

APAN Conference 2000, Beijing



Ninf Project

Kento Aida⁽⁴⁾, Atsuko Takefusa⁽⁴⁾, Hirotaka Ogawa⁽⁴⁾,
Osamu Tatebe⁽¹⁾, Hidemoto Nakada⁽¹⁾, Hiromitsu Takagi⁽¹⁾,
Yoshio Tanaka ⁽¹⁾, Satoshi Matsuoka⁽⁴⁾, Mitsuhisa Sato⁽²⁾,
Satoshi Sekiguchi⁽¹⁾, Umpei Nagashima⁽³⁾

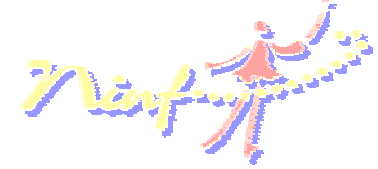
Electrotechnical Laboratory⁽¹⁾

Real World Computing Partnership⁽²⁾

National Institute of Materials and Chemical Research⁽³⁾

Tokyo Institute of Technology⁽⁴⁾

URL: <http://ninf.etl.go.jp>



Towards Global Computing

○ Rapid increase in speed and availability of WAN enables:

○ sharing distributed data resources

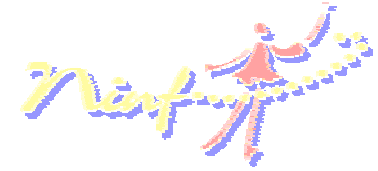
☞ email, file archives, WWW

○ sharing distributed computational power

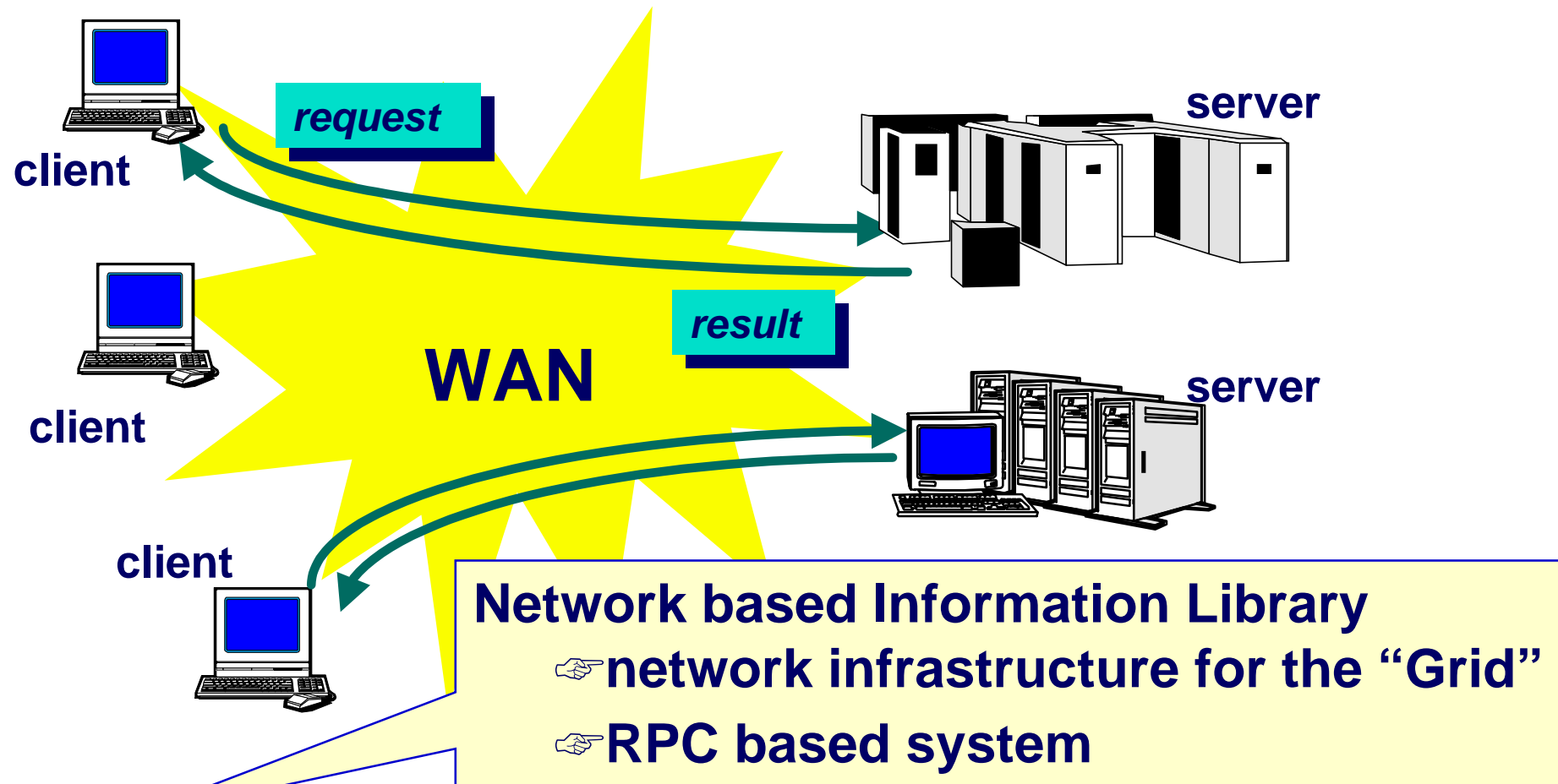
☞ “Grid”, Global Computing



Starting Grid / global computing projects



Global Computing



○ Ninf, NetSolve, RCS, Legion, Javelin, Globus,...

