Continuous Adaptation for High Performance Throughput Computing across Distributed Clusters

Edward Walker

Agenda

- MyCluster
  - Interface
  - Architecture
- Job proxy migration technique
  - Liability-adjusted throughput measure
  - Paxos master selection algoritm
- Experimental results
  - Simulation
  - Real-world implementation

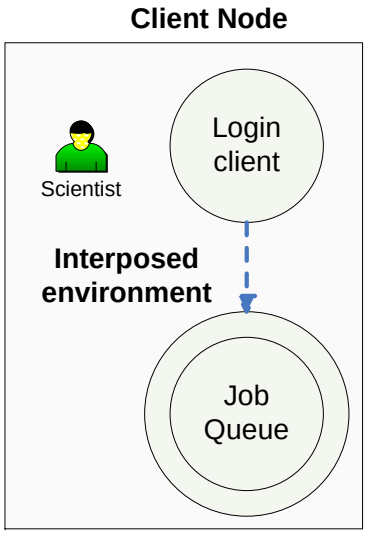
MyCluster

- Builds personal clusters on-demand for scientists
- Uses the concept of job proxies to …
  - Aggregates resources across sites.
  - Provisions resources for some period of time.
  - Provides a single job management interface.
    - I.e. Condor, OpenPBS, Sun Grid Engine, etc.
  - Maximizes user job throughput by reacting adaptively to changing cluster load conditions

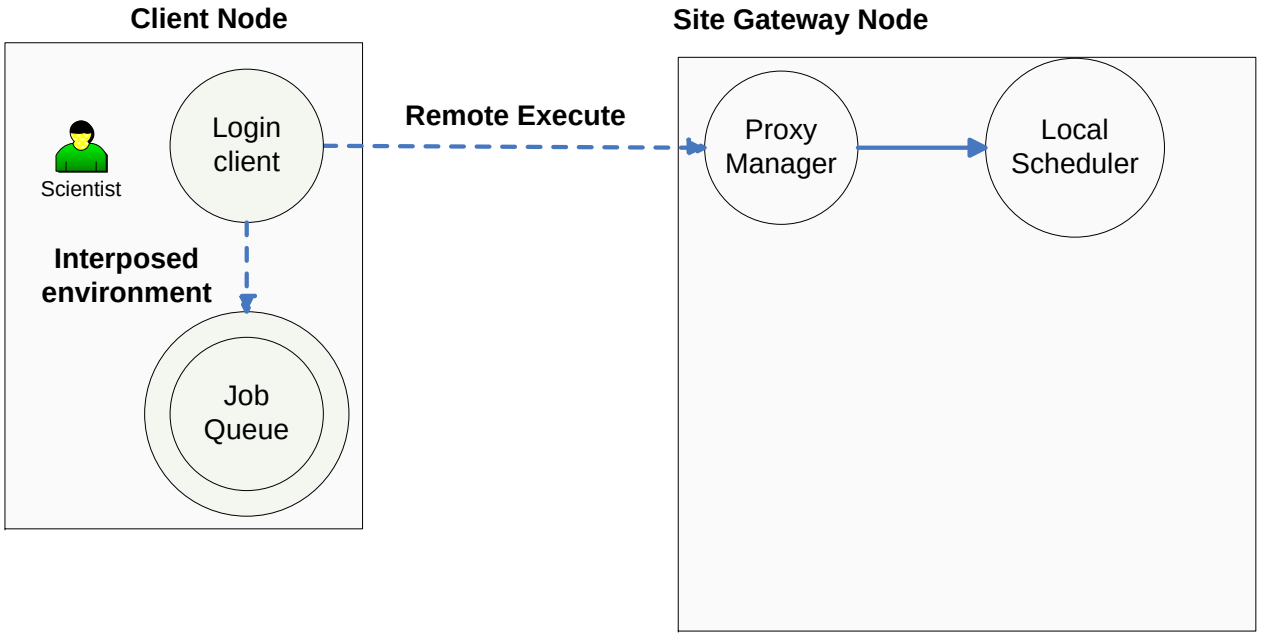
The **virtual login** session

```
% vo-login ### or ec2_pool
Enter GRID passphrase:                  ← GRAM or SSH login
Spawning on lonestar.tacc.utexas.edu
Spawning on tg-login2.ncsa.teragrid.org
Setting up VO participants ......Done
Welcome to your MyCluster/Condor environment
To shutdown environment, type "gexit"
To detach from environment, type "detach"

mycluster(gtcsh.9676)%
mycluster(gtcsh.9676)% condor_status

Name          OpSys       Arch   State      Activity   LoadAv Mem   ActvtyTime

32020@compute LINUX       INTEL  Unclaimed  Idle       0.000  2026[?????]
…
32021@tg-c383 LINUX       IA64   Unclaimed  Idle       0.000  2026[?????]

                     Machines Owner Claimed Unclaimed Matched Preempting

          INTEL/LINUX        2     0       0         2       0          0
           IA64/LINUX        2     0       0         2       0          0

                 Total        4     0       0         4       0          0
mycluster(gtcsh.9676)% gexit
%
```

