Bricks: A Performance Evaluation System for Scheduling Algorithms on the Grids

Atsuko Takefusa

Tokyo Institute of Technology

Japan Society for the Promotion of Science

http://ninf.is.titech.ac.jp/bricks/

Scheduling Studies on the Grids

- Application Level Scheduling
  - APST, AMWAT (AppLeS)
  - MW (Condor)
  - Prophet, stochastic scheduling, performance surface, ...
- Job Scheduling
  - Match-making (Condor)
  - Scheduler for network enabled servers (Ninf, NetSolve)
  - Computational economy (Nimrod, G-Commerce)

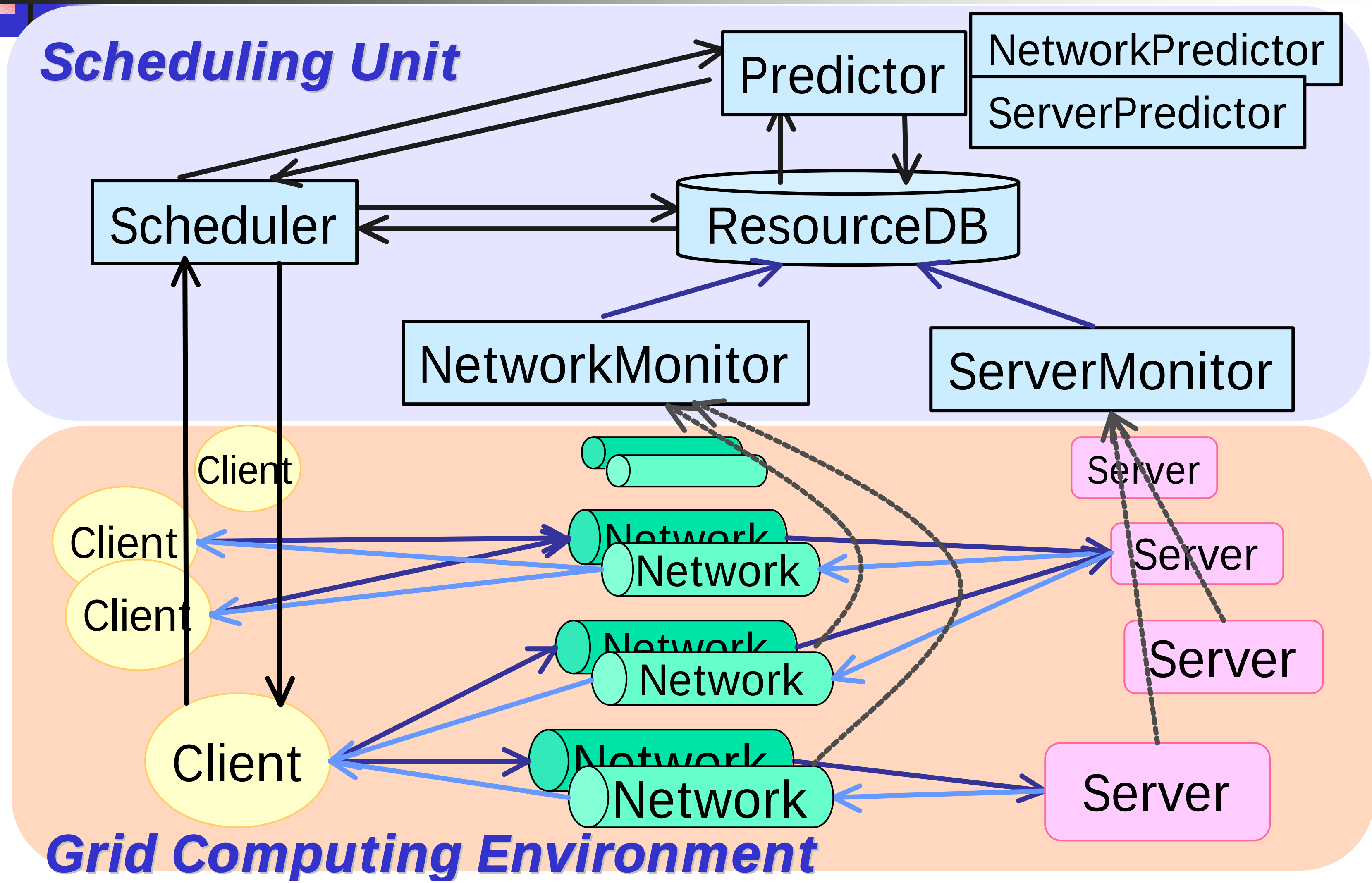
Evaluation of the Scheduling Algorithms

- Unrealistic to compare scheduling algorithms w/physical benchmarks
  - Reproducible large scale benchmarks are too difficult under various
    - Networks - topology, bandwidth, congestion, variance
    - Servers - architecture, performance, load, variance
- Validity of scheduling framework modules have not been well-investigated.
  - Benchmarking cost of monitor / predictor under real environment *HIGH*

