Toward Loosely Coupled Programming on Petascale Systems with Falkon

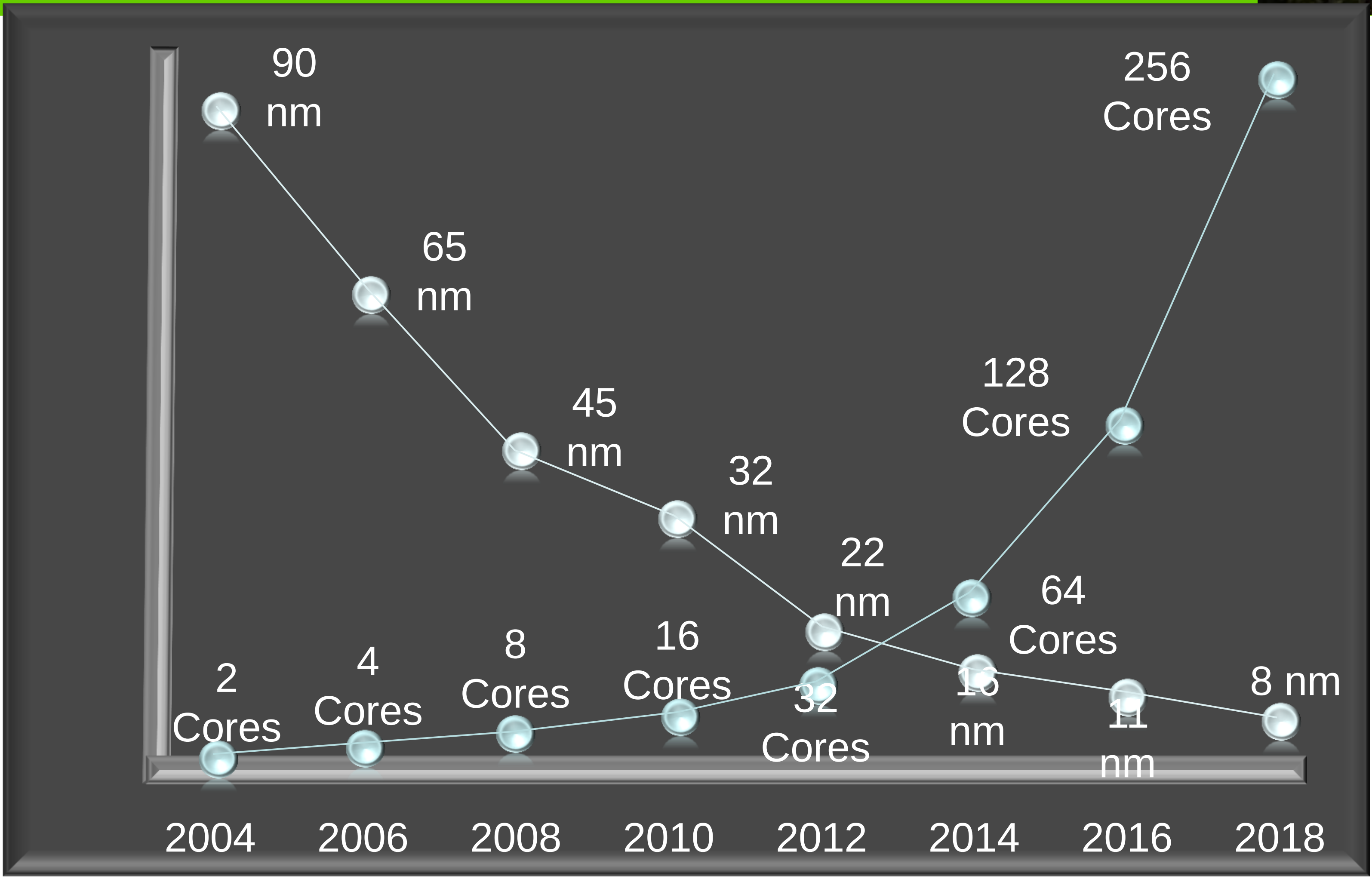
Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Based on Slides given at IEEE/ACM Supercomputing 2008
October 16th, 2013

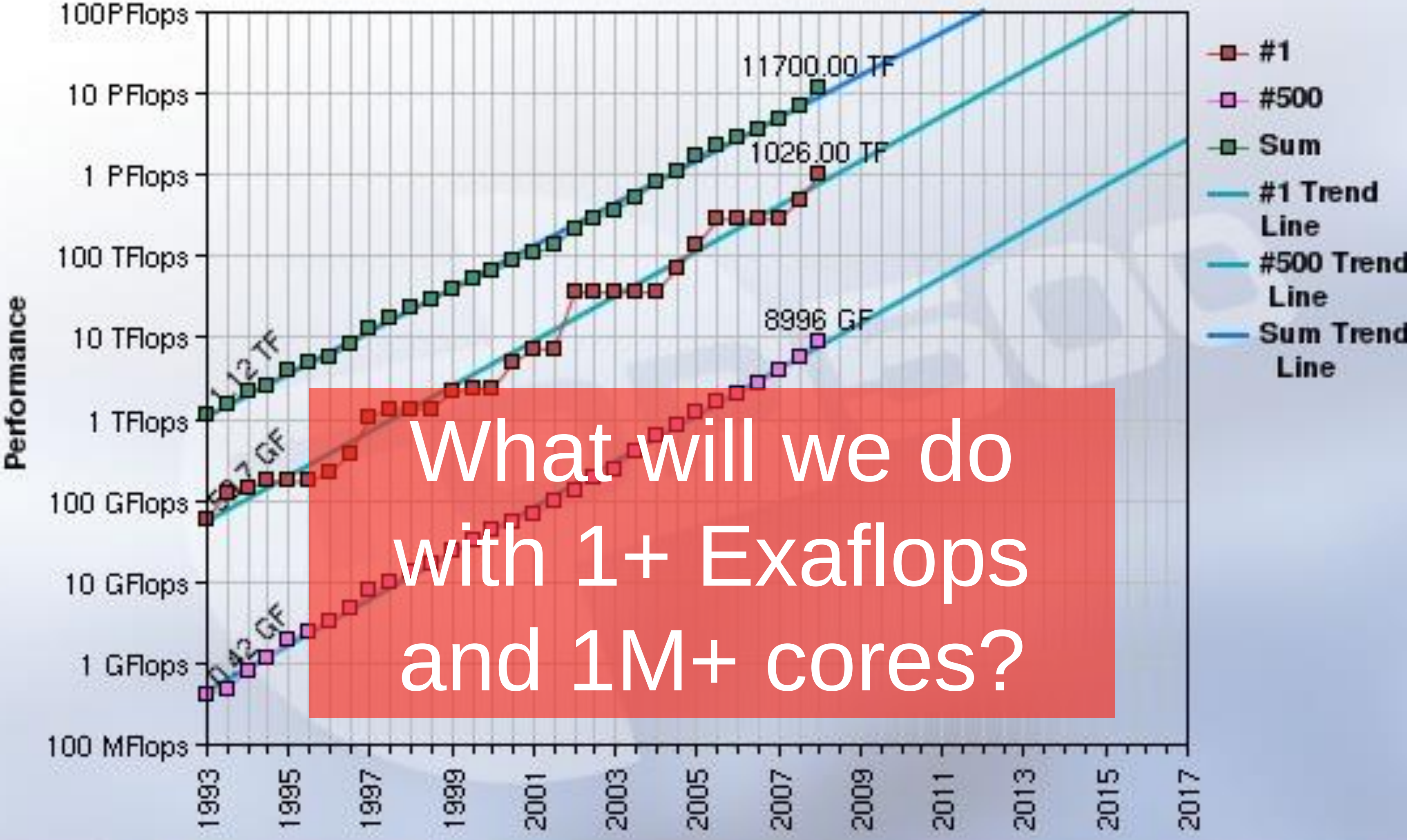
PART I

Motivation

Many-Core Growth Rates



Projected Performance Development

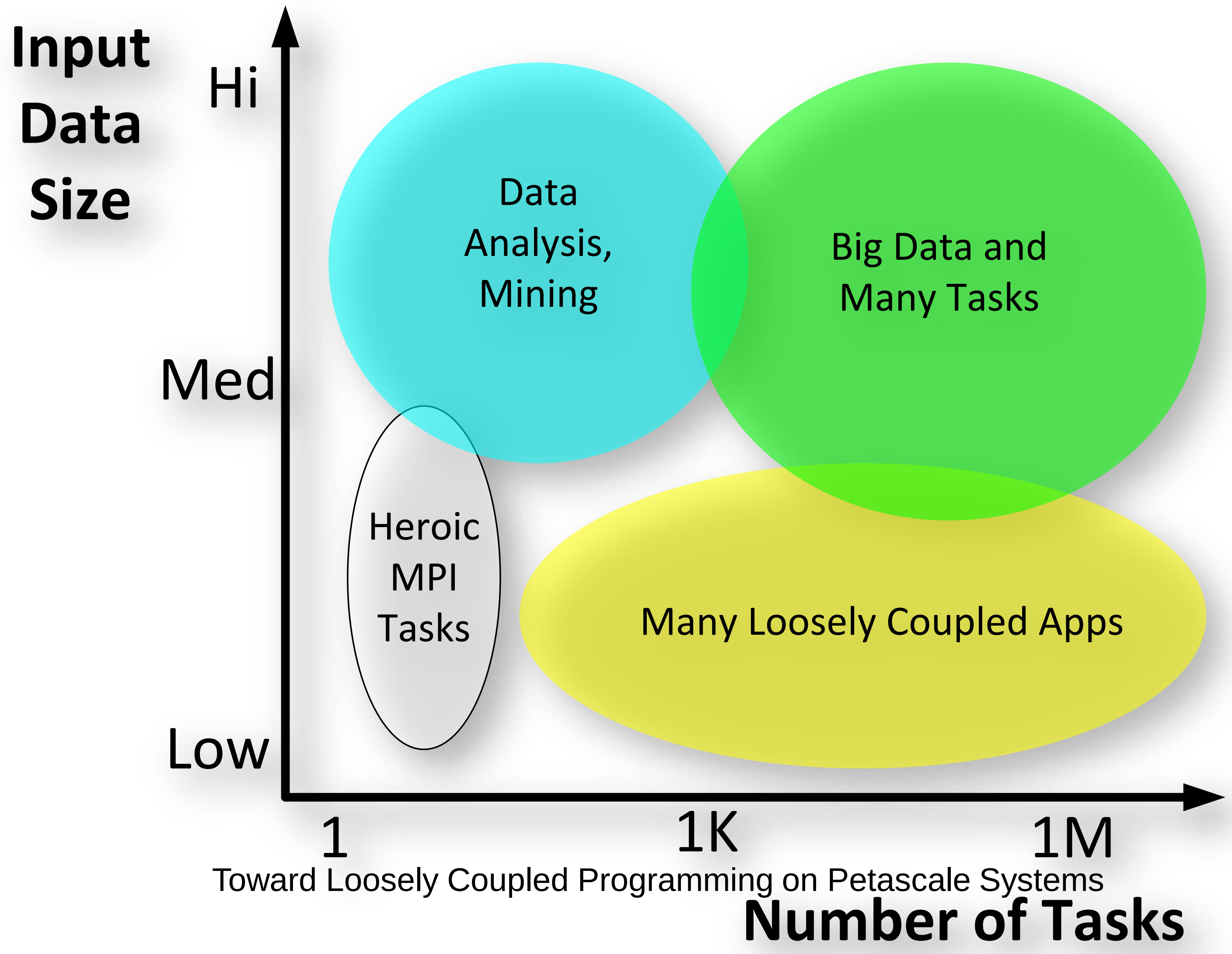


What will we do with 1+ Exaflops and 1M+ cores?

Programming Model Issues

- **Multicore/Manycore** processors
- Massive **task parallelism**
- Massive **data parallelism**
- Integrating **black box applications**
- Complex **task dependencies** (task graphs)
- **Failure**, and other execution management issues
- **Dynamic task graphs**
- Documenting **provenance** of data products
- **Data management**: input, intermediate, output
- **Dynamic data access** over large amounts of data

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)

MTC: Many Task Computing

- Bridge the gap between HPC and HTC
- Loosely coupled applications with HPC orientations
- HPC comprising of multiple distinct activities, coupled via file system operations or message passing
- Emphasis on many resources over short time periods
- Tasks can be:
  - small or large, independent and dependent, uniprocessor or multiprocessor, compute-intensive or data-intensive, static or dynamic, homogeneous or heterogeneous, loosely or tightly coupled, large number of tasks, large quantity of computing, and large volumes of data…

Growing Interest on enabling HTC/MTC on Supercomputers

- Project Kittyhawk
  - IBM Research
- HTC-mode in Cobalt/BG
  - IBM
- Condor on BG
  - University of Wisconsin at Madison, IBM
- Grid Enabling the BG
  - University of Colorado, National Center for Atmospheric Research
- Plan 9
  - Bell Labs, IBM Research, Sandia National Labs
- Falkon/Swift on BG/P and Sun Constellation
  - University of Chicago, Argonne National Laboratory

Many Large Systems available for Open Science Research

- Jaguar (#2) *[to be announced in 90 minutes]*
  - DOE, Oak Ridge National Laboratory
- Intrepid (#5)
  - DOE, Argonne National Laboratory
- Ranger (#6)
  - University of Texas / NFS TeraGrid

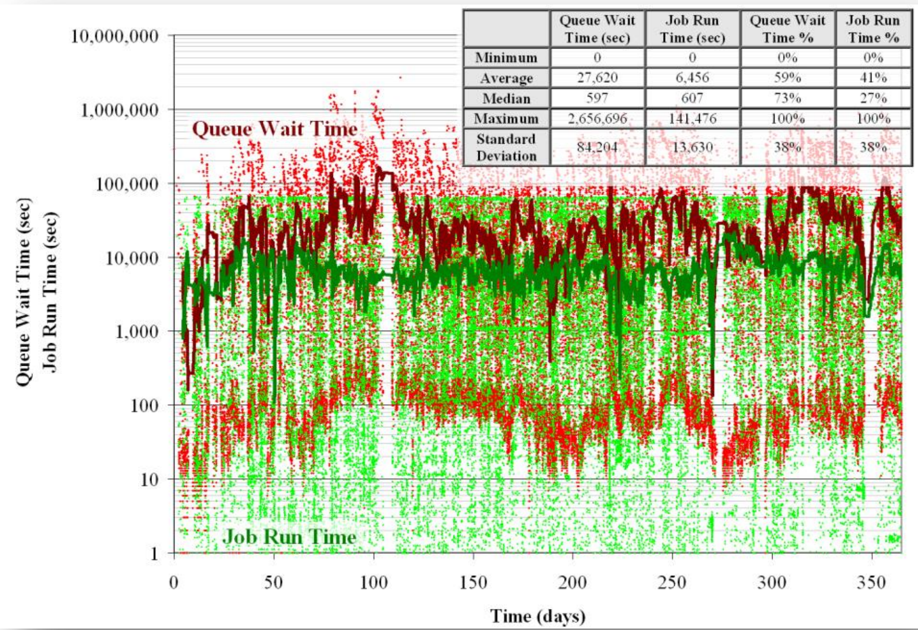
Why Petascale Systems for MTC Applications?

