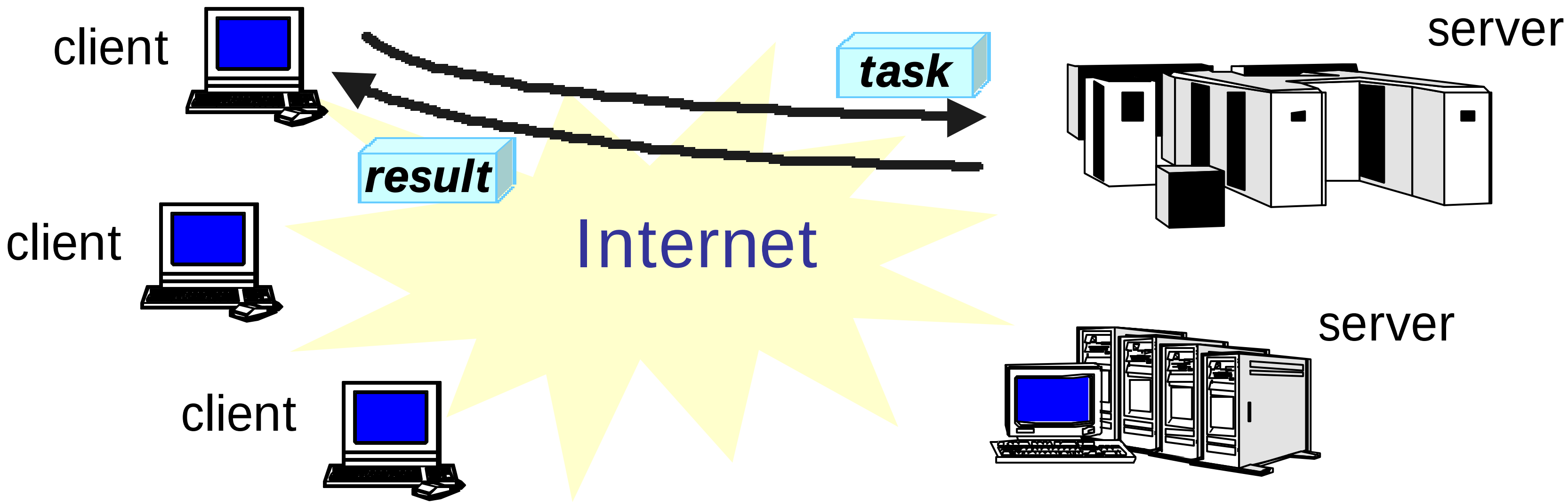
Performance Issues in Client-Server Global Computing

Atsuko Takefusa
Ochanomizu University
Satoshi Matsuoka
Tokyo Institute of Technology

Global Computing System (a.k.a. the "Grid")

- A global-wide high performance computing environment on the Internet
  - Client-Server systems (e.g., Ninf, NetSolve, Nimrod)
  - Middleware model systems (e.g., Globus, Legion)
  - Java-based systems (e.g., Ninflet, Javelin++)



Performance Issues

- Various parameters govern Grid performance
  - Execution environment
    - # of clients, # of servers, network topology
    - Hardware of clients, servers, networks
  - Application and data sets
  - System implementations
  - Scheduling schemes
- Reproducible and controlled environments
  - Large-scale benchmarks
  - Low benchmarking cost
  - Objective comparison between different systems or scheduling frameworks

Our Approach

- Benchmarks under different parameters [SC97]
  - Multiple client
  - LAN/WAN
  - Applications: Linpack, EP, SDPA
  - Sytems: Ninf, NetSolve, CORBA [IPDPS2000]
- Performance evaluation system for the Grid: Bricks [HPDC99]
  - A discrete event simulator
  - Reproducible and controlled environments
  - Flexible setups of simulation environments
  - Evaluation environment for existing global computing components (ex. NWS)

Outline

- Benchmark results of client-server global computing systems
  - Multi-client benchmark using Ninf
  - Comparison of various client-server systems
    Ninf, NetSolve, CORBA
- Performance evaluation system: Bricks
  - Overview of the Bricks system
  - Incorporating existing global computing components (ex. NWS)
  - Bricks experiments
- Conclusions and future work

Ninf Multi-client Benchmarks

- Communication and overall performance

  → LAN, WAN (Single-site, Multi-site)
- Robustness of computational server

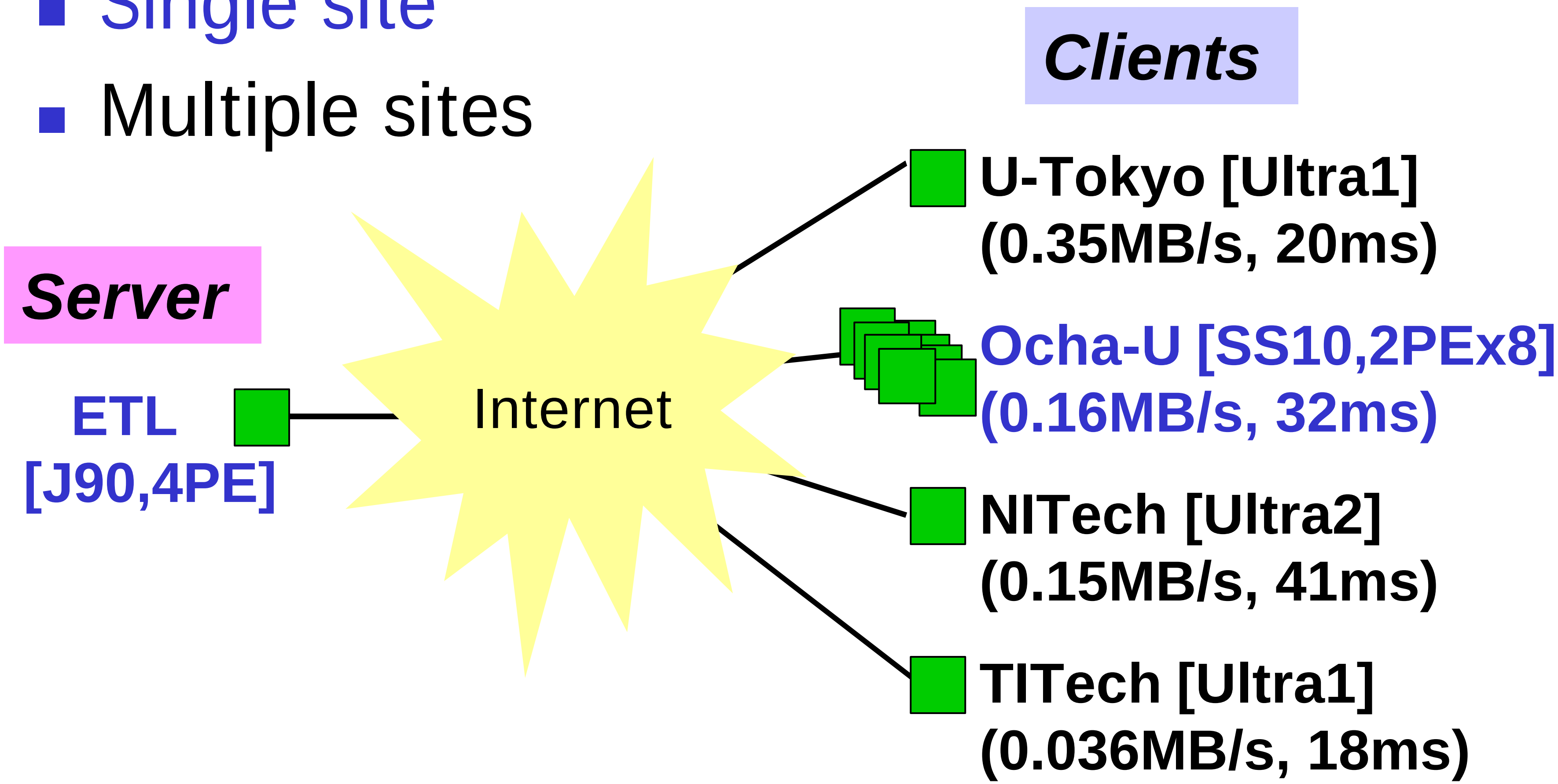
  → vector parallel server (Cray J90, 4PE)
- Remote library design and reuse

