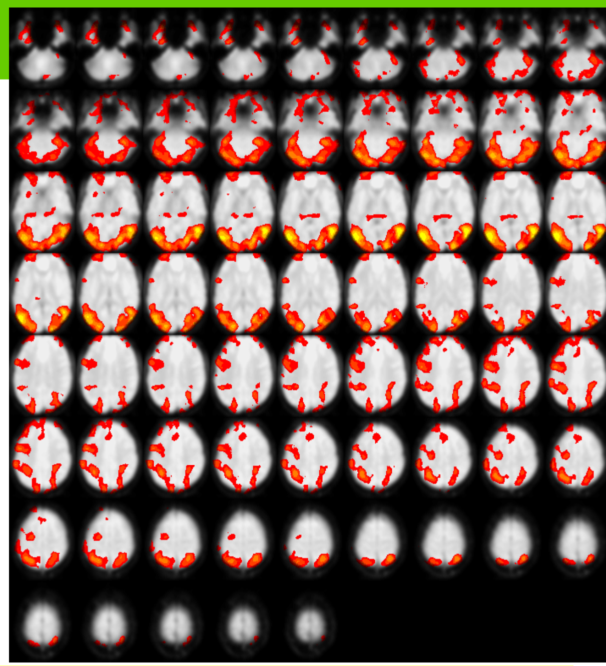
	Swift	BPEL	XPDL	MS Workflow Fundation	DAGMan	Tavena	Triana	Kepler	Mapreduce	Star-P
Scales to Grids	++	-	-	-	++	-	-	-	+	+
Typing	++	++	++	++	-	-	-	+	-	+
Iteration	++	-/+	-	+	-	-	-	+	+	+
Scripting	++	-	-	+	+	+	-	-	-	++
Dataset Mapping	+	-	-	-	-	-	-	-	-	-
Service Interop	+	-	+	-	-	-	-	+	-	-
Subflow/comp.	+	-	+	+	-	-	+	+	-	+
Provenance	+	-	-	+	-	+	-	+	-	-
Open source	+	+	+	-	+	+	+	+	-	-

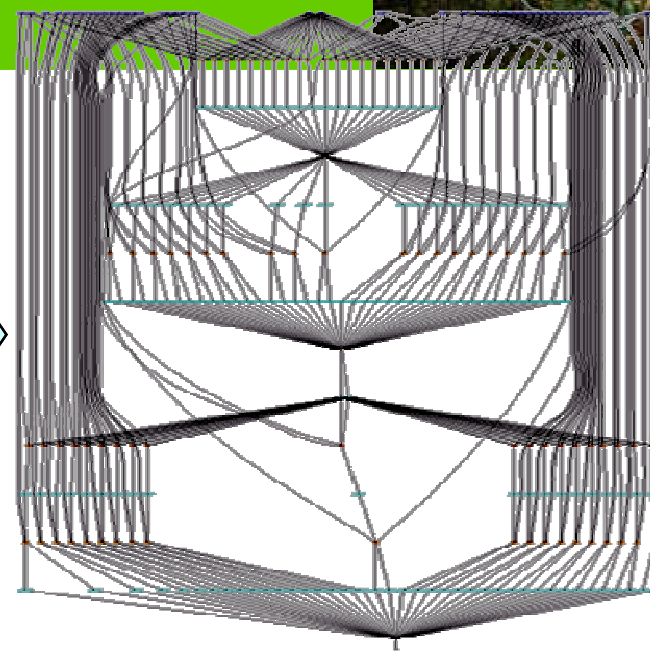
Swift Architecture

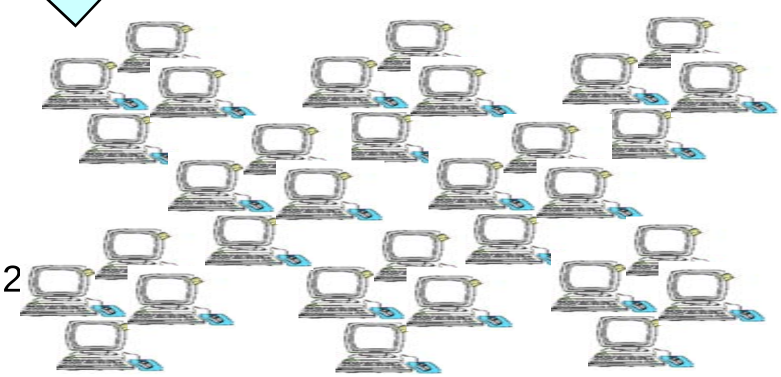


Functional MRI (fMRI)