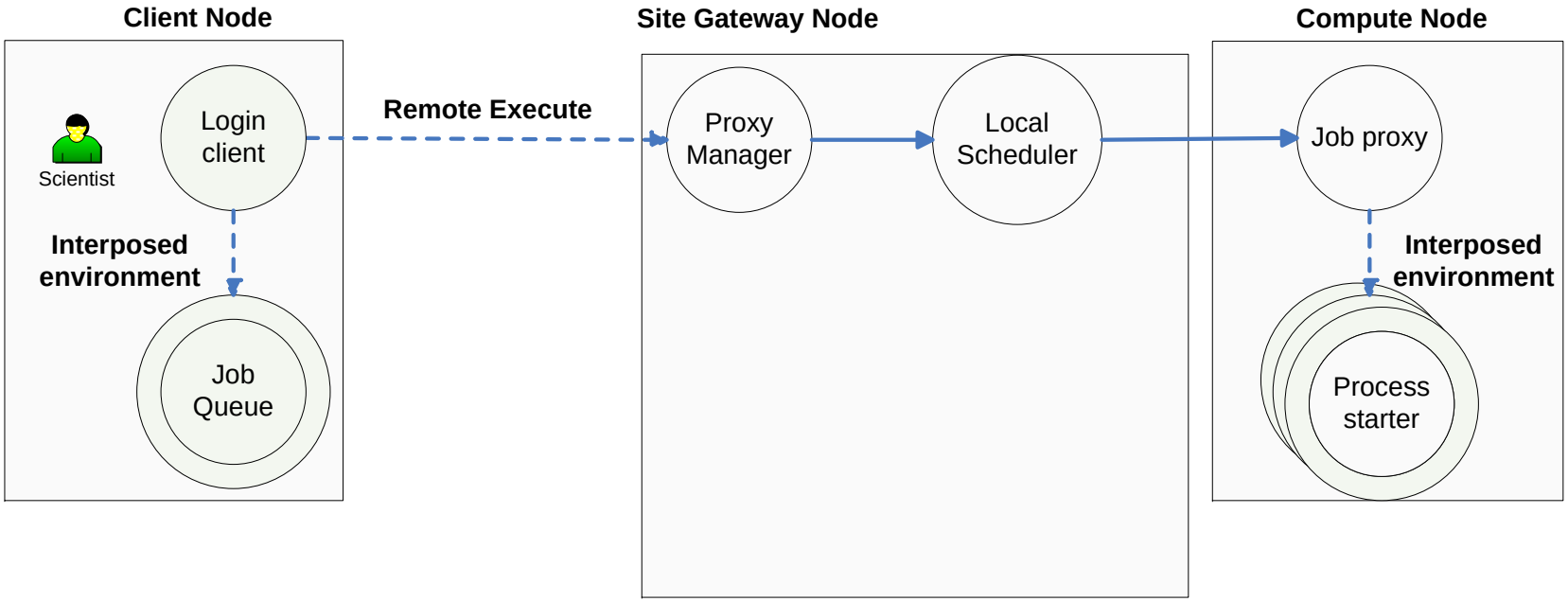
MyCluster Process Architecture



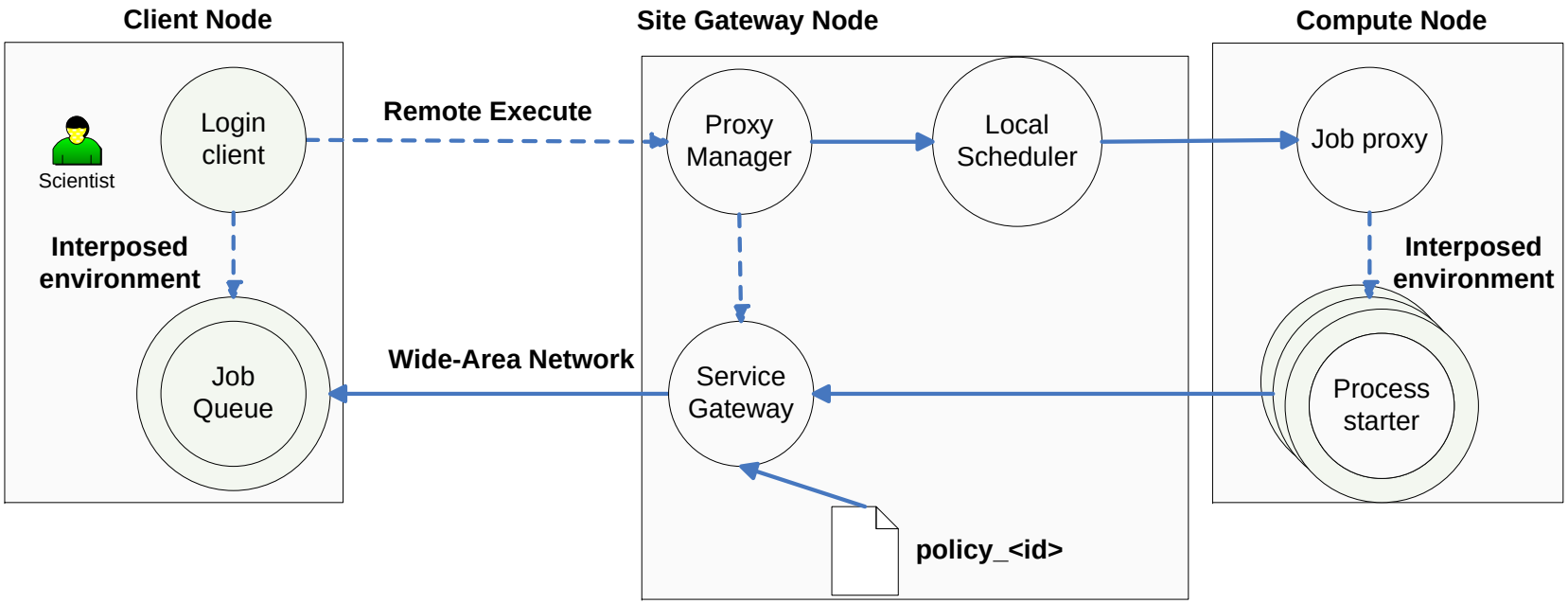
MyCluster Process Architecture



MyCluster Process Architecture



MyCluster Process Architecture



Current job proxy distribution strategy

- Default is static:
  - User needs to specify how many job proxies, and what their job sizes are, should be submitted at each contributing site.
- Dynamic job proxy migration can be enabled:
  - Moves job proxies from sites with pending job proxies to sites with all its current job proxies running.
  - Does not take into account the actual performance contributed by each site.
  - E.g.
    - Site A has 1 running job proxy and no pending job proxies (job proxies were migrated away due to past poor performance).
    - Site B has 10 running job proxies (and 2 pending job proxies)
    - The current technique will migrate the pending proxy from site B to site A (not desirable because of site A's historic poor performance)