A Performance Evaluation System: *Bricks*

- Performance evaluation system for
  - Scheduling algorithms
  - Scheduling framework components (e.g., sensors, predictors)
- Bricks provides
  - Reproducible and controlled evaluation environments
  - Flexible setups of simulation environments
  - Evaluation environment for external Grid components (e.g., NWS forecaster)

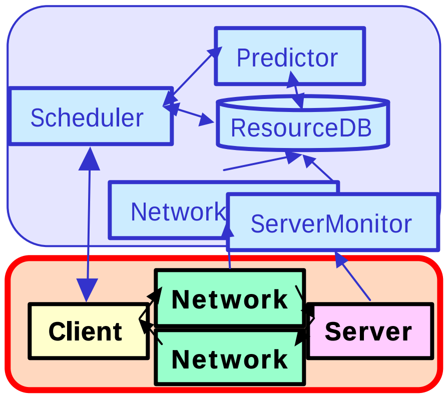
The Bricks Architecture



Grid Computing Environment

- Client
  - Represents user of the Grid system
  - Invokes Grid computing Jobs
    Amount of data transmitted to/from server,
    # of executed instructions
- Server
  - Represents computational resources
- Network
  - Represents the network interconnecting the Client and the Server

Represented using queues



Communication/Server Models using queues in Bricks

- Extraneous data/job model

  Congestion represented by adjusting the amount of arrival data/jobs from other nodes/users