1. The I/O subsystem of petascale systems offers unique capabilities needed by MTC applications
2. The cost to manage and run on petascale systems is less than that of conventional clusters or Grids
3. Large-scale systems that favor large jobs have utilization issues
4. Some problems are intractable without petascale systems

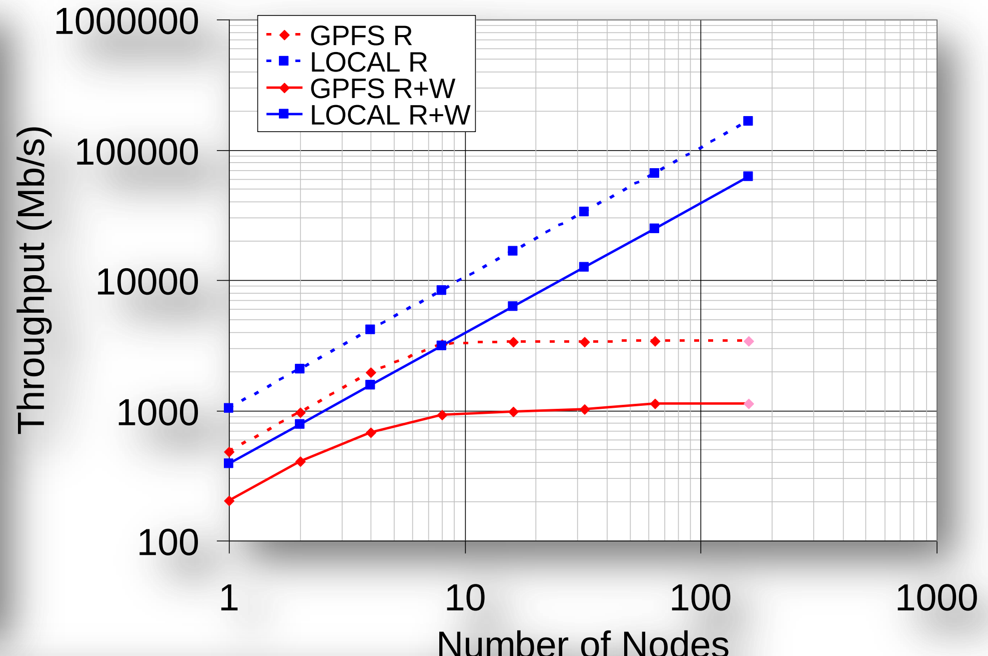
PART II

Some context on systems we used as building blocks

Obstacles running MTC apps in Clusters/Grids



	Queue Wait Time (sec)	Job Run Time (sec)	Queue Wait Time %	Job Run Time %
Minimum	0	0	0%	0%
Average	27,620	6,456	59%	41%
Median	597	607	73%	27%
Maximum	2,656,696	141,476	100%	100%
Standard Deviation	84,204	13,630	38%	38%

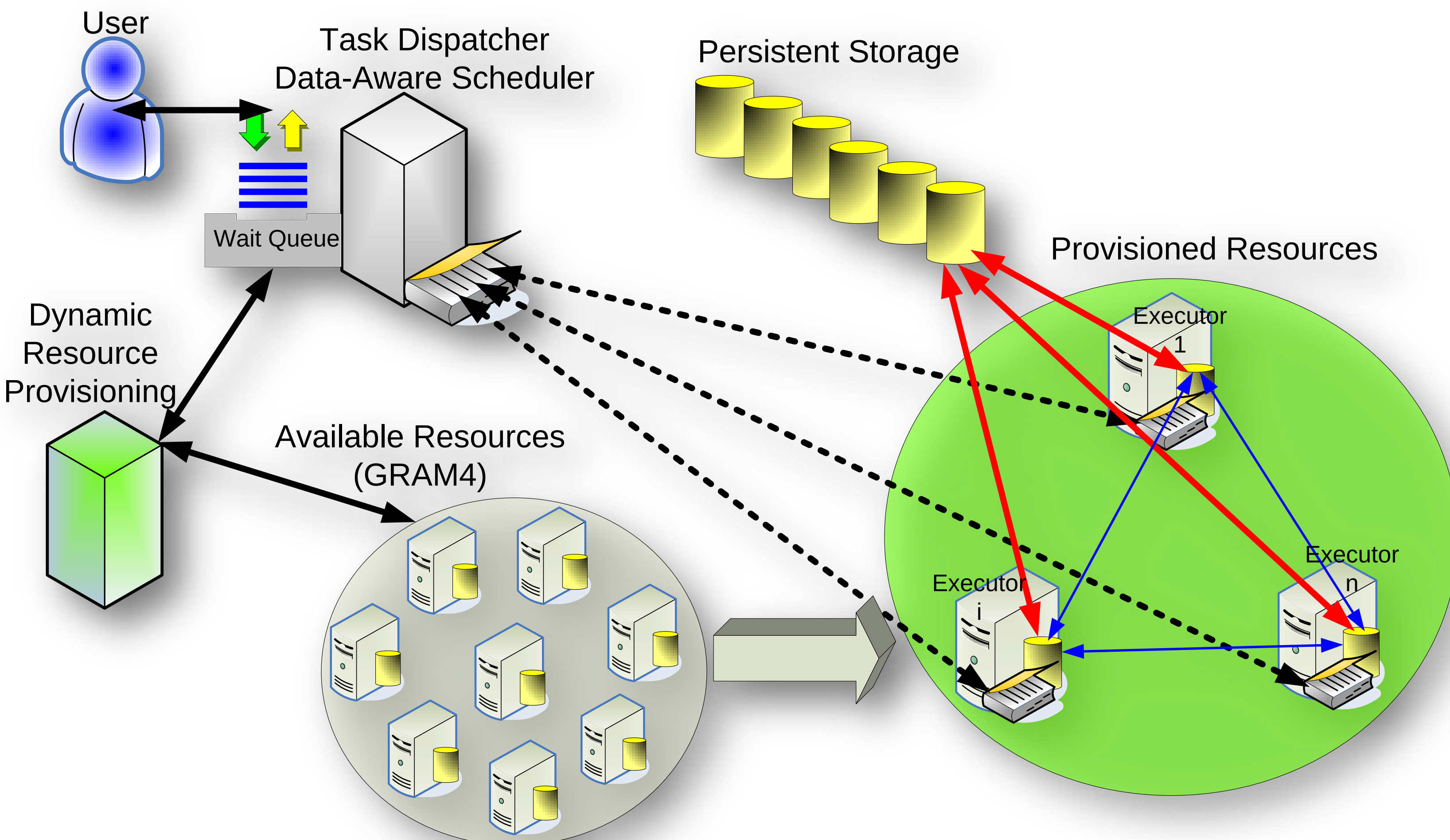


System	Comments	Throughput (tasks/sec)
Condor (v6.7.2) - Production	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8) - Production	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) - Production	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) - Production		0.42
Condor (v6.9.3) - Development		11
Condor-J2 - Experimental	Quad Xeon 3 GHz, 4GB	22

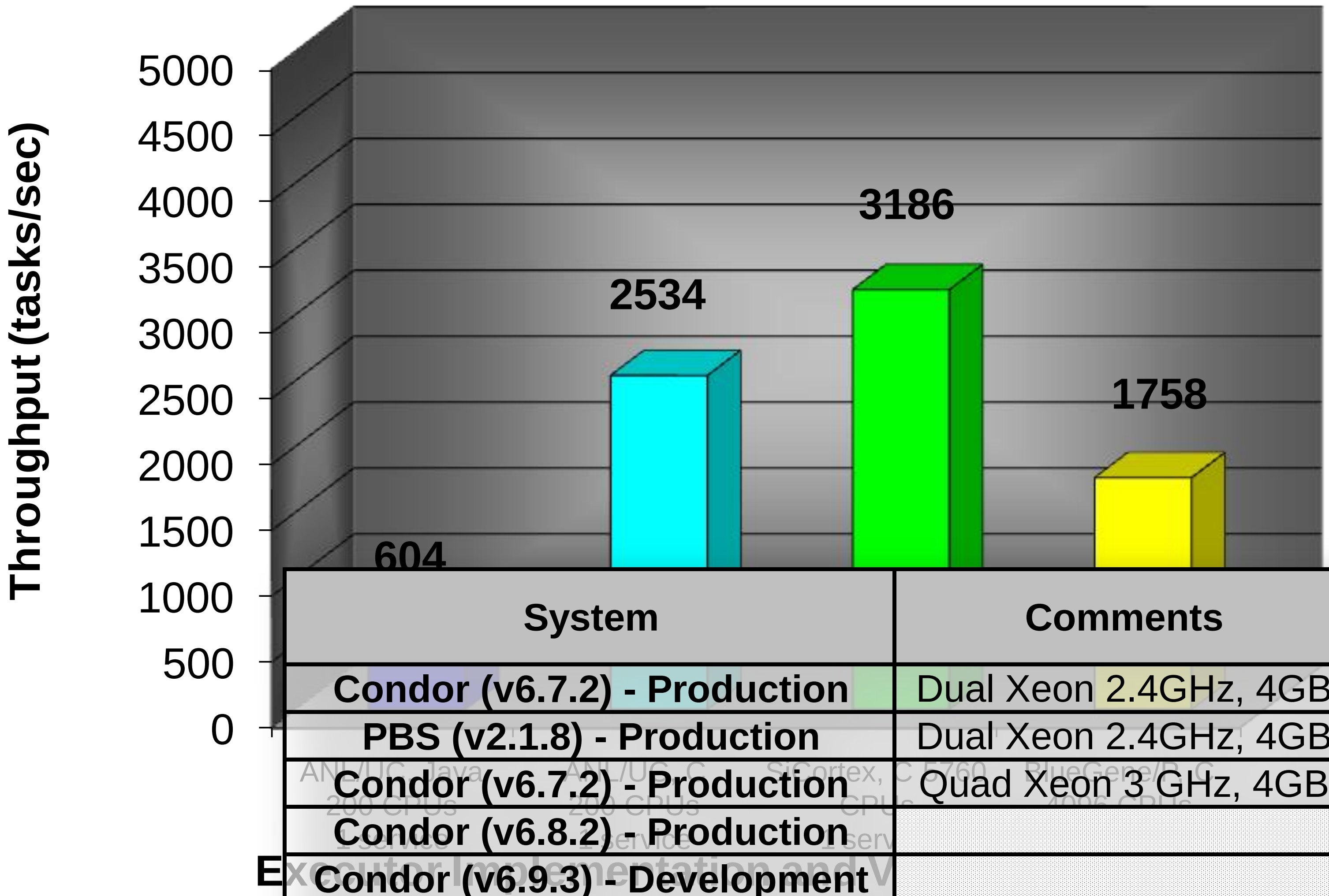
Solutions

- Falkon: A Fast and Light-weight tasK executiON framework
  - ***Goal:*** enable the ***rapid and efficient*** execution of many independent jobs on large compute clusters
  - Combines three components:
    - A ***streamlined task dispatcher***
    - ***Resource provisioning*** through multi-level scheduling techniques
    - ***Data diffusion*** and data-aware scheduling to leverage the co-located computational and storage resources
- Swift: A parallel programming system for loosely coupled applications
  - Applications cover many domains: Astronomy, astro-physics, medicine, chemistry, economics, climate modeling, data analytics