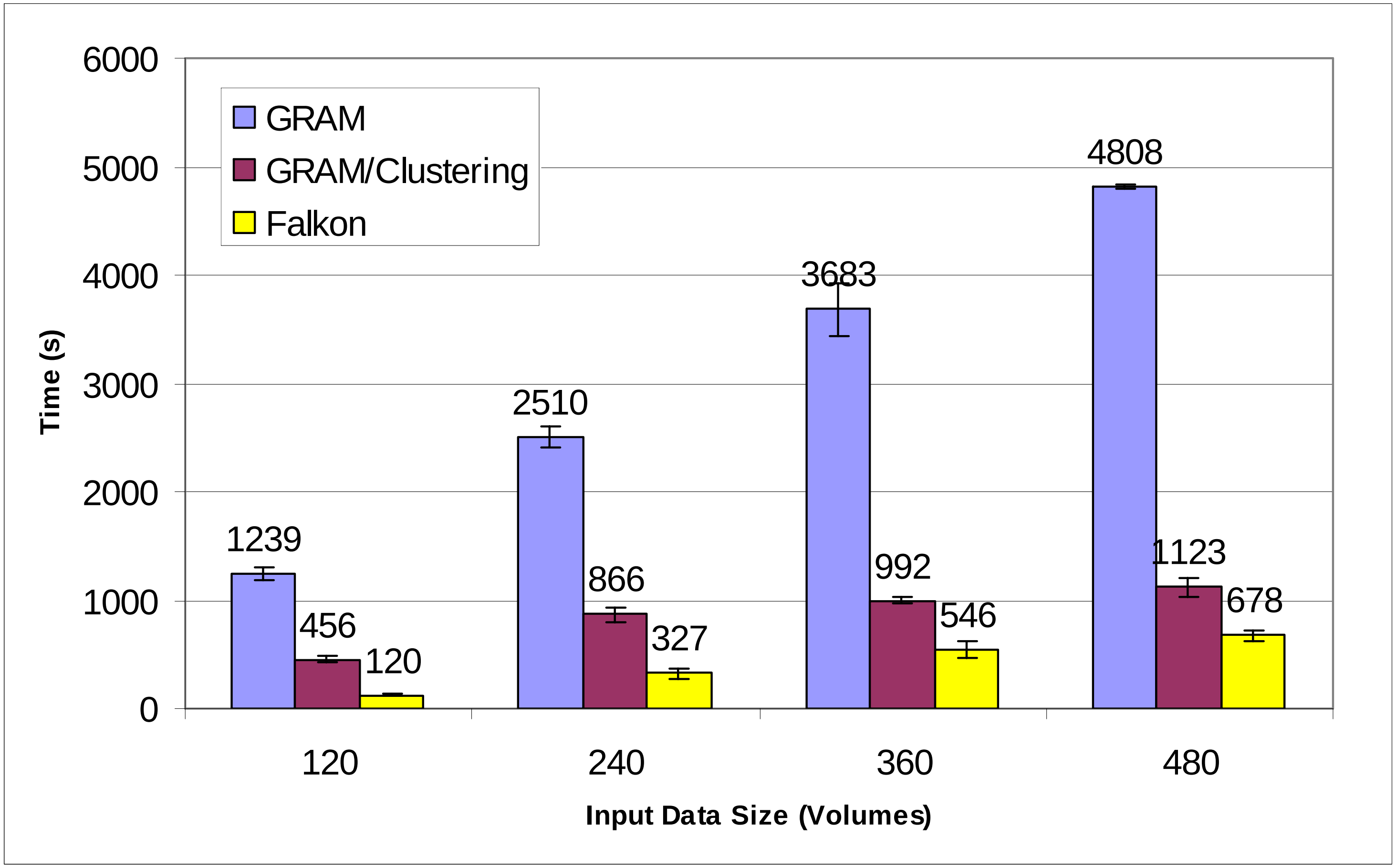


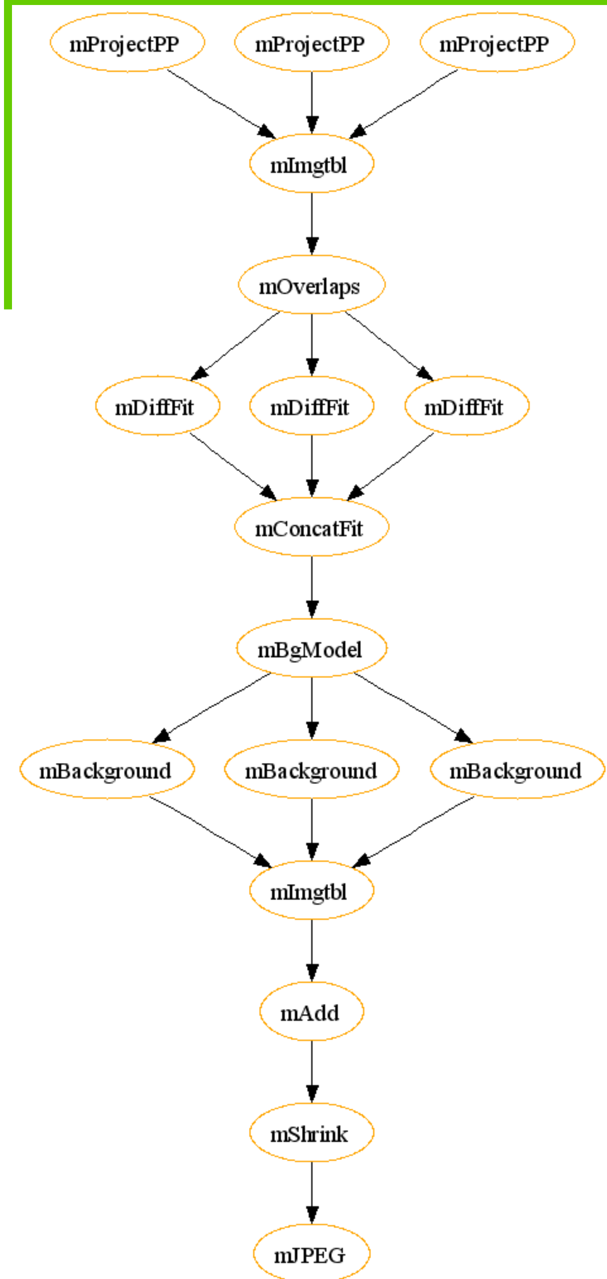


- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies

Completed Milestones: fMRI Application

- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time



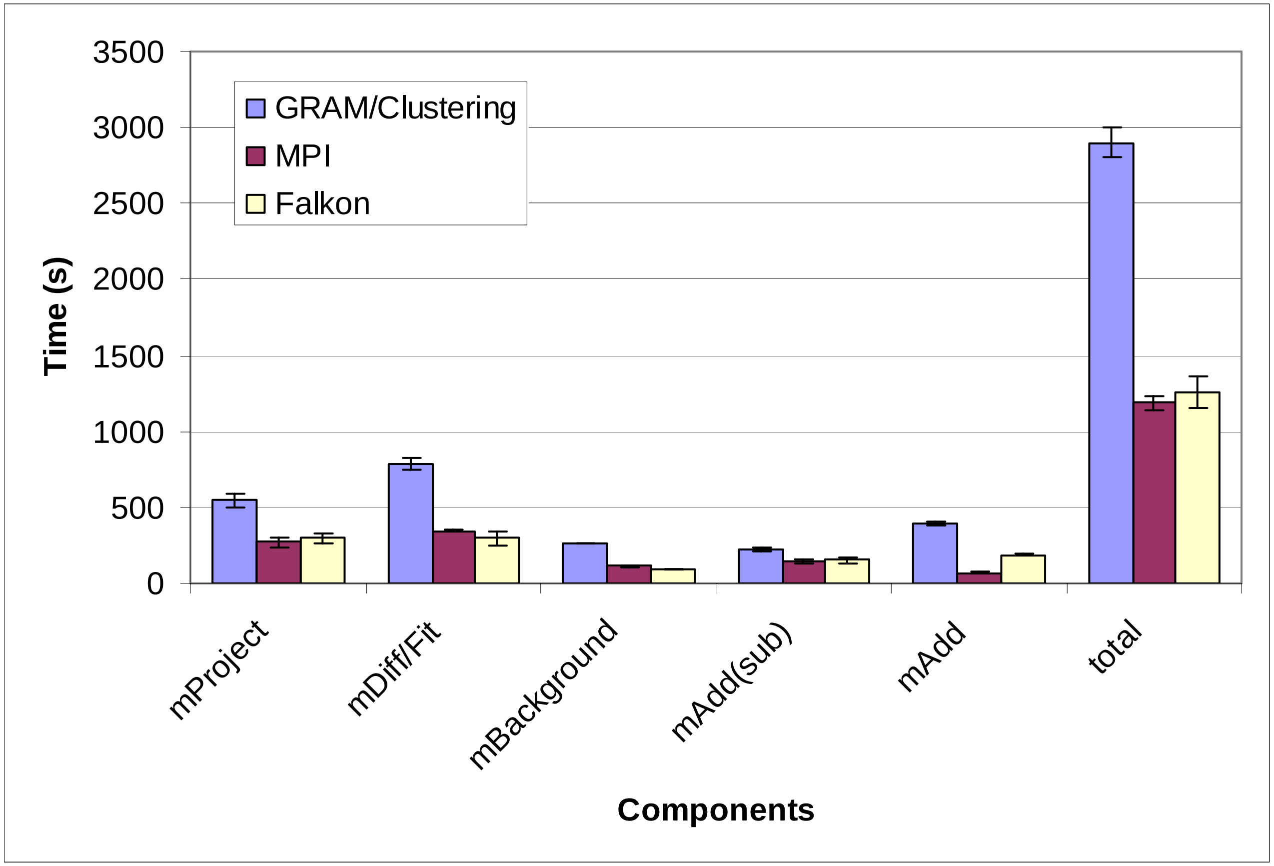


Montage

B. Berriman, J. Good (Caltech)
J. Jacob, D. Katz (JPL)

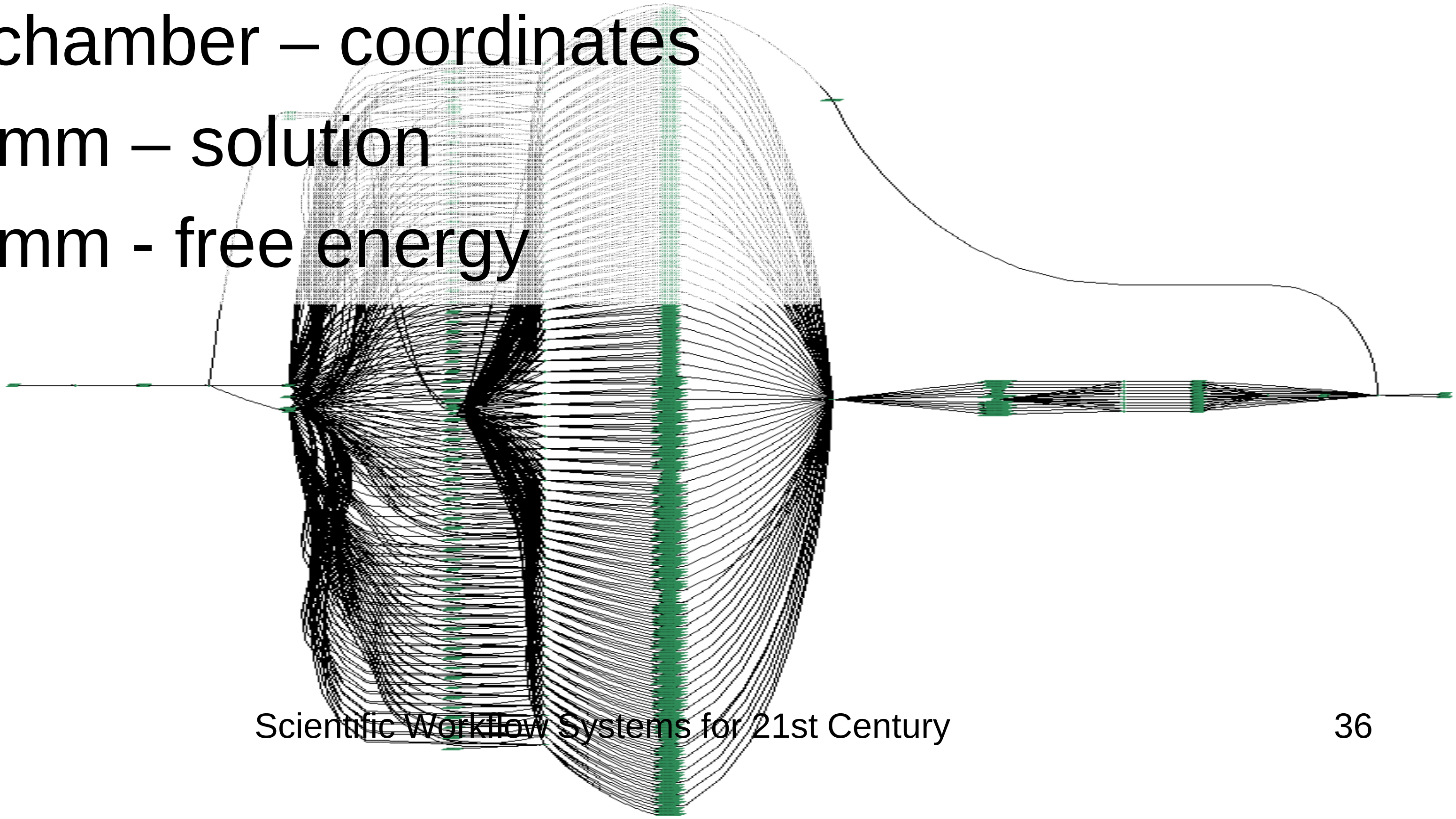
Completed Milestones: Montage Application

- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



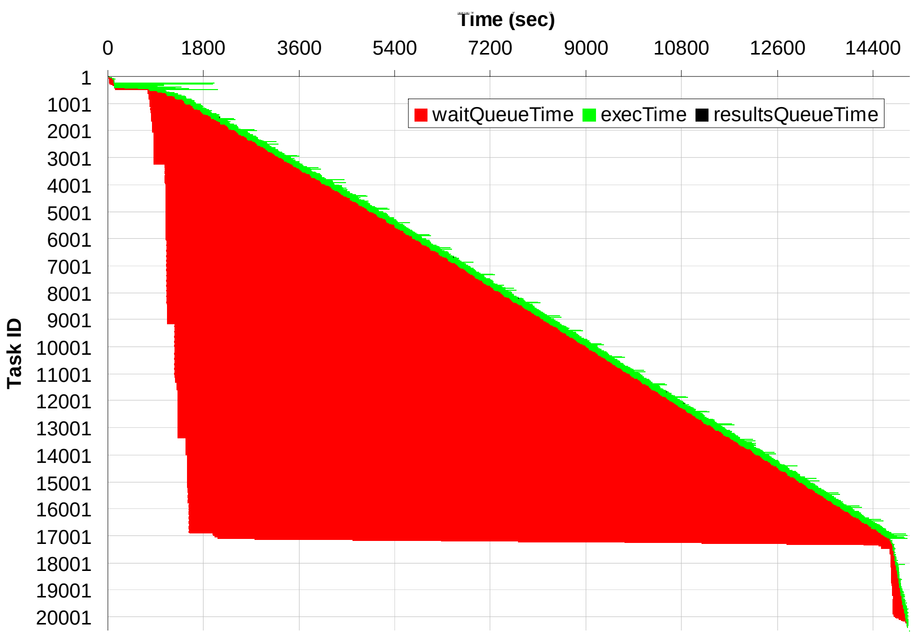
Molecular Dynamics

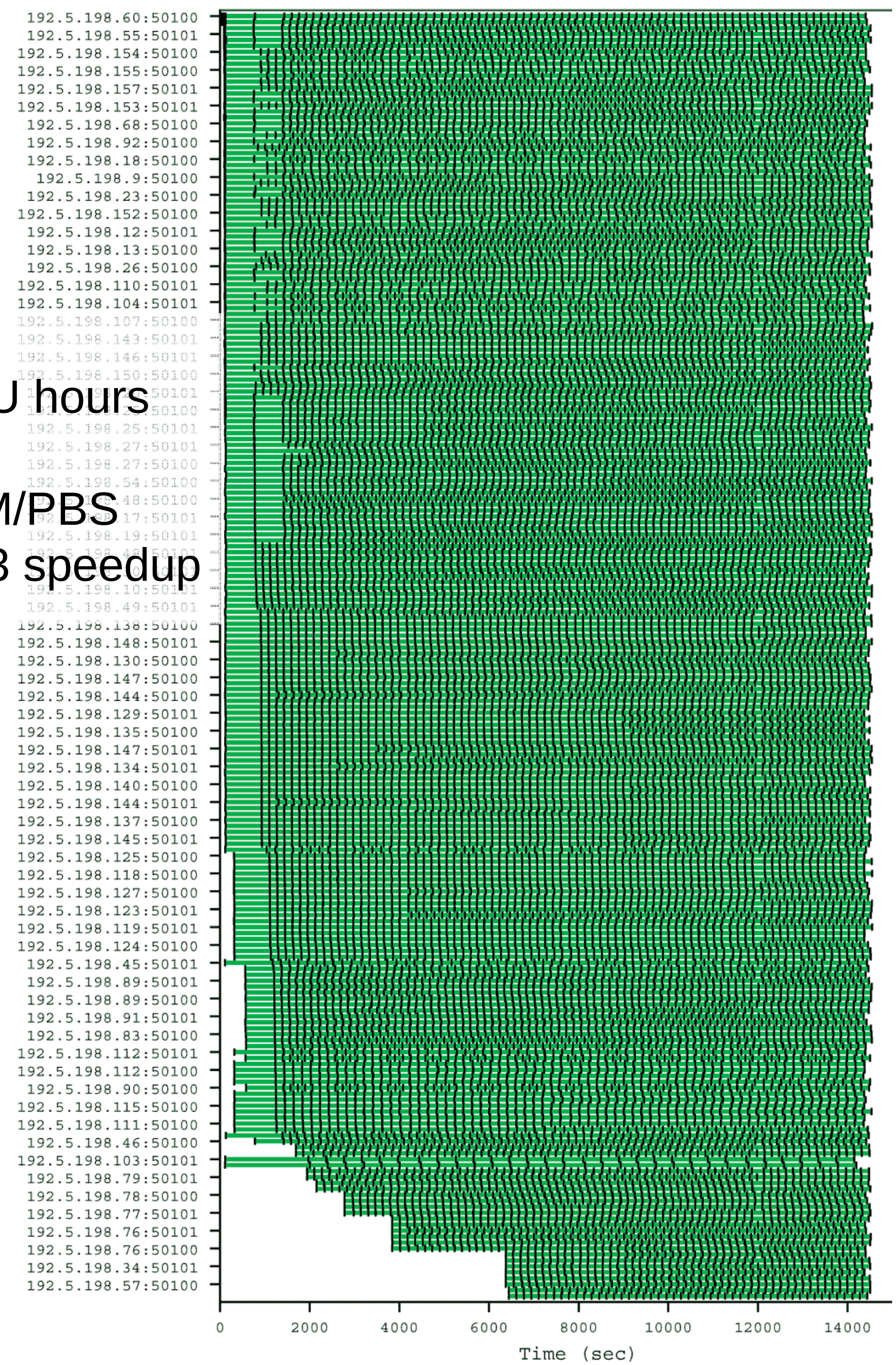
- Determination of free energies in aqueous solution
  - Antechamber – coordinates
  - Charmm – solution
  - Charmm - free energy



MolDyn Application

- 244 molecules → 20497 jobs
- 15091 seconds on 216 CPUs → 867.1 CPU hours
- Efficiency: 99.8%
- Speedup: 206.9x → 8.2x faster than GRAM/PBS
- 50 molecules w/ GRAM (4201 jobs) → 25.3 speedup