  → Task Parallel(1PE lib), Data Parallel(4PE lib)
- Interaction between computation and communication for remote libraries

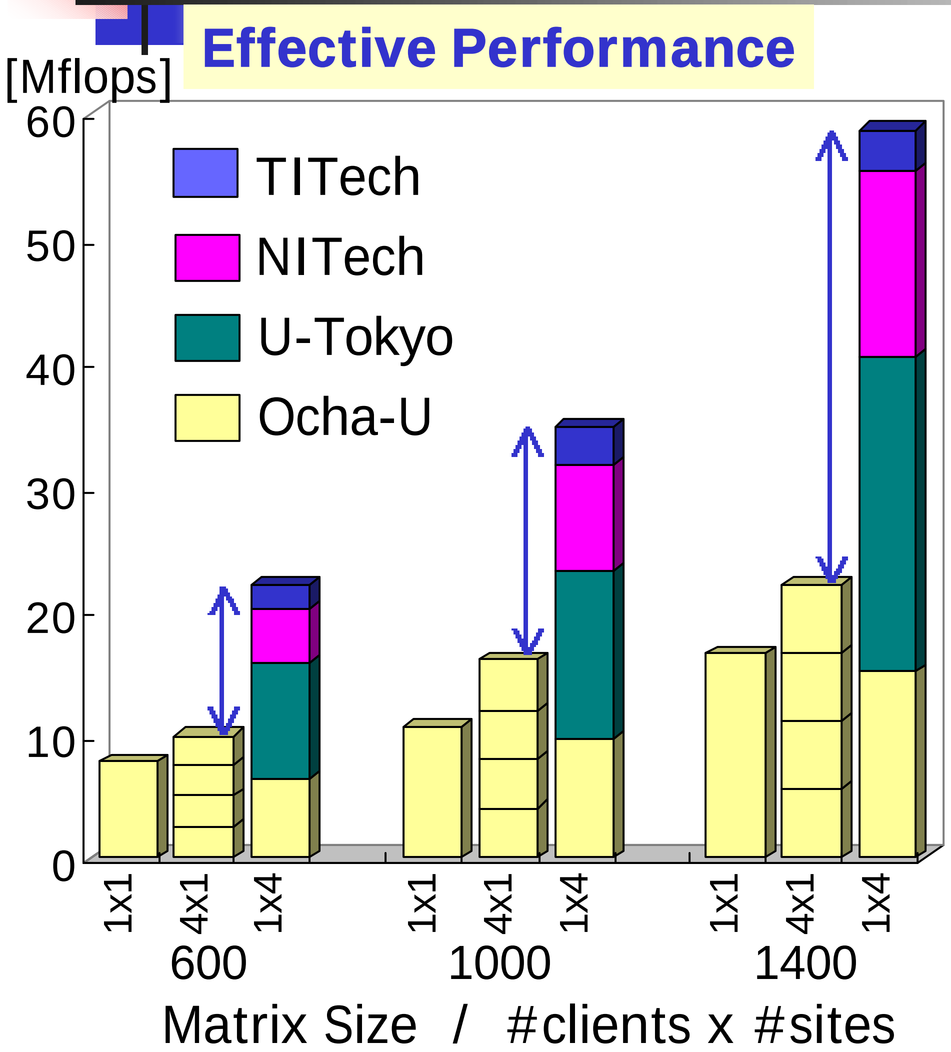
  → Linpack, EP

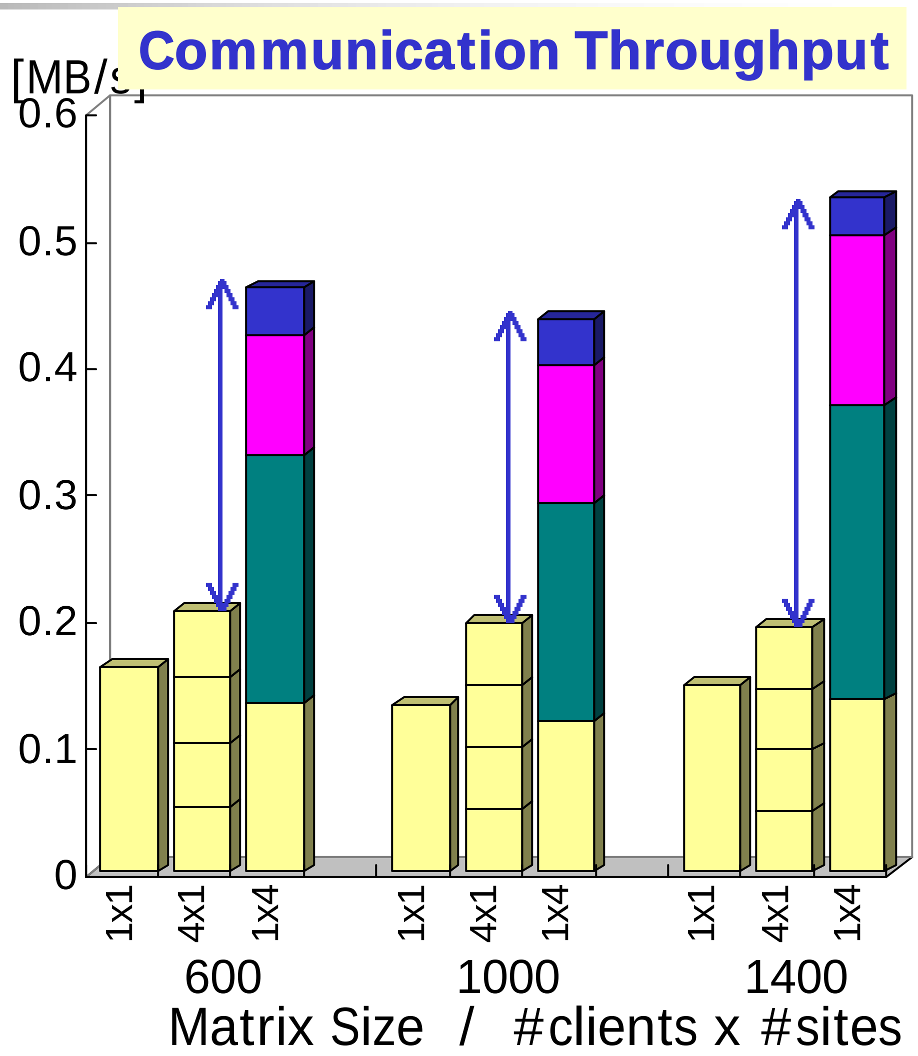
WAN Multi-client Benchmarking Environment