Falkon Overview

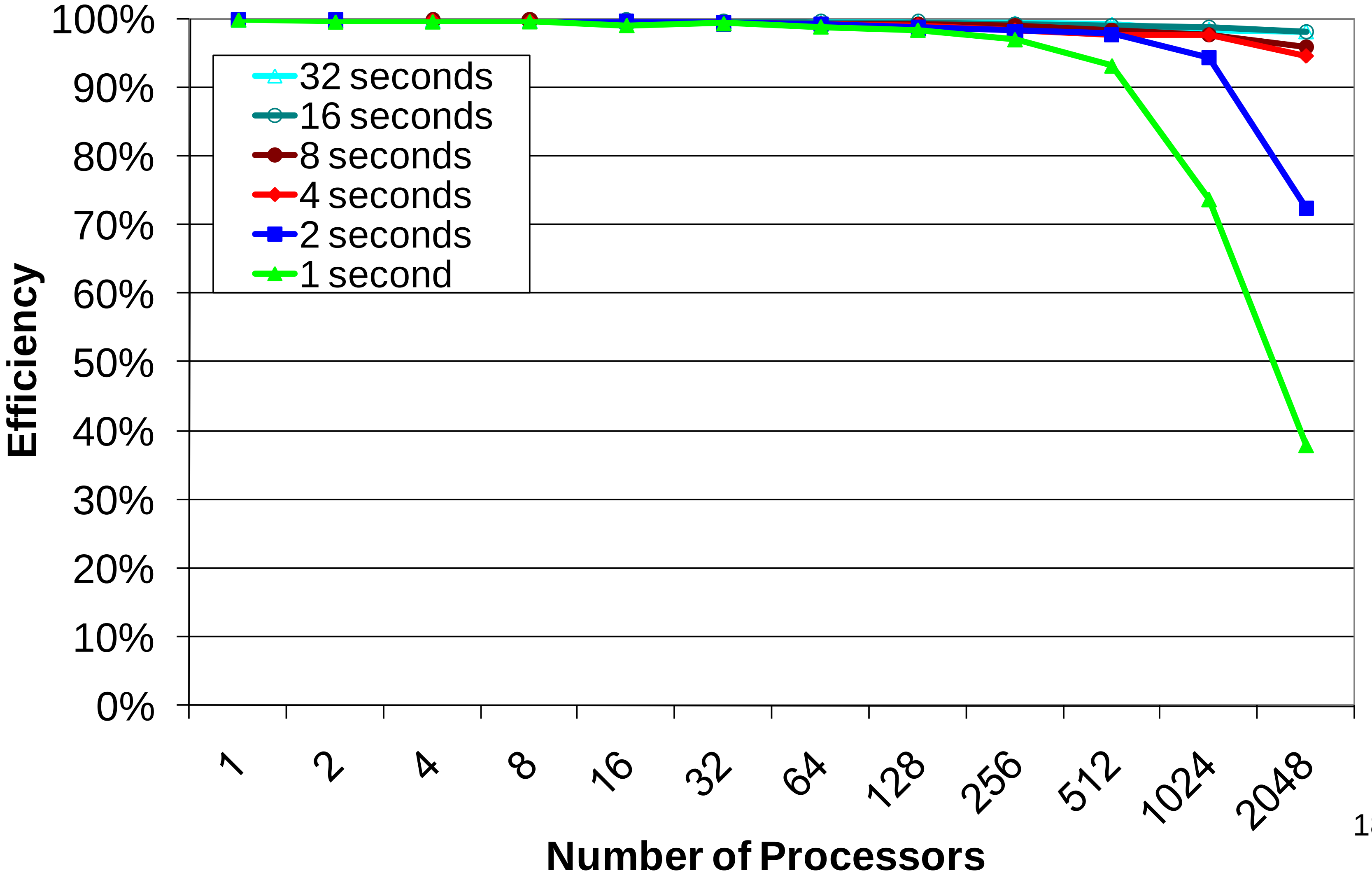


Dispatch Throughput



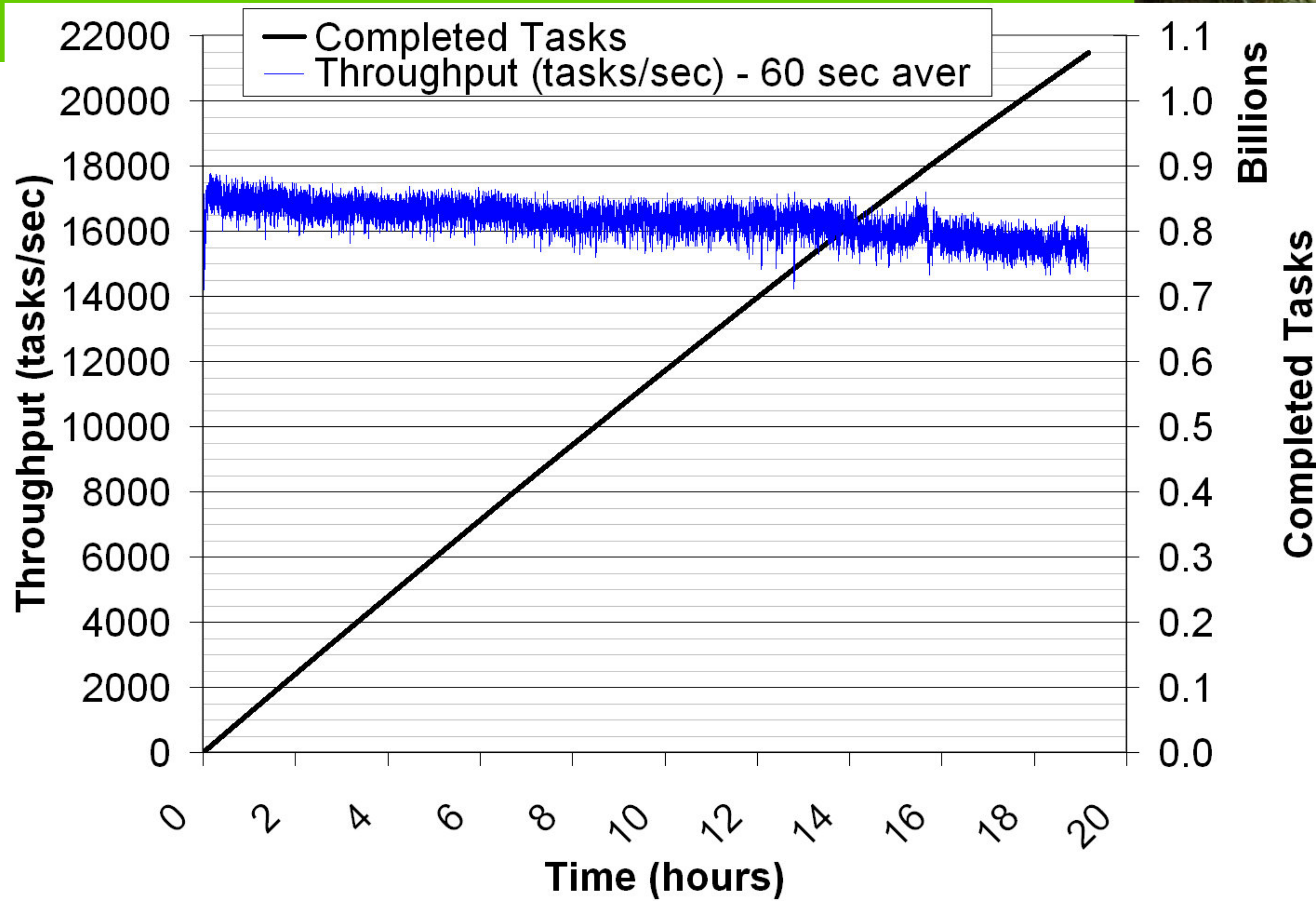
System	Comments	Throughput (tasks/sec)
Condor (v6.7.2) - Production	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8) - Production	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) - Production	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) - Production		0.42
Condor (v6.9.3) - Development		11
Condor-J2 - Experimental	Quad Xeon 3 GHz, 4GB	22

Efficiency

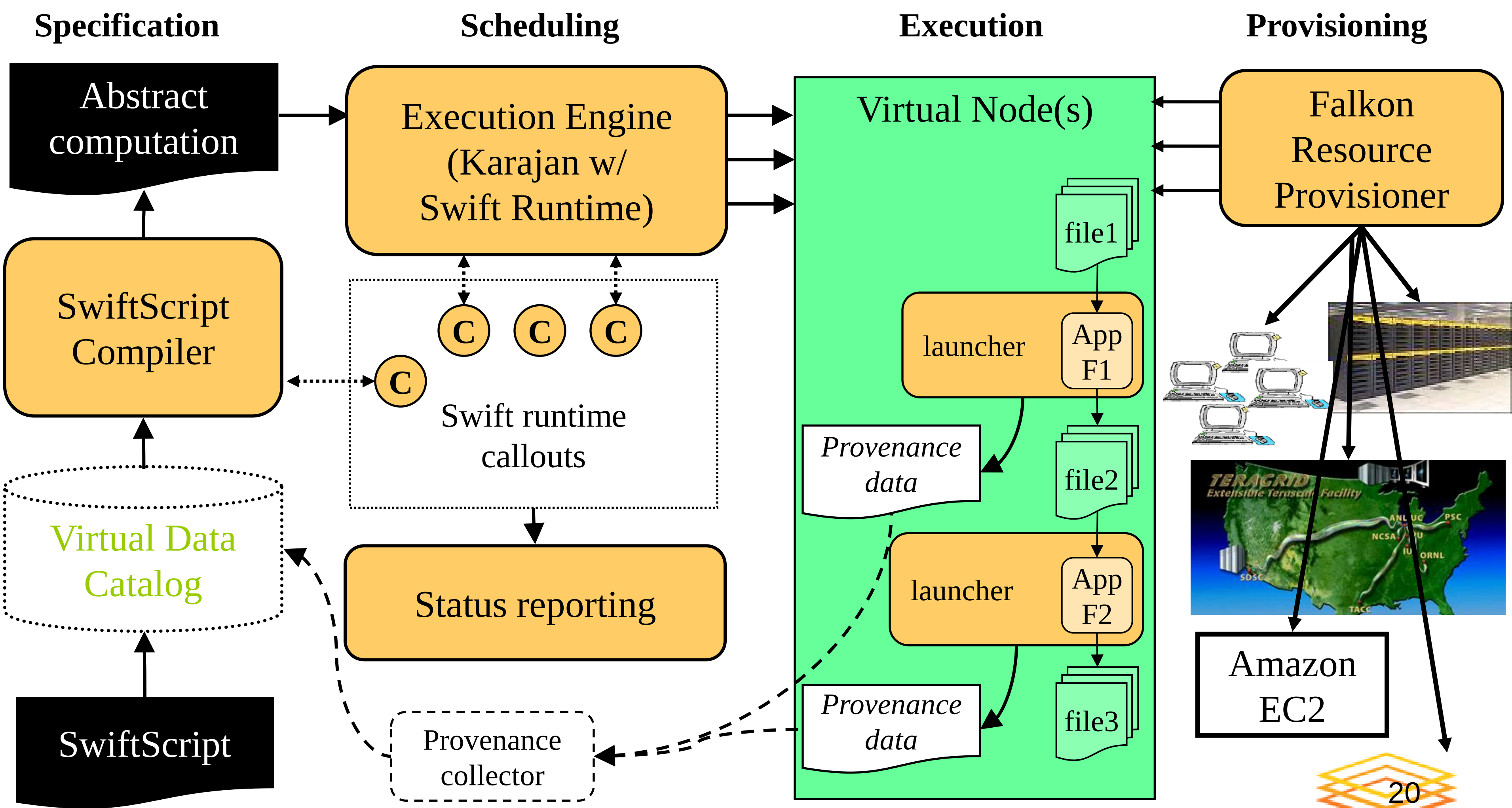


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Efficiency
32 seconds
16 seconds
8 seconds
4 seconds
2 seconds
1 second
1
2
4
8
16
32
64
128
256
512
1024
2048
Number of Processors