Proposed new job proxy migration technique

- Intuition
  1. We want to send job proxies to clusters exhibiting the largest throughput
  2. We do not want to send job proxies to clusters that already have many pending job proxies
  3. We want to ensure that recent past performance is always taken into account (filter out “noise”)

Calculating the throughput and liability of a cluster

$$C(t) = \mathrm{R(t)} * \mathrm{S} * \mathrm{F}$$

Throughput at time instance t

$$\mathrm{L(t)} = \mathrm{P(t)} * \mathrm{S} * \mathrm{F}$$

Liability at time instance t

Where at time instance t

R(t)	**number of running jobs**
P(t)	**number of pending jobs**
S	**job proxy size**
F	**normalized FLOPS rate of cluster processors**

Liability-adjusted throughput

$C(t) = \text{running CPUs at time t}$

$L(t) = \text{pending CPUs at time t}$

$$\tau(t) = C(t) - L(t)$$

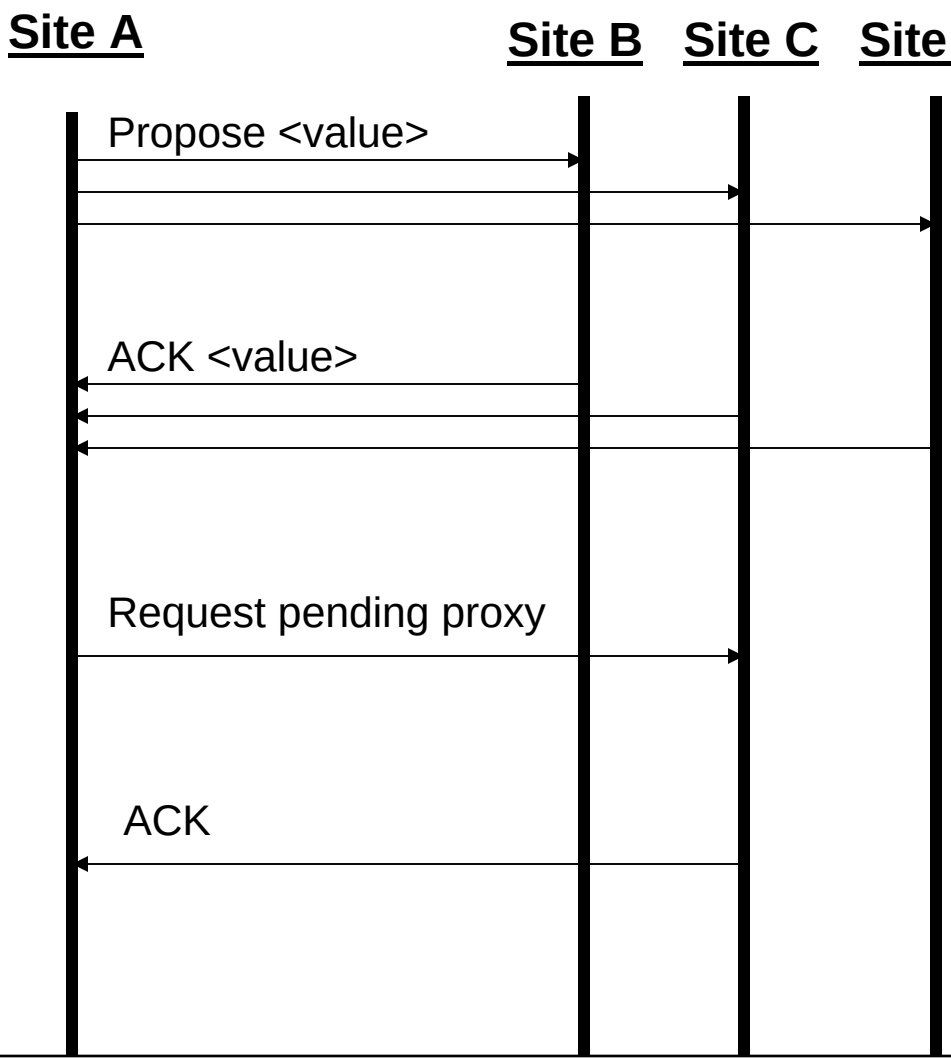
← Liability-adjust throughput

$$T(t) = \tau(t) + \frac{1}{2}\tau(t-1) + \frac{1}{3}\tau(t-2) + \frac{1}{4}\tau(t-3) + \ldots$$