- Single site
- Multiple sites

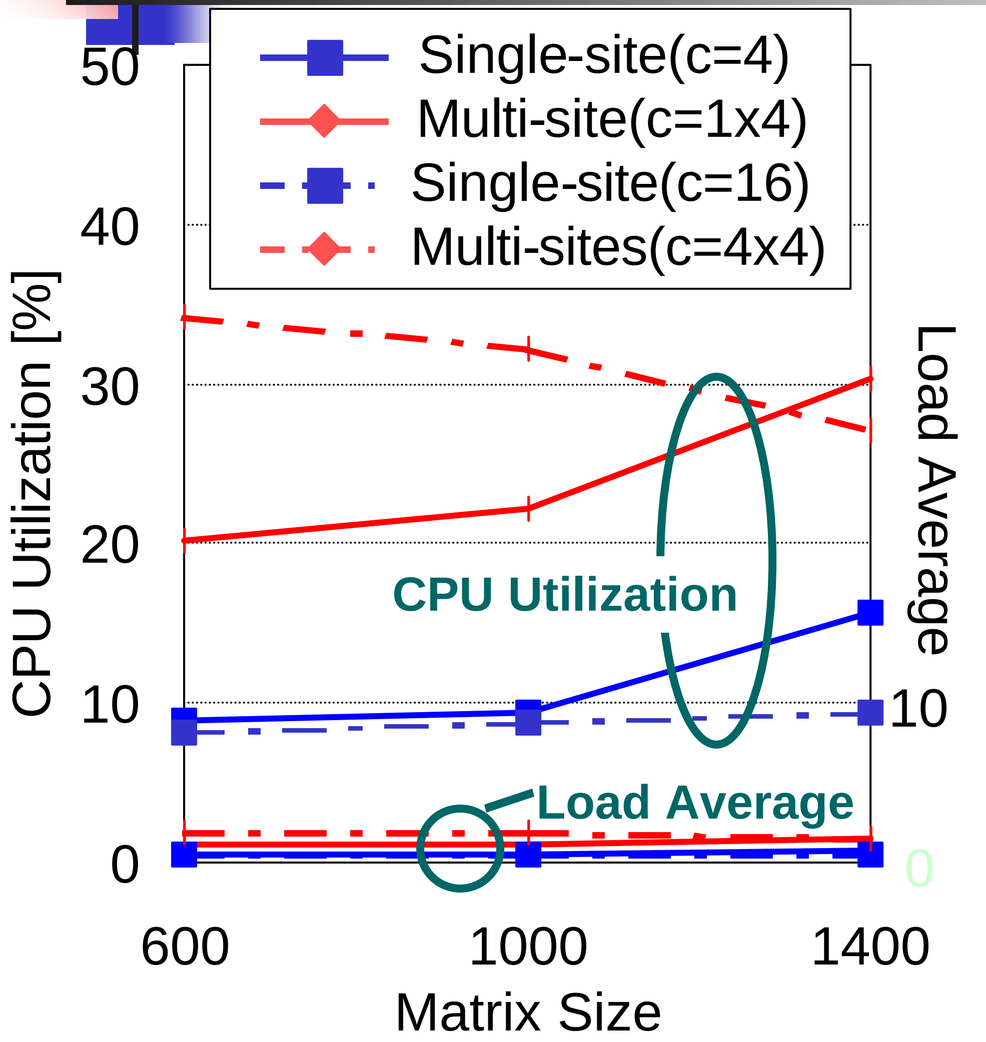


WAN(Single-site, Multi-site) Multi-client Benchmark Results





WAN Benchmark Results
Interaction betw. Comp. and Comm.
- CPU Utilization and Load Average-