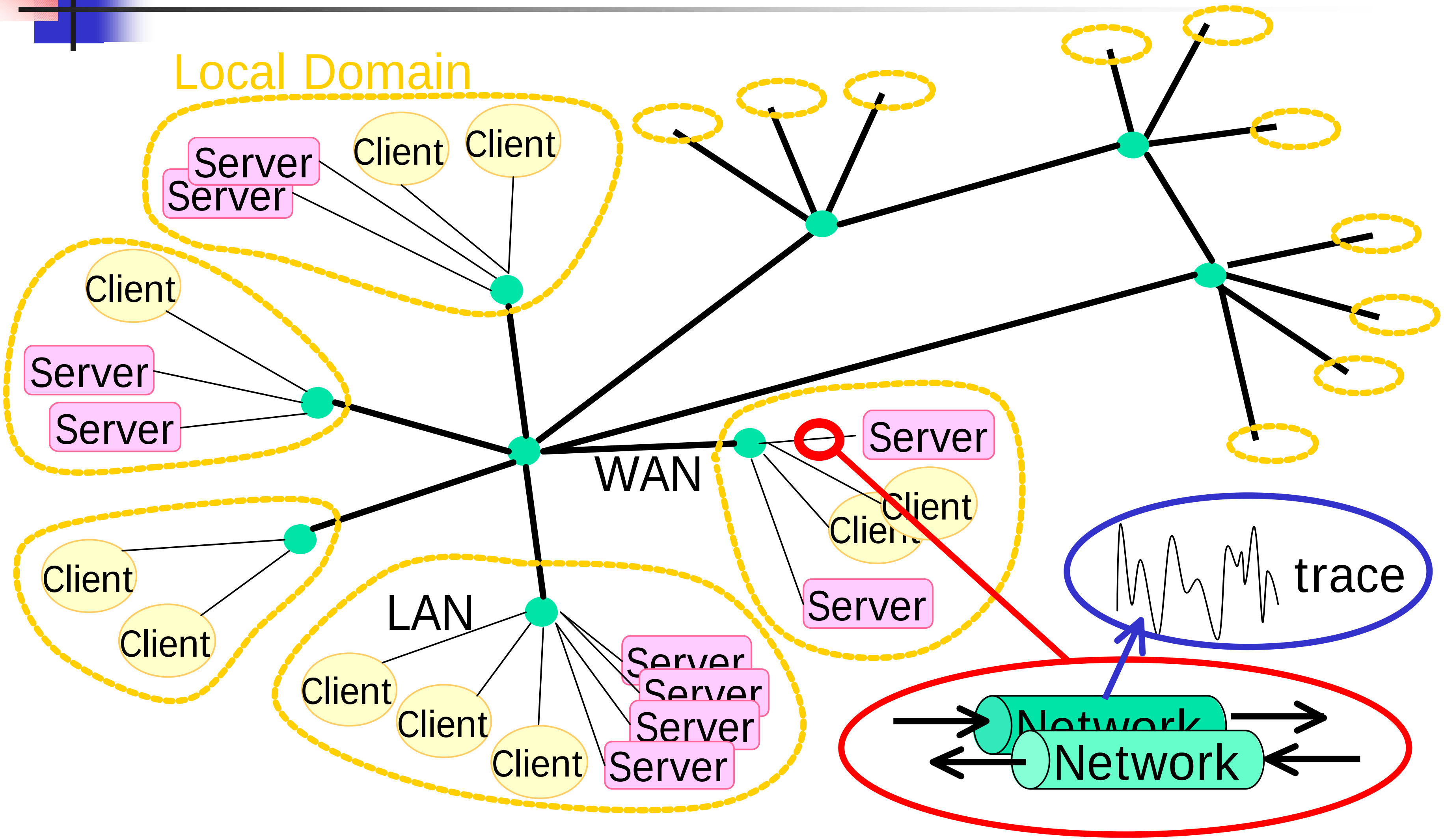
  ○ Need to specify only several parameters
  ✕ Greater accuracy requires larger simulation cost

- Trace model

  Bandwidth/performance at each step = trace data such as observed parameters of real environment.

  ○ Network/Server behaves as if real network/server
  ○ Simulation cost lower than the previous model
  ✕ Need to accumulate the measurements