Falkon Endurance Test



Swift Architecture



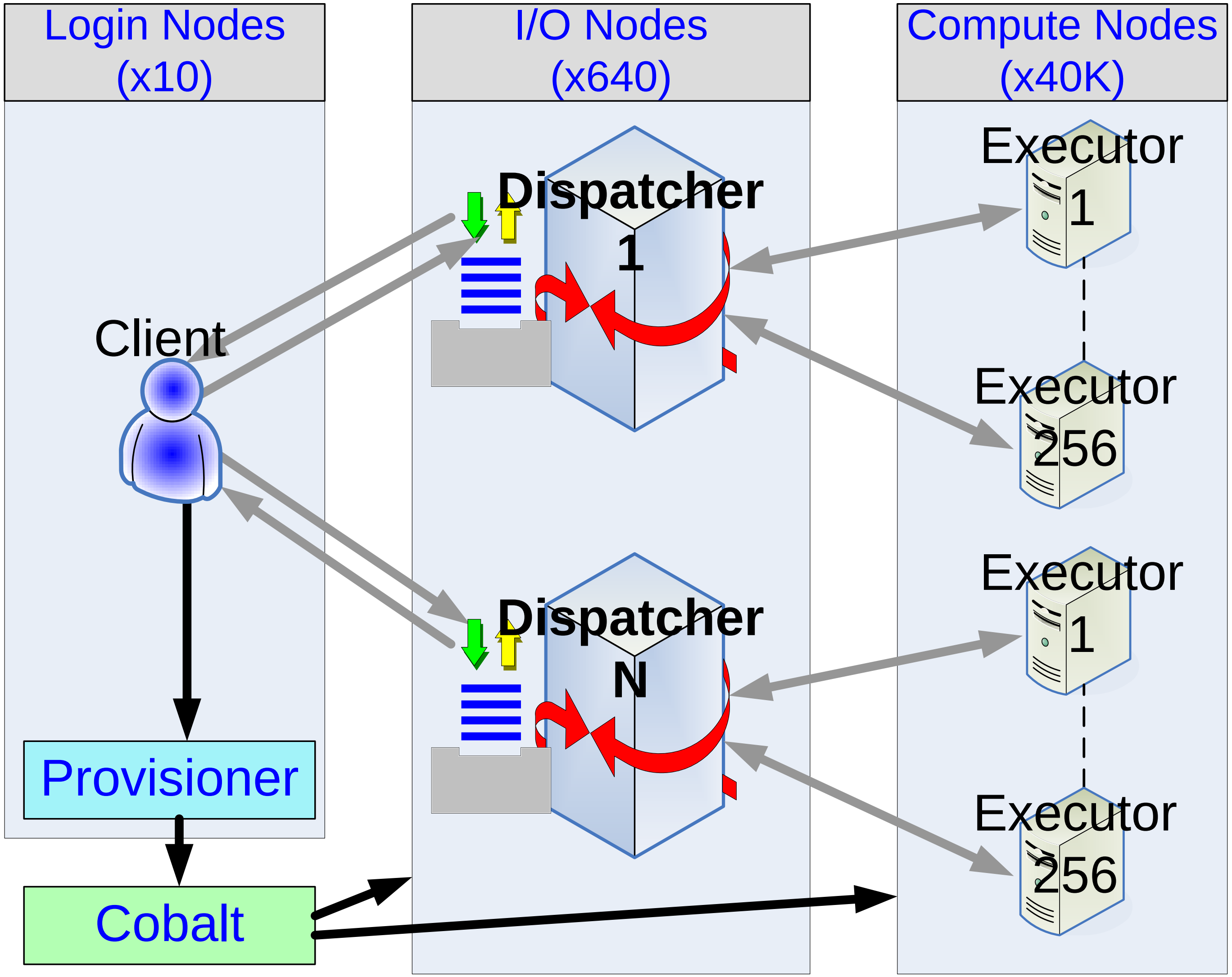
PART III

Contributions:
Proposed Changes & Results

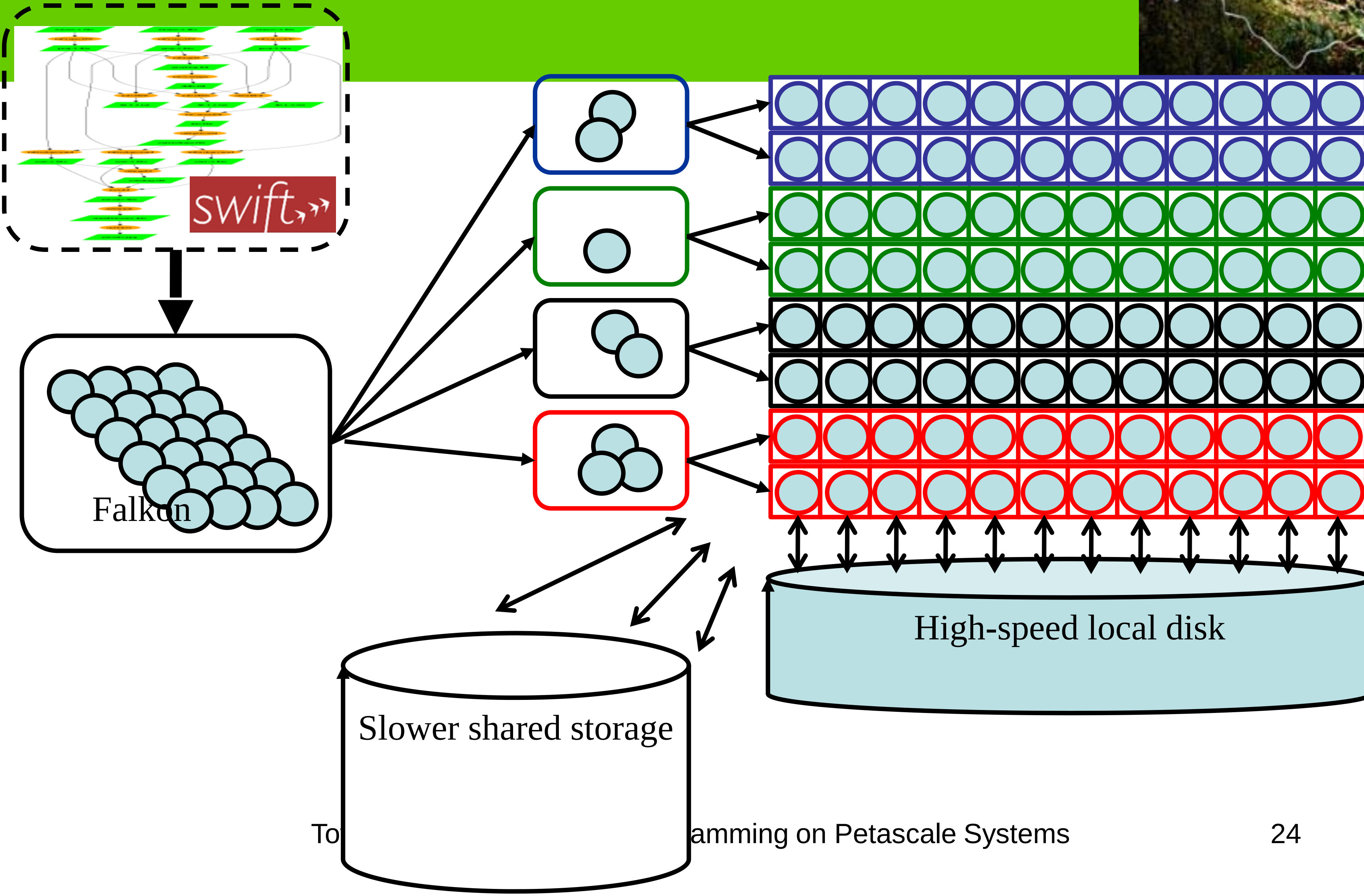
Scaling from 1K to 100K CPUs

- At 1K CPUs:
  - 1 Server to manage all 1K CPUs
  - Use shared file system extensively
    - Invoke application from shared file system
    - Read/write data from/to shared file system
- At 100K CPUs:
  - N Servers to manage 100K CPUs (1:256 ratio)
  - Don’t trust the application I/O access patterns to behave optimally
    - Copy applications and input data to RAM
    - Read input data from RAM, compute, and write results to RAM
    - Archive all results in a single file in RAM
    - Copy 1 result file from RAM back to GPFS
  - Use collective I/O primitives to make app logic simpler
  - Leverage all networks (Ethernet, Tree, and Torus) for high aggregate bandwidth