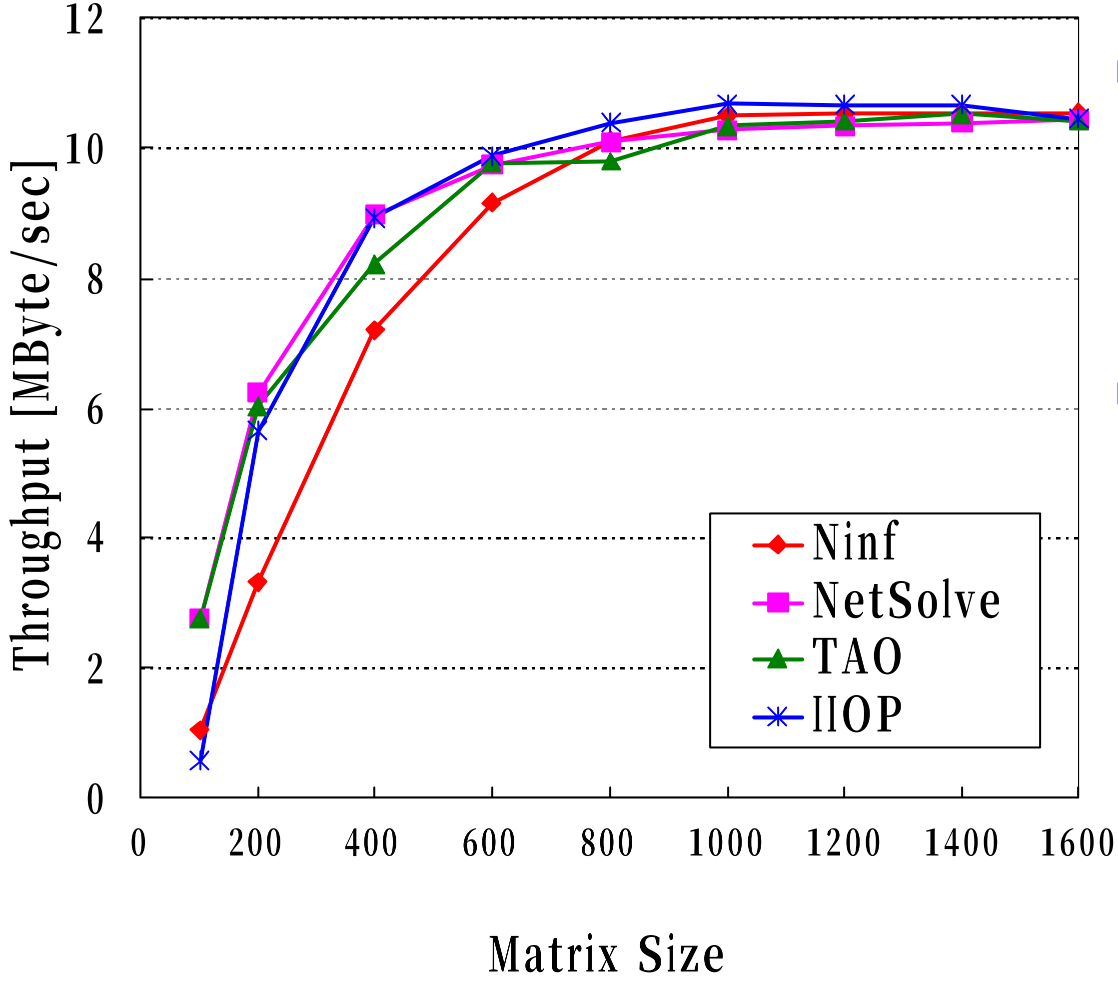
- The J90 server does not saturate for *n* and *c*.
  - Network bandwidth saturation the cause.
- Utilization is low due to network congestion, not lack of jobs.

→ **Utilization and Load alone are NOT suitable criteria for global computing scheduling.**

Comparison between Client Server Systems [IPDPS2000]

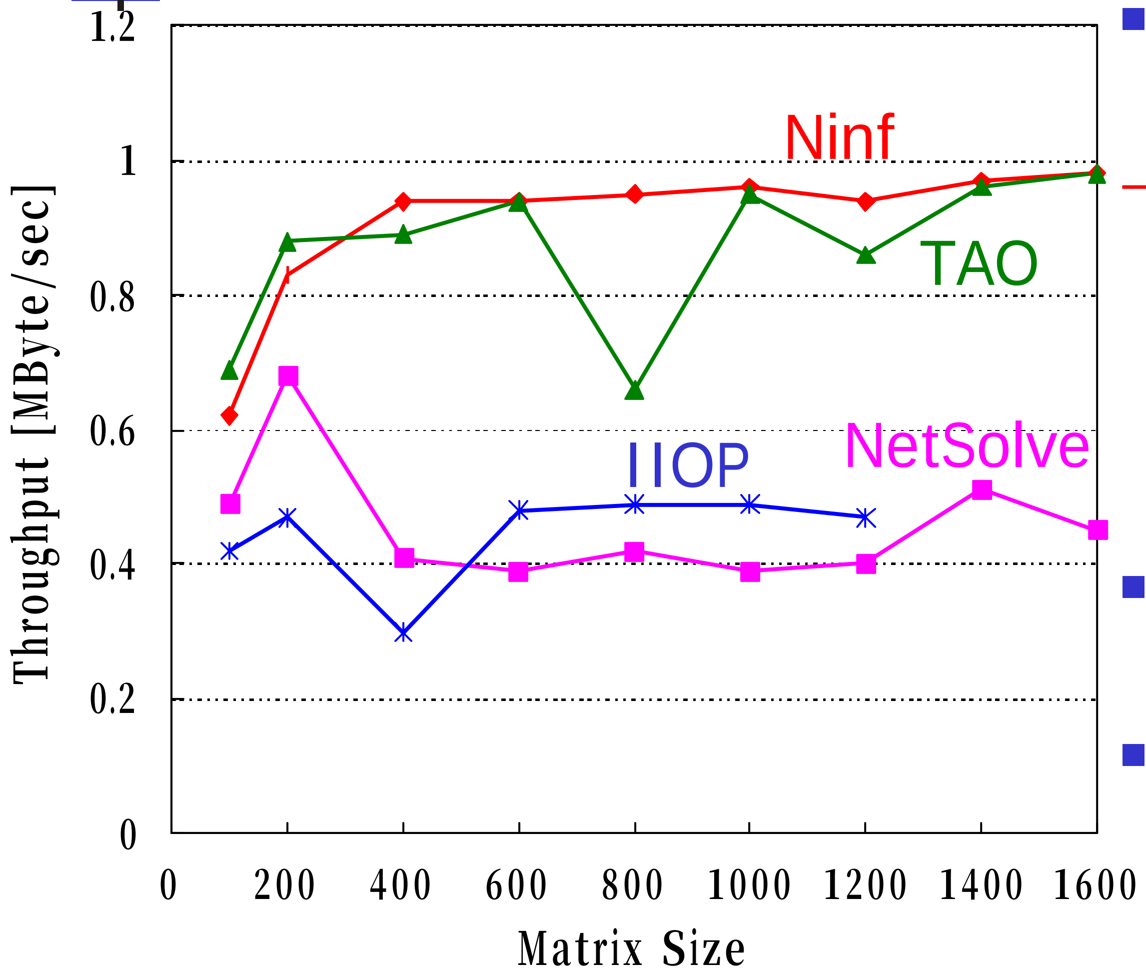
- Systems
  - Ninf, NetSolve, CORBA(TAO, IIOP[TAO-OmniORB])
- LAN (100Base-TX)
  - Server: Ultra60[300MHz×2, 256MB] at TITECH
  - Client: Ultra2[200MHz×2, 256MB] at TITECH
- WAN (ave. 0.6[Mbyte/s])
  - Server: Ultra60 at TITECH, Tokyo
  - Client: SS5[85MHz, 32MB] at ETL, Tsukuba
- Benchmark routine: Linpack

LAN Communication Throughput



- Throughput differences between systems are quite small
- For smaller problem size, Ninf seems to be slightly slower

WAN Communication Throughput



- TAO closely matches Ninf

→TAO uses a private comm. Protocol, which seems to match the efficiency of that of Ninf

- Unlike LAN, IIOP and NetSolve were slower
- High interoperability does not come at the expense of performance.