A Hierarchical Network Topology on Bricks



Scheduling Unit

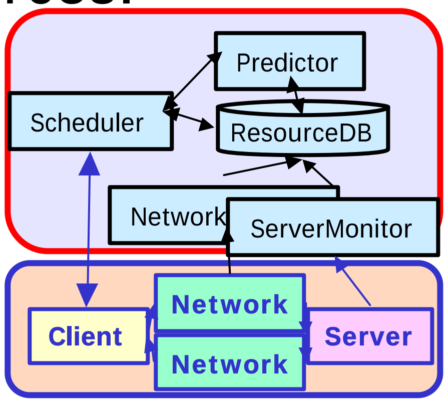
- Network/ServerMonitor

  Measures/monitors network/server status on the Grid
- ResourceDB

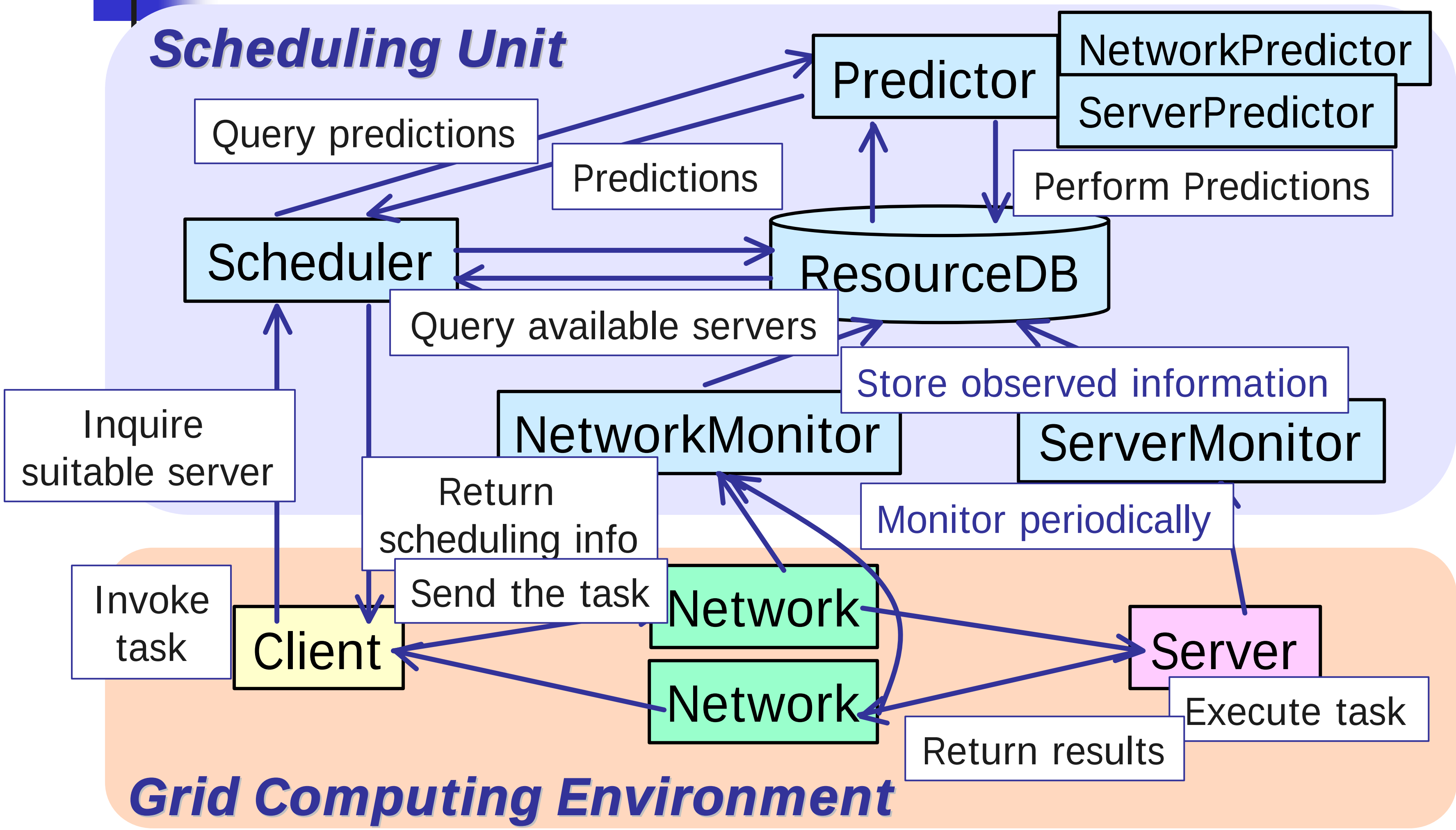
  Serves as scheduling-specific database, storing the values of various measurements.
- Predictor

  Reads the measured resource information from ResourceDB, and predicts availability of resources.
- Scheduler

  Allocates a new task invoked by a client on suitable server machine(s)