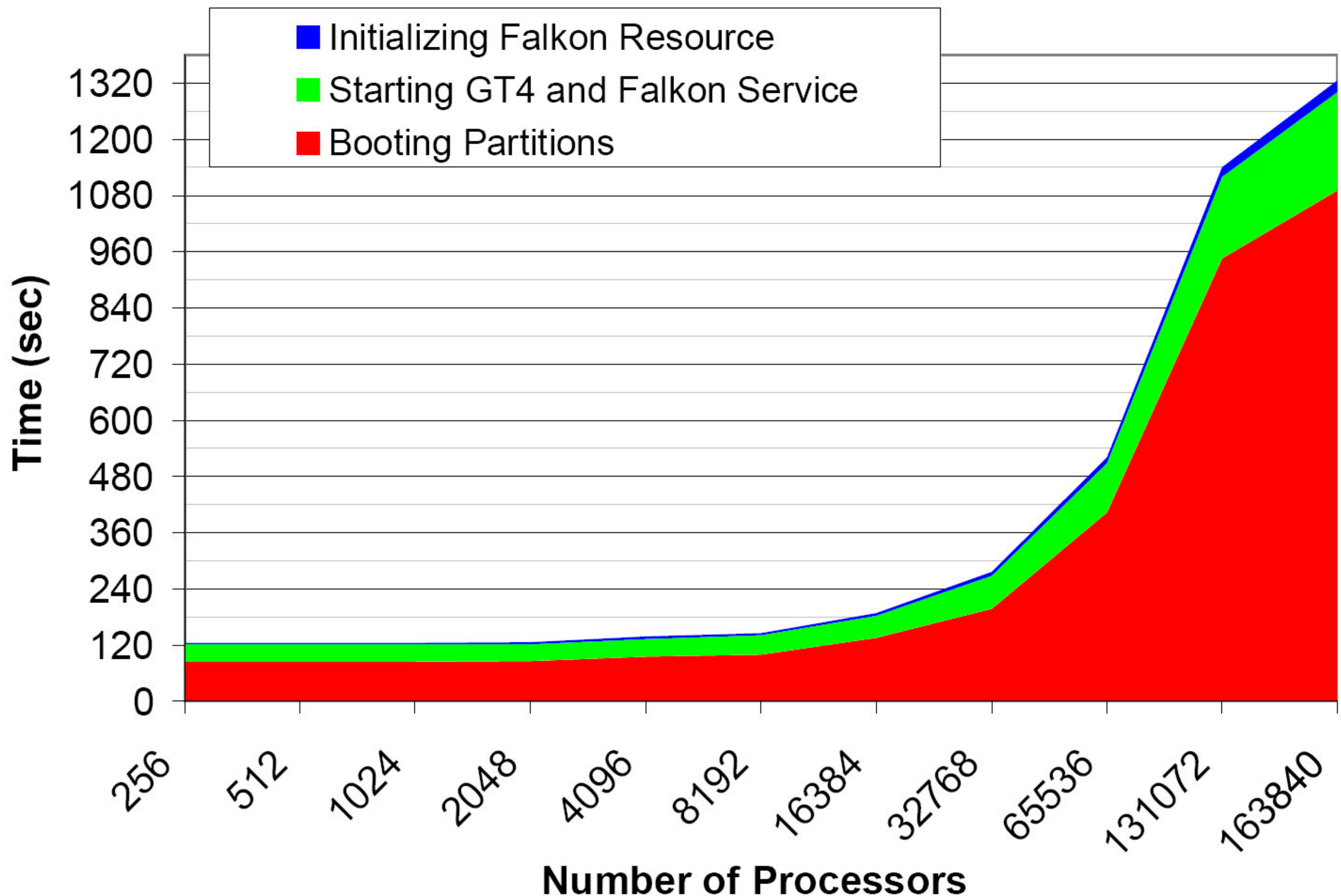
Distributed Falkon Architecture



Managing 160K CPUs

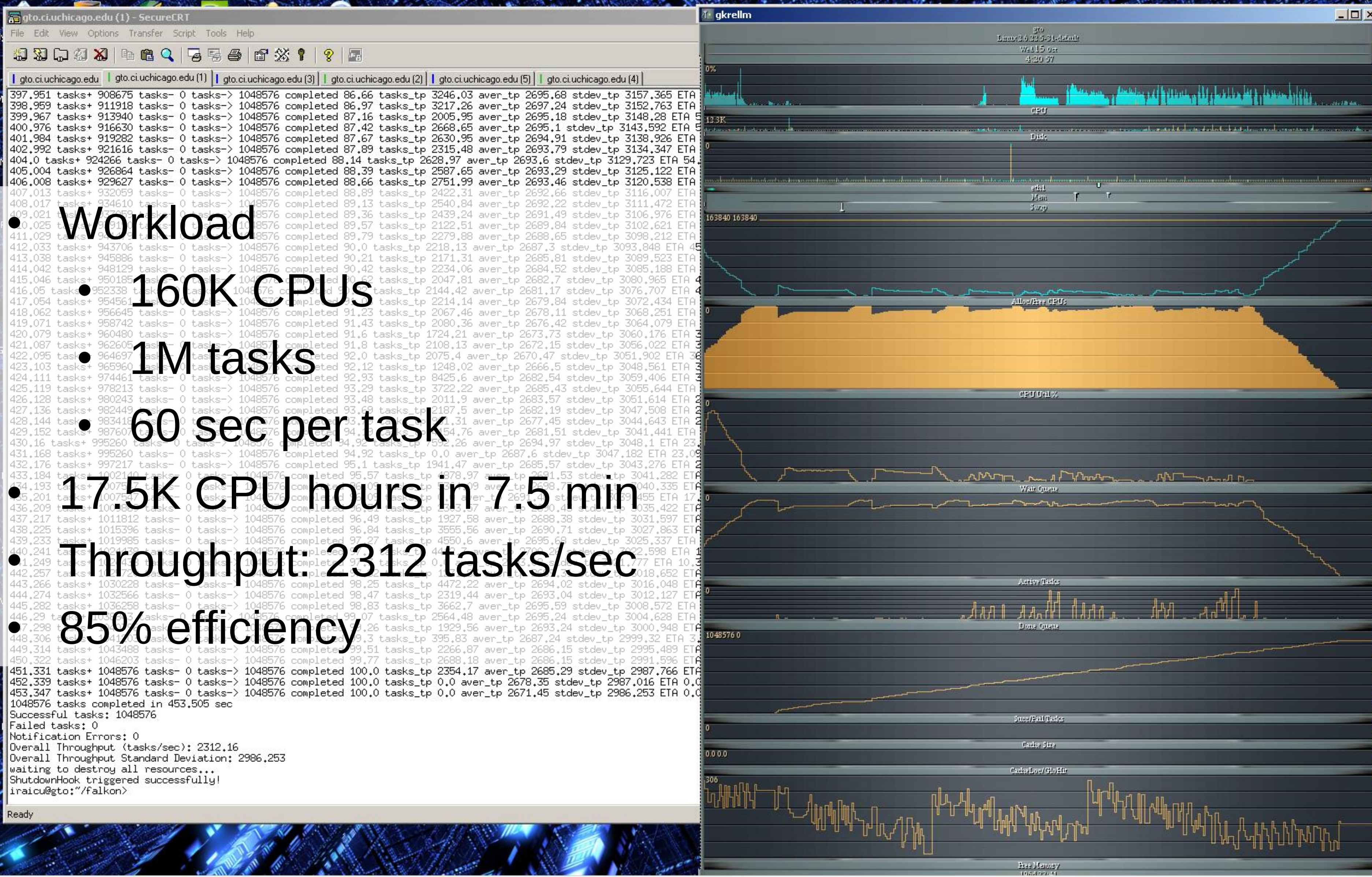


Falkon Bootstrapping

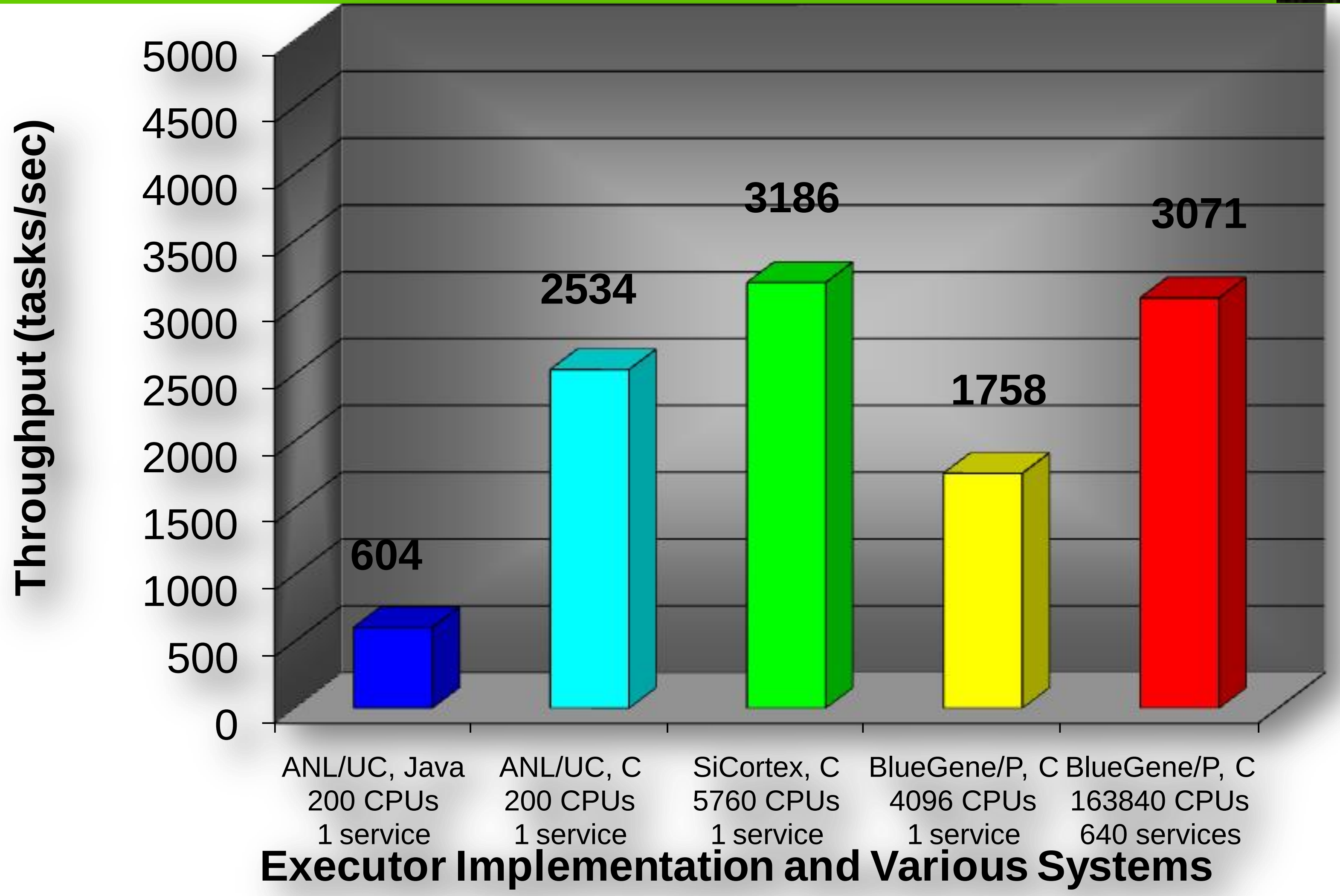


Falkon Monitoring

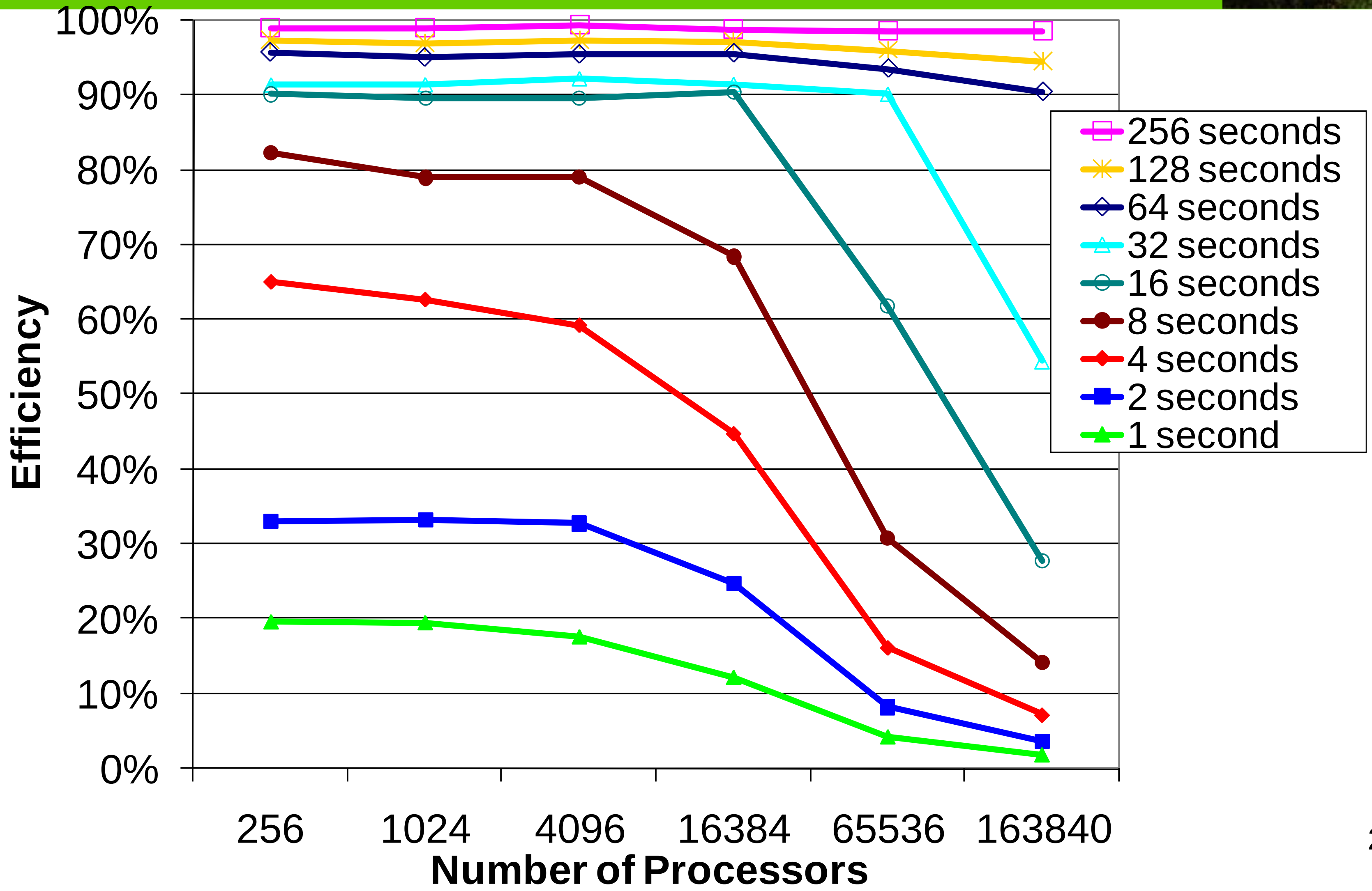
- Workload
  - 160K CPUs
  - 1M tasks
  - 60 sec per task
- 17.5K CPU hours in 7.5 min
- Throughput: 2312 tasks/sec
- 85% efficiency



Dispatch Throughput



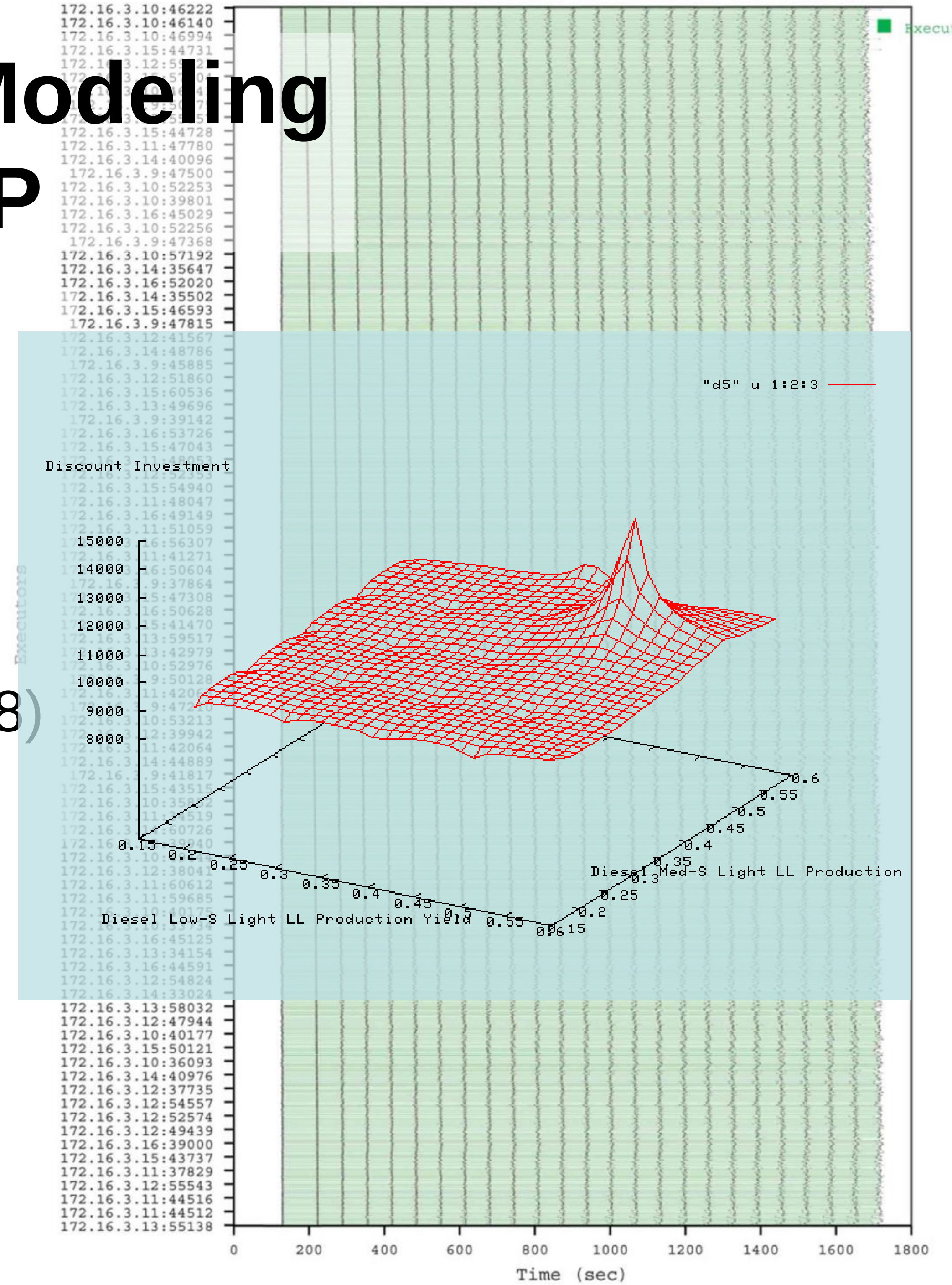
Efficiency



100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Efficiency
256
1024
4096
16384
65536
163840
Number of Processors
256 seconds
128 seconds
64 seconds
32 seconds
16 seconds
8 seconds
4 seconds
2 seconds
1 second

MARS Economic Modeling on IBM BG/P

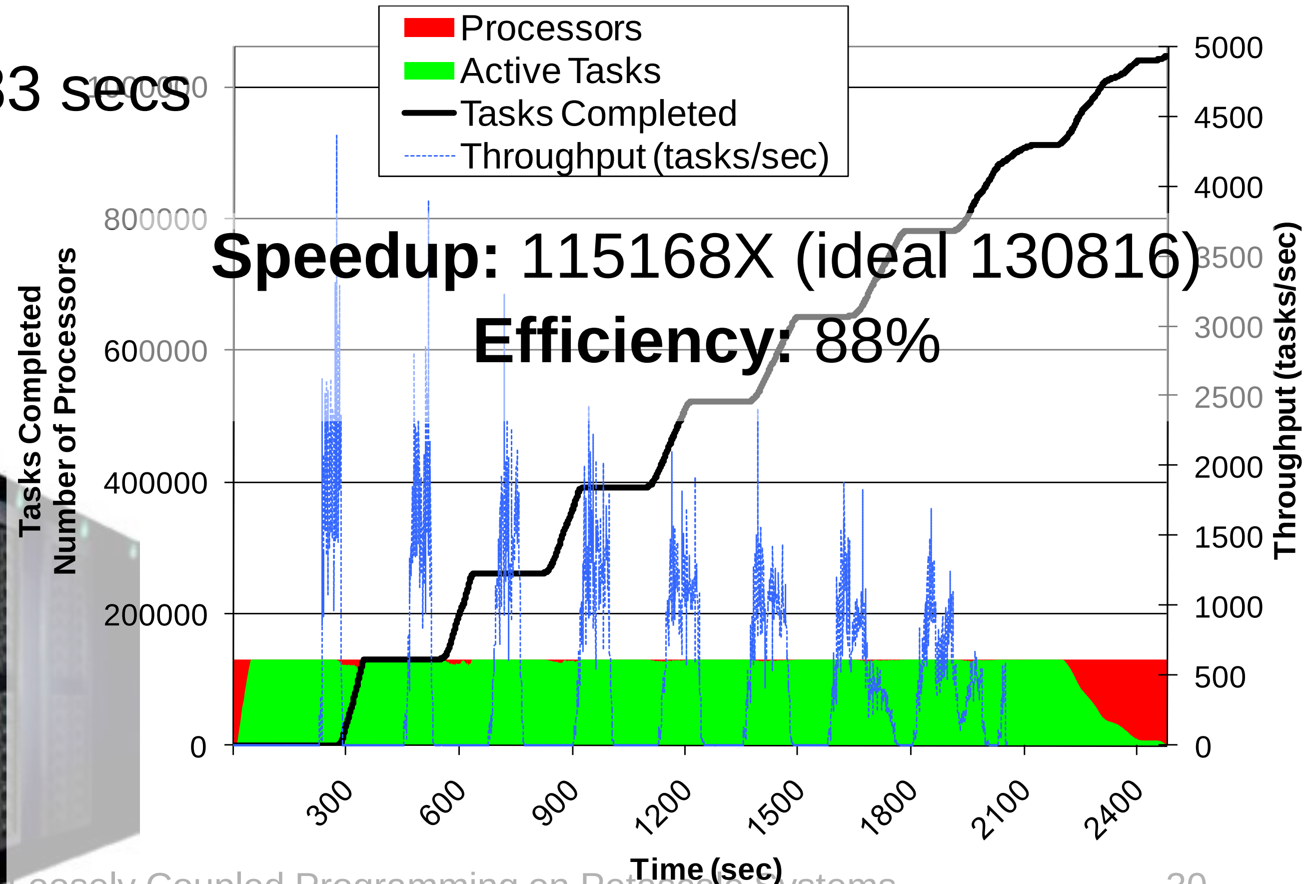
- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%