Summary of Benchmarks

- In WAN, limitation of communication throughput is more significant
- We expect multiple client requests will be issued from different sites, causing “false” lowering of load ave.

→The scheme which properly dispatches comm.-/comp.-intensive jobs to the servers is important.

- Ninf and TAO are comparable in LAN and WAN

→However, ease-of-programming, availability of Grid services, etc., differentiate dedicated Grid systems and general systems such as CORBA.

Outline

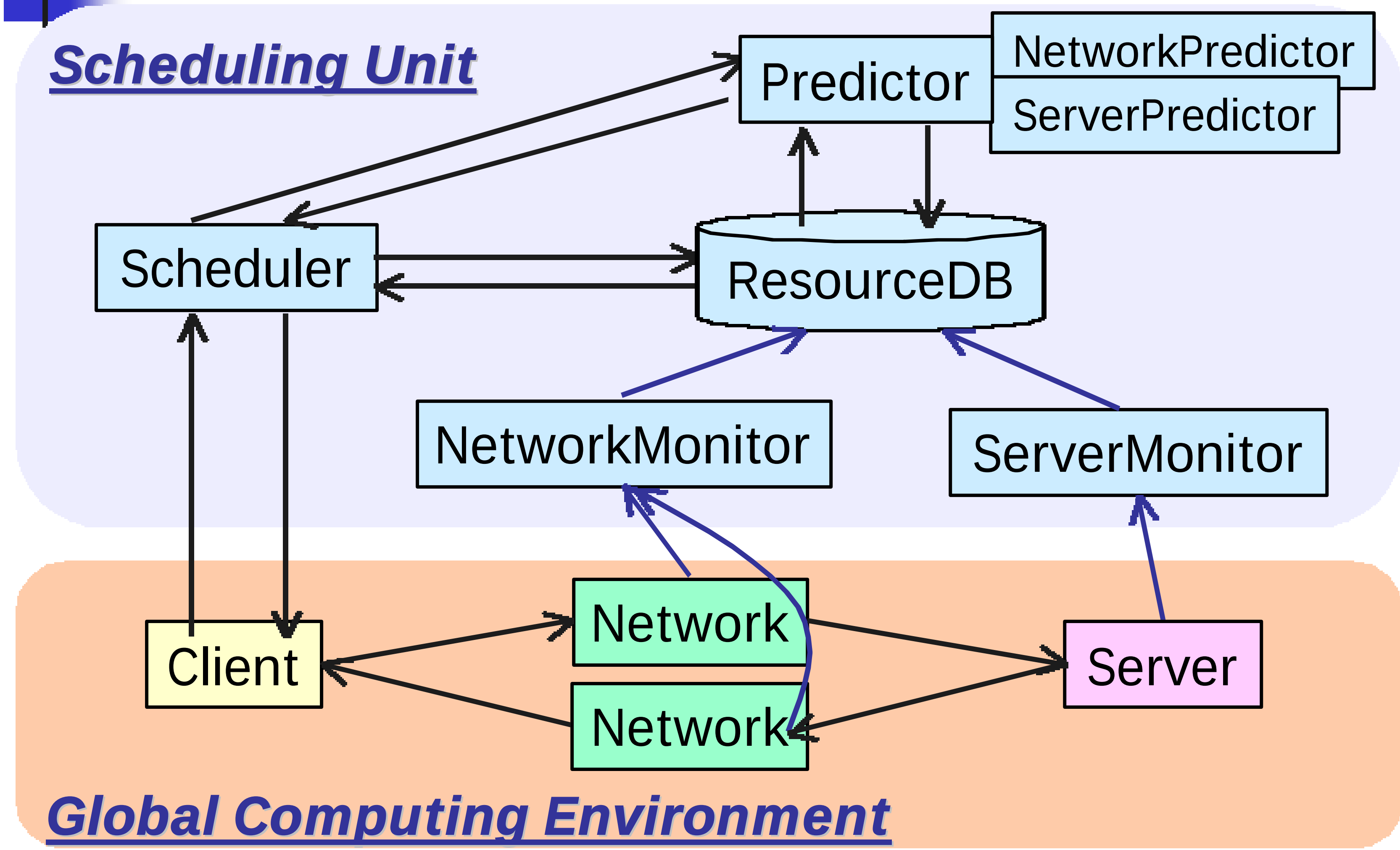
- Benchmark results of client-server global computing systems
  - Multi-client benchmark using Ninf
  - Comparison of various client-server systems
    Ninf, NetSolve, CORBA
- Performance evaluation system: Bricks
  - Overview of the Bricks system
  - Incorporating existing global computing components (ex. NWS)
  - Bricks experiments
- Conclusions and future work

Overview of Bricks

- Consists of simulated *Global Computing Environment* and *Scheduling Unit.*
- Allows simulation of various behaviors of
  - resource scheduling algorithms
  - programming modules for scheduling
  - network topology of clients and servers
  - processing schemes for networks and servers (various queuing schemes)

  using the *Bricks script*.
- Makes benchmarks of existing global scheduling components available

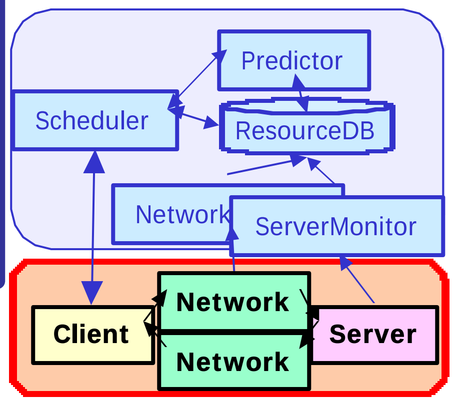
The Bricks Architecture



Global Computing Environment

- Client
  - represents user of global computing system
  - invokes global computing Tasks
    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



Scheduling Unit

- NetworkMonitor/ServerMonitor

  measures/monitors network/server status in global computing environments

- ResourceDB

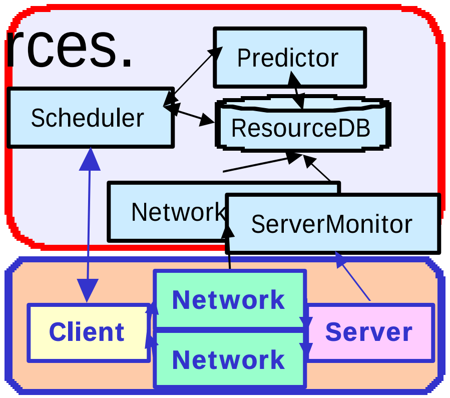
  serves as scheduling-specific database, storing the values of various measurements.