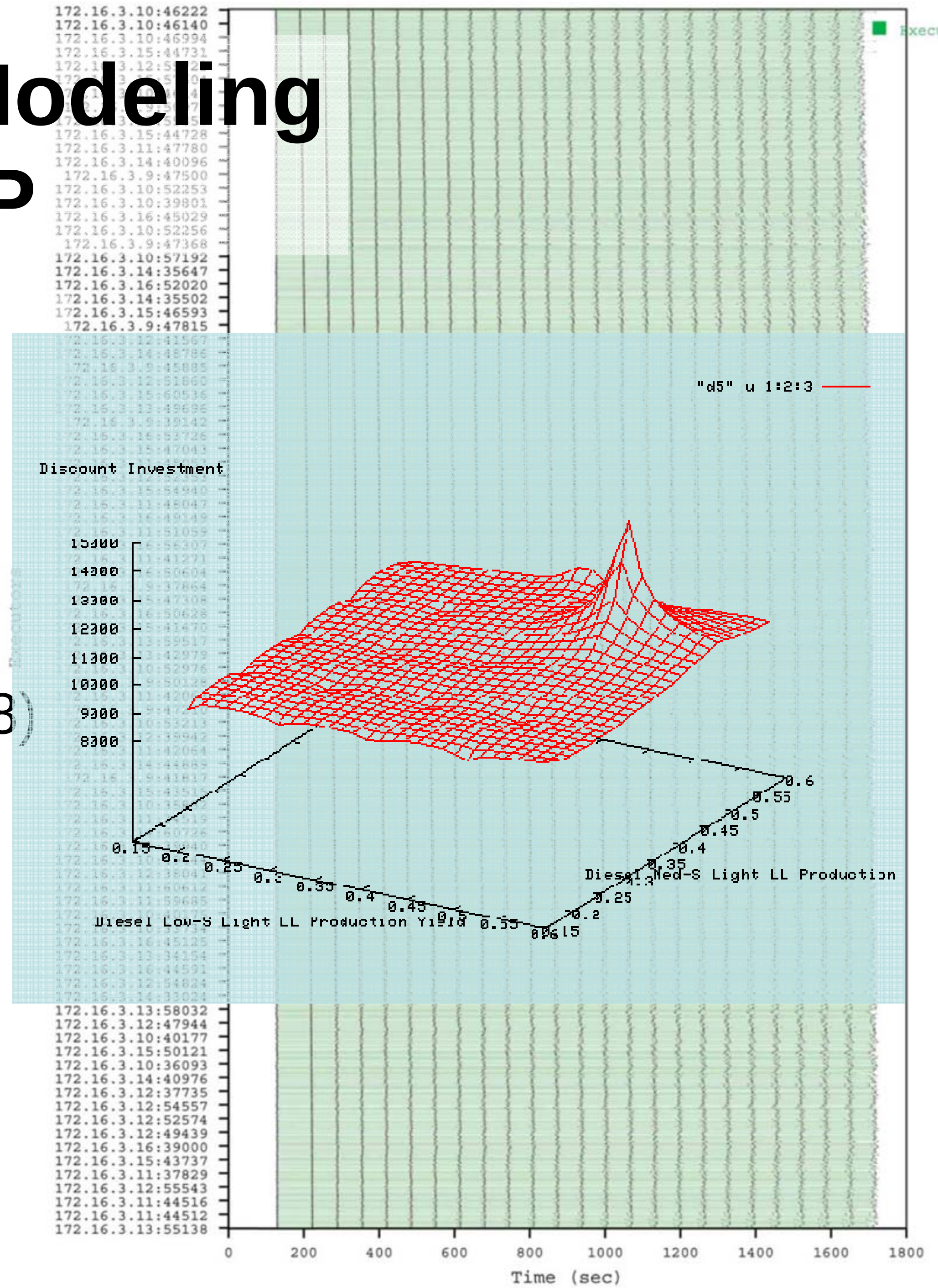


Scientific Workflow Sys

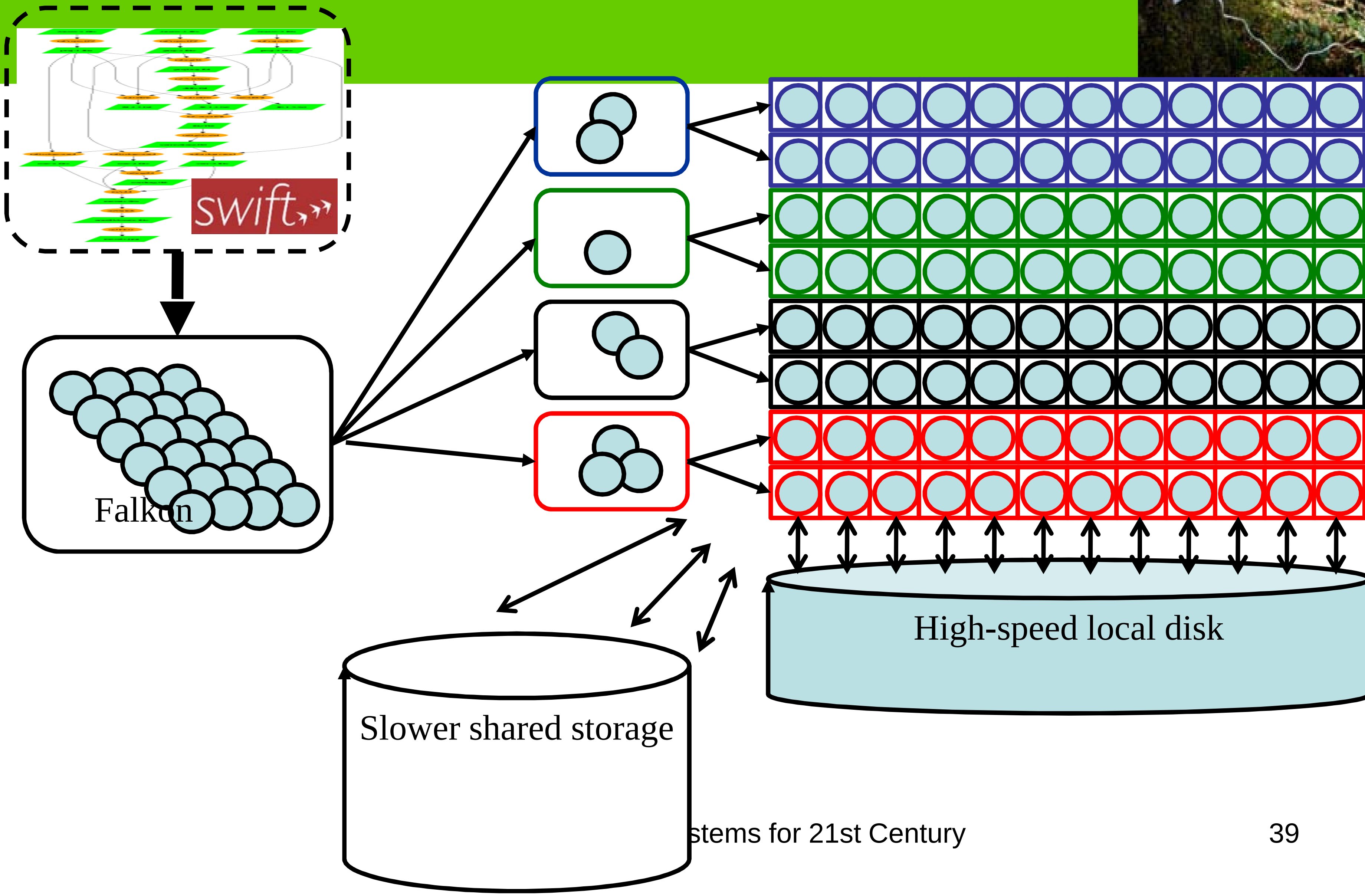
MARS Economic Modeling on IBM BG/P

- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%

ic Workf

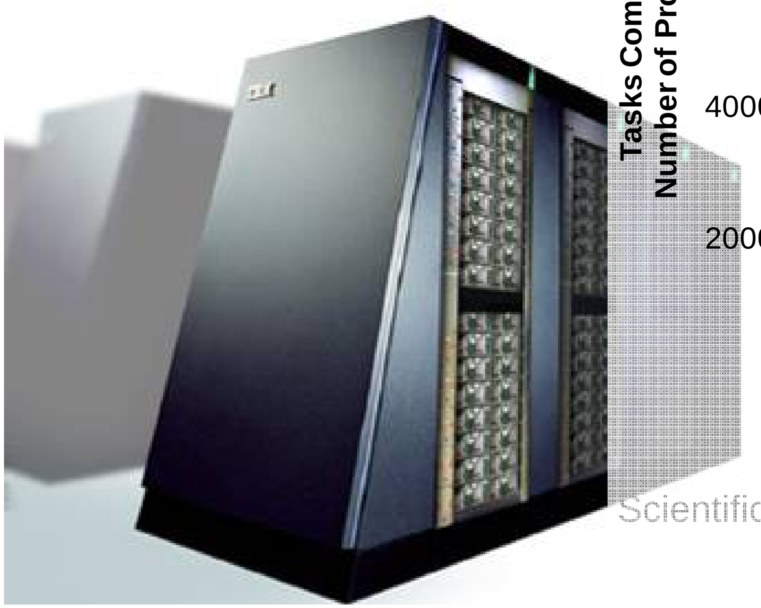


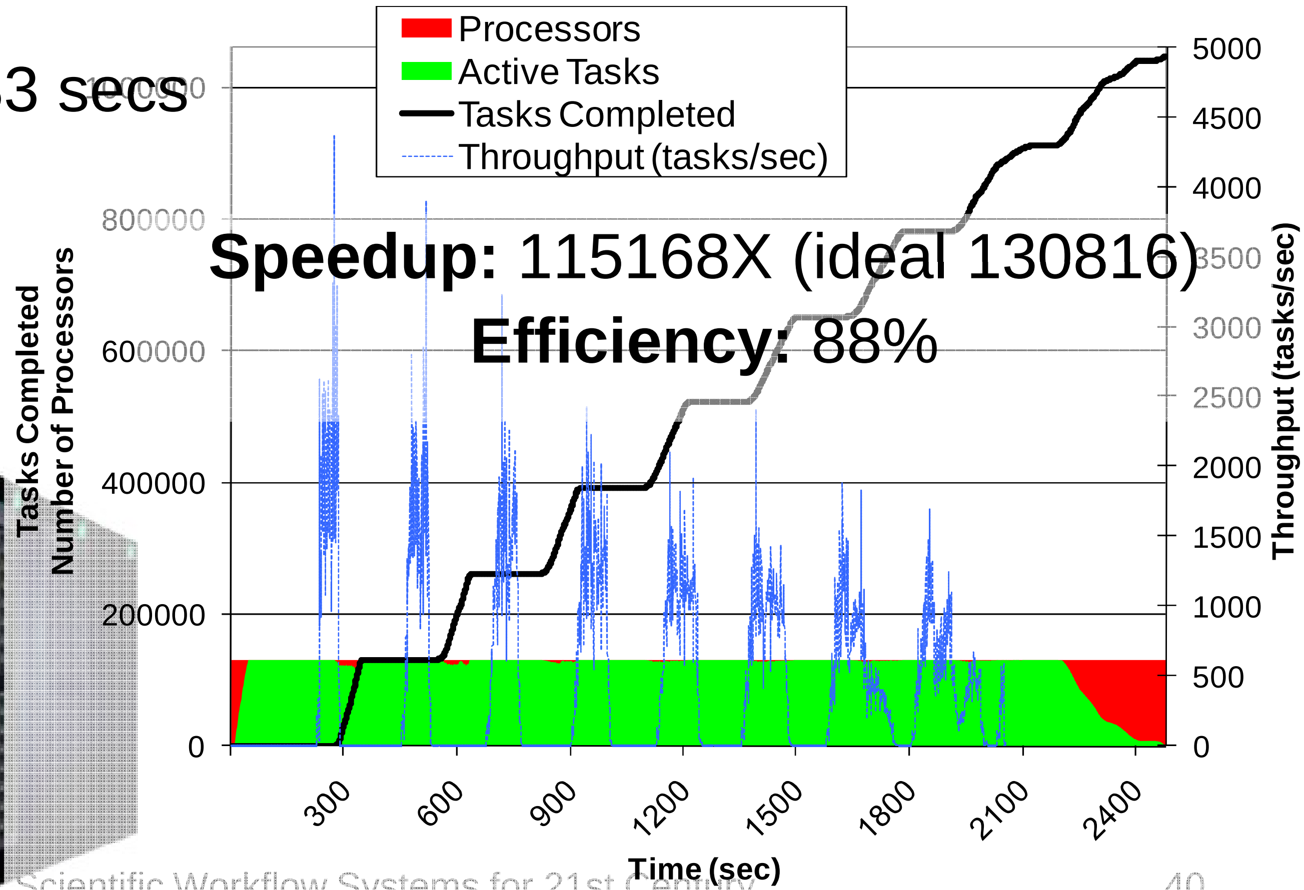
Managing 120K CPUs



MARS Economic Modeling on IBM BG/P (128K CPUs)

- CPU Cores: 130816
- Tasks: 1048576
- Elapsed time: 2483 secs
- CPU Years: 9.3



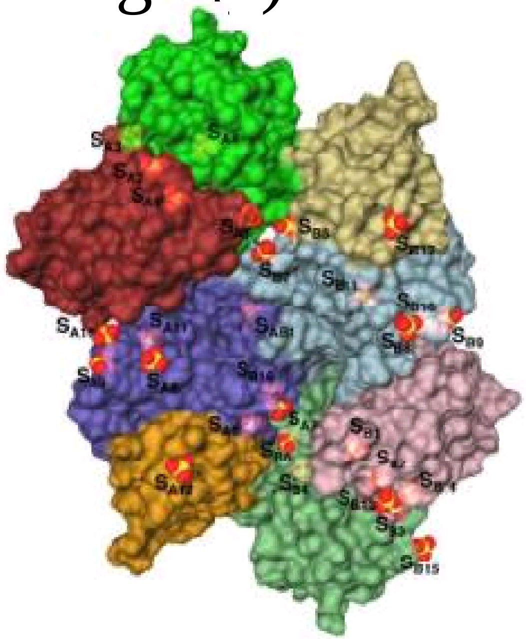


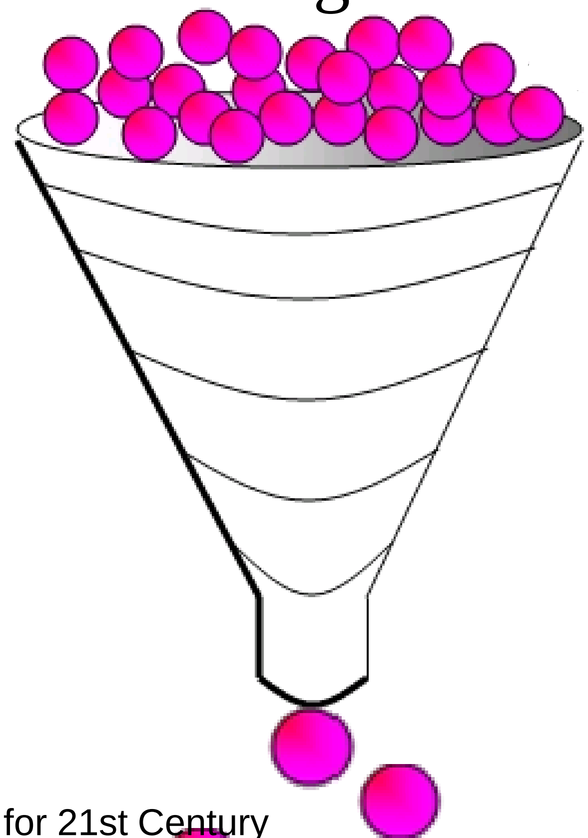
Speedup: 115168X (ideal 130816)

