THE UNIVERSITY OF CHICAGO

Scientific Workflow Systems for 21st Century, New Bottle or New Wine?

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Collaborators:
Yong Zhao, Microsoft Corporation
Ian Foster, University of Chicago and Argonne National Laboratory

Argonne NATIONAL LABORATORY

Scientific Workflows (SWF 2008)
July 8th, 2008

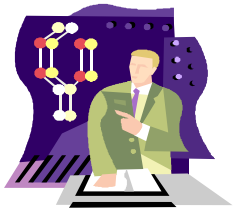
Workflow Systems

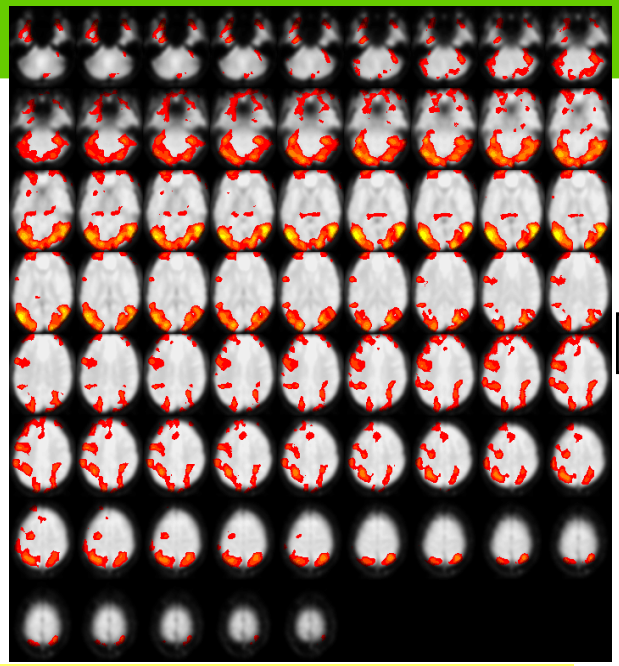
Describes computation components, ports and channels

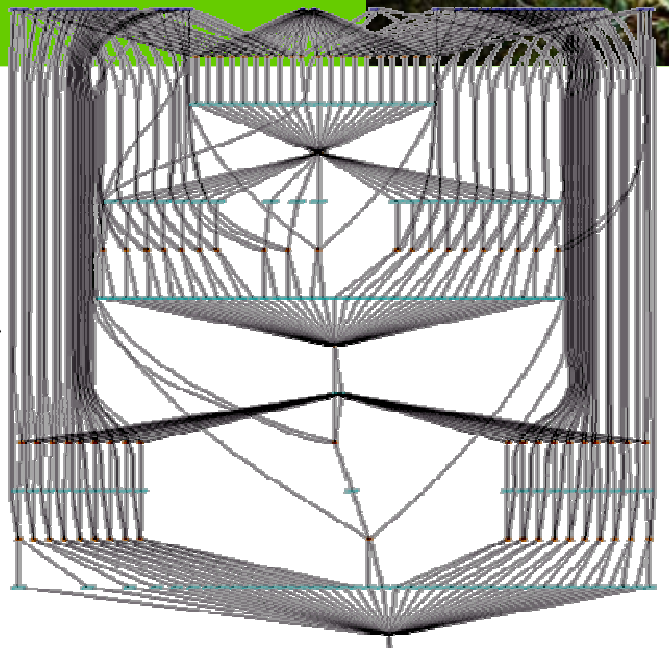
Describes data and event flow

Coordinate the execution of the components

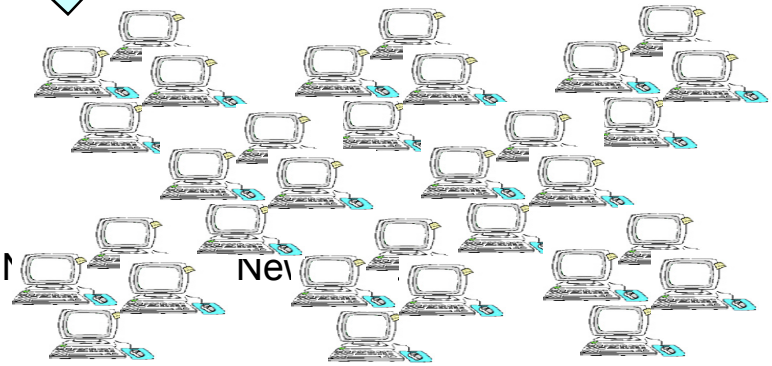
Functional MRI (fMRI)

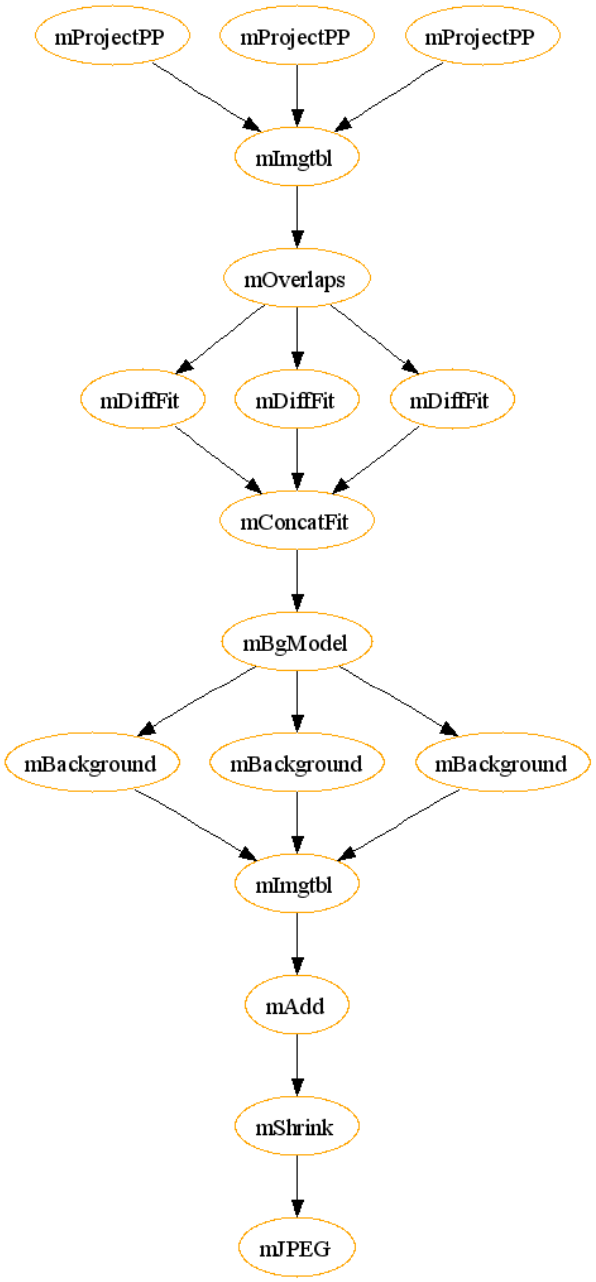






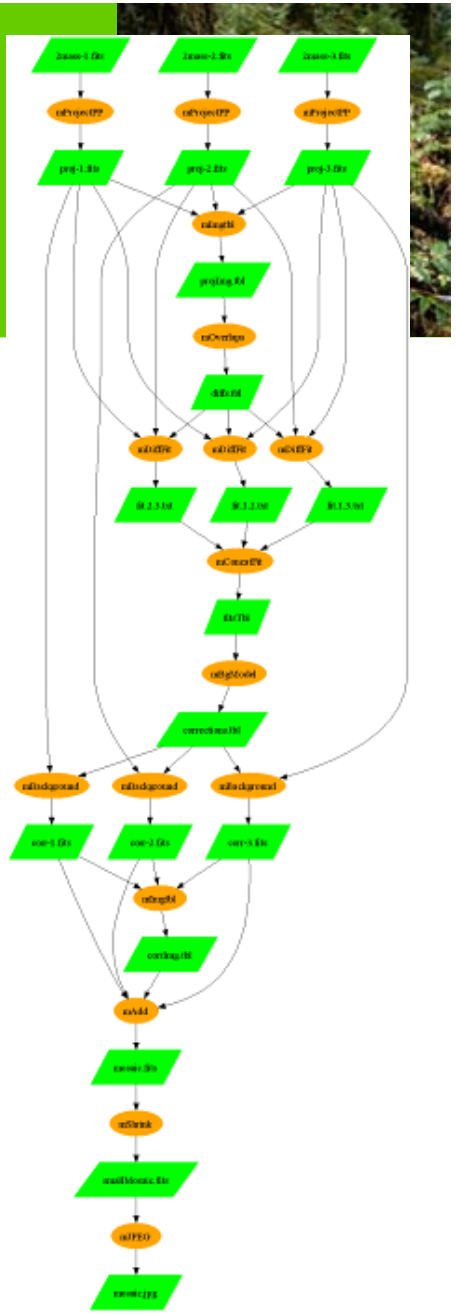
- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies





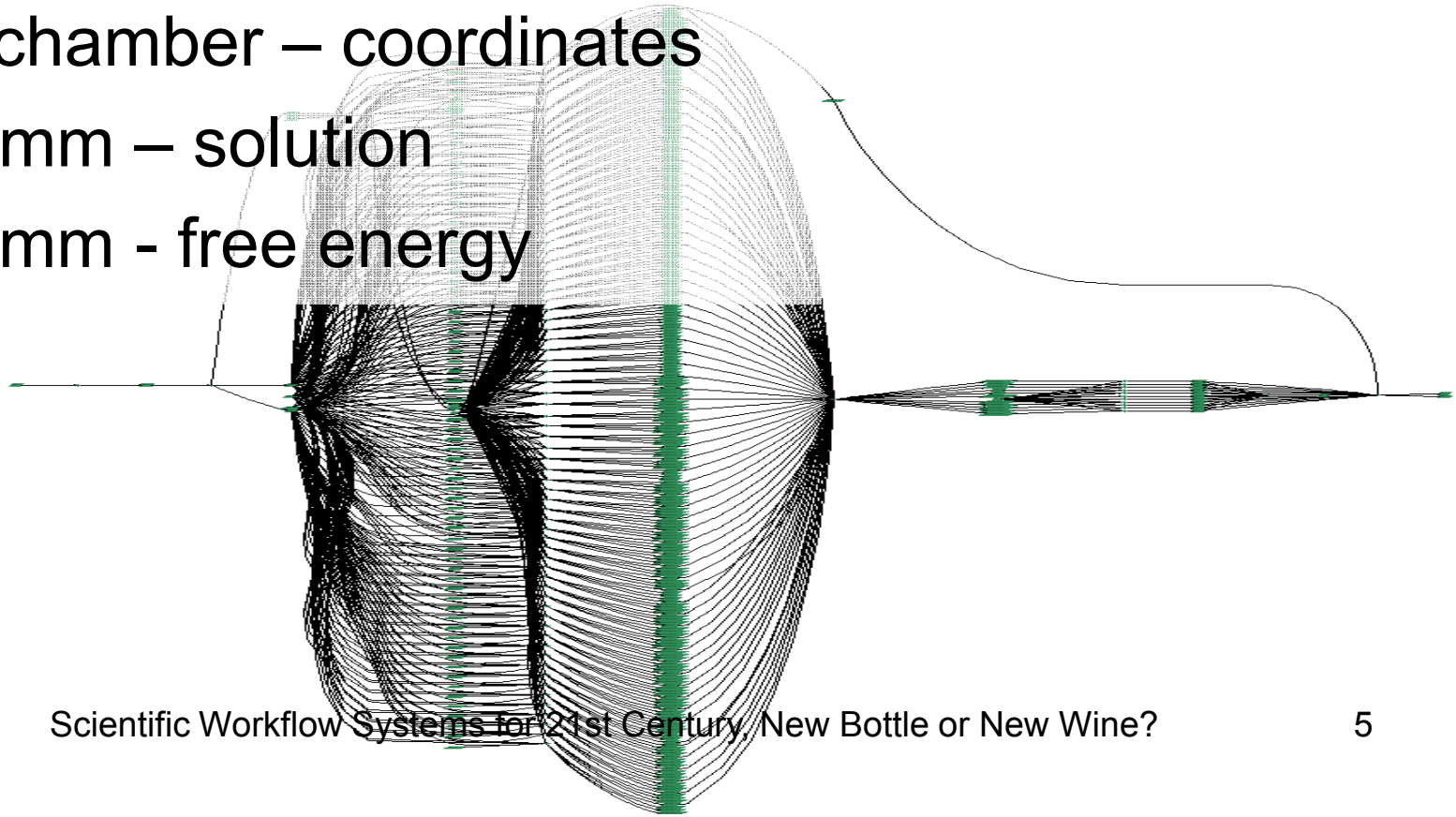
Montage

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

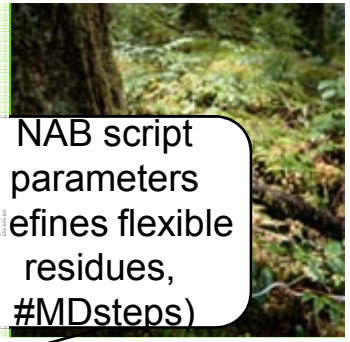


Molecular Dynamics

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



Many Many Tasks: Identifying Potential Drug Targets



PDB protein descriptions

1 protein (1MB)