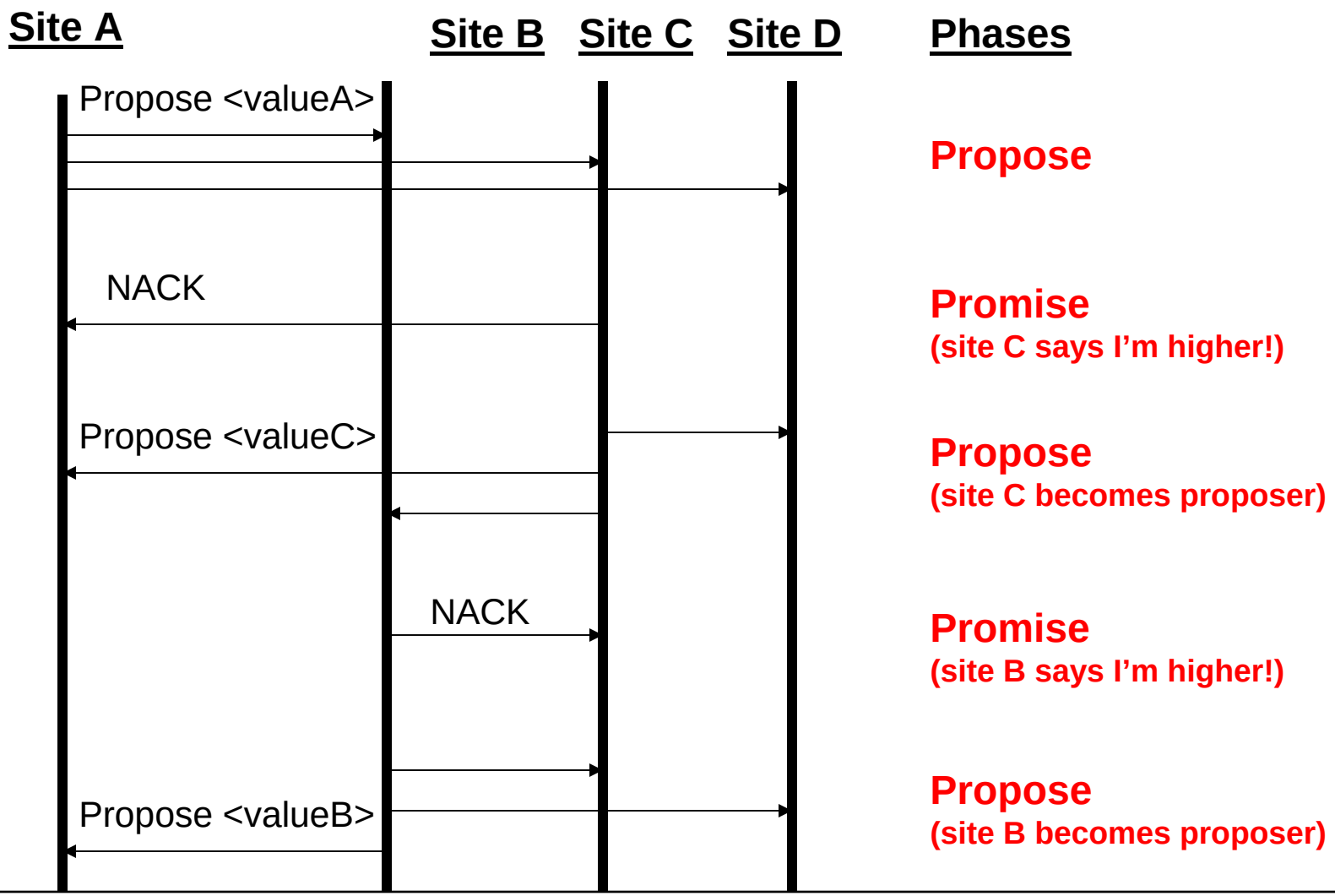
Incorporate past throughput performance

$$\Rightarrow T(t) = \sum_{n=1}^{k} \frac{1}{n}\tau(t-n+1)$$

Paxos master selection algorithm

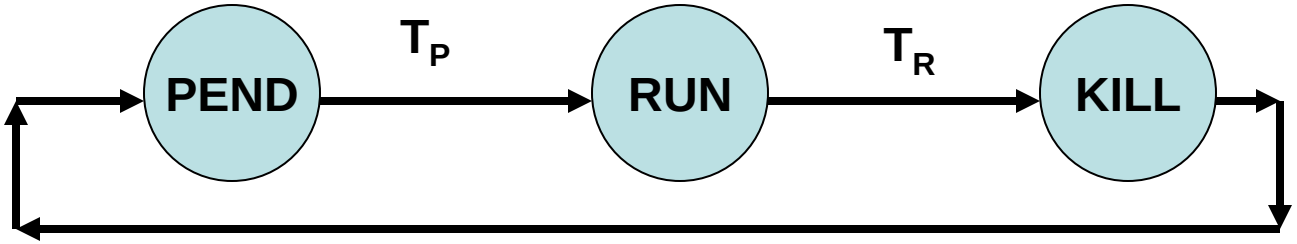


Competing proposers



Simulation Results

Simulator: probability of job dispatch



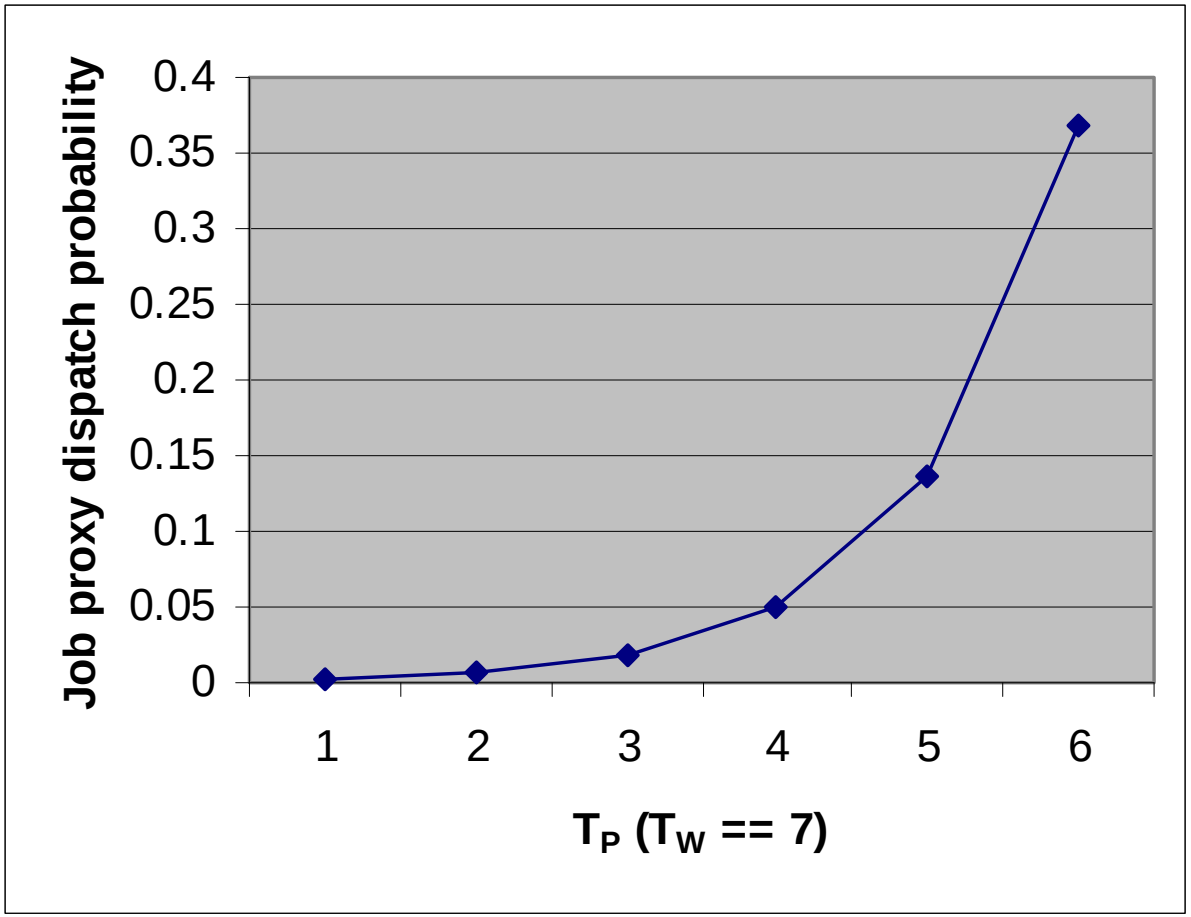
Job pending time

$$E(J) = \begin{cases} \dfrac{1}{e^{T_W - T_P}} & \text{if } T_P < T_W \\ \dfrac{1}{e} + \dfrac{1}{N_R + 2} & \text{if } T_P >= T_W \end{cases}$$