Couplec

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

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

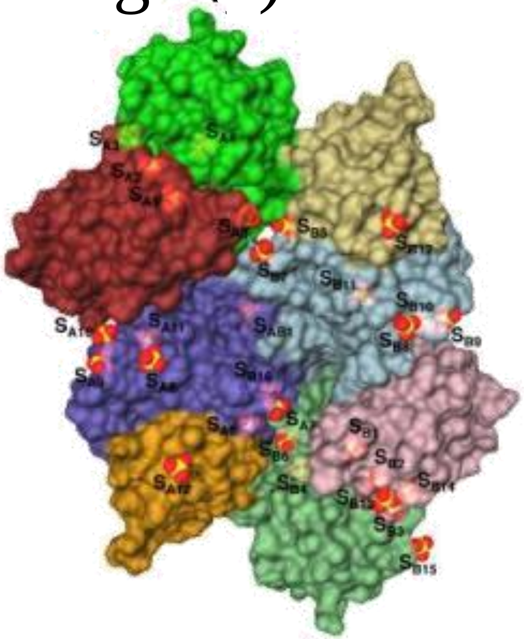


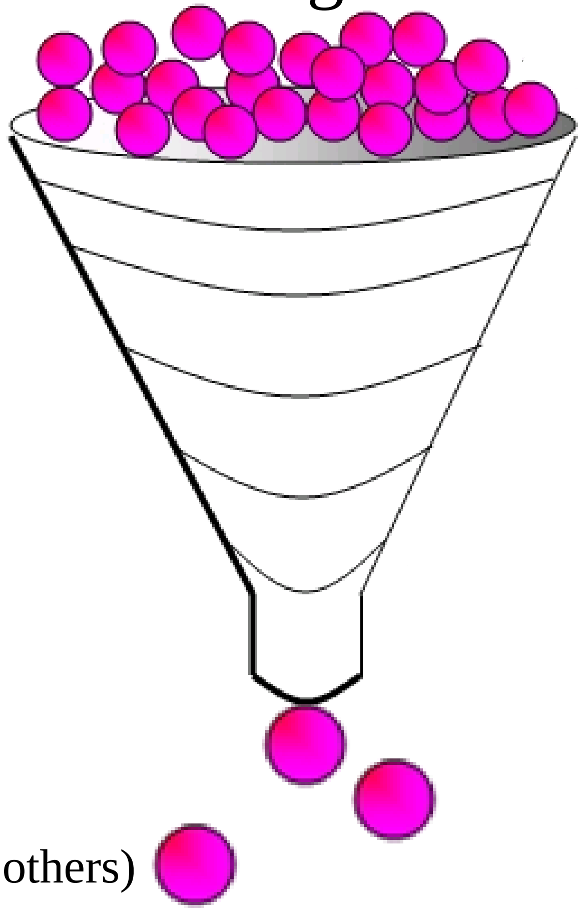
Speedup: 115168X (ideal 130816)

Efficiency: 88%

Many Many Tasks: Identifying Potential Drug Targets

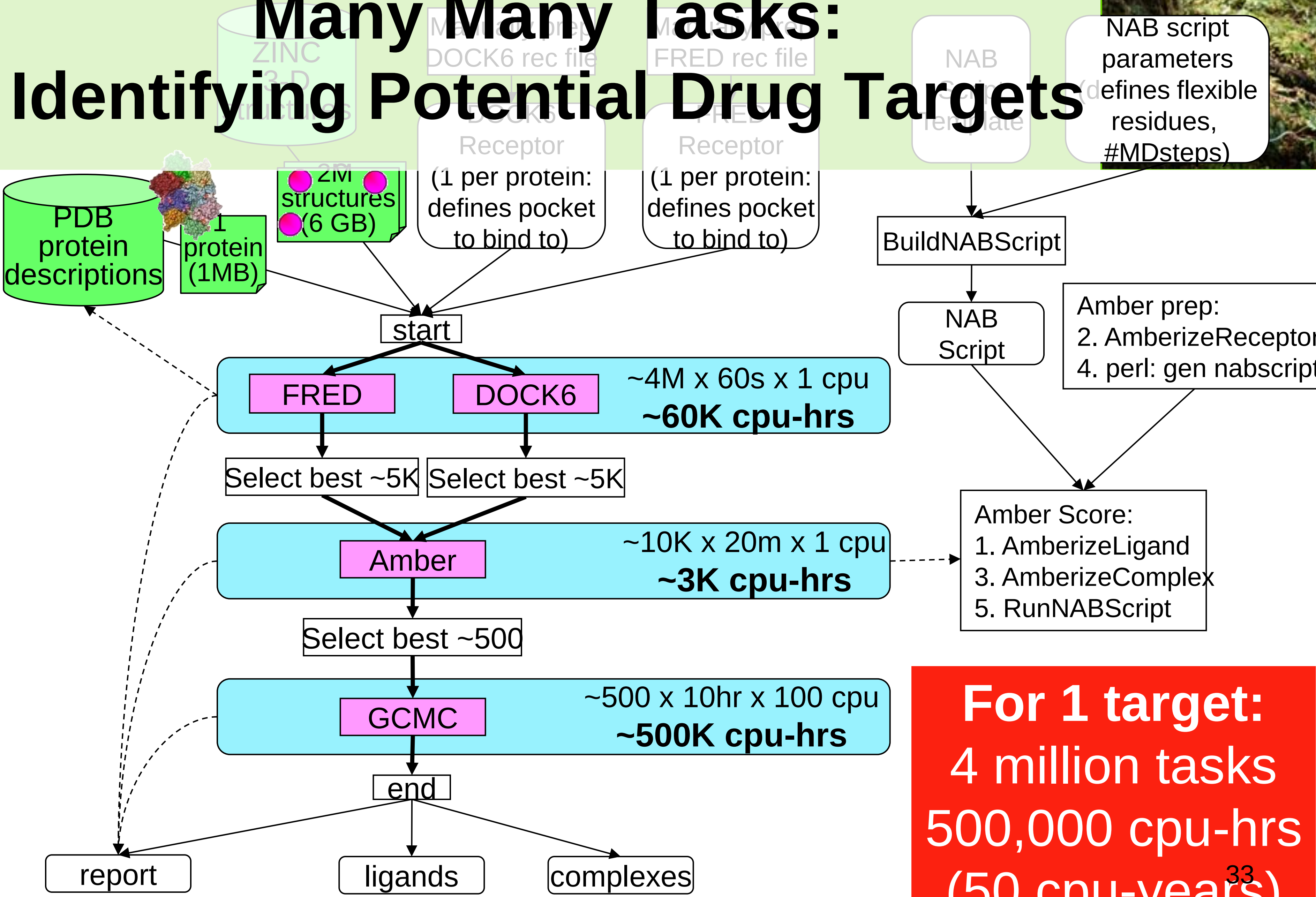
Protein target(s) x 2M+ ligands





(Mike Kubal, Benoit Roux, and others)

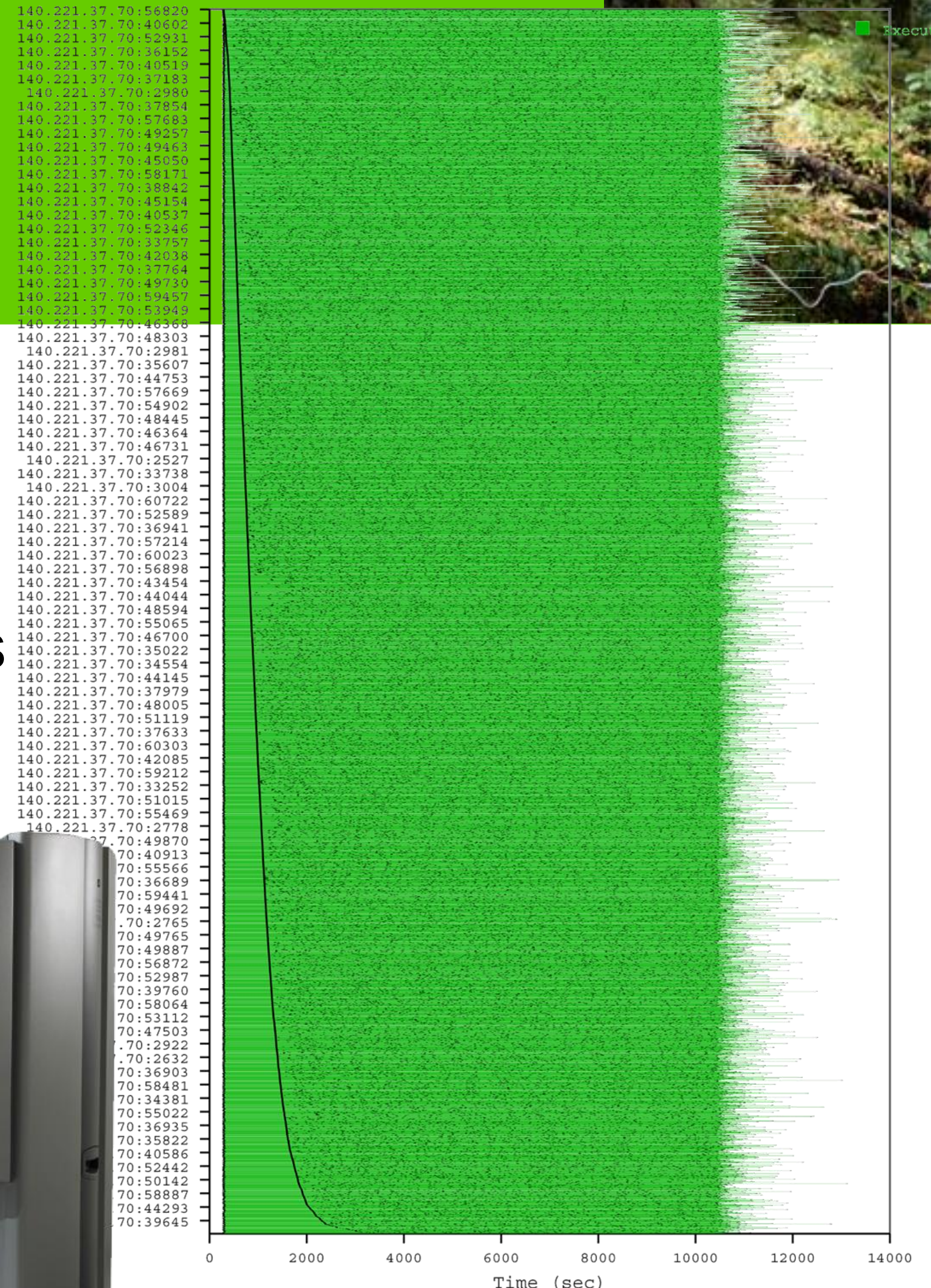
Many Many Tasks: Identifying Potential Drug Targets



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

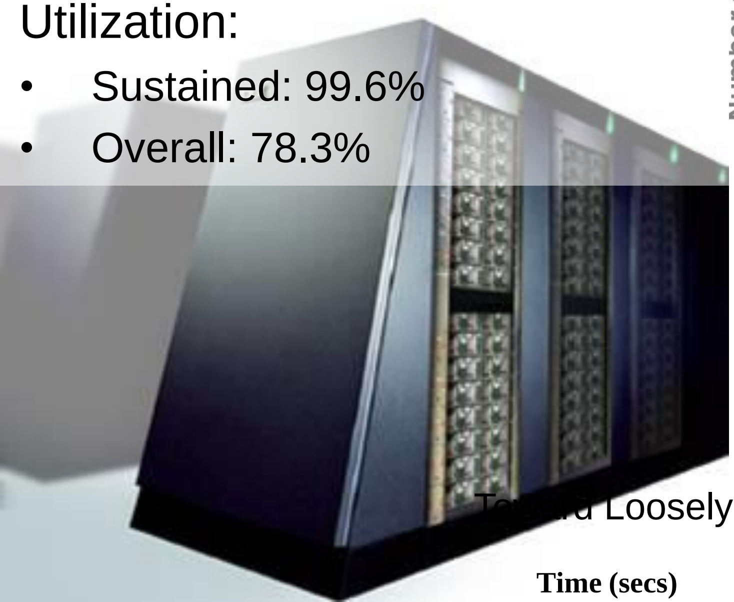
Average task time: 667 sec

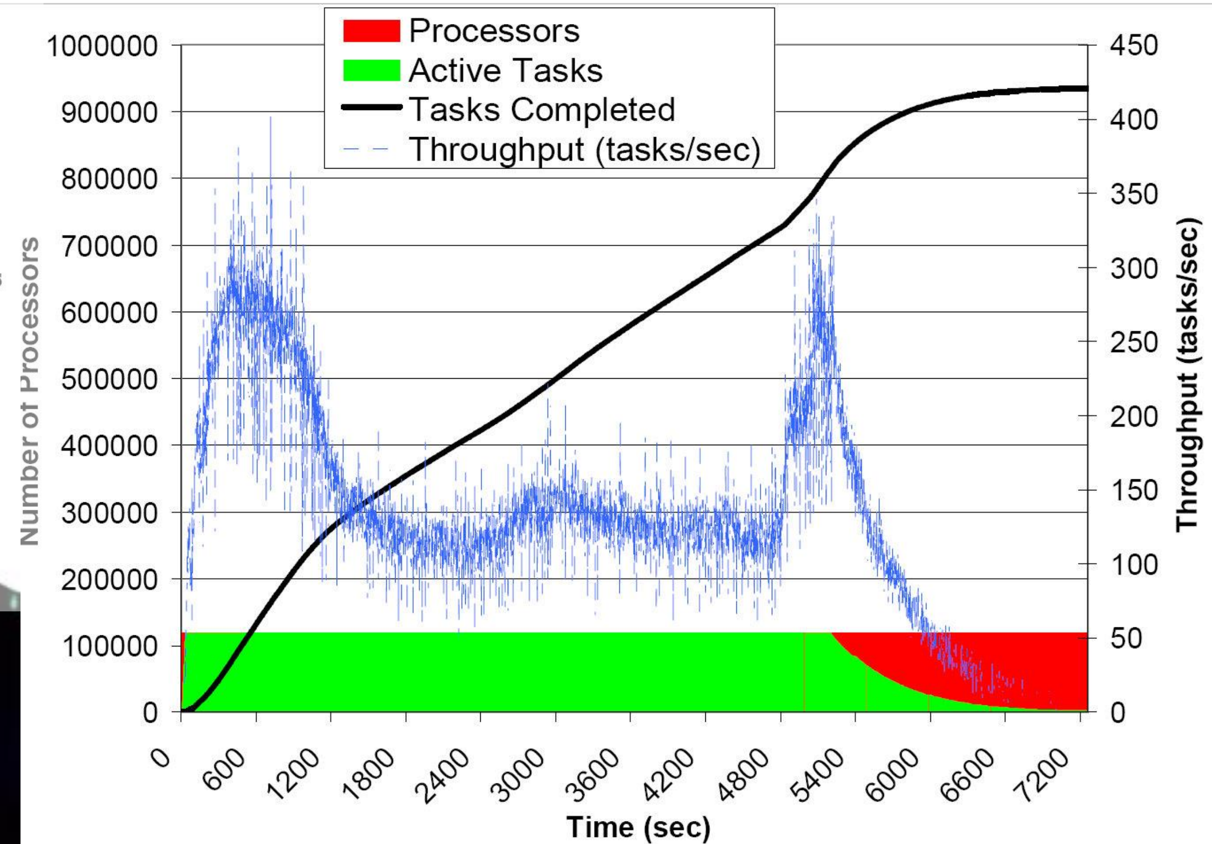
Relative Efficiency: 99.7%

(from 16 to 32 racks)