2M structures (6 GB)

DOCK6 Receptor (1 per protein: defines pocket to bind to)

FRED Receptor (1 per protein: defines pocket to bind to)

NAB script parameters defines flexible residues, #MDsteps)

BuildNABScript

NAB Script

Amber prep:
2. AmberizeReceptor
4. perl: gen nabscript

start

FRED

DOCK6

~4M x 60s x 1 cpu
~60K cpu-hrs

Select best ~5K

Select best ~5K

Amber

~10K x 20m x 1 cpu
~3K cpu-hrs

Amber Score:
1. AmberizeLigand
3. AmberizeComplex
5. RunNABScript

Select best ~500

GCMC

~500 x 10hr x 100 cpu
~500K cpu-hrs

end

report

ligands

complexes

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

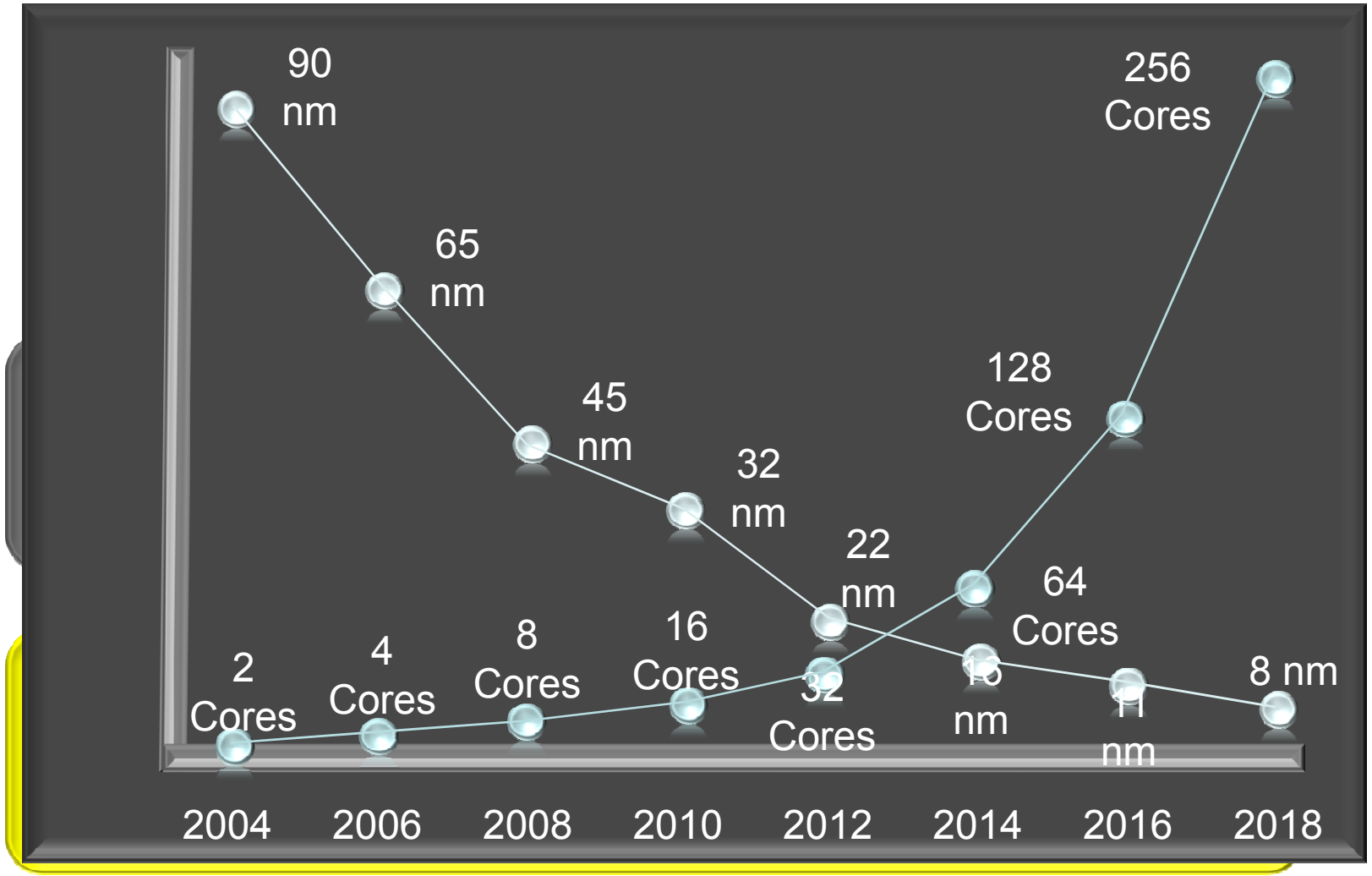
Characterizing Scientific Workflows

- 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

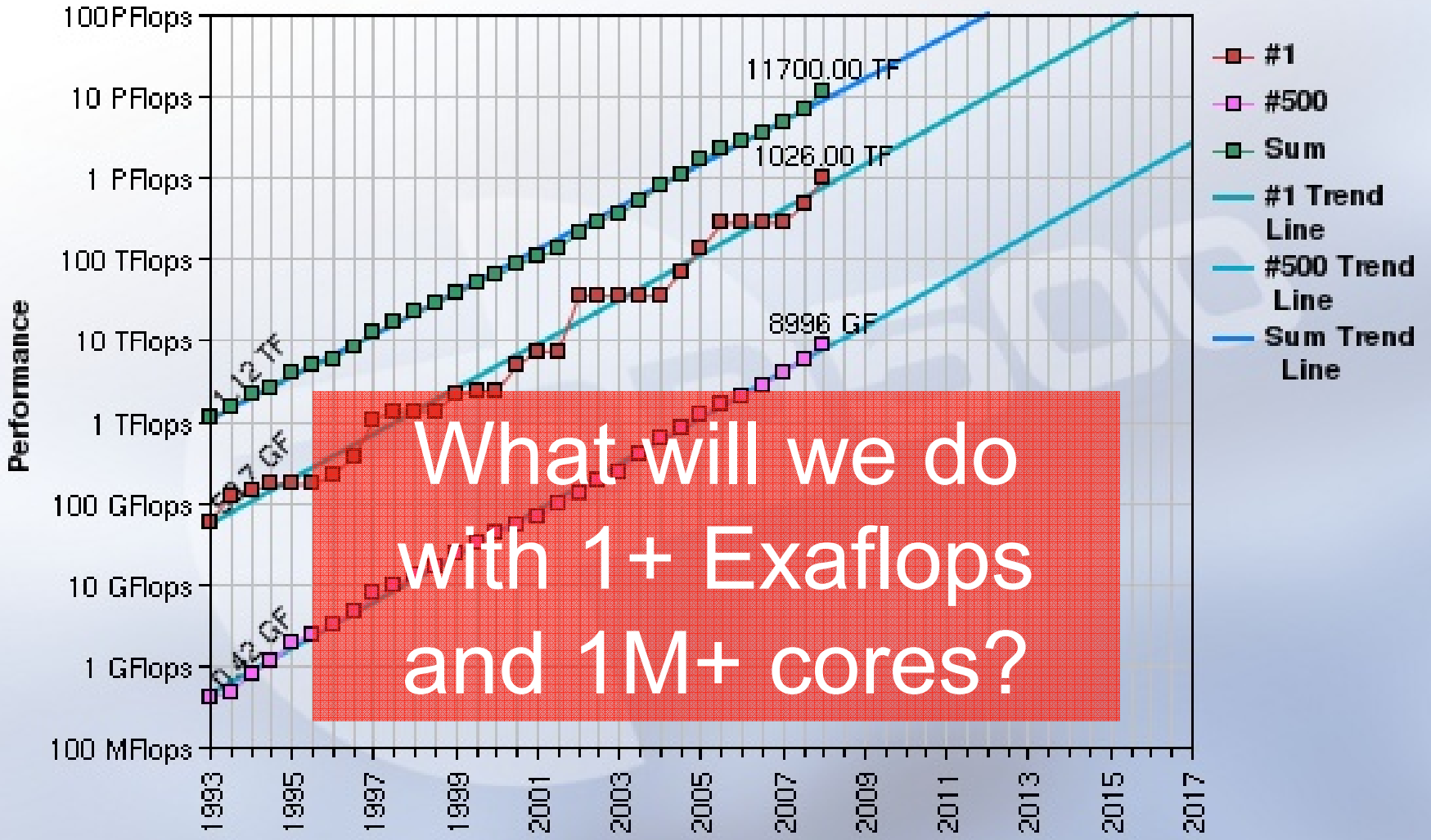
Characterizing Scientific Applications

- 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

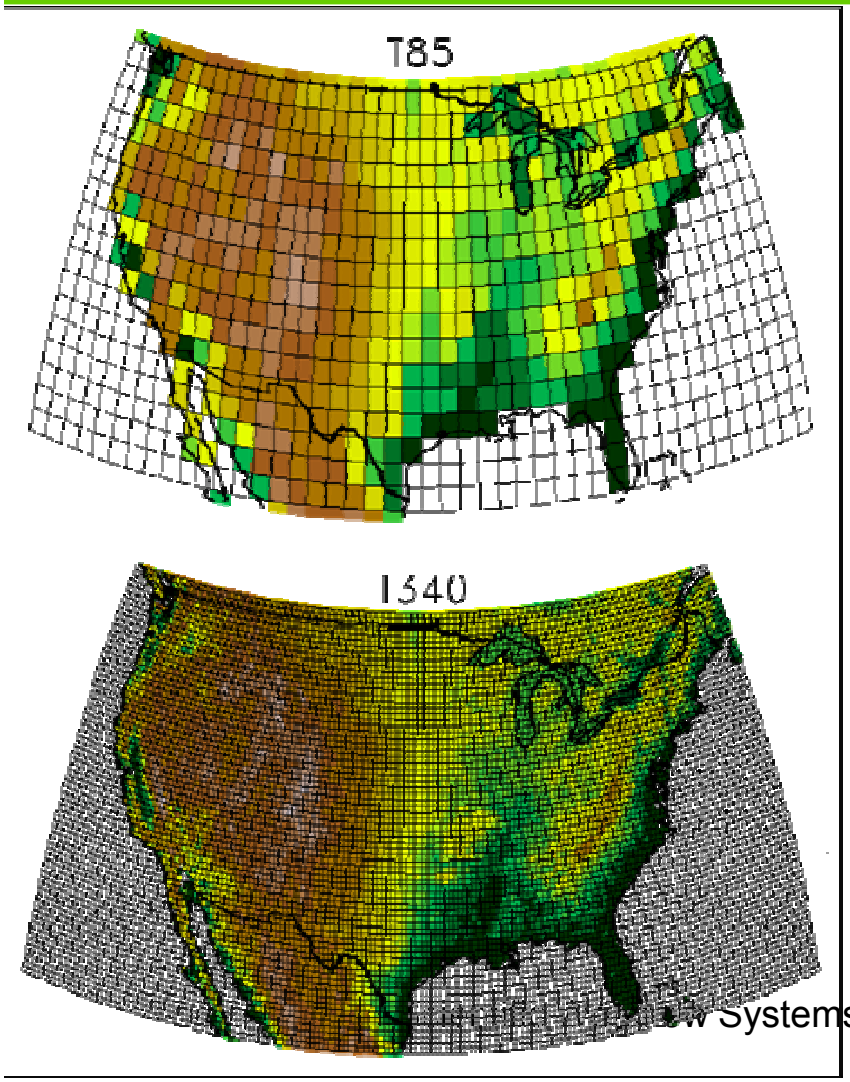


Projected Performance Development



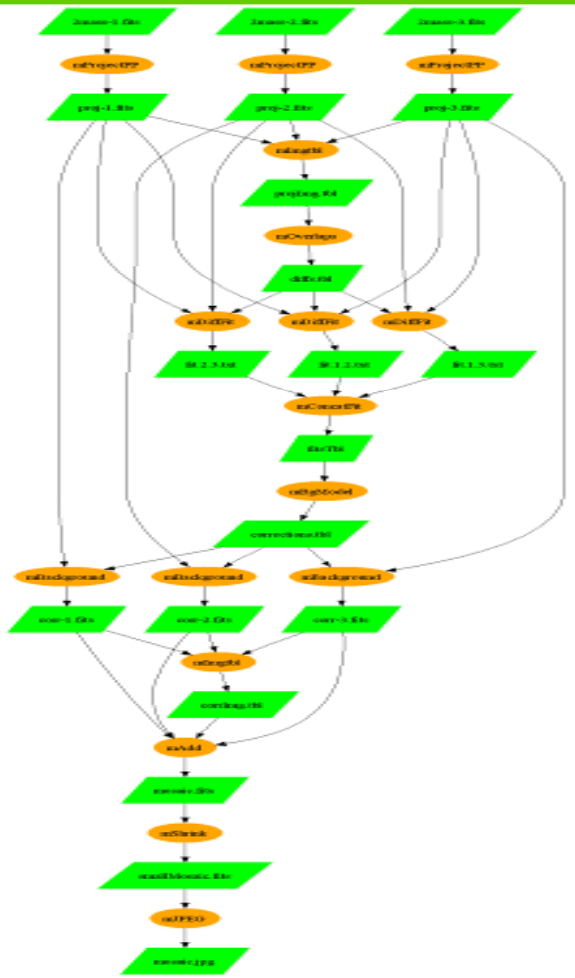