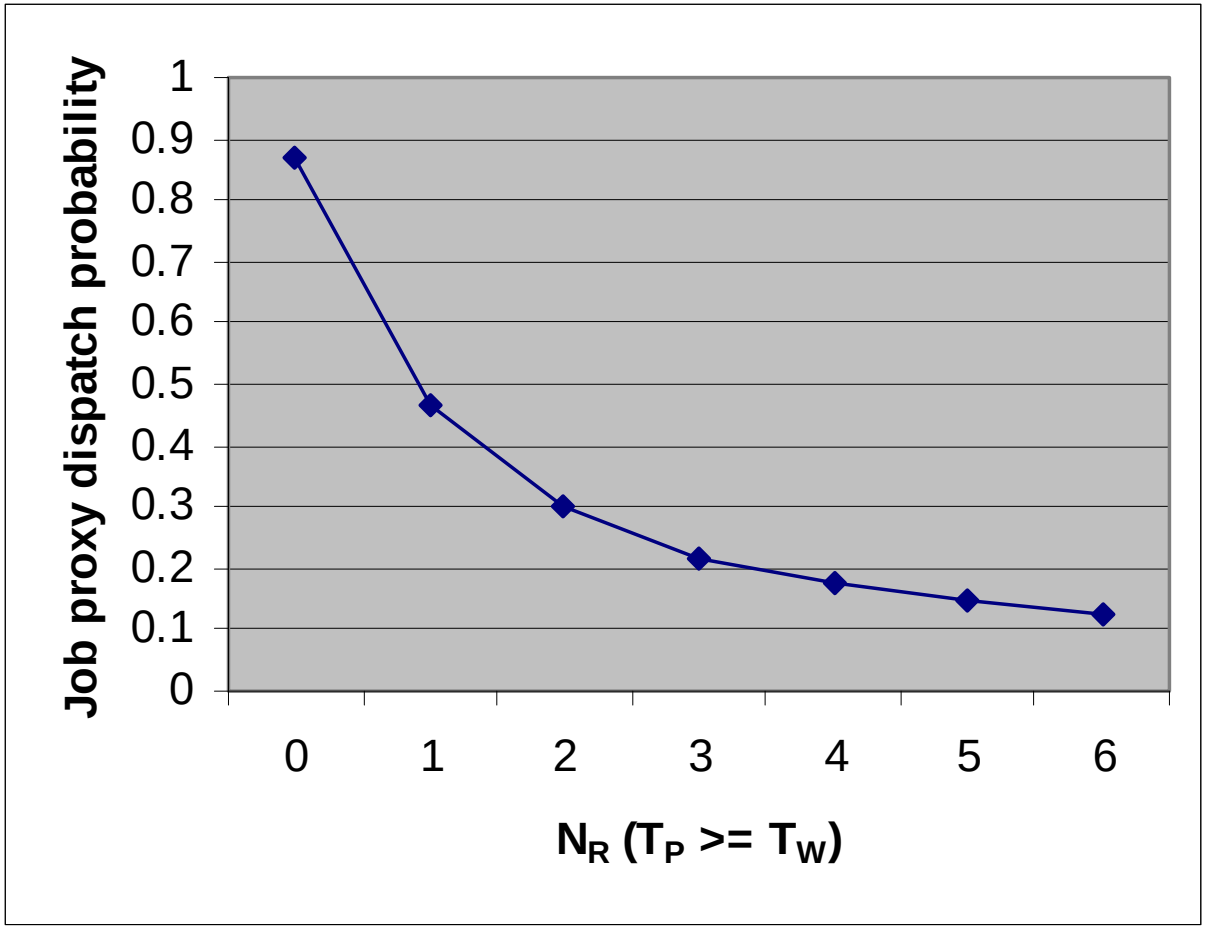
Expected queue wait time

Number of running jobs

Similator: Probability of job dispatch ($T_P < T_W$)



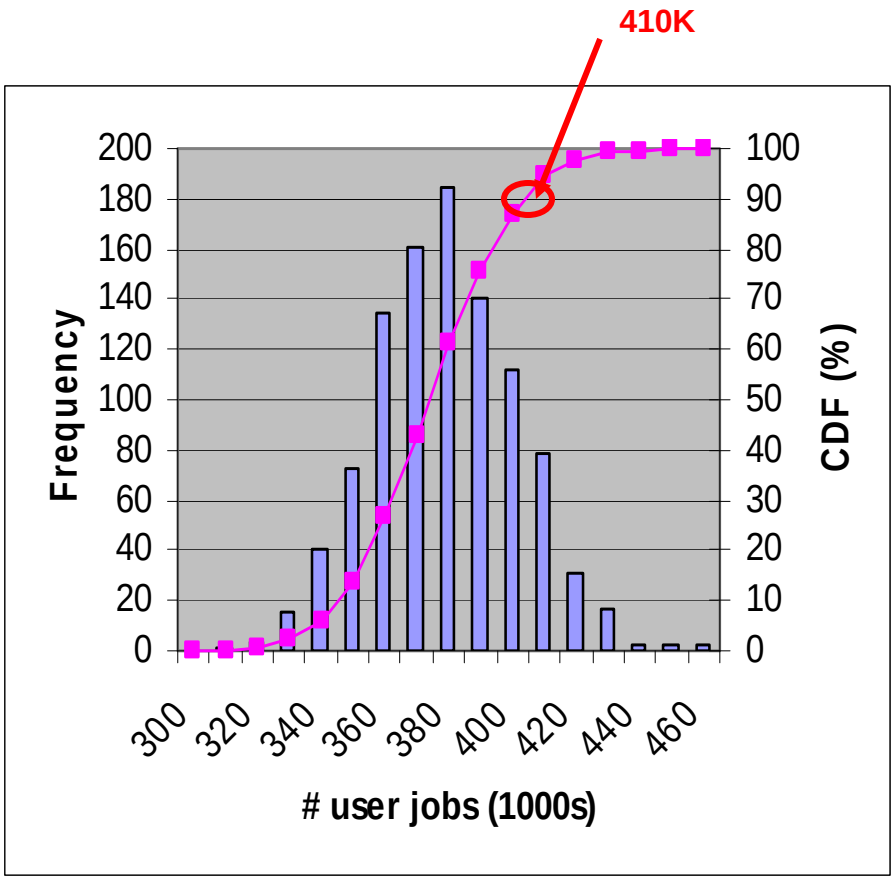
Simulator: Probability of job dispatch ($T_P >= T_W$ and N_R=[0,6])