Overview of Grid Computing Simulation with Bricks



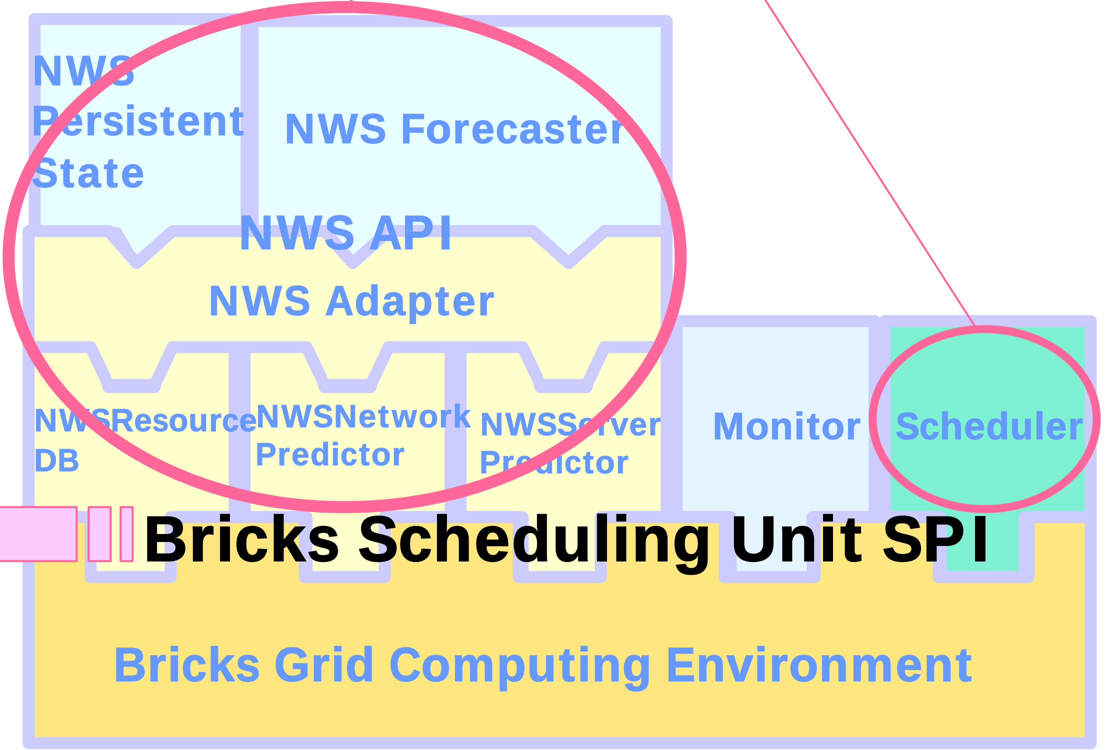
Incorporating External Components

- Scheduling Unit module *replacement*
  - Replaceable with other Java scheduling components
  - Components could be external - in particular, real Grid scheduling components
  - ? Allowing their validation and benchmarking under simulated and reproducible environments
- Bricks provides the Scheduling Unit SPI.

Scheduling Unit SPI

```
interface ResourceDB {
    void putNetworkInfo();
    void putServerInfo();
    NetworkInfo getNetworkInfo();
    ServerInfo getServerInfo();
}
interface NetworkPredictor {
    NetworkInfo getNetworkInfo();
}
interface ServerPredictor {
    ServerPredictor getServerInfo();
}
interface Scheduler {
    ServerAggregate selectServers();
}
```

Real / Original grid computing scheduling components are available!