What will we do with 1+ Exaflops and 1M+ cores?

1) Tackle Bigger and Bigger Problems



Computational Scientist as **Hero**

2) Tackle Increasingly Complex Problems



Computational Scientist as **Logistics Officer**

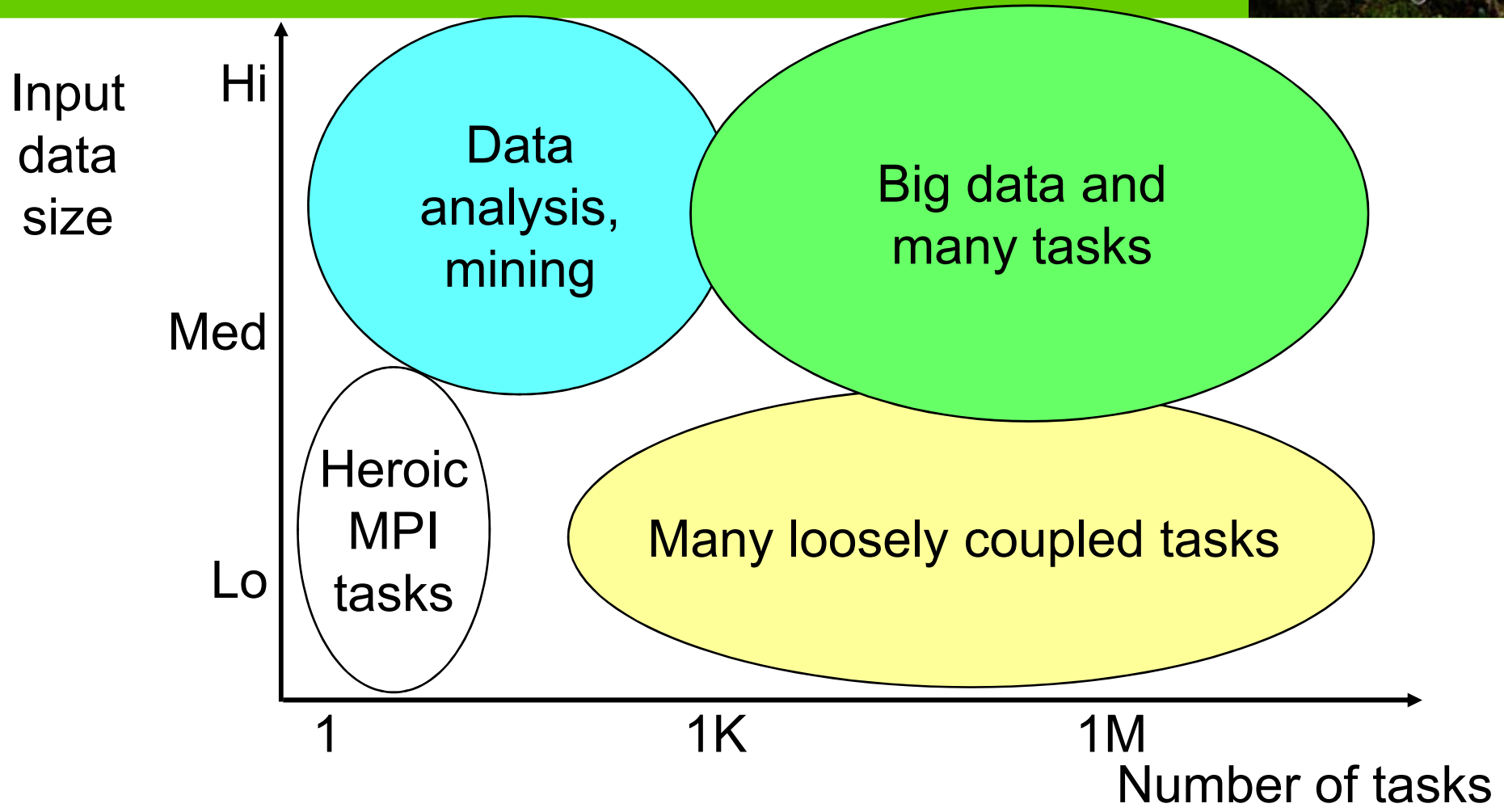
“More Complex Problems”

- 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., etc., 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)

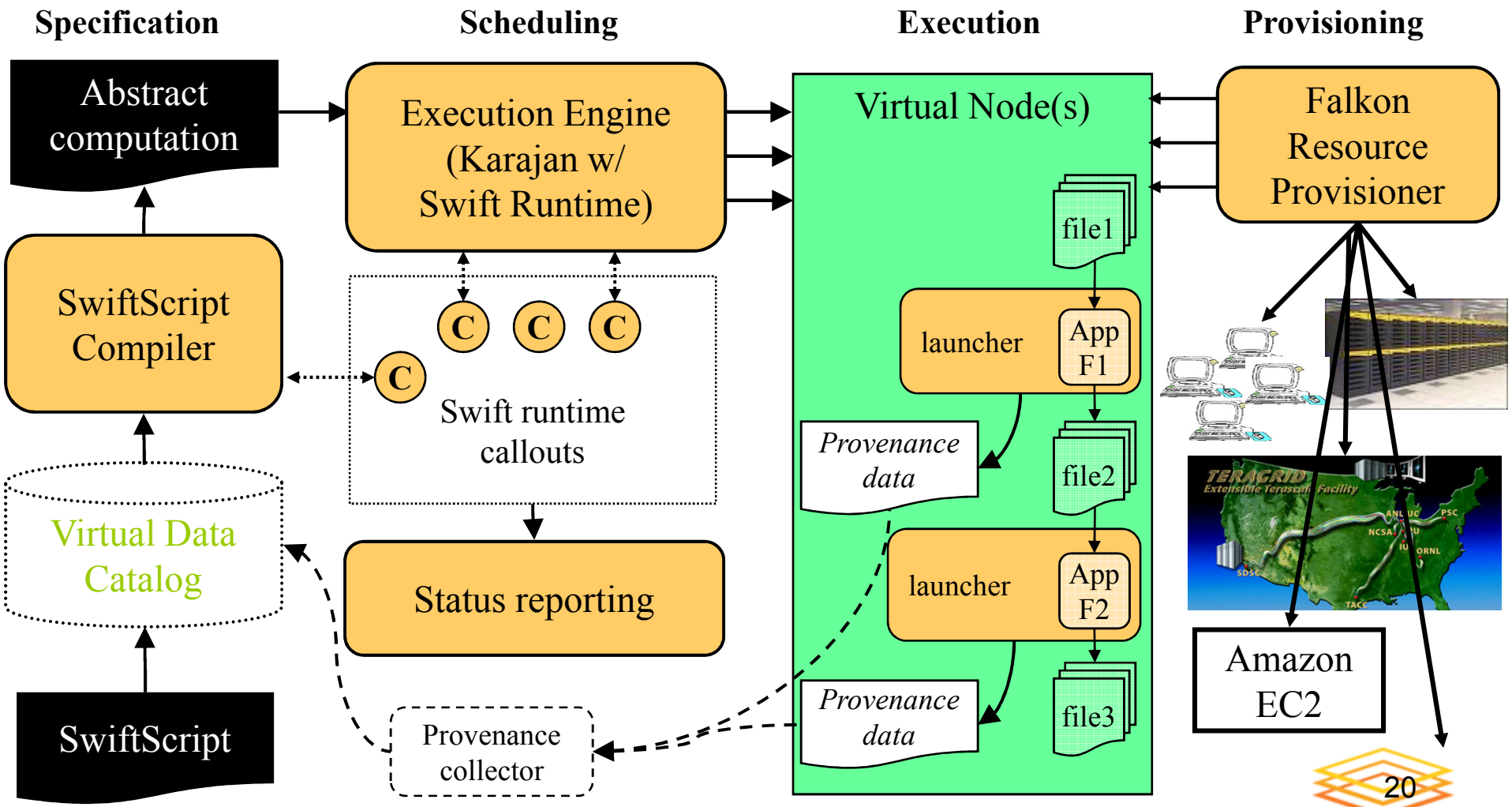
Major Challenges to Large Scale Scientific Computation

- 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
- 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

Existing and emerging workflow technologies

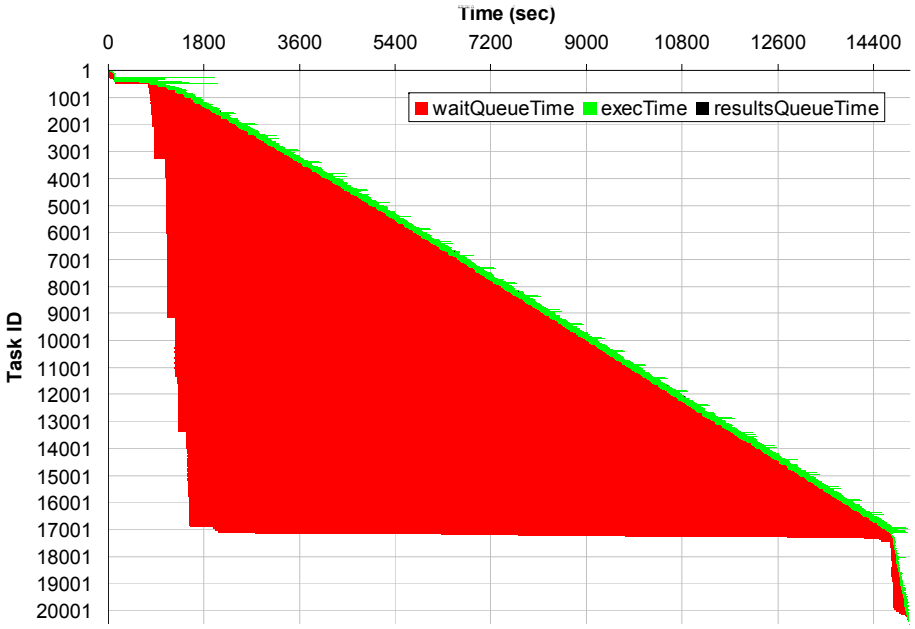
	Swift	BPEL	XPDL	MS Workflow Fundation	DAGMan	Tavena	Triana	Kepler	Mapreduce	Star-P
Scales to Grids	++	-	-	-	++	-	-	-	+	+
Typing	++	++	++	++	-	-	-	+	-	+
Iteration	++	-/+	-	+	-	-	-	+	+	+
Scripting	++	-	-	+	+	+	-	-	-	++
Dataset Mapping	+	-	-	-	-	-	-	-	-	-
Service Interop	+	-	+	-	-	-	-	+	-	-
Subflow/comp.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