Utilization:

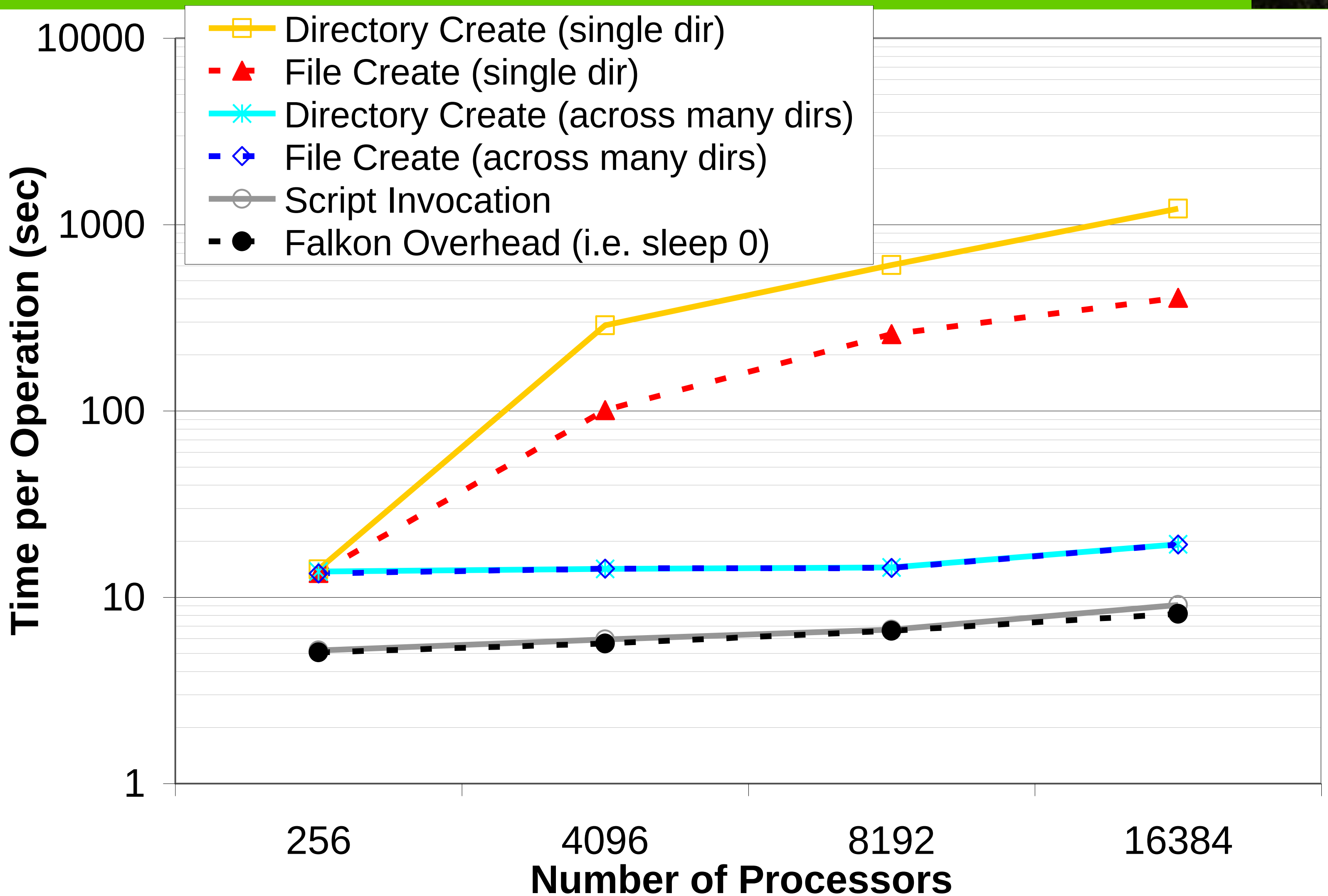
- Sustained: 99.6%
- Overall: 78.3%



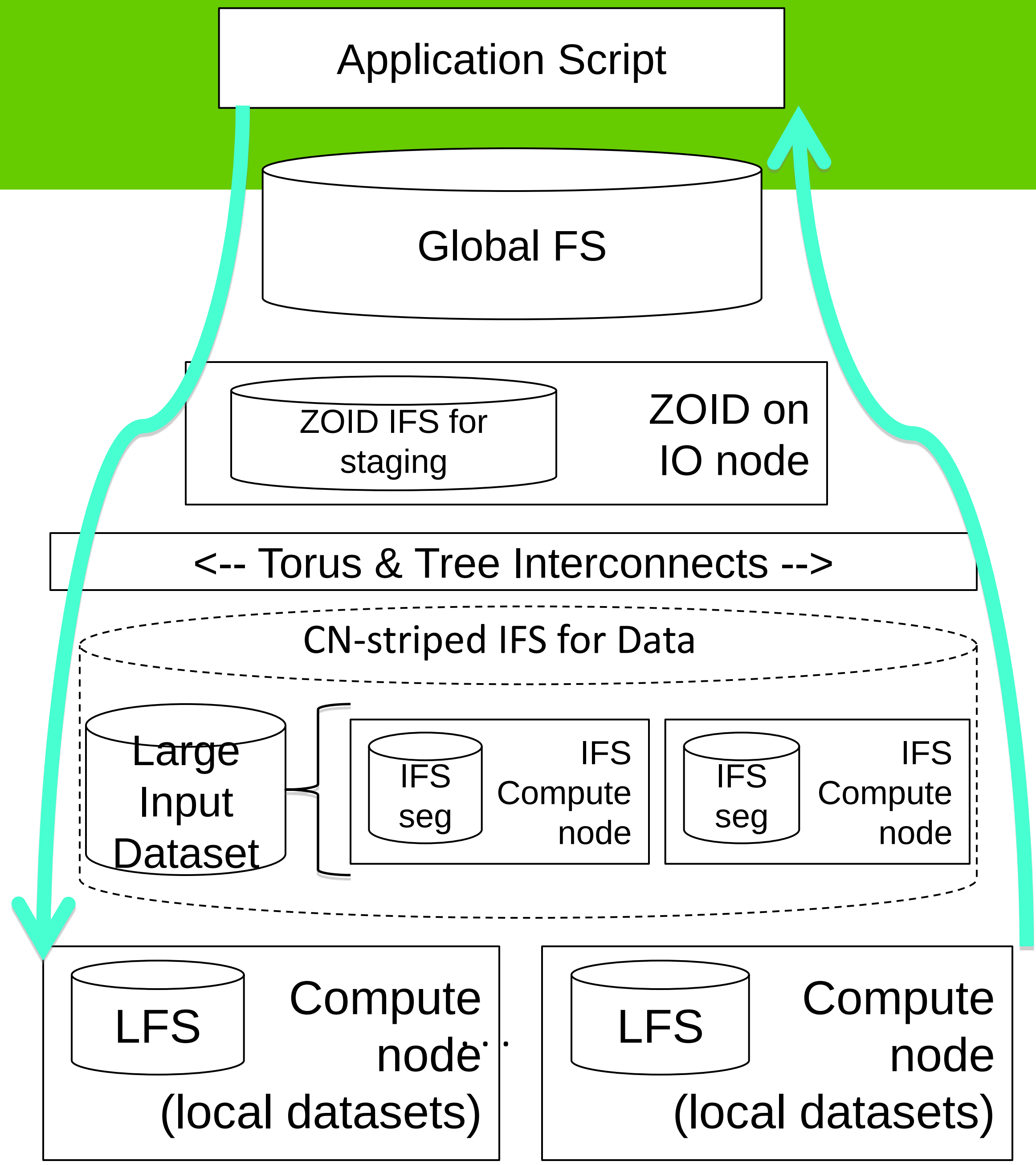


Time (secs)