Performance of a Deadline Scheduling Scheme

- Traditional scheduling ? deadline scheduling
  - Charging mechanisms will be adopted.
    - The Grids consists of various resources.
    - The resources are shared by various users.
  - Grid Users want the lowest cost machines which process jobs in a fixed period of time
- Deadline scheduling
  - meets a deadline for returning the results of jobs

A Deadline Scheduling Algorithm

1. Compute available processing time

   $T_{elapsed} = T_{deadline} - T_{start}$

2. Compute target processing time

   $T_{target} = T_{elapsed} \times Opt\ (0 < Opt\ ?\ 1.0)$

3. Estimate processing time on each server

   $T_{si} = W_{send}/P_{send} + W_{recv}/P_{recv} + W_{s}/P_{serv}\ (0\ ?\ i < n)$

   W_{send}, W_{recv}, W_{s}: send/recv data size, and # of instructions

   P_{send}, P_{recv}, P_{serv}: estimated send/recv network throughput, and server performance

4. Select suitable server i

   Conditions: $Diff = T_{target} - T_{si}\ ?\ 0$ && Min(*Diff*)

   If $Diff < 0$ (? i) then Min(|*Diff*|)

Evaluation of The Deadline Scheduling Algorithms

- Scheduling algorithms
  - Deadline: Opt = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
  - LOTH: select server i such as $Min(T_{si})$
- Environment
  - Grid Computing Environment
    - # of local domain: 10, # of local domain nodes: 5-10
    - Ave. LAN bandwidth: 50-100[Mbits/s]
    - Ave. WAN bandwidth: 500-1000[Mbits/s]
    - Ave. server performance: 100-500[Mops/s]
    - Ave. server Load: 0.1
    - Job processing manner on servers: FCFS
  - Characteristics of client jobs
    - Send/recv data size: 100-5000[Mbits]
    - # of instructions: 1.5-1080[Gops]
    - Ave. interval of invoking: 60(**high load**), 90(**lower load**) [sec]