- Predictor

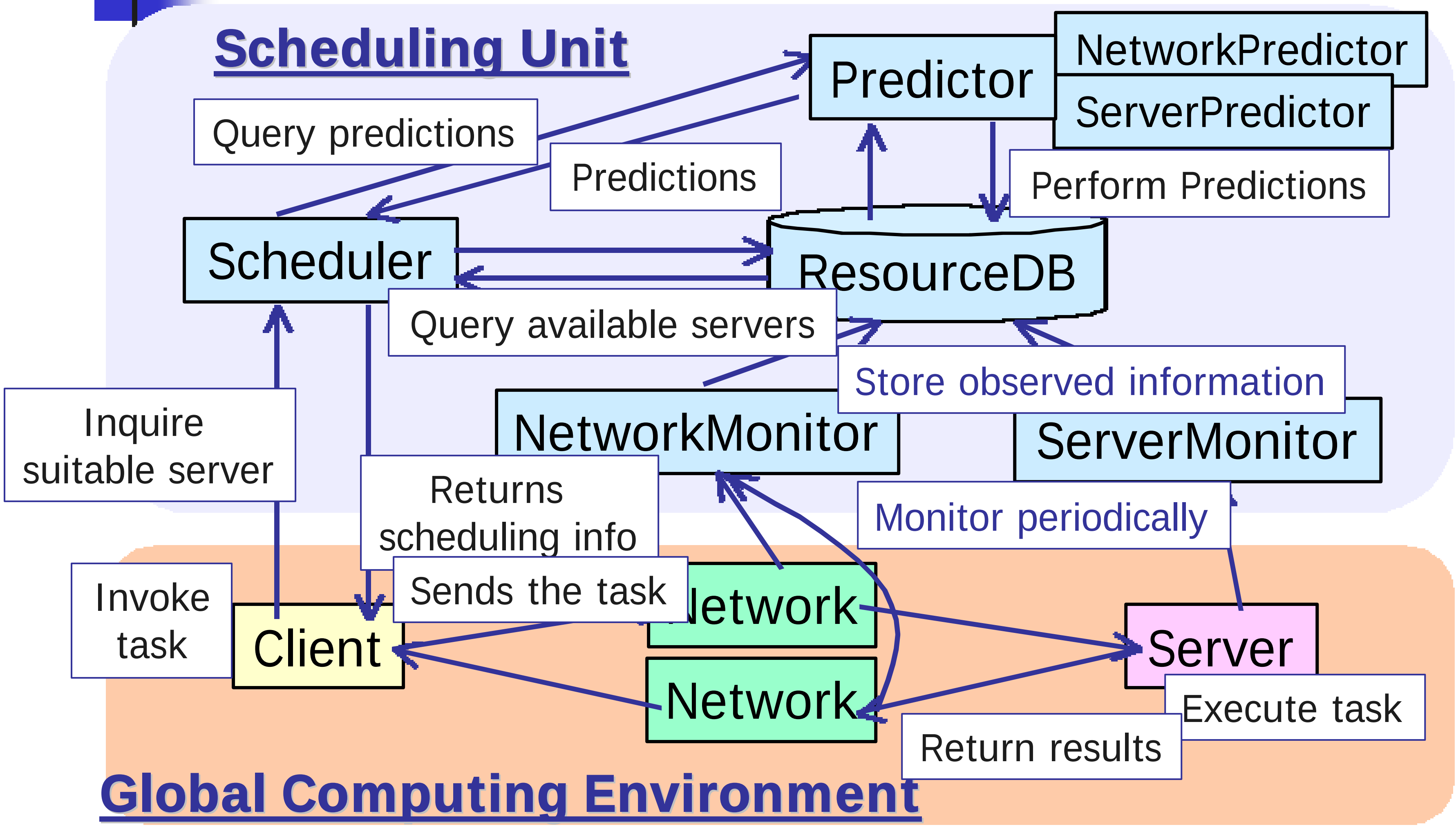
  reads the measured resource information from ResourceDB, and predicts availability of resources.