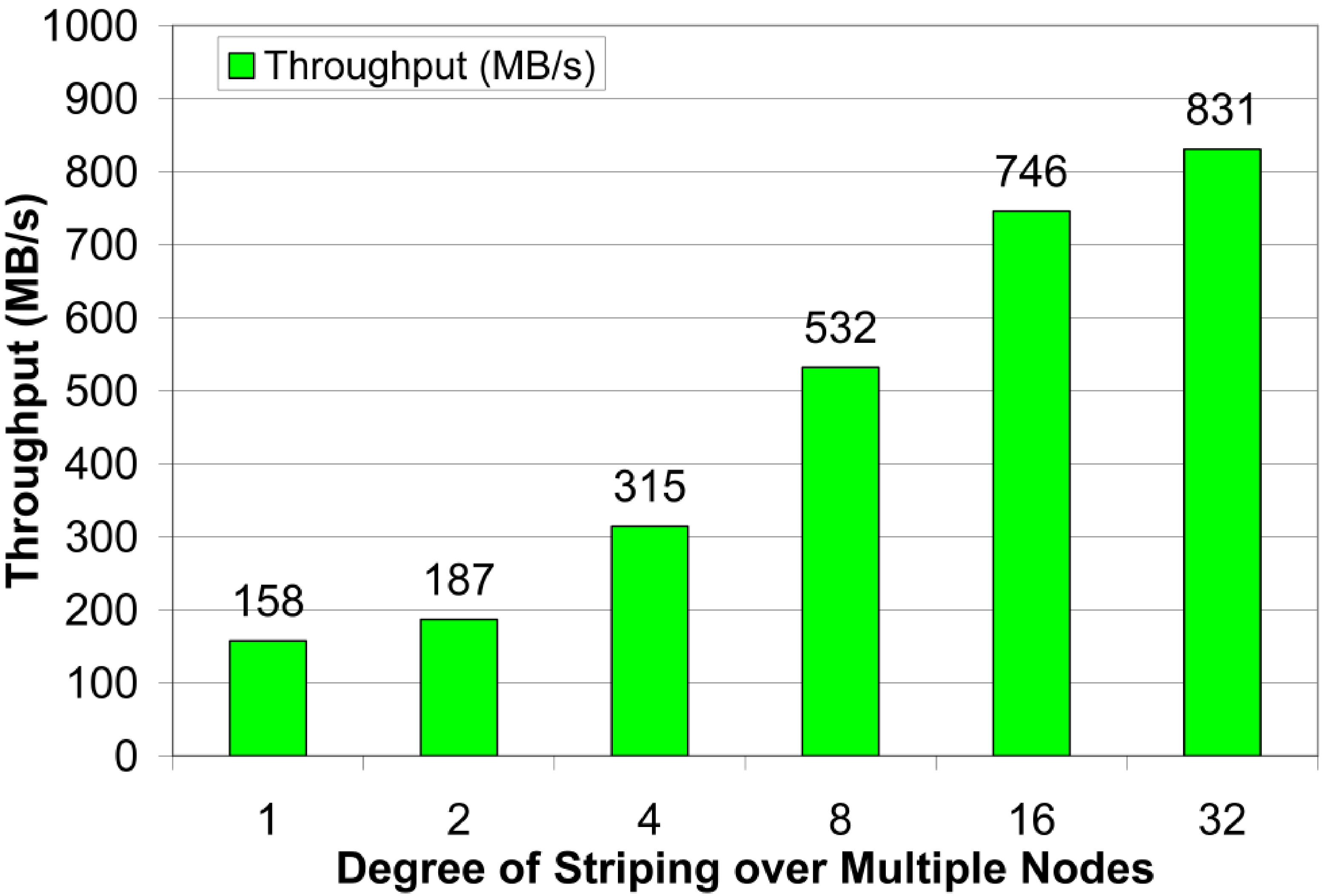
Costs to interact with GPFS



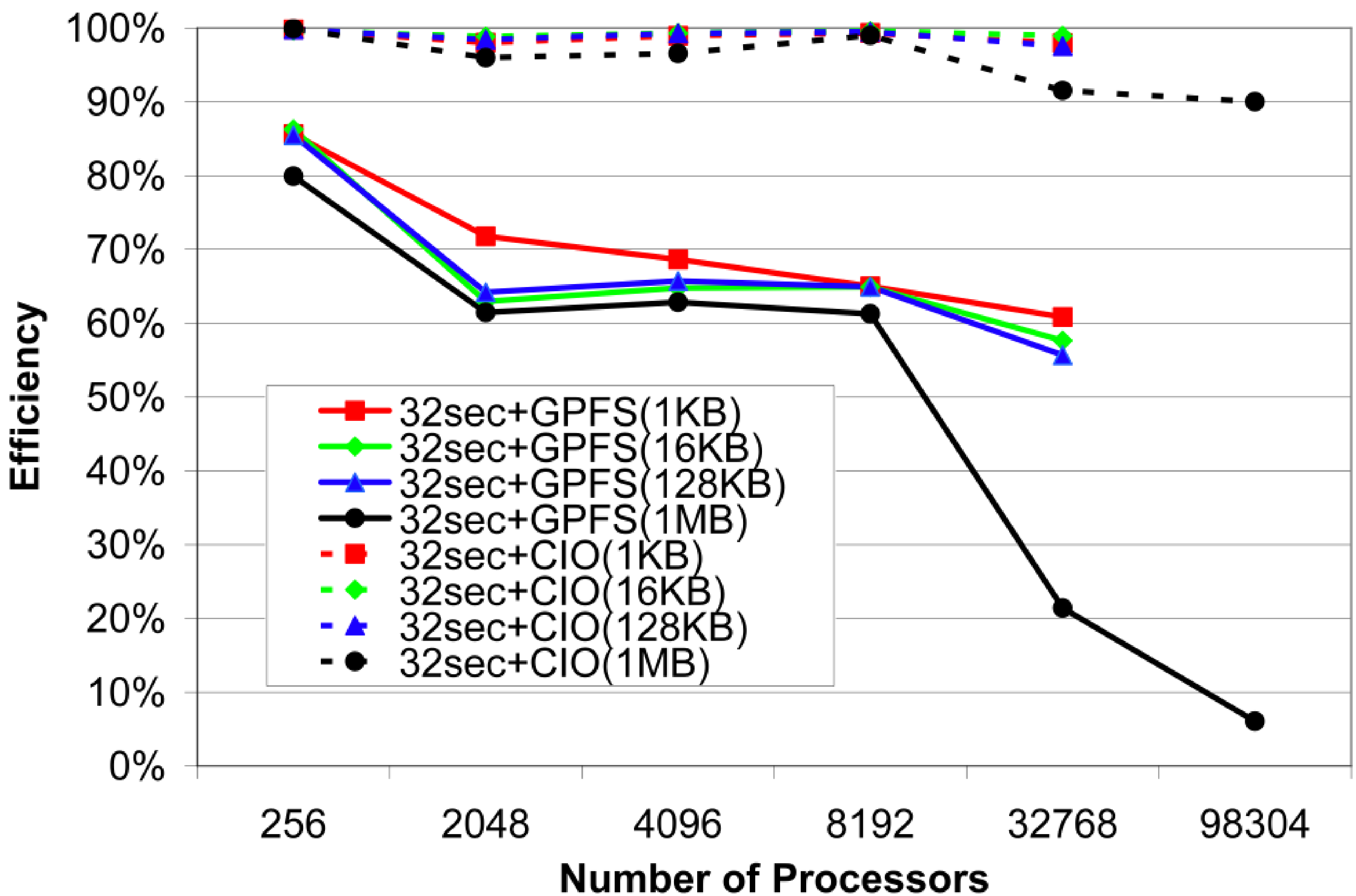
LCP Collective IO Model



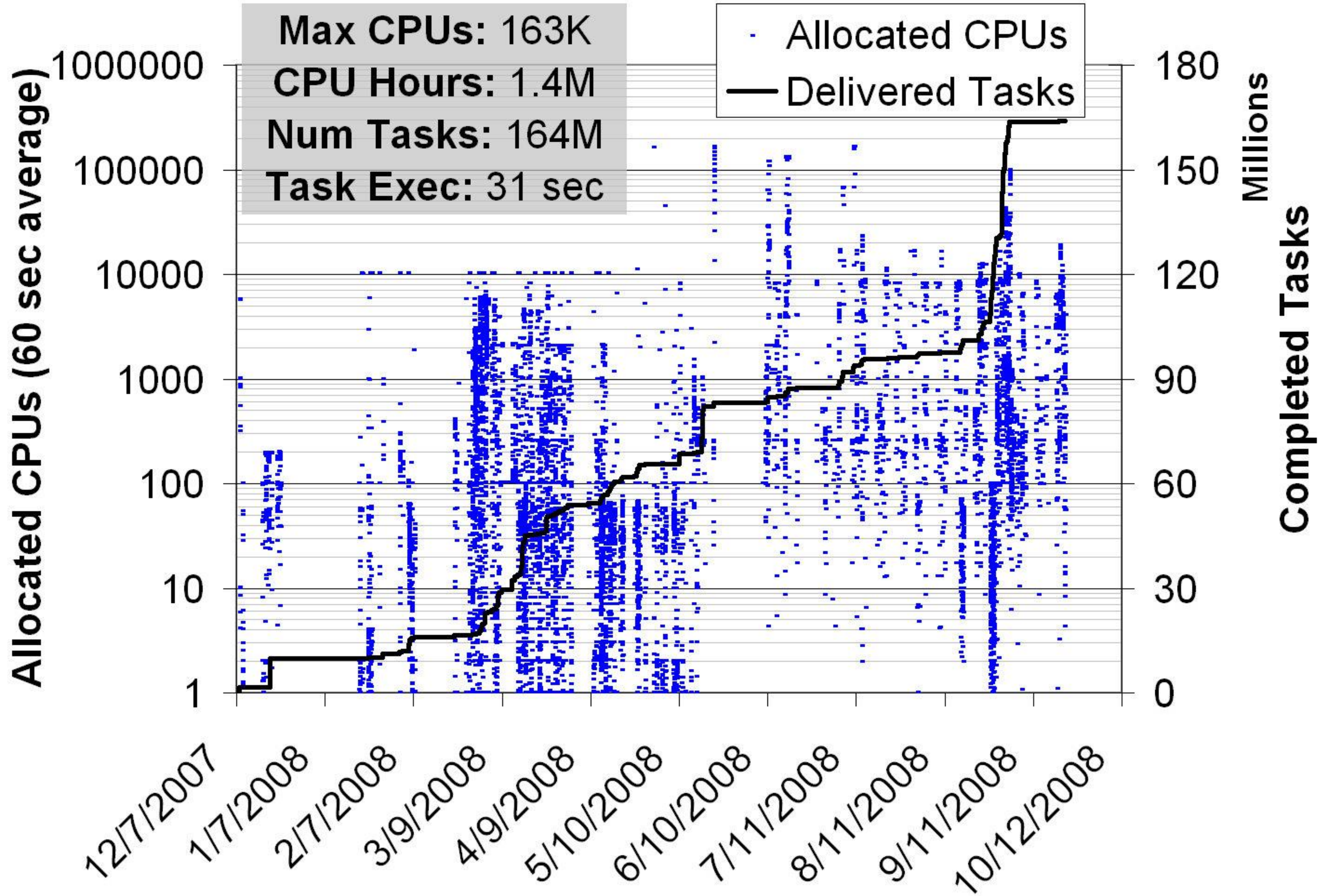
Read performance from IFS



Write Performance
CIO vs. GFS efficiency



Falkon Activity History (10 months)



PART IV

Conclusions and Future Work

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

Conclusions & Contributions

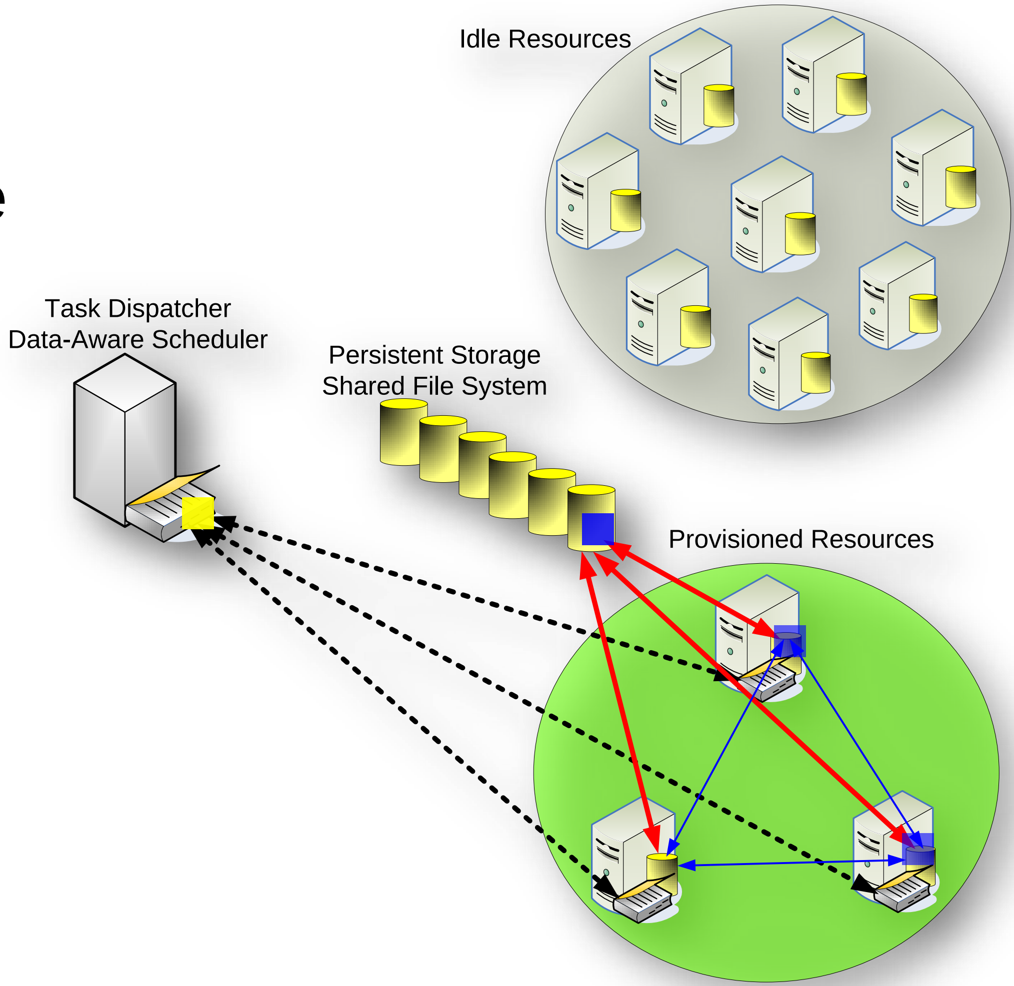
- Defined a new class of applications: MTC
- Proved that MTC applications can be executed efficiently on supercomputers at full scale
- Extended Falkon by distributing the dispatcher/scheduler
- Falkon installed and configured on the BG/P for anyone to use

Future Work: Other Supercomputers

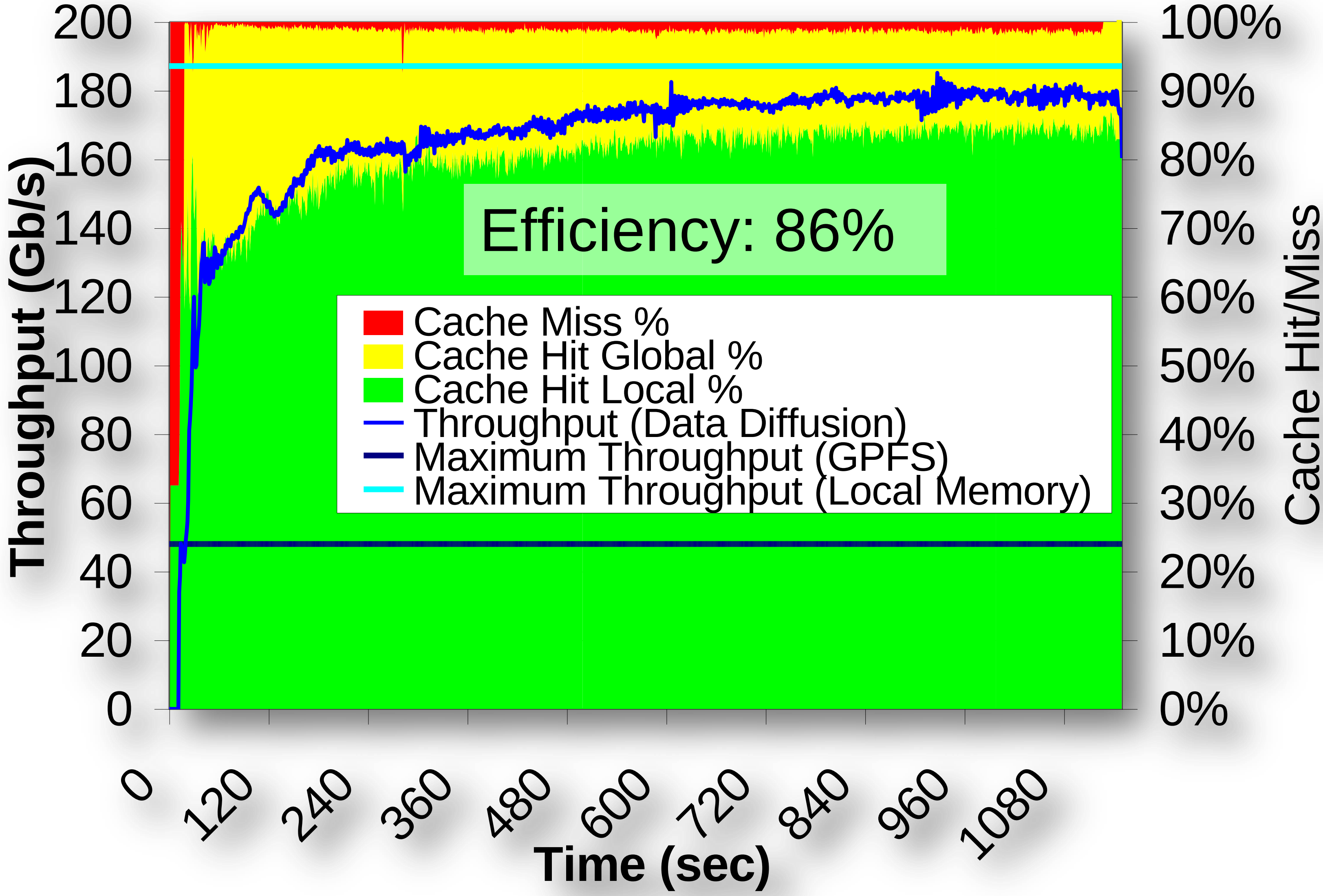
- Ranger: Sun Constellation
  - Basic mechanisms in place, and have started testing
- Jaguar: Cray
  - Plan to get accounts on machine as soon as its online
- Future Blue Gene machines (Q?)
  - Discussions underway between IBM, ANL and UChicago

Future Work: 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



All-Pairs Workload 1000x1000 on 4K emulated CPUs



More Information

- More information: http://people.cs.uchicago.edu/~iraicu/
- Related Projects:
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - Swift: http://www.ci.uchicago.edu/swift/index.php
- Funding:
  - **NASA**: Ames Research Center, Graduate Student Research Program
    - Jerry C. Yan, NASA GSRP Research Advisor
  - **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