Efficiency: 88%

Many Many Tasks: Identifying Potential Drug Targets

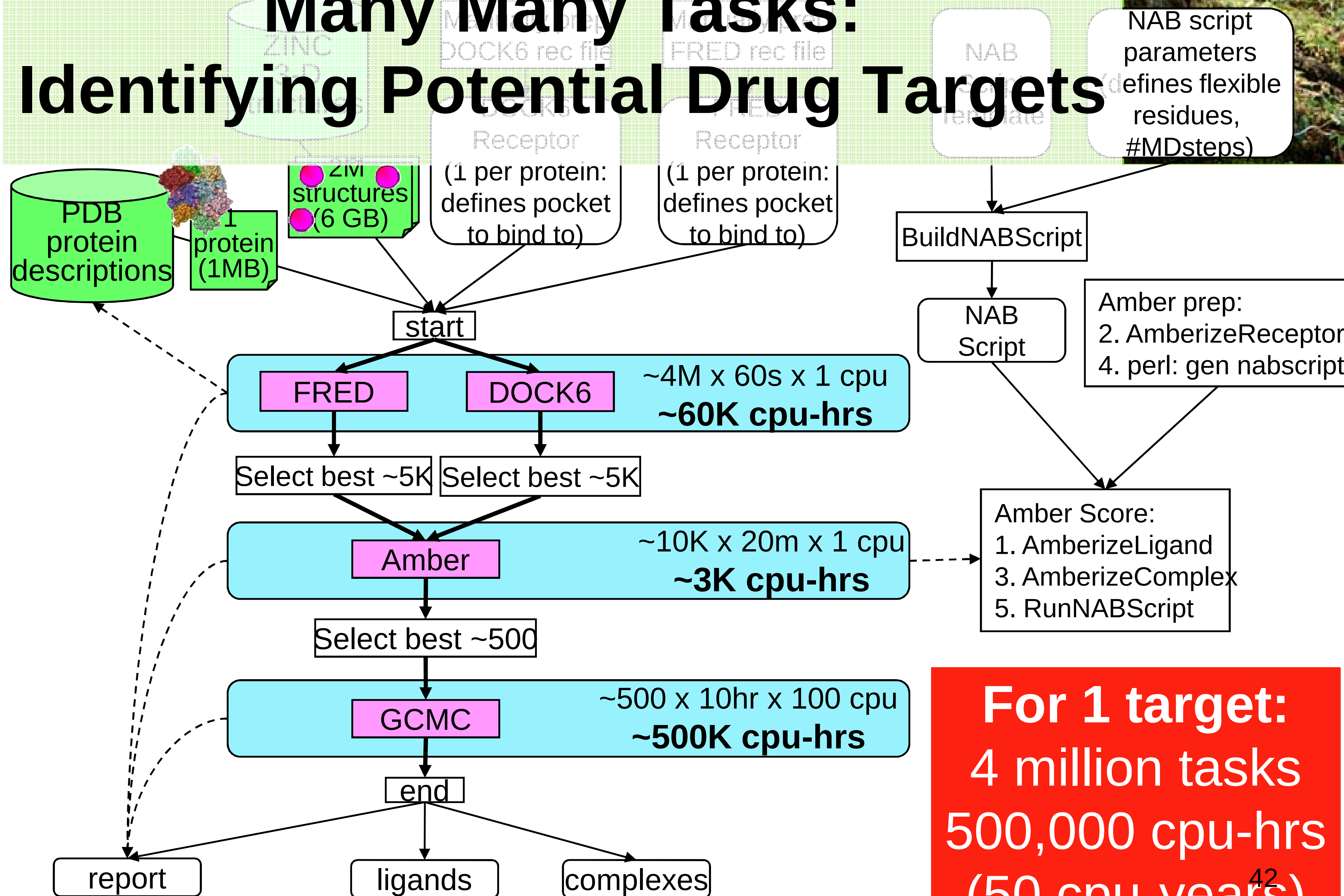
Protein target(s) x 2M+ ligands





(Mike Kubal, Benoit Roux, and others)

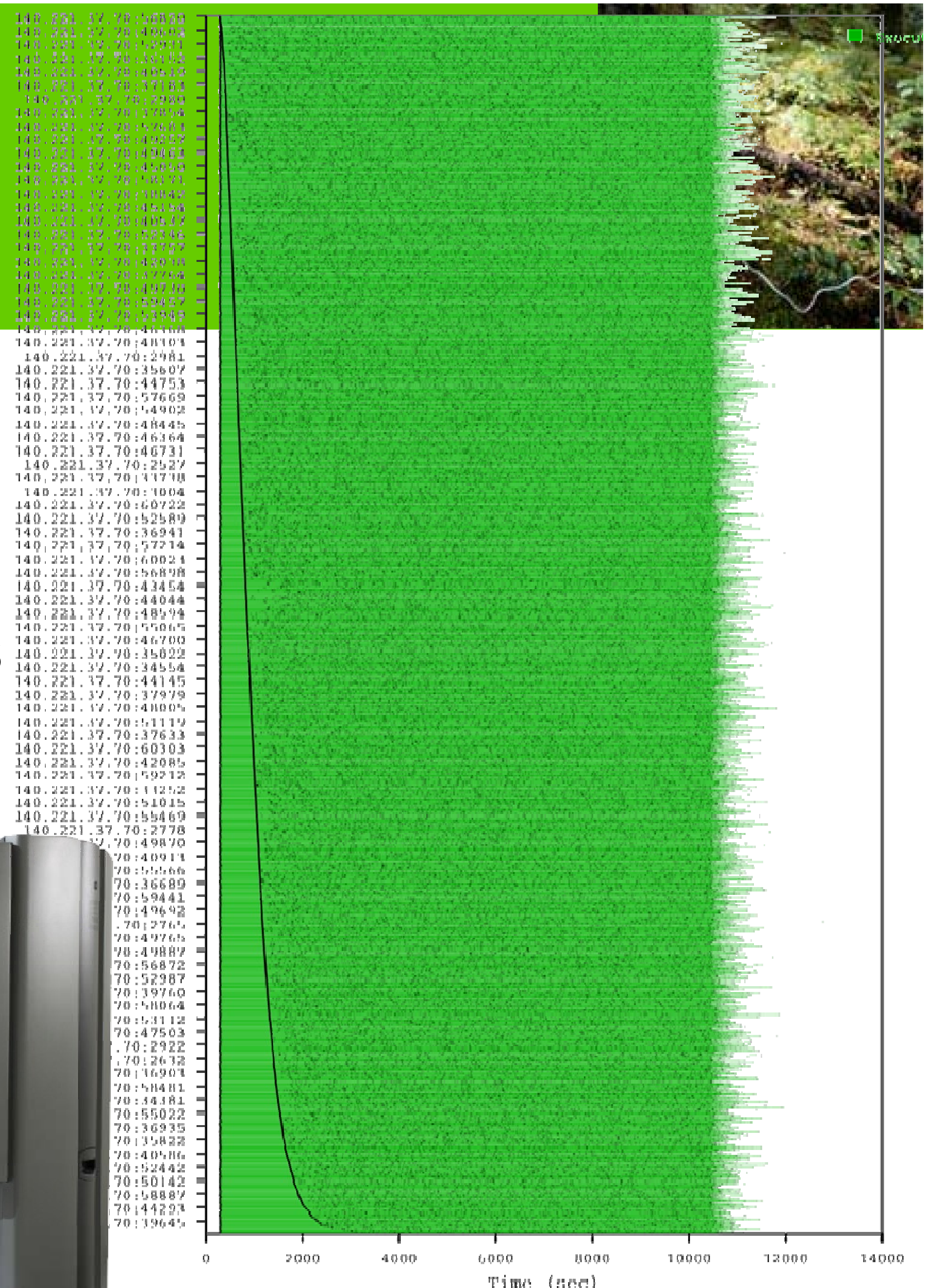
Many Many Tasks: Identifying Potential Drug Targets



For 1 target:
4 million tasks
500,000 cpu-hrs
(50 cpu-years)

DOCK on SiCortex

- CPU cores: 5760
- Tasks: 92160
- Elapsed time: 12821 sec
- Compute time: 1.94 CPU years
- Average task time: 660.3 sec
- Speedup: 5650X (ideal 5760)
- Efficiency: 98.2%