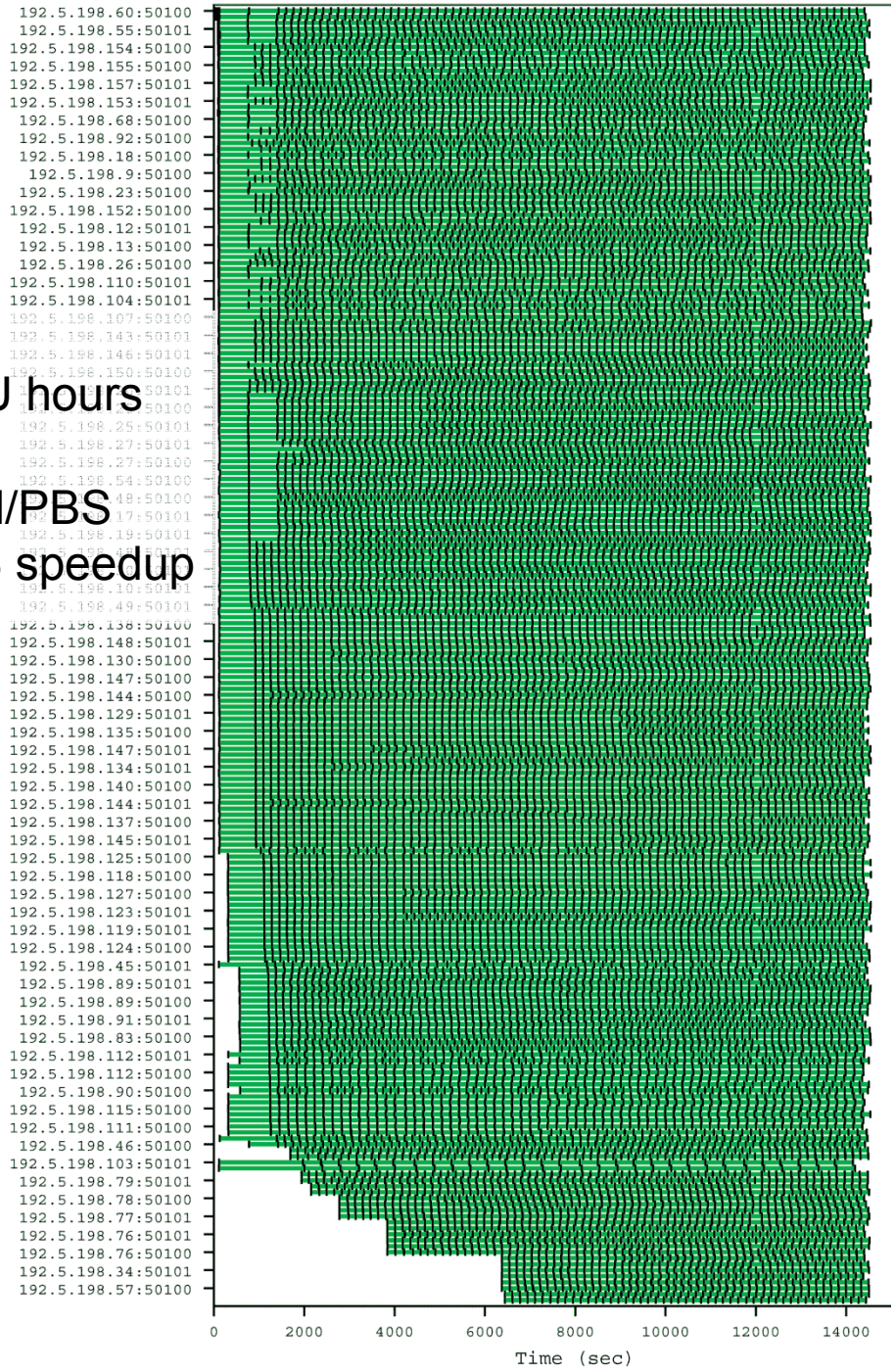
Swift Architecture



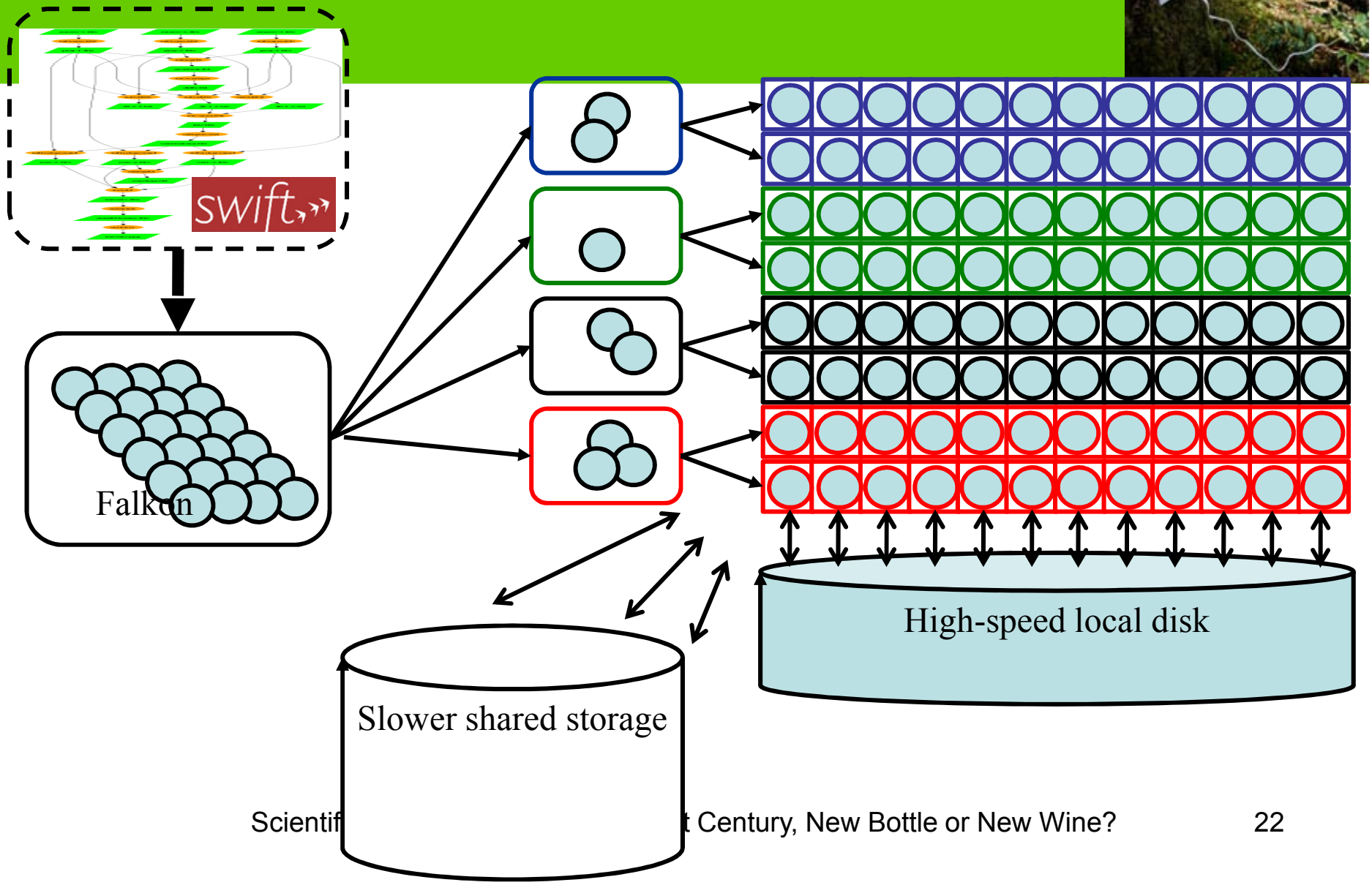
MolDyn Application

- 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





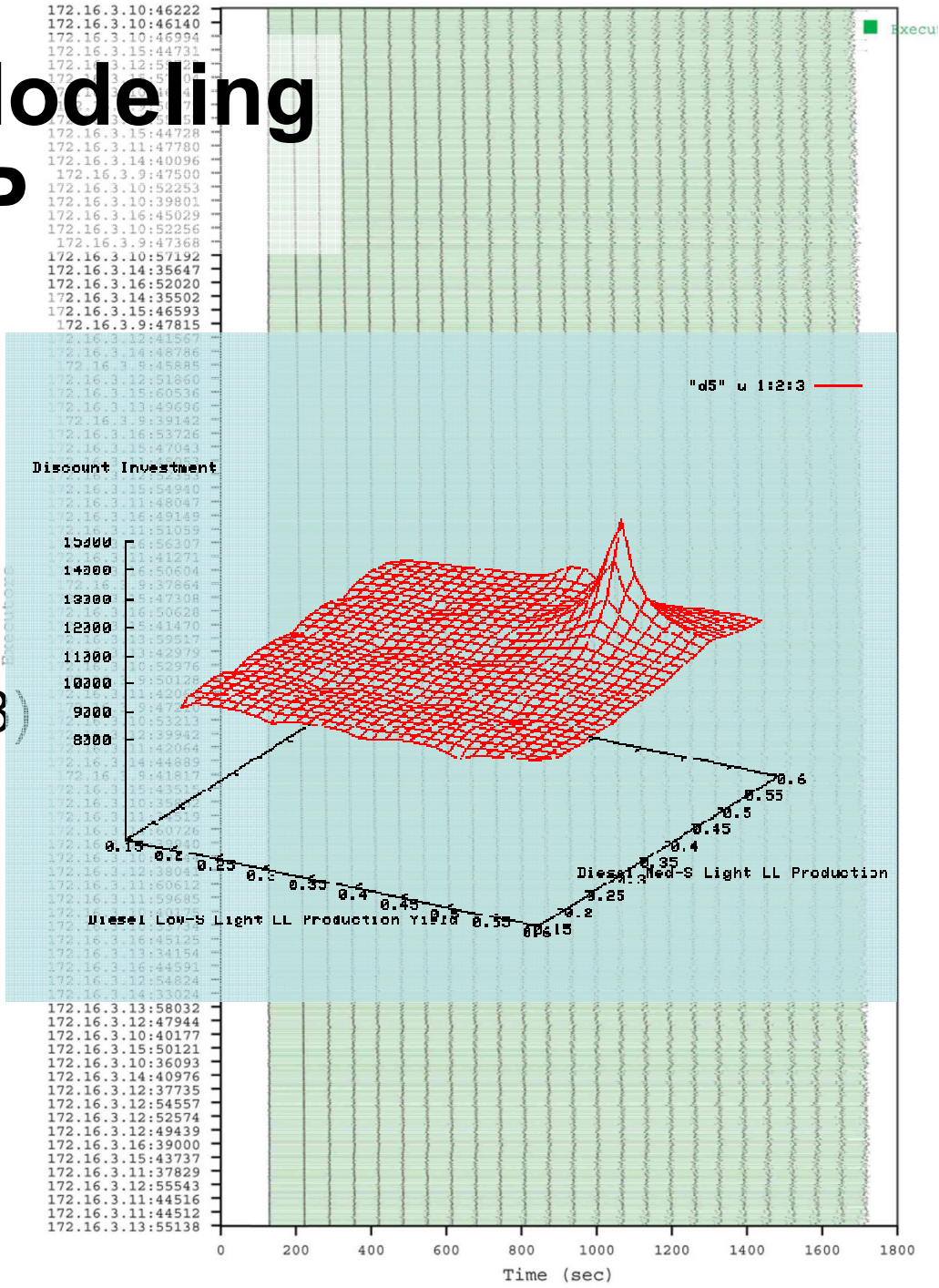
Managing 120K CPUs