- Scheduler

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



Overview of Global Computing Simulation with Bricks



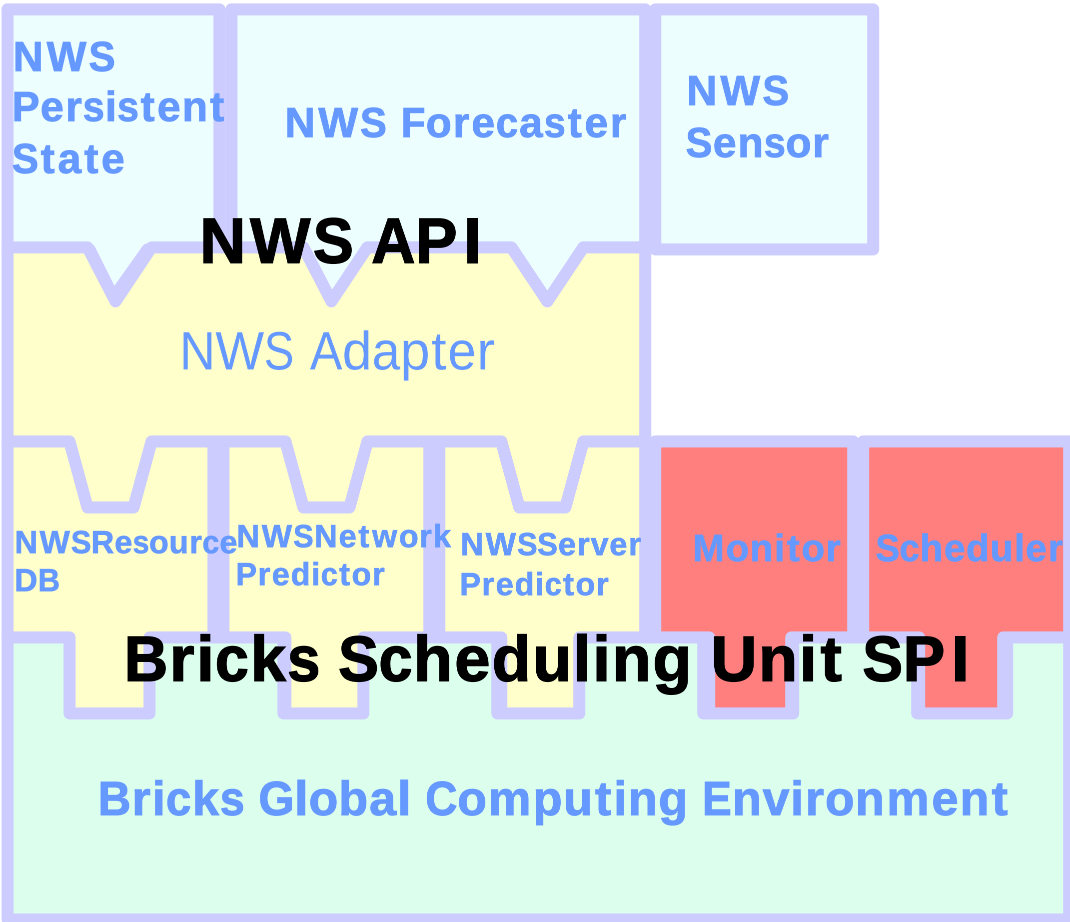
Incorporating External Components

- Scheduling Unit module *replacement*
  - Replaceable with other Java scheduling components
  - Components could be external --- in particular, real global computing 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();
}
```



Case study

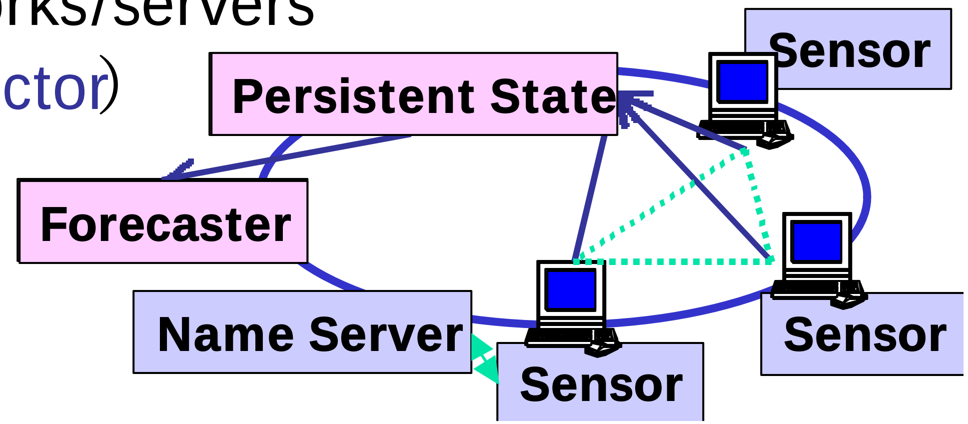
- NWS[UCSD] integration into Bricks
  - monitors and predicts the behavior of global computing resources
  - has been integrated into several systems, such as AppLeS, Globus, Legion, Ninf
  - Orig. C-based API

    → NWS Java API development

    → NWS run under Bricks

The NWS Architecture

- Persistent State (→Replace ResourceDB)
  is storage for measurements
- Name Server
  manages the correspondence between the IP/domain address for each independently-running modules of NWS
- Sensor (→Network/ServerMonitor)
  monitors the states of networks/servers
- Forecaster (→ Replace Predictor)
  predicts availability of
  the resources



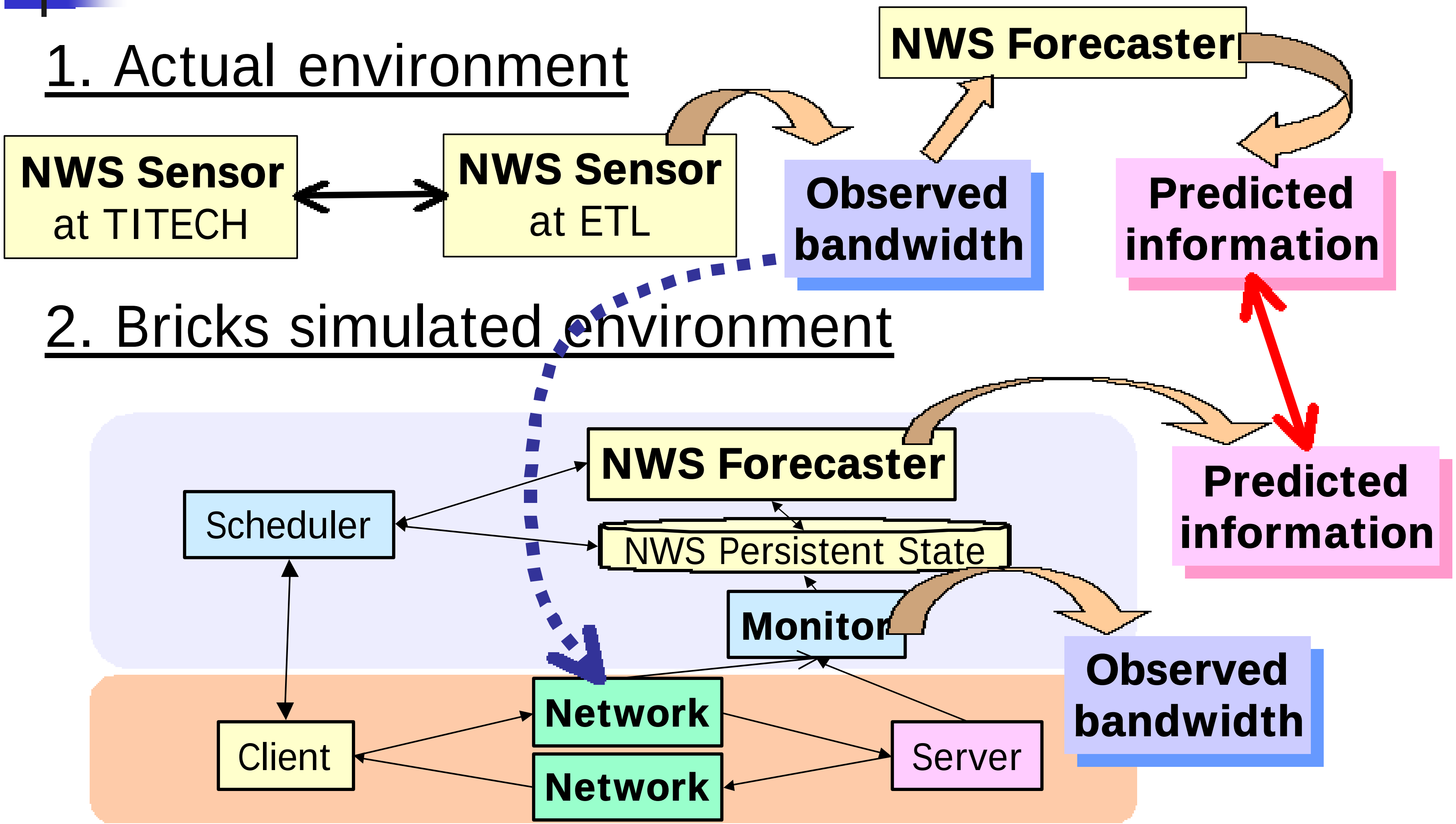
Bricks Experiments

- The experiments conducted by running NWS under a real environment vs. Bricks environment

Whether Bricks can provide

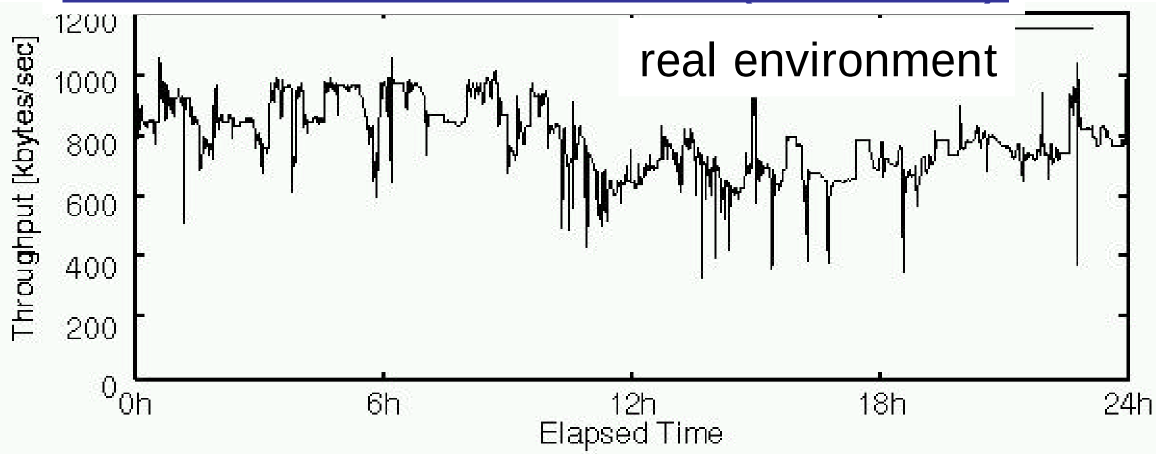
- A simulation environment for global computing with reproducible results?
- A benchmarking environment for existing global computing components?

Overview of Experiments

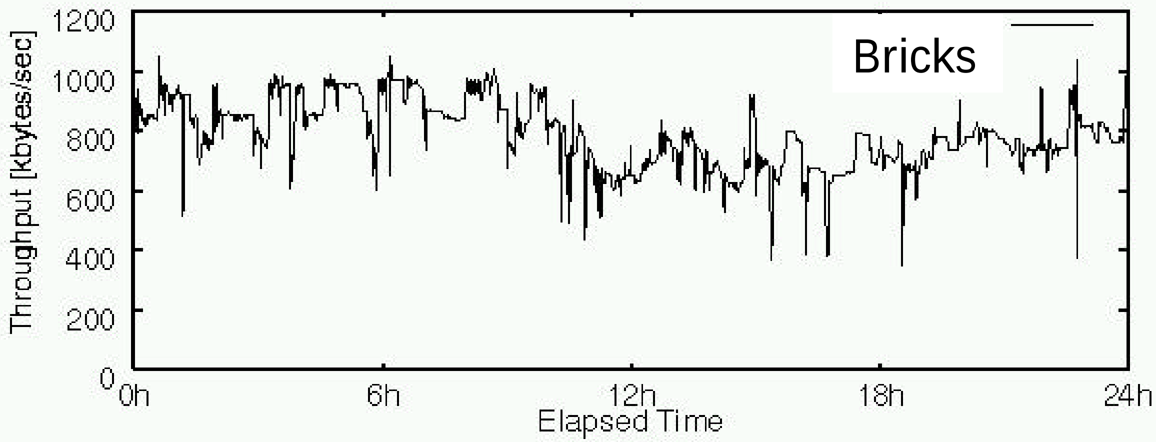


Bricks Experimental Results: Comparison of Predicted Bandwidth

Under real environment(24hours)



Under Bricks (24hours)