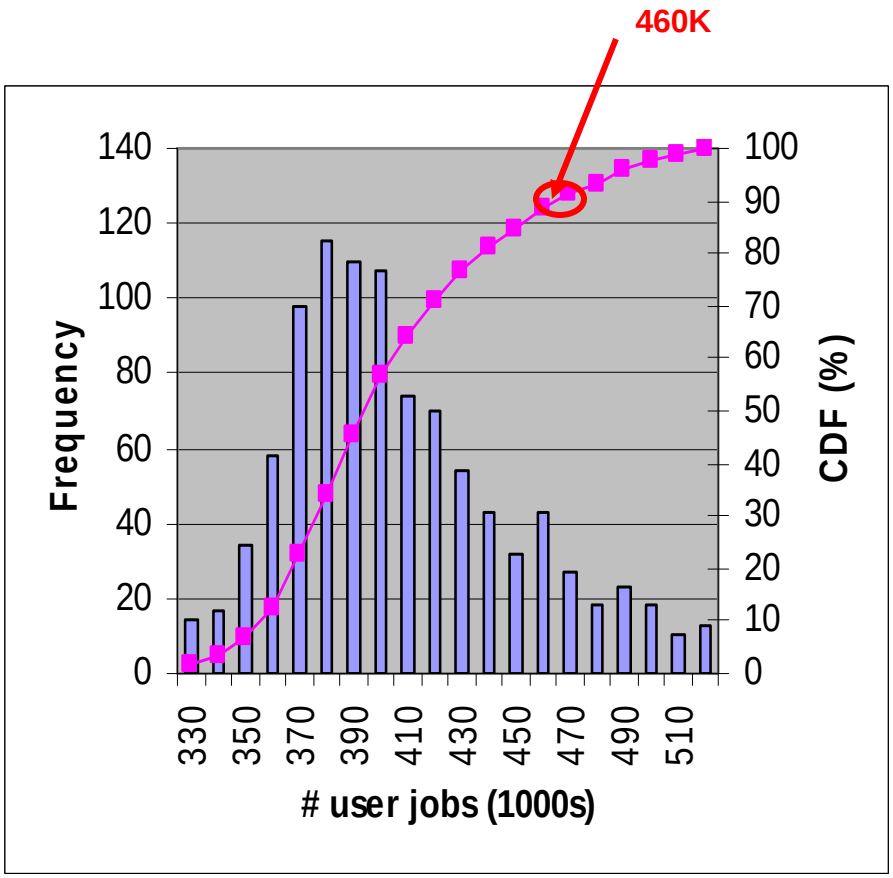
Simulation parameters

Description	Value
Number of clusters	16
Job proxy size	32
Job proxy runtime	4 ticks
Average user job runtime	1 tick
User job size	4
Average queue wait time	[2,20] ticks
Cluster normalized FLOPS	[1,3]
Total simulation	600 ticks

CDF of simulation runs (1000 runs)