MARS Economic Modeling on IBM BG/P

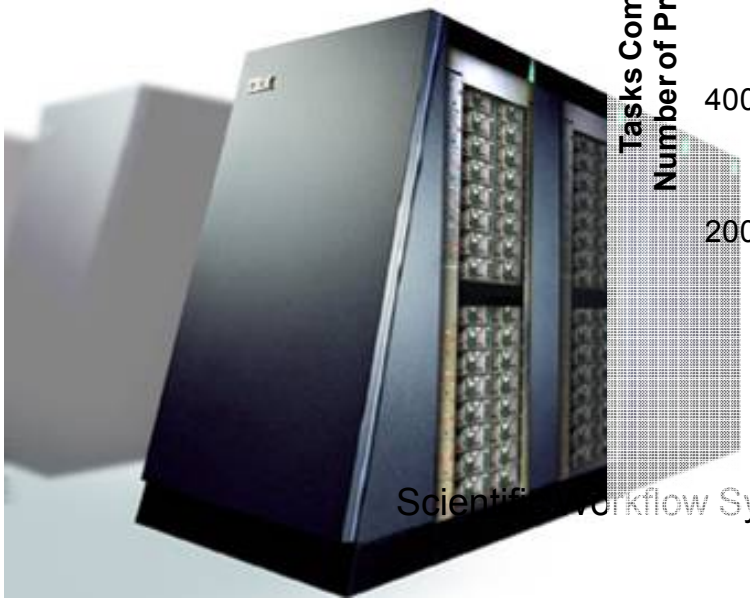
- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%

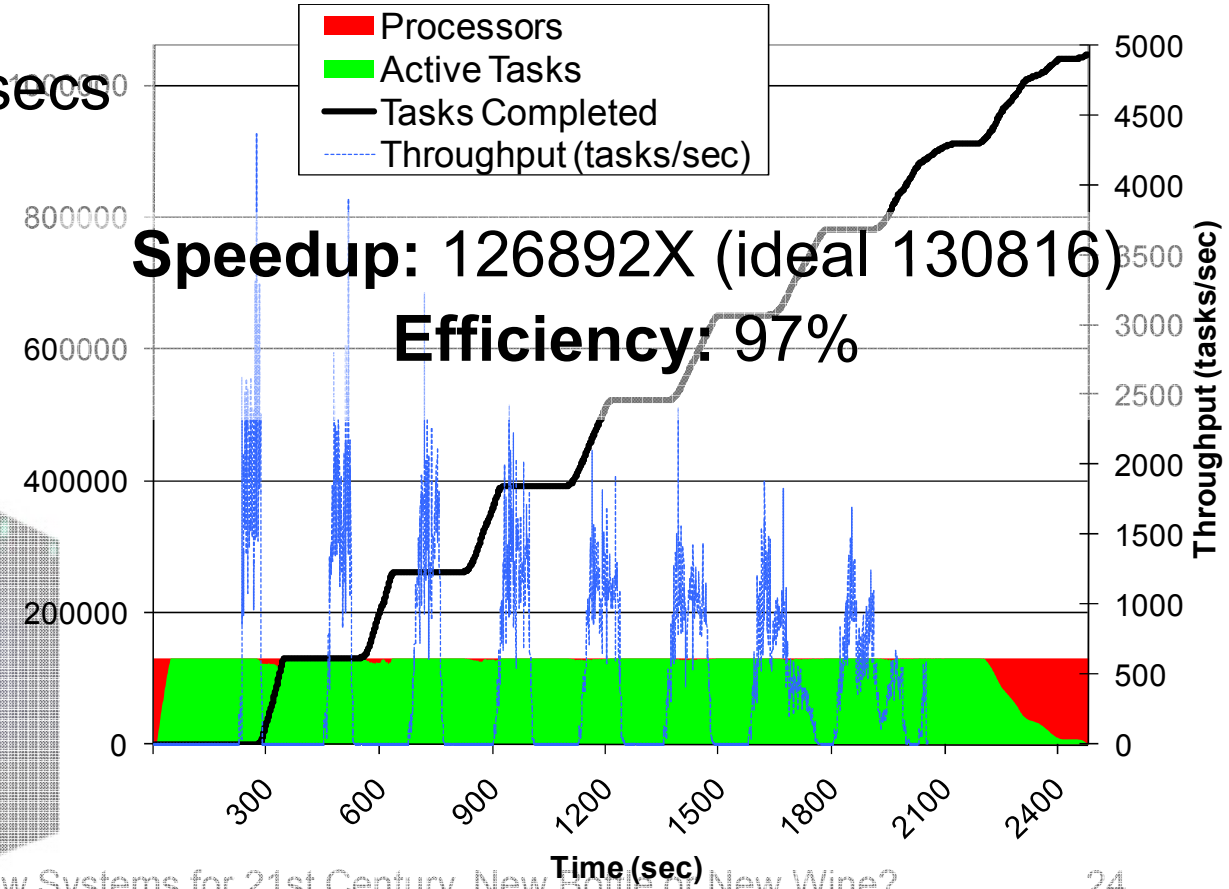
ystems

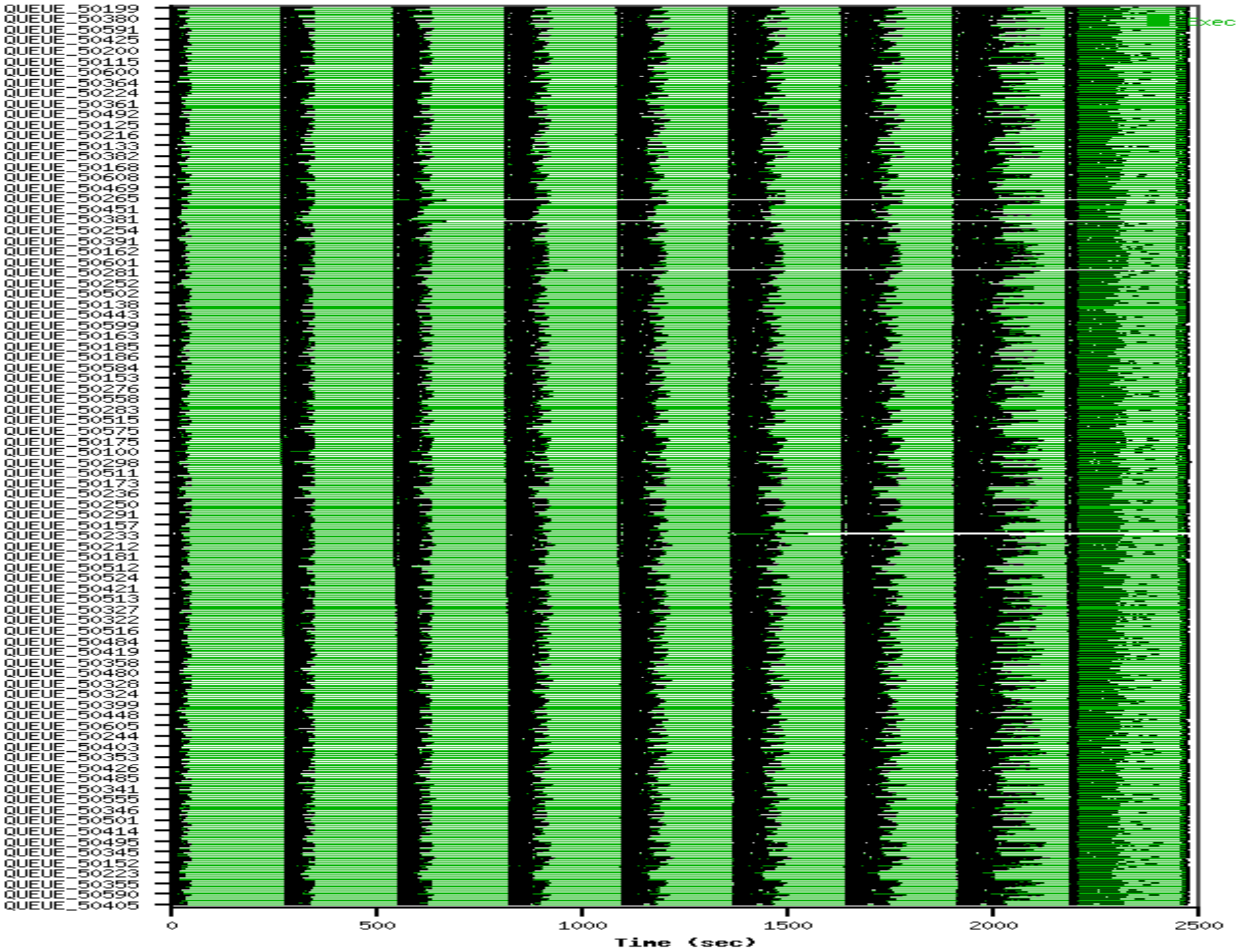


MARS Economic Modeling on IBM BG/P (128K CPUs)