DOCK on the BG/P

CPU cores: 118784

Tasks: 934803

Elapsed time: 2.01 hours

Compute time: 21.43 CPU years

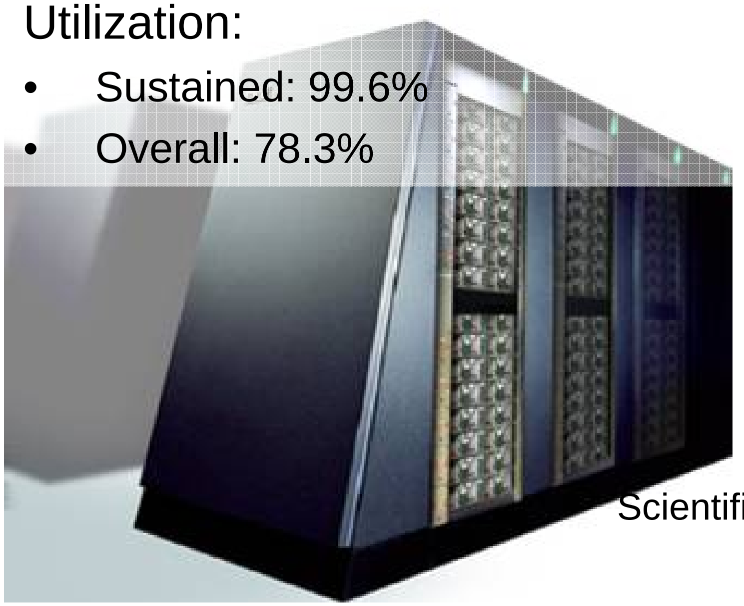
Average task time: 667 sec

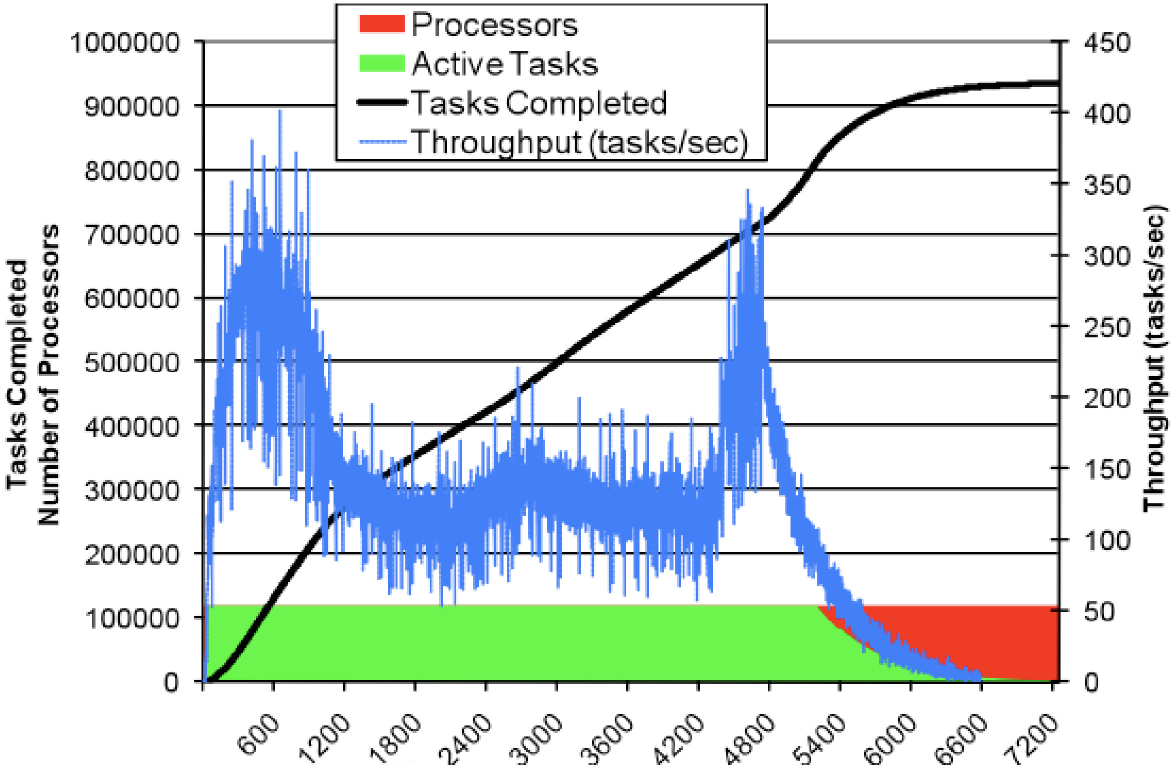
Relative Efficiency: 99.7%

(from 16 to 32 racks)

Utilization:

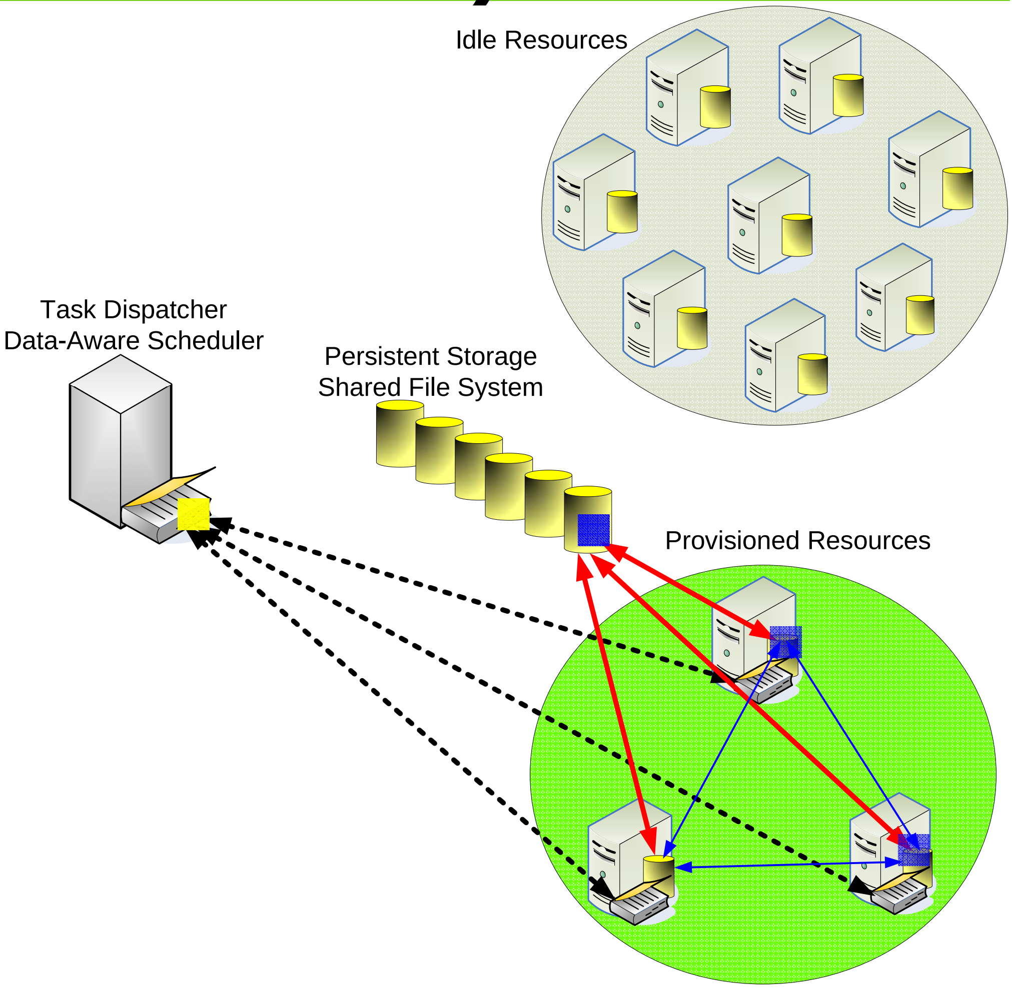
- Sustained: 99.6%
- Overall: 78.3%





Support for Data Intensive Applications (Falkon and Data Diffusion)

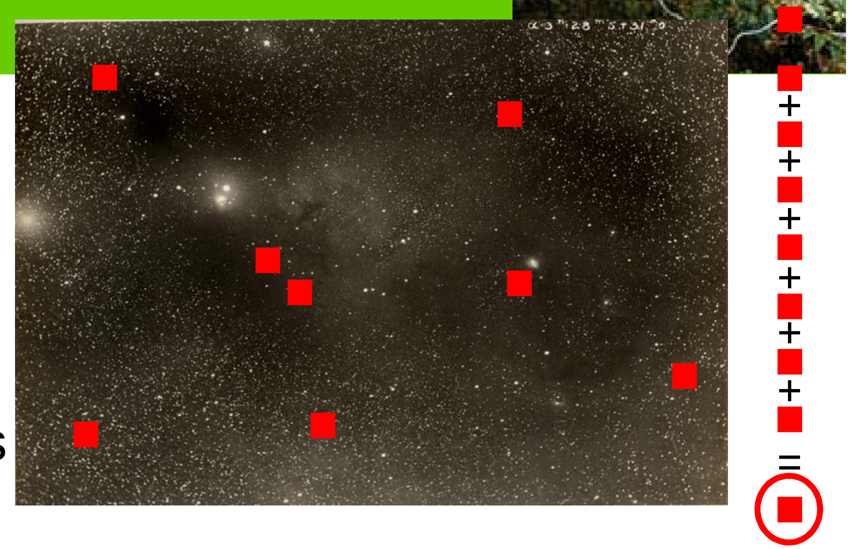
- Resource acquired in response to demand
- Data and applications diffuse from archival storage to newly acquired resources
- Resource "caching" allows faster responses to subsequent requests
  - Cache Eviction Strategies: RANDOM, FIFO, LRU, LFU
- Resources are released when demand drops

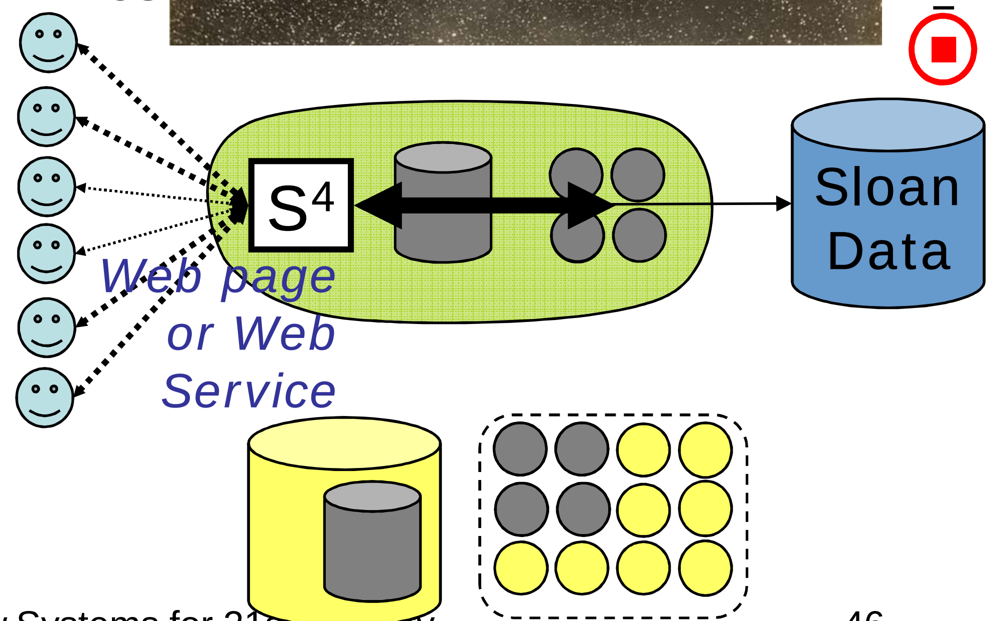


AstroPortal Stacking Service

- Purpose
  - On-demand "stacks" of random locations within ~10TB dataset
- Challenge
  - Rapid access to 10-10K "random" files
  - Time-varying load
- Sample Workloads