- The NWS Forecaster functions and behaves normally under Bricks
- Both prediction are very similar

Bricks provides existing global computing components with a benchmarking environment

Related Work

- Osculant Simulator[Univ. of Florida]
  - evaluates Osculant: bottom-up scheduler for heterogeneous computing environment
  - makes various simulation settings available
- WARMstones [Syracuse Univ.]
  - is similar to Bricks, although it seems not have been implemented yet.
  - will provide an interface language(MIL) and libraries based on the MESSIAHS system to represent various scheduling algorithms →Bricks provides SPI
  - has no plan to provide a benchmarking environment for existing global computing components

Conclusions

- We conducted benchmarks under various environment such as multiple clients, systems.
- We proposed the Bricks performance evaluation system for global computing scheduling
  - multiple simulated reproducible benchmarking environments for
    - Scheduling algorithms
    - Existing global computing components
- Bricks experiments showed
  - Evaluation of existing global computing components now possible

Future Work

- Performance evaluation under various parameters
- Simulation model of Bricks needs to be more sophisticated and robust
  - Task model for parallel application tasks
  - Server model for various server machine architectures (e.g., SMP, MPP) and scheduling schemes (e.g., LSF)
- Standardization of interface and data representations of Bricks
- Communication component integration (e.g., direct support for IP)
- Providing benchmarking set of the Bricks simulation
- Investigation of various scheduling algorithms on Bricks