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

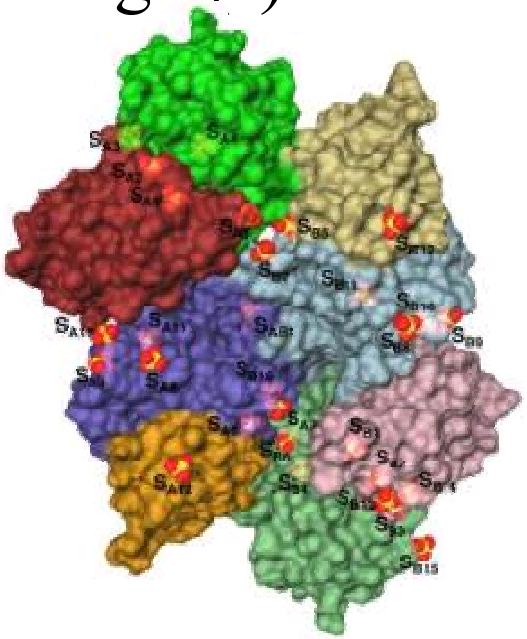


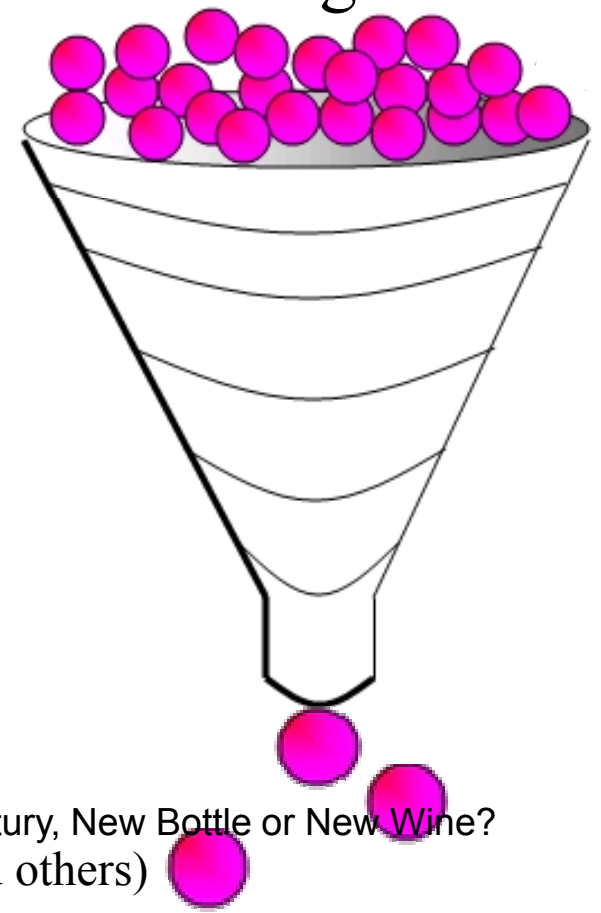



QUEUE_50199
QUEUE_50380
QUEUE_50591
QUEUE_50425
QUEUE_50200
QUEUE_50115
QUEUE_50600
QUEUE_50364
QUEUE_50224
QUEUE_50361
QUEUE_50492
QUEUE_50125
QUEUE_50216
QUEUE_50133
QUEUE_50382
QUEUE_50168
QUEUE_50608
QUEUE_50469
QUEUE_50265
QUEUE_50451
QUEUE_50381
QUEUE_50254
QUEUE_50391
QUEUE_50162
QUEUE_50601
QUEUE_50281
QUEUE_50252
QUEUE_50502
QUEUE_50138
QUEUE_50443
QUEUE_50599
QUEUE_50163
QUEUE_50185
QUEUE_50186
QUEUE_50584
QUEUE_50153
QUEUE_50276
QUEUE_50558
QUEUE_50283
QUEUE_50515
QUEUE_50575
QUEUE_50175
QUEUE_50100
QUEUE_50298
QUEUE_50511
QUEUE_50173
QUEUE_50236
QUEUE_50250
QUEUE_50291
QUEUE_50157
QUEUE_50233
QUEUE_50212
QUEUE_50181
QUEUE_50512
QUEUE_50524
QUEUE_50421
QUEUE_50513
QUEUE_50327
QUEUE_50322
QUEUE_50516
QUEUE_50484
QUEUE_50419
QUEUE_50358
QUEUE_50480
QUEUE_50328
QUEUE_50324
QUEUE_50399
QUEUE_50448
QUEUE_50605
QUEUE_50244
QUEUE_50403
QUEUE_50353
QUEUE_50426
QUEUE_50485
QUEUE_50341
QUEUE_50555
QUEUE_50346
QUEUE_50501
QUEUE_50414
QUEUE_50495
QUEUE_50345
QUEUE_50152
QUEUE_50223
QUEUE_50355
QUEUE_50590
QUEUE_50405
Exec
0
500
1000
1500
2000
2500
Time (sec)

Many Many Tasks: Identifying Potential Drug Targets

Protein target(s) x 2M+ ligands

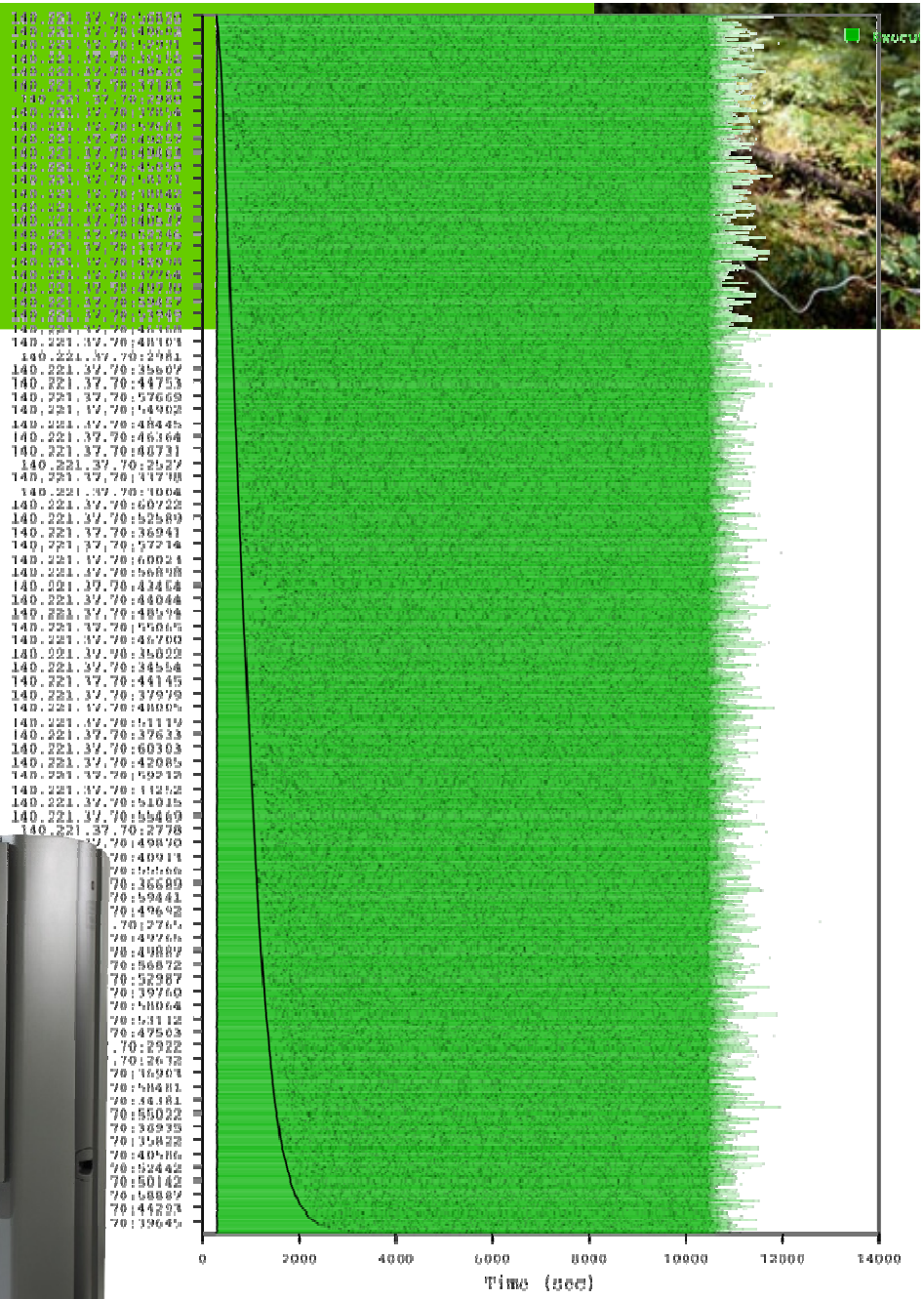




(Mike Kubal, Benoit Roux, and others)