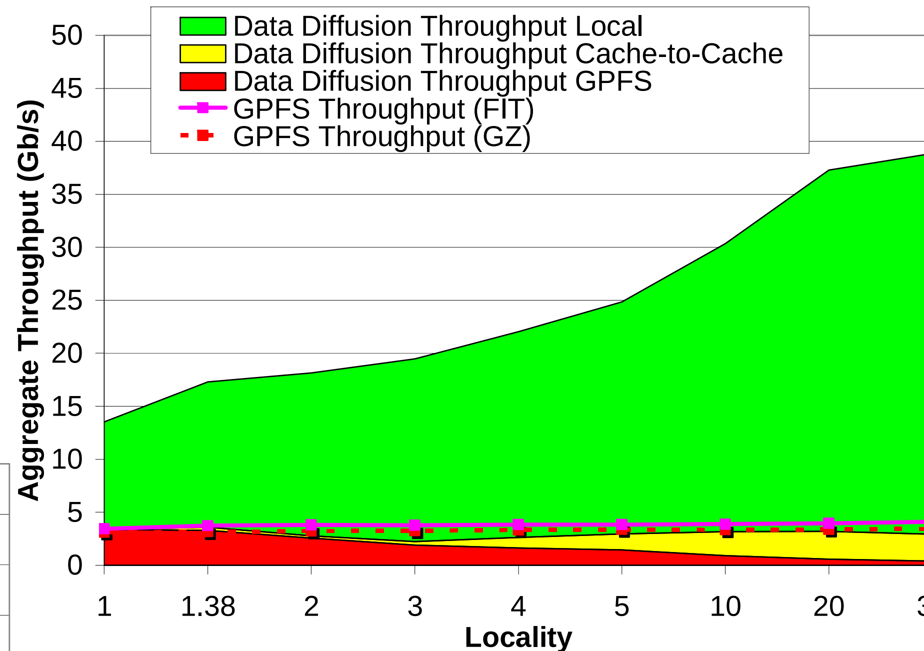
Locality	Number of Objects	Number of Files
1	111700	111700
1.38	154345	111699
2	97999	49000
3	88857	29620
4	76575	19145
5	60590	12120
10	46480	4650
20	40460	2025
30	23695	790

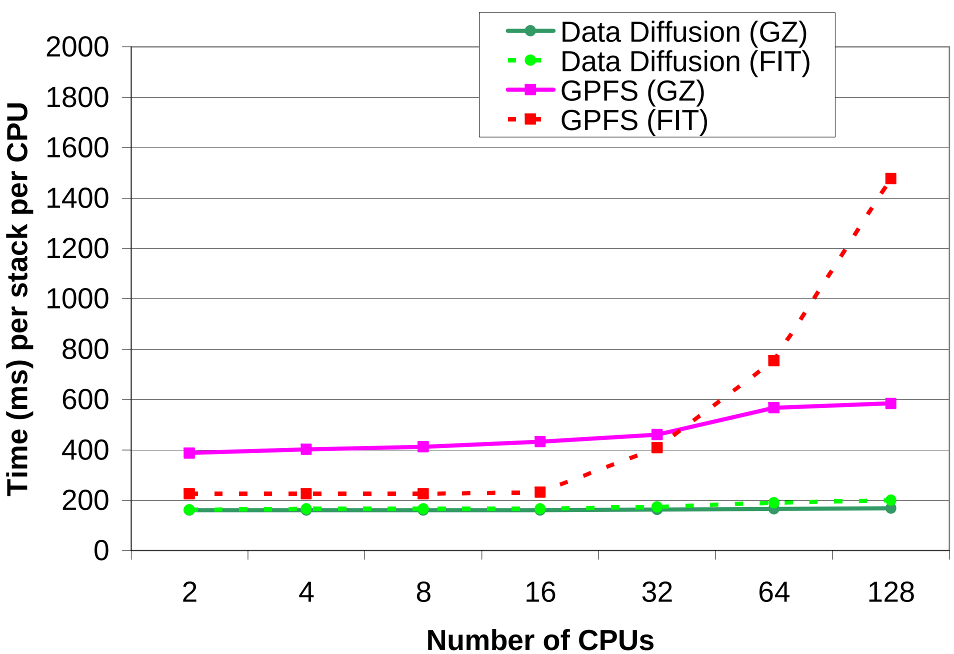




AstroPortal Stacking Service with Data Diffusion

- Aggregate throughput:
  - 39Gb/s
  - 10X higher than GPFS
- Reduced load on GPFS
  - 0.49Gb/s
  - 1/10 of the original load

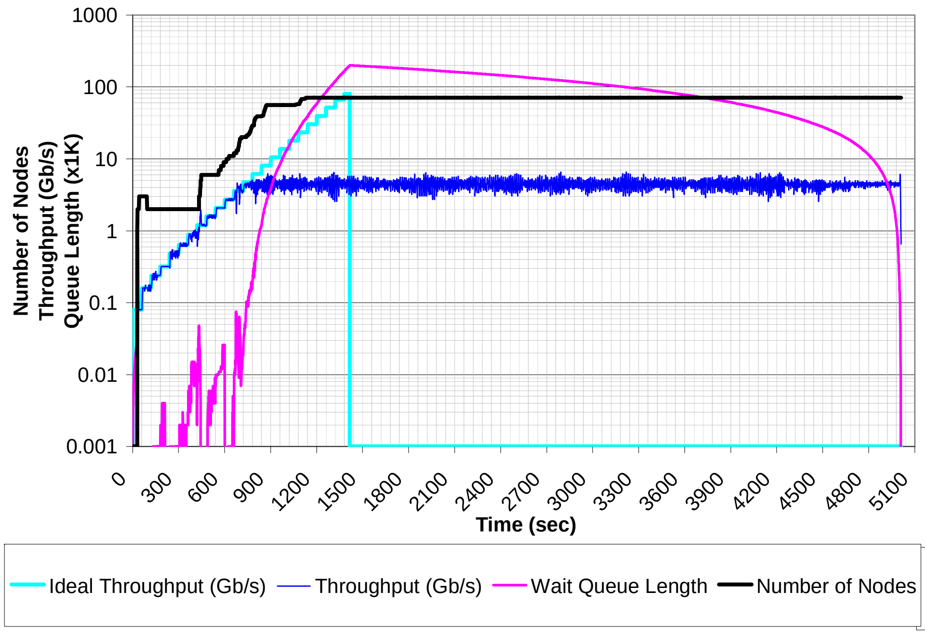


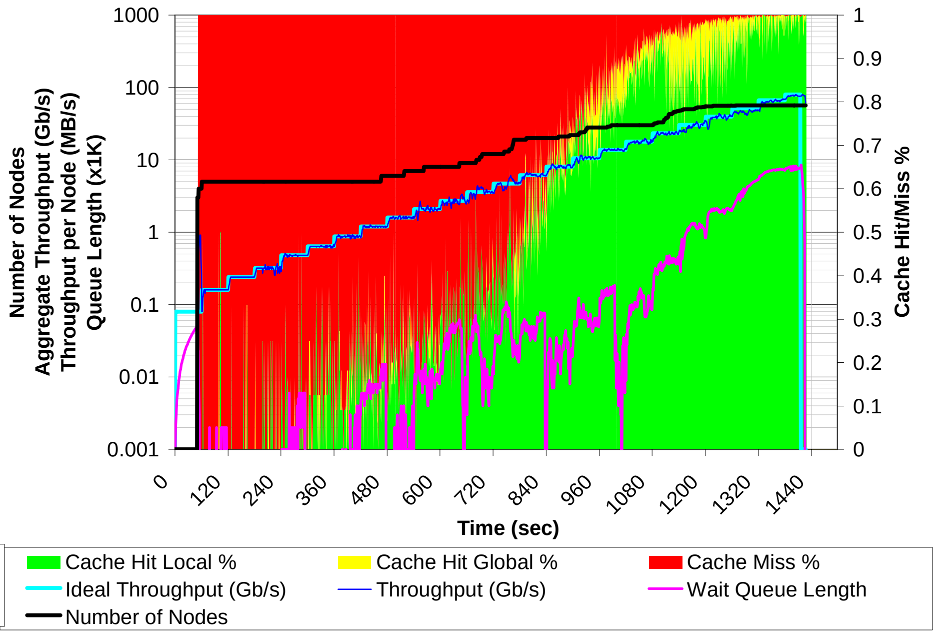


← High data locality
- Near perfect scalability

Data Diffusion: Data-Intensive Workload

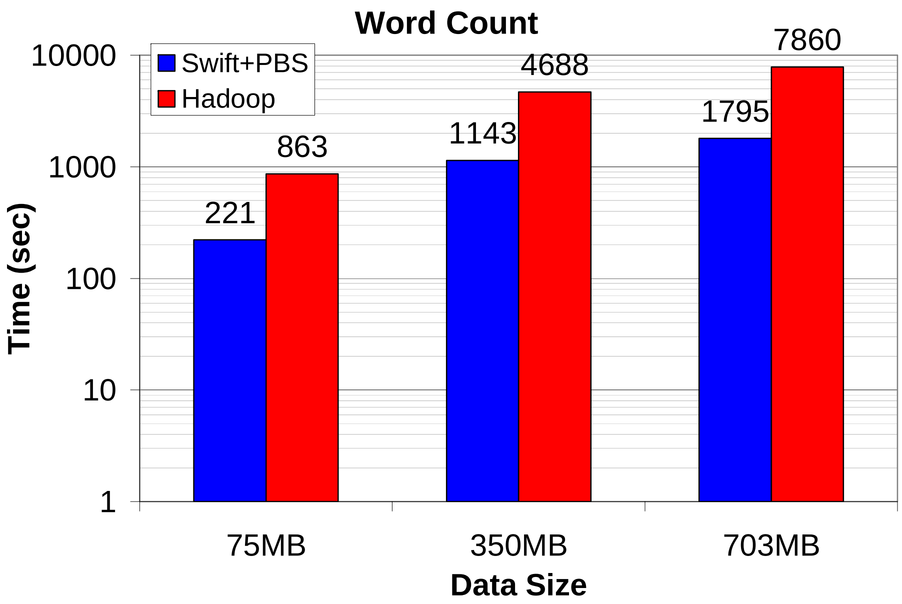
- 250K tasks on 128 processors
  - 10MB read, 10ms compute
- Comparing GPFS with data diffusion
  - 5011 sec vs. 1427 sec (ideal is 1415 sec)

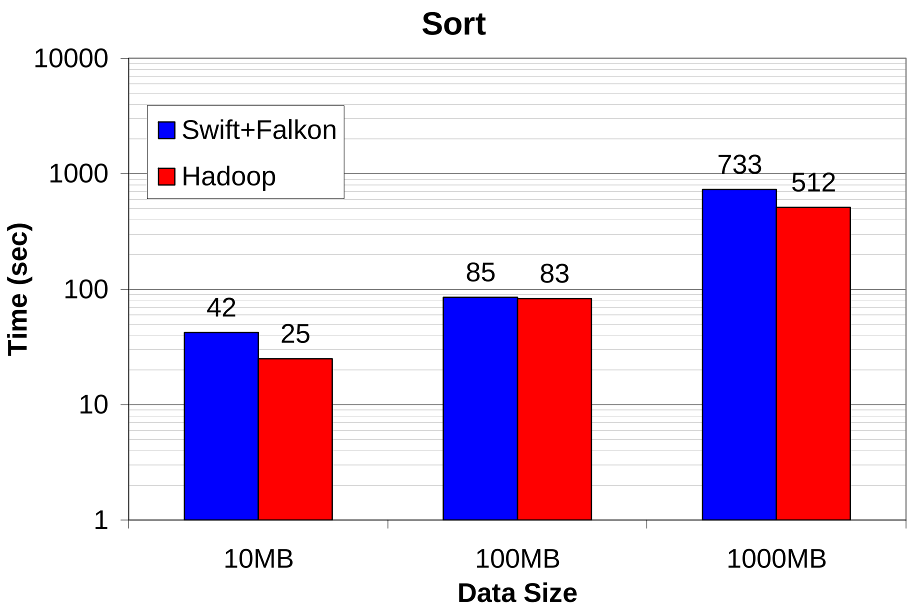




Hadoop vs. Swift

- Classic benchmarks for MapReduce
  - Word Count
  - Sort
- Swift performs similar or better than Hadoop (on 32 processors)





Mythbusting

- ~~Embarrassingly~~ Happily parallel apps are trivial to run
  - Logistical problems can be tremendous
- Loosely coupled apps do not require “supercomputers”
  - Total computational requirements can be enormous
  - Individual tasks may be tightly coupled
  - Workloads frequently involve large amounts of I/O
  - Make use of idle resources from “supercomputers” via backfilling
  - Costs to run “supercomputers” per FLOP is among the best
    - BG/P: 0.35 gigaflops/watt **(higher is better)**
    - SiCortex: 0.32 gigaflops/watt
    - BG/L: 0.23 gigaflops/watt
    - x86-based HPC systems: an order of magnitude lower
- Loosely coupled apps do not require specialized system software
- Shared file systems are good for all applications
  - They don’t scale proportionally with the compute resources
  - Data intensive applications don’t perform and scale well

Features Scientific Workflow Systems should Have!

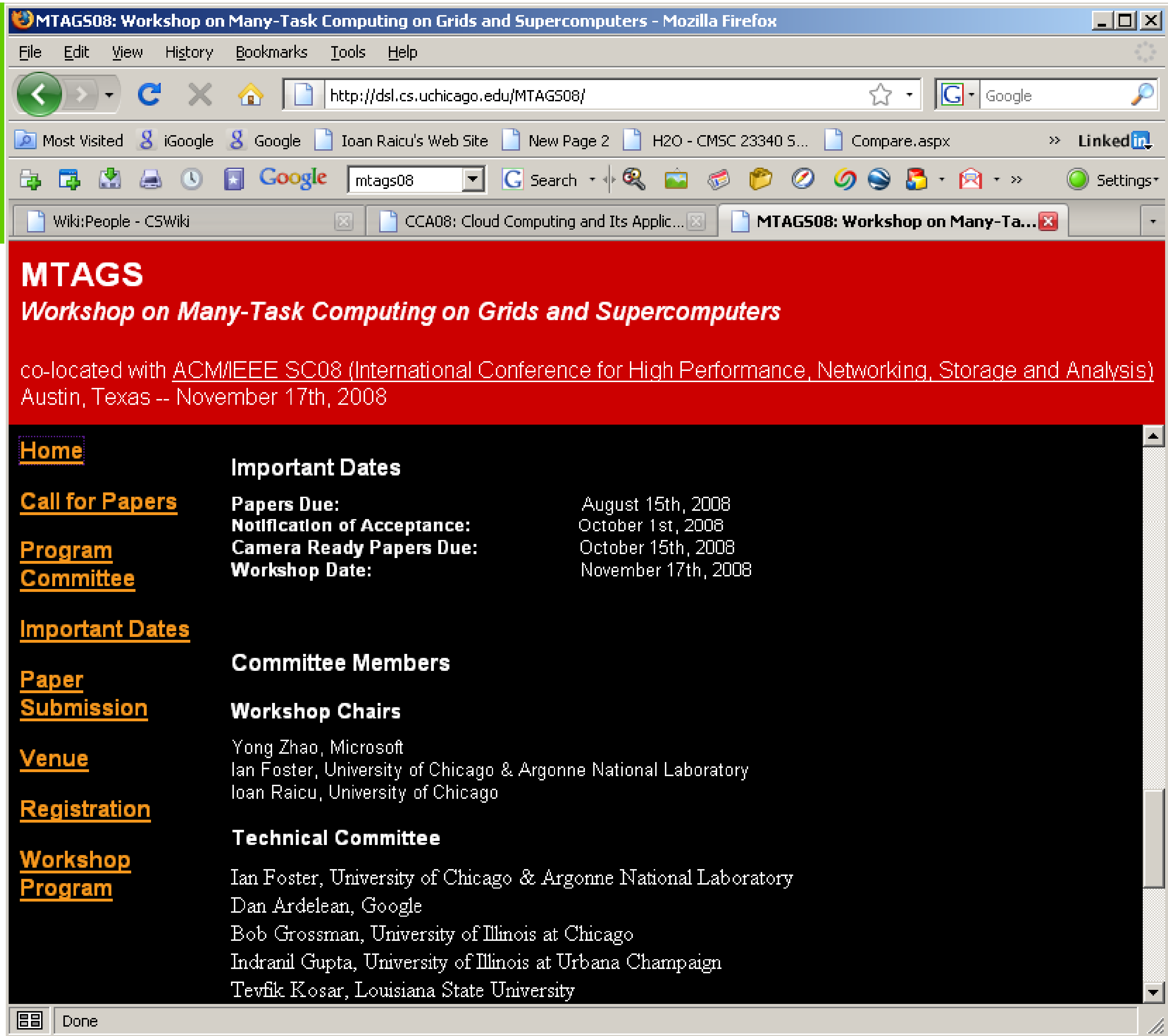
- **Parallelism**
  - Support for both explicit and implicit parallelism
- **Performance and Scalability**
  - Million to billions of tasks
  - Handle 100s~1000s of tasks/sec
- **Data management**
  - Reduce reliance on shared file systems
  - Scale with processing power
  - Data-aware scheduling
- **Reliability**
  - Self healing
  - Efficient and scalable monitoring
- **Provenance**

Solutions
(we have experience with)

- Falkon
  - A Fast and Light-weight tasK executiON framework
  - Globus Incubator Project
  - http://dev.globus.org/wiki/Incubator/Falkon
- Swift
  - Parallel programming tool for rapid and reliable specification, execution, and management of large-scale science workflows
  - http://www.ci.uchicago.edu/swift/index.php
- Environments:
  - *Clusters*: TeraPort (TP)
  - *Grids*: Open Science Grid (OSG), TeraGrid (TG)
  - *Specialized large machines*: SiCortex 5732
  - *Supercomputers*: IBM BlueGene/P (BG/P)

More Information

- More information:
  - Personal research page: http://people.cs.uchicago.edu/~iraicu/
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - Swift: http://www.ci.uchicago.edu/swift/index.php
- Collaborators:
  - Ian Foster, The University of Chicago & Argonne National Laboratory
  - Alex Szalay, The Johns Hopkins University
  - Rick Stevens, The University of Chicago & Argonne National Laboratory
  - Yong Zhao, Microsoft
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Catalin Dumitrescu, Fermi National Laboratory
  - Zhao Zhang, The University of Chicago
  - Jerry C. Yan, NASA, Ames Research Center
  - Kamil Iskra, Argonne National Laboratory
  - Pete Beckman, Argonne National Laboratory
  - Mihael Hategan, The University of Chicago
  - Ben Clifford, The University of Chicago
  - Veronika Nefedova, Argonne National Laboratory
  - Tiberiu Stef-Praun, The University of Chicago
  - Daniela Stan Raicu, DePaul University
  - Gabriela Turcu, The University of Chicago
  - Atilla S. Balkir, The University of Chicago
  - Jing Tie, The University of Chicago
  - Quan T. Pham, The University of Chicago
  - Sarah Kenny, The University of Chicago
  - Gregor von Laszewski, Rochester Institute of Technology
  - Jim Gray, Microsoft Research
  - Julian Bunn, California Institute of Technology


MTAGS08: Workshop on Many-Task Computing on Grids and Supercomputers - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dsl.cs.uchicago.edu/MTAGS08/
Most Visited iGoogle Google Ioan Raicu's Web Site New Page 2 H2O - CMSC 23340 S... Compare.aspx
Wiki:People - CSWiki
CCA08: Cloud Computing and Its Applic...
MTAGS08: Workshop on Many-Ta...
MTAGS
Workshop on Many-Task Computing on Grids and Supercomputers
co-located with ACM/IEEE SC08 (International Conference for High Performance, Networking, Storage and Analysis)
Austin, Texas -- November 17th, 2008
Home
Call for Papers
Program Committee
Important Dates
Paper Submission
Venue
Registration
Workshop Program
Important Dates
Papers Due: August 15th, 2008
Notification of Acceptance: October 1st, 2008
Camera Ready Papers Due: October 15th, 2008
Workshop Date: November 17th, 2008
Committee Members
Workshop Chairs
Yong Zhao, Microsoft
Ian Foster, University of Chicago & Argonne National Laboratory
Ioan Raicu, University of Chicago
Technical Committee
Ian Foster, University of Chicago & Argonne National Laboratory
Dan Ardelean, Google
Bob Grossman, University of Illinois at Chicago
Indranil Gupta, University of Illinois at Urbana Champaign
Tevfik Kosar, Louisiana State University
Done

Handling Megajobs BOF at SC08

- More and more people need to run thousands to millions of closely related jobs that are associated with individual projects. Scientists seek convenient means to specify and manage many jobs, arranging inputs, aggregating outputs, identifying successful and failed jobs and repairing failures. System administrators seek methods to process extraordinary numbers of jobs for multiple users without overwhelming queuing systems or disrupting fair-share usage policies. And, grid developers are producing a new generation of queuing and scheduling systems as well as auxiliary systems for use with existing queuing and scheduling systems. This Birds-of-feather session provides a venue for the exchange of information about processing large numbers of jobs. Short presentations of an invited sample of projects will be followed by discussion.
- For more information, contact:
  - Marlon Pierce: mpierce@cs.indiana.edu
  - Dick Repasky: rrepasky@indiana.edu
  - Ioan Raicu: iraicu@cs.uchicago.edu