Migration disabled



Migration enabled

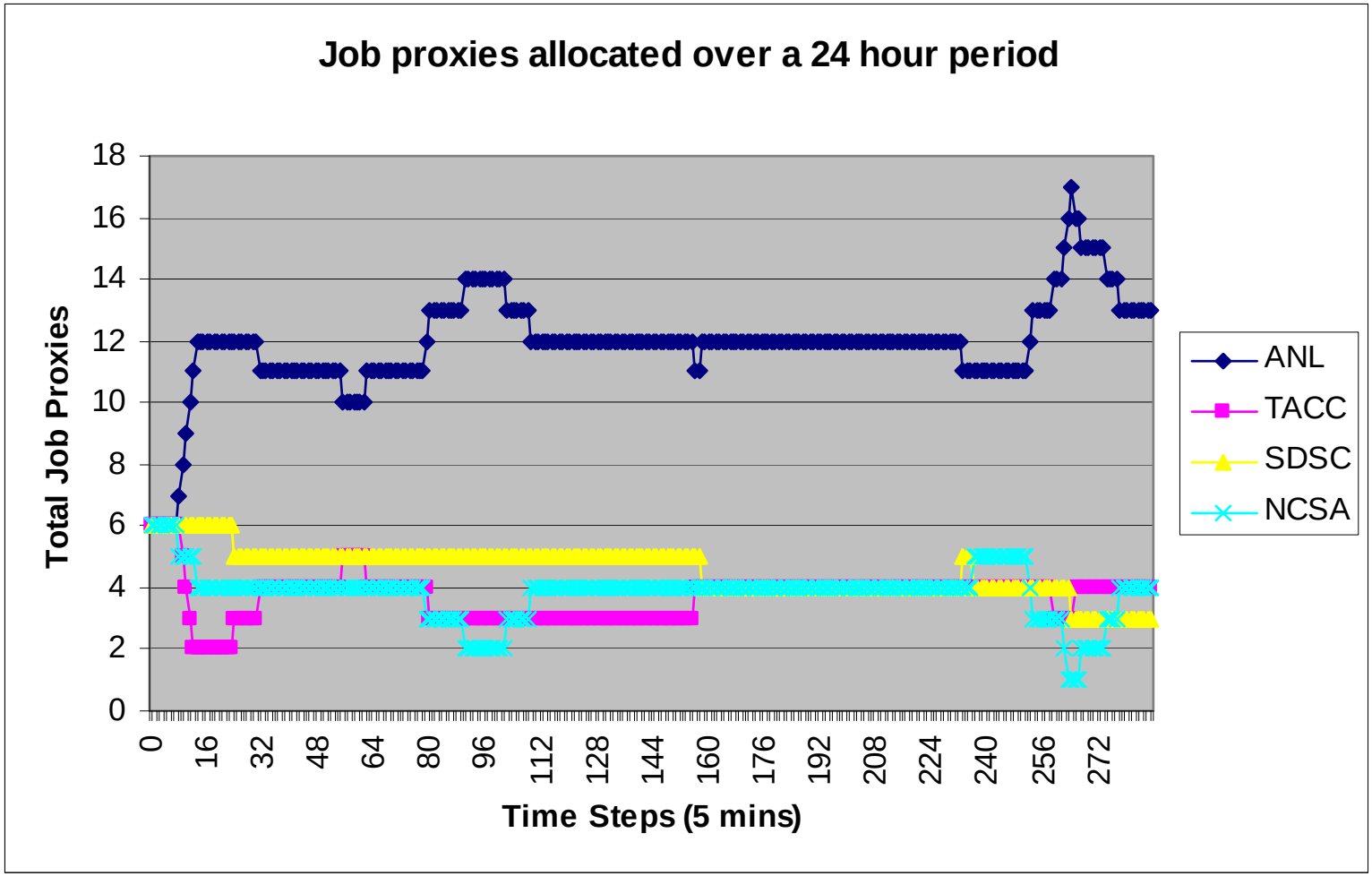
TACC

Real-world experiment

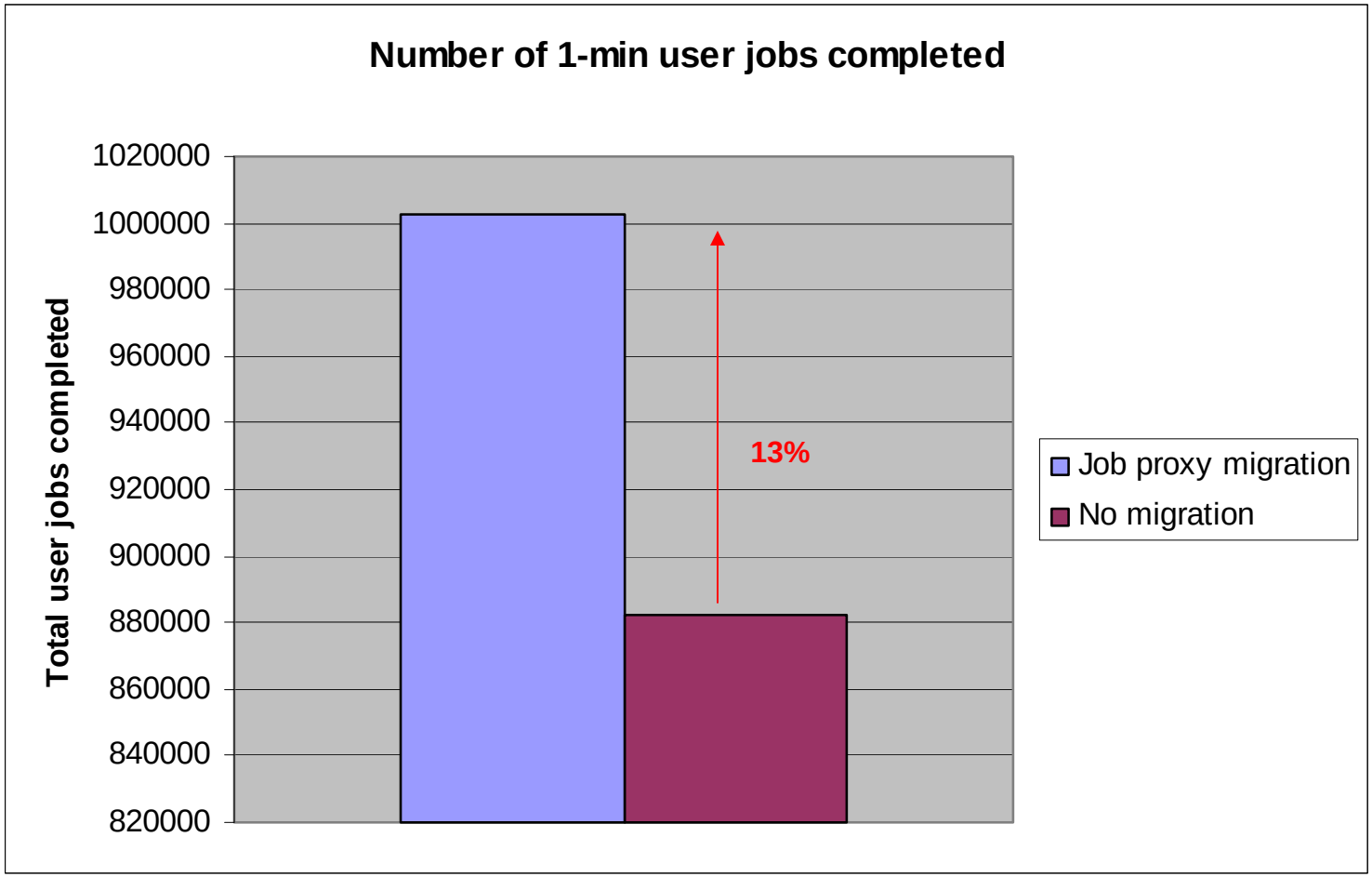
Experiment parameters

- Initial distribution:
  - 6 job proxies at each site of job size of 32 CPUs
- Running time:
  - 24 hours
- Frequency of migration:
  - Every 5 mins
- Job:
  - “sleep 60”
  - Therefore normalized FLOPS => F == 1

Experiment: proxy migration initiated every 5 mins over a 24 hour period



Aggregate throughput (# of 1 minute jobs through personal cluster) with and without proxy migration



Related work

- Singh et. al [1] attempts to provision CPU slots by solving a multi-objective genetic algorithm
  - Assumes the available CPU slots from contributing clusters are known ahead of time
- Raicu et. al. [2] have implemented a provisioner component in the Falkon system.
  - Implemented policies for CPU acquisition and retirement.
  - However, they do not take into account dynamic information like throughput in their policies.
- Sotomayor et. al. [3] (in the *virtual workspace* system) provision CPU resources using a combination of advanced reservation and best-effort scheduling.
  - However, they do not take into account dynamic information.

References

1. G. Singh, C. Kesselman, and E. Deelman, “A Provisioning Model and its Comparison with Best-Effort for Performance-Cost Optimization in Grids”, HPDC’07.
2. I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, “Falkon: a Fast and Light-weight tasK executiON framework”, SC’07
3. B. Sotomayor, K. Keahey, and I. Foster, “Combining Batch Execution and Leasing Using Virtual Machines”, HPDC’08.