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

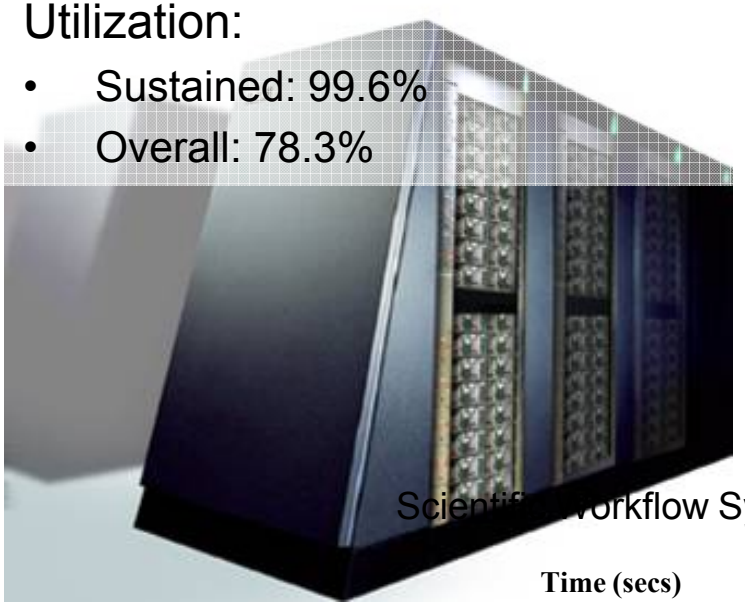
Average task time: 667 sec

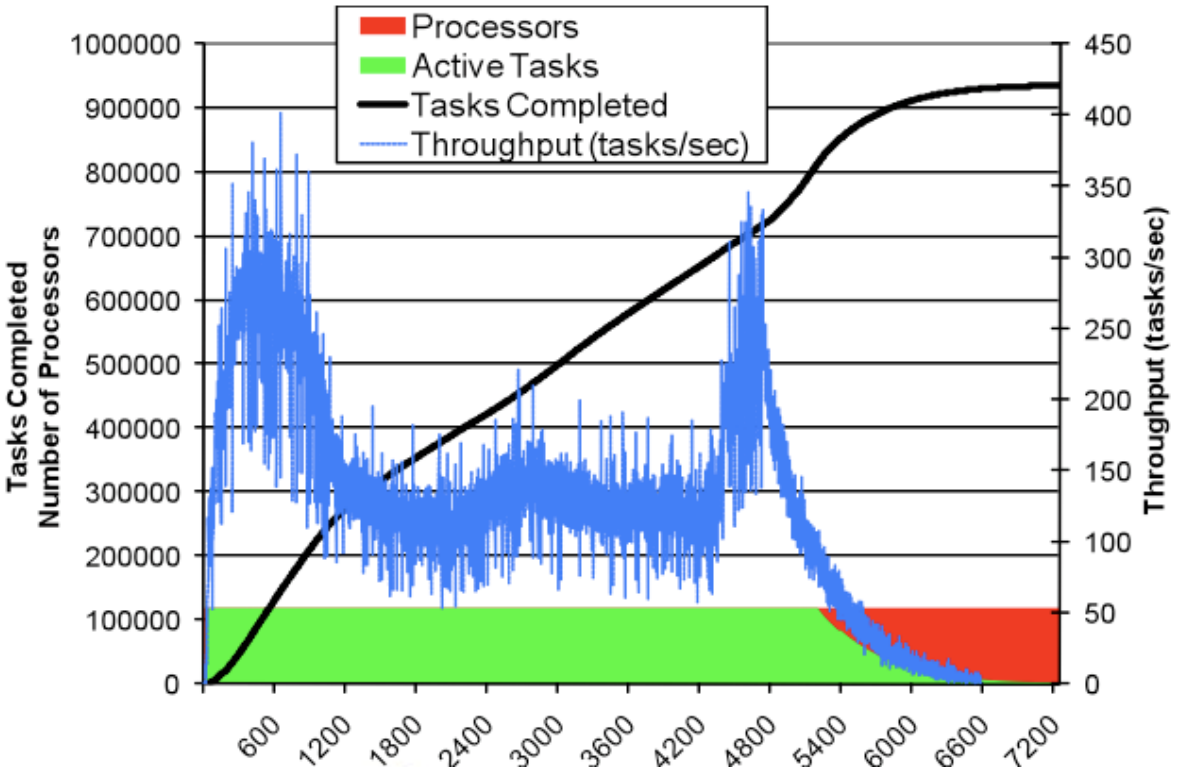
Relative Efficiency: 99.7%

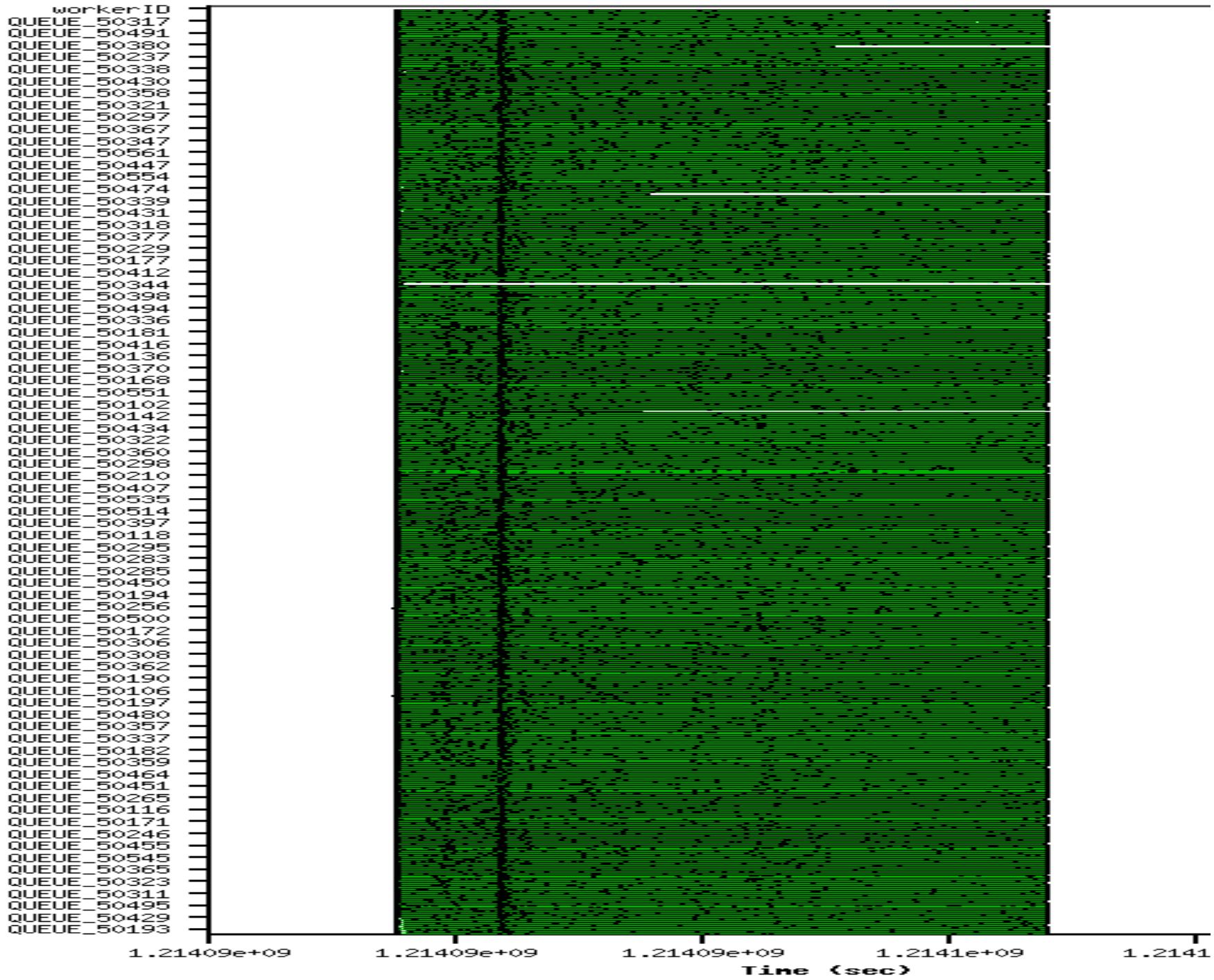
(from 16 to 32 racks)

Utilization:

- Sustained: 99.6%
- Overall: 78.3%

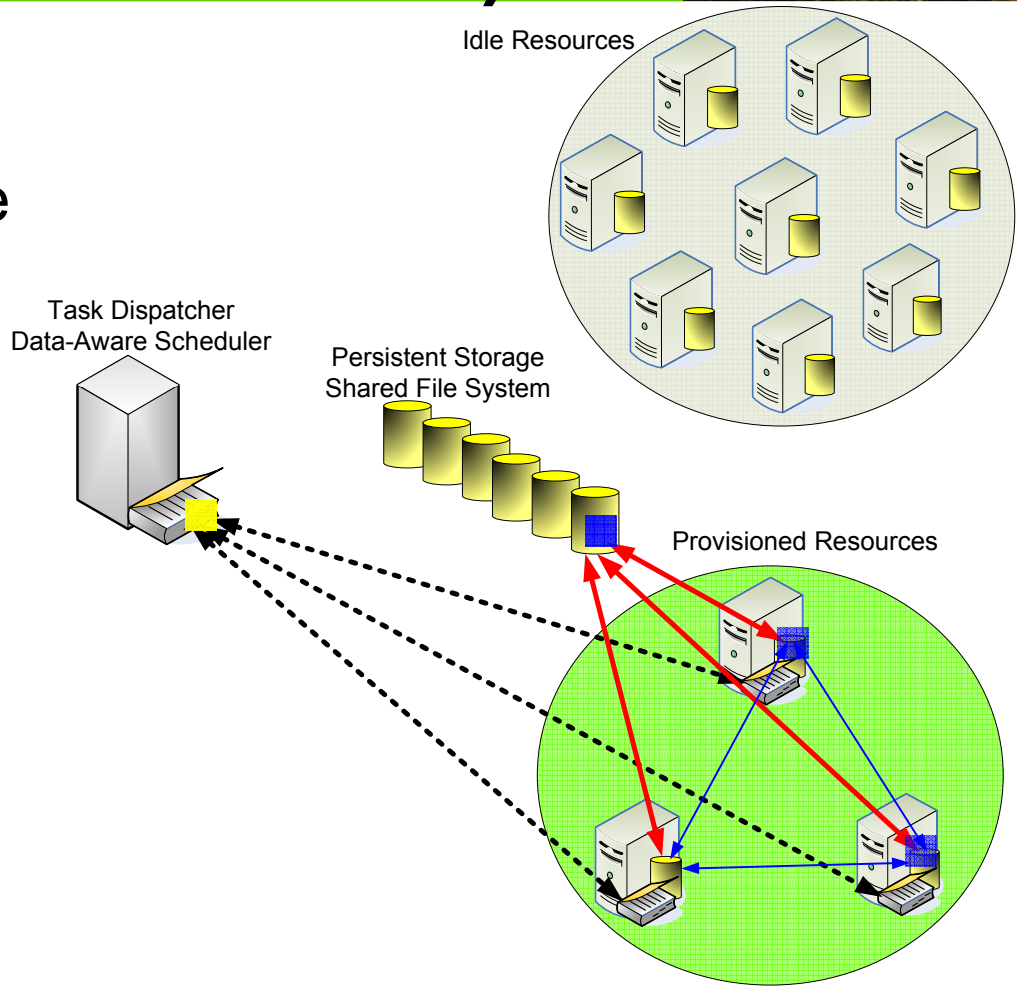





workerID
QUEUE_50317
QUEUE_50491
QUEUE_50380
QUEUE_50237
QUEUE_50338
QUEUE_50430
QUEUE_50358
QUEUE_50321
QUEUE_50297
QUEUE_50367
QUEUE_50347
QUEUE_50561
QUEUE_50447
QUEUE_50554
QUEUE_50474
QUEUE_50339
QUEUE_50431
QUEUE_50318
QUEUE_50377
QUEUE_50229
QUEUE_50177
QUEUE_50412
QUEUE_50344
QUEUE_50398
QUEUE_50494
QUEUE_50336
QUEUE_50181
QUEUE_50416
QUEUE_50136
QUEUE_50370
QUEUE_50168
QUEUE_50551
QUEUE_50102
QUEUE_50142
QUEUE_50434
QUEUE_50322
QUEUE_50360
QUEUE_50298
QUEUE_50210
QUEUE_50407
QUEUE_50535
QUEUE_50514
QUEUE_50397
QUEUE_50118
QUEUE_50295
QUEUE_50283
QUEUE_50285
QUEUE_50450
QUEUE_50194
QUEUE_50256
QUEUE_50500
QUEUE_50172
QUEUE_50306
QUEUE_50308
QUEUE_50362
QUEUE_50190
QUEUE_50106
QUEUE_50197
QUEUE_50480
QUEUE_50357
QUEUE_50337
QUEUE_50182
QUEUE_50359
QUEUE_50464
QUEUE_50451
QUEUE_50265
QUEUE_50116
QUEUE_50171
QUEUE_50246
QUEUE_50455
QUEUE_50545
QUEUE_50365
QUEUE_50323
QUEUE_50311
QUEUE_50495
QUEUE_50429
QUEUE_50193
1.21409e+09
1.21409e+09
1.21409e+09
1.2141e+09
1.2141
Time (sec)

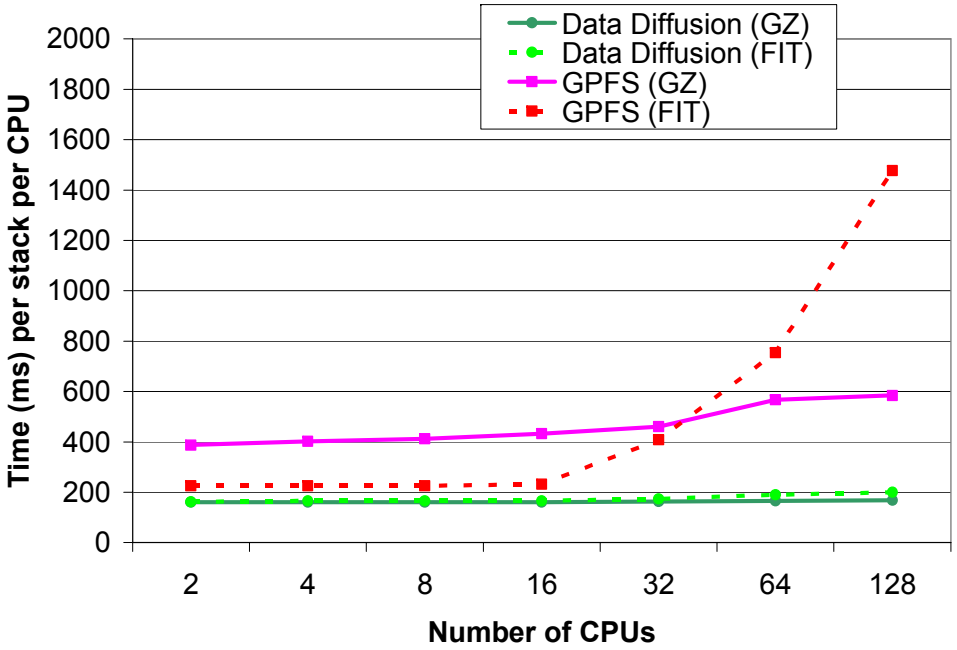
Support for Data Intensive Applications (Falkon and Data Diffusion)

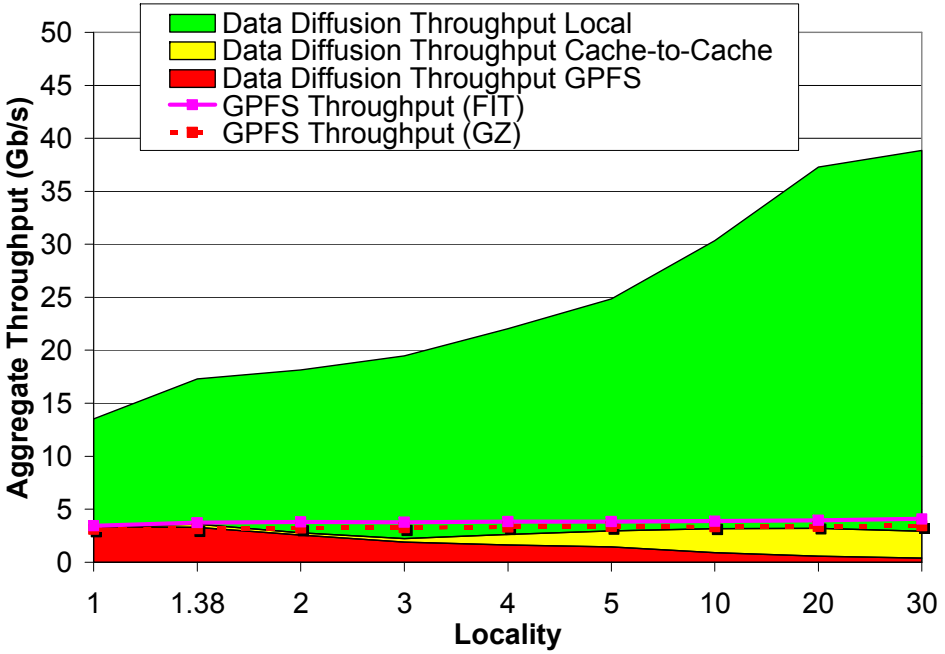
- Resource acquired in response to demand
- Data and applications diffuse from archival storage to newly acquired resources
- Resource "caching" allows faster responses to subsequent requests
  - Cache Eviction Strategies: RANDOM, FIFO, LRU, LFU
- Resources are released when demand drops