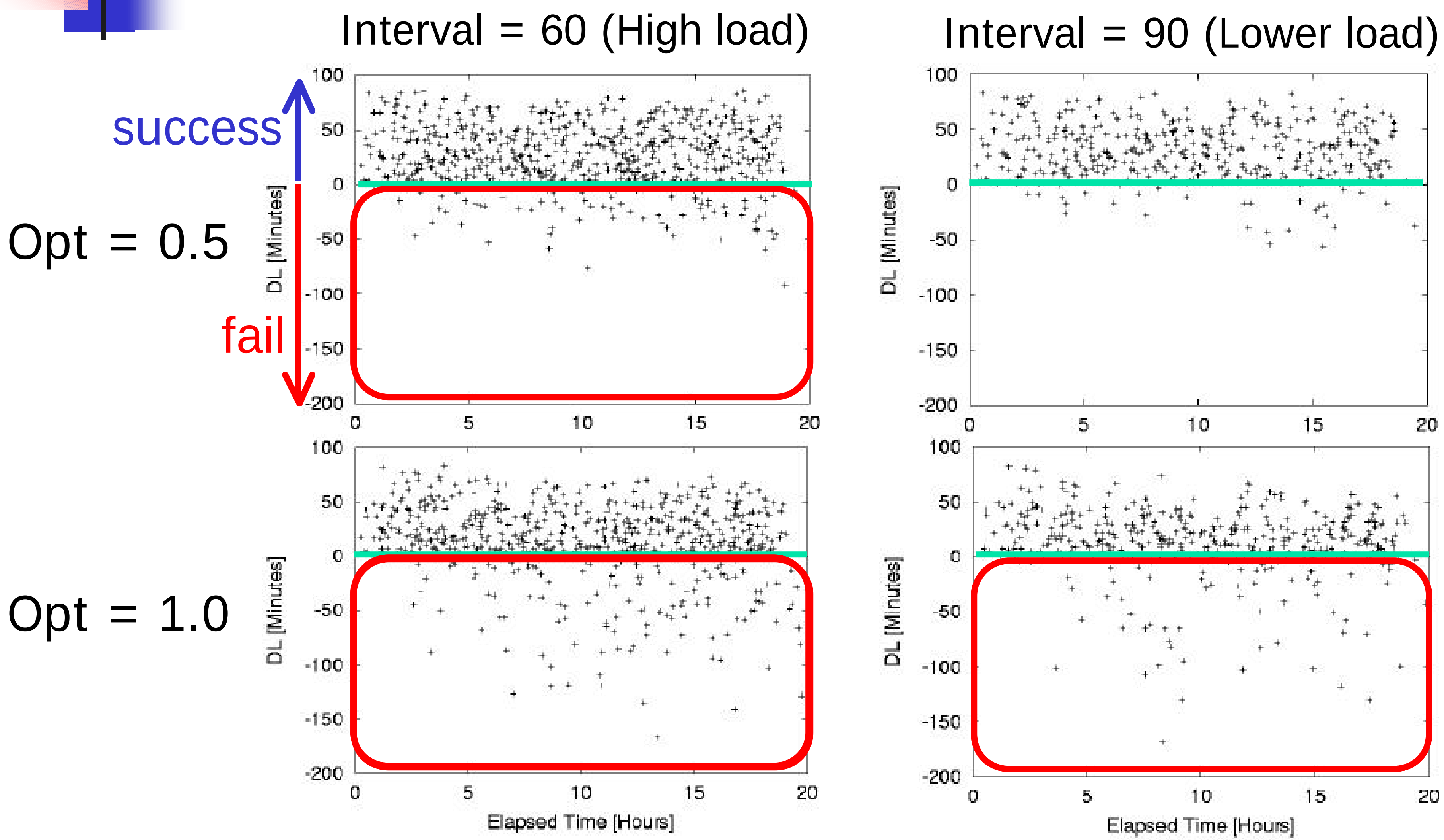
Simulation Environment

The Prospero cluster:

66PE cluster at Matsuoka Lab., Tokyo Institute of Technology.

- Dual Pentium III 800MHz
- Memory: 640MB
- Network: 100Base/TX

Experimental Results: differences (DL=Tdeadline – Tend)



Experimental Results

Interval = 90(Lower load), # of jobs = 445

- For smaller Opt, fail rates become smaller while costs become higher.
- Estimation of prediction error is important.
- Cost of LOTH is NOT suitable

Opt	Ave [min]	DL+ [min]	DL- [min]	#of Fail	Fail [%]	Cost
0.5	30.83	33.92	-20.98	30	6.7	382.3
0.6	34.77	30.37	-21.60	35	7.9	338.8
0.7	38.83	28.56	-26.54	51	11.5	317.3
0.8	43.65	26.01	-34.78	63	14.2	287.5
0.9	47.82	24.68	-47.19	71	16.0	273.4
0	49.26	22.82	-40.14	78	17.5	256.3
TH	29.67	34.93	-13.01	33	7.4	405.7

Sharing scheduling info. and fallback mechanisms are important.

Related Work: Performance Evaluation Environments

- Coarse-grained simulator
  - Bricks
  - Simgrid [UCSD]
- Emulator
  - MicroGrid [UCSD]
- Actual testbed
  - APGrid, Grid (US), eGrid, etc.

Accuracy of the experiments

Usability, Scalability, Reproducibility, ease to control, long-term experiments

Conclusions

- The Bricks performance evaluation system for Grid scheduling
  - multiple simulated reproducible benchmarking environments for
    - Scheduling algorithms
    - External Grid components
- Experiments of a Deadline scheduling scheme
  - The Accuracy of prediction affects deadline scheduling performance
  - LOTH is not suitable under charging mechanisms.
  - To avoid remarkable delay, sharing scheduling history and fallback mechanisms are important.

Future Work

- Simulation model
  - Server model for various architectures (e.g., SMP, MPP) and local scheduling schemes (e.g., LSF)
  - representation of parallel application tasks (Parameter-sweep applications are available)
- System Issues
  - Reconsideration of the Scheduling Unit design, interfaces, and data formats (c.f. Global Grid Forum)
  - Providing benchmarking sets of Bricks simulations
- Evaluation
  - Investigation of various job/task scheduling schemes on Bricks (e.g. computational economy)
  - Performance evaluation under real environment