AstroPortal Stacking Service with Data Diffusion

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



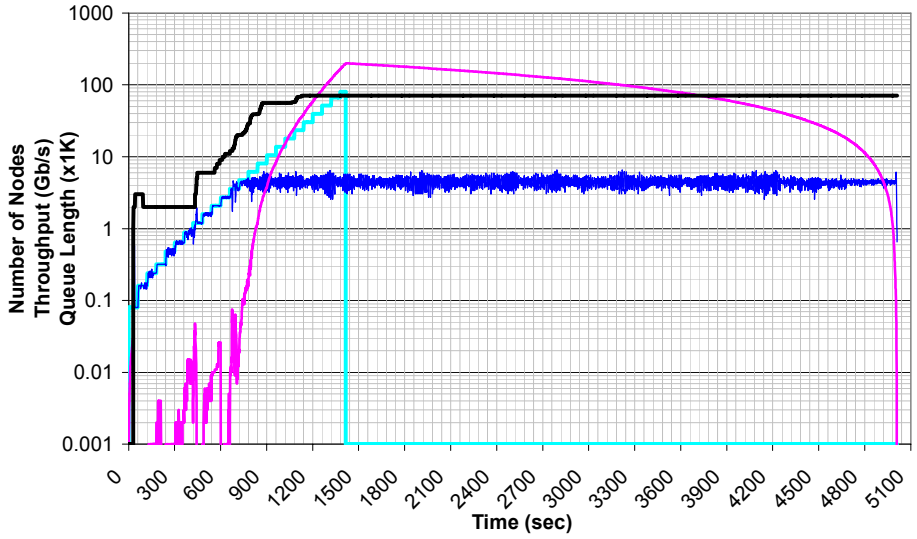


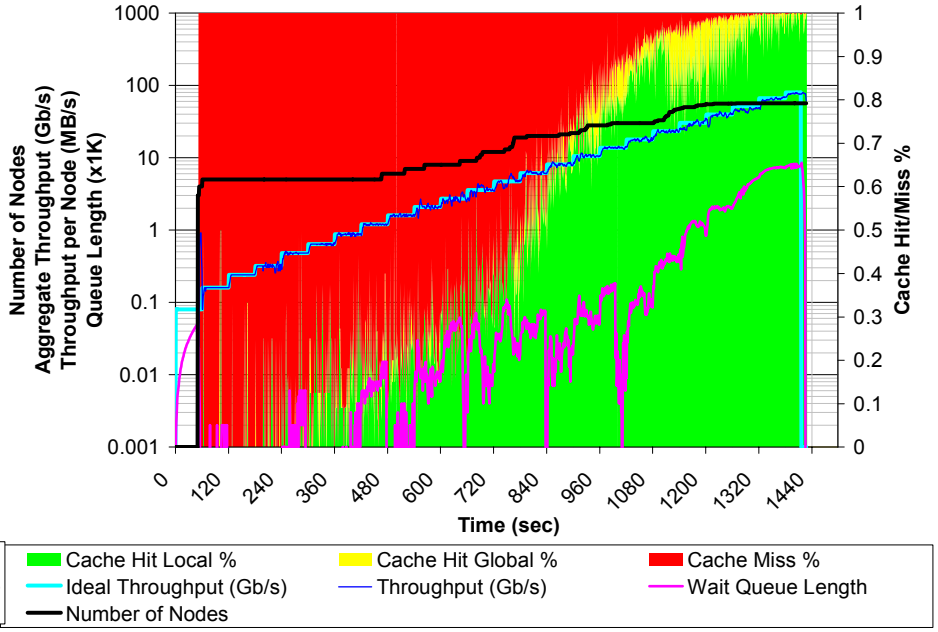
←High data locality

- Near perfect scalability

Data Diffusion: Data-Intensive Workload

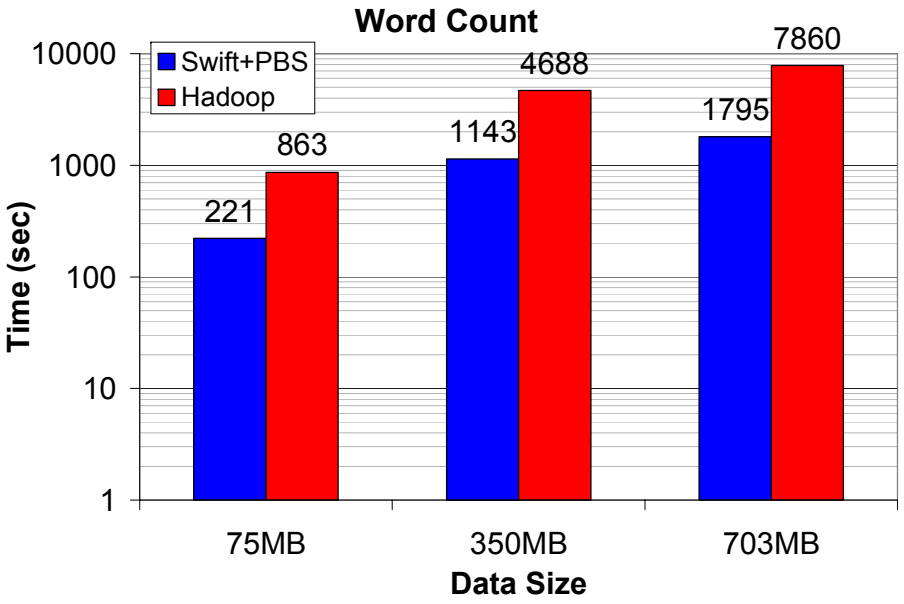
- 250K tasks on 128 processors
  - 10MB read, 10ms compute
- Comparing GPFS with data diffusion
  - 5011 sec vs. 1427 sec (ideal is 1415 sec)

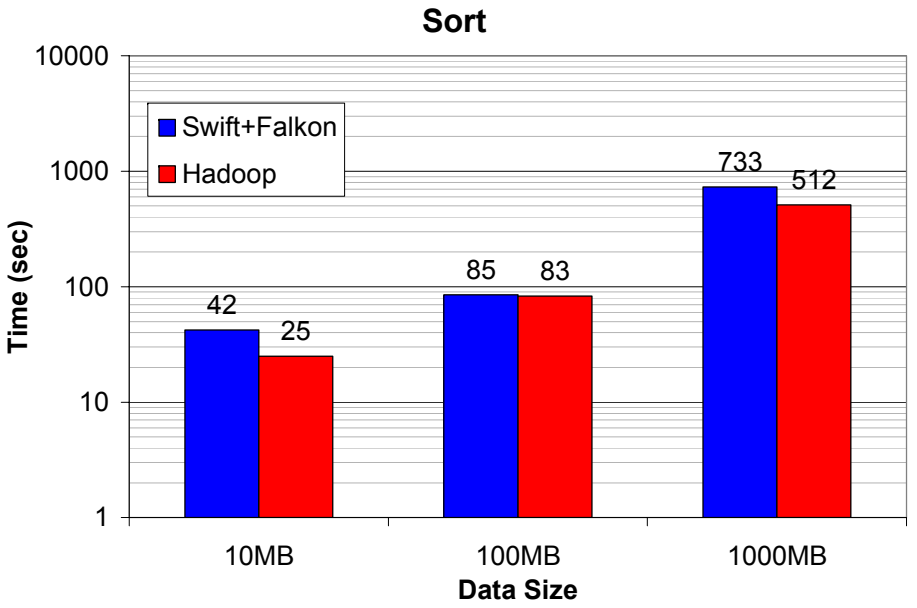




Hadoop vs. Swift

- 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!

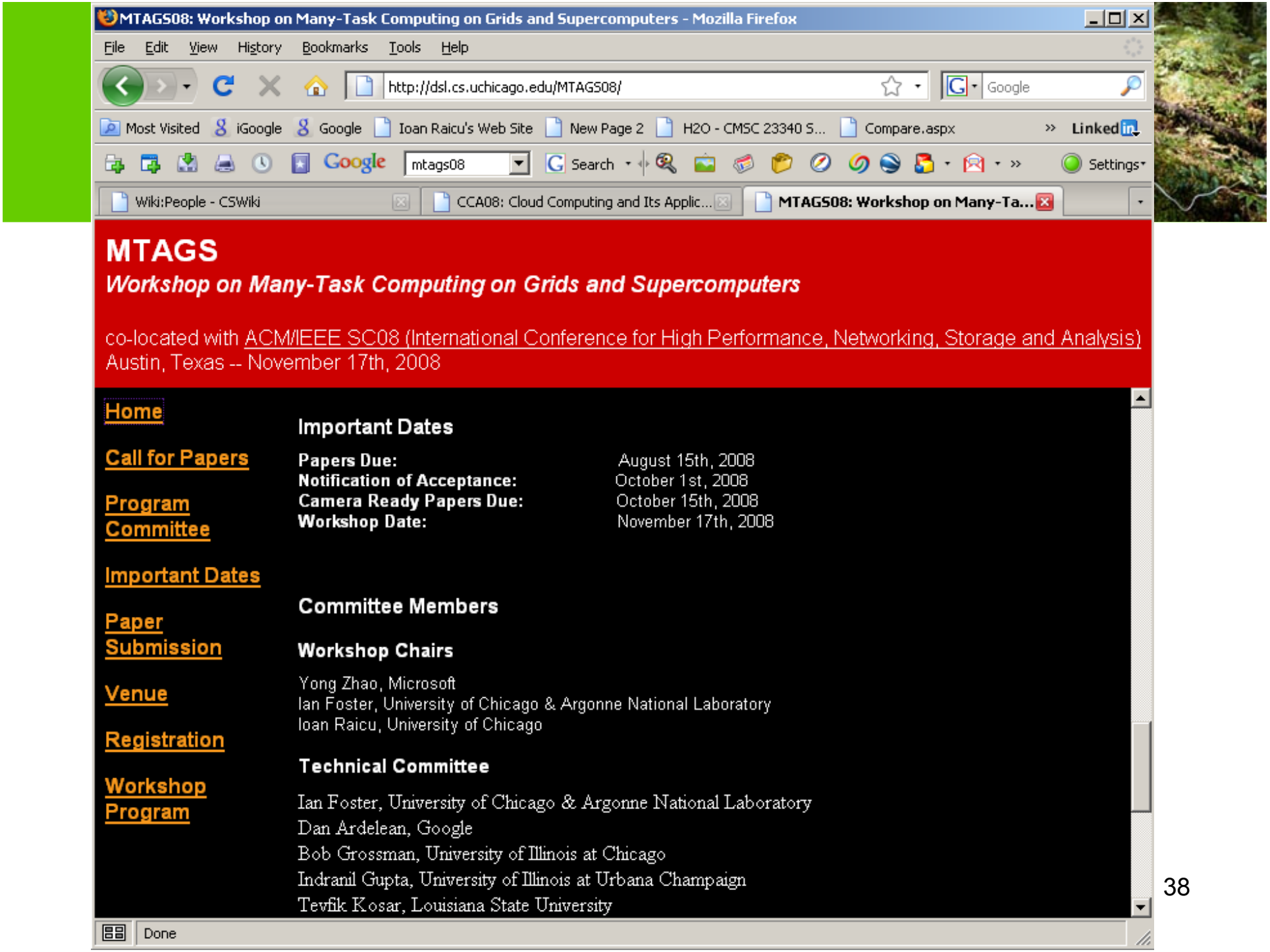
- **Parallelism**
  - Support for both explicit and implicit parallelism
- **Performance and Scalability**
  - Million to billions of tasks
  - Handle 100s~1000s of tasks/sec
- **Data management**
  - Reduce reliance on shared file systems
  - Scale with processing power
  - Data-aware scheduling
- **Reliability**
  - 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 (relevant to this proposal):
  - Ian Foster, The University of Chicago & Argonne National Laboratory
  - Alex Szalay, The Johns Hopkins University
  - 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
- Funding:
  - NASA: Ames Research Center, Graduate Student Research Program (GSRP)
  - DOE: Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy
  - NSF: TeraGrid


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/
Google
Most Visited iGoogle Google Ioan Raicu's Web Site New Page 2 H2O - CMSC 23340 S... Compare.aspx LinkedIn
mtags08 Search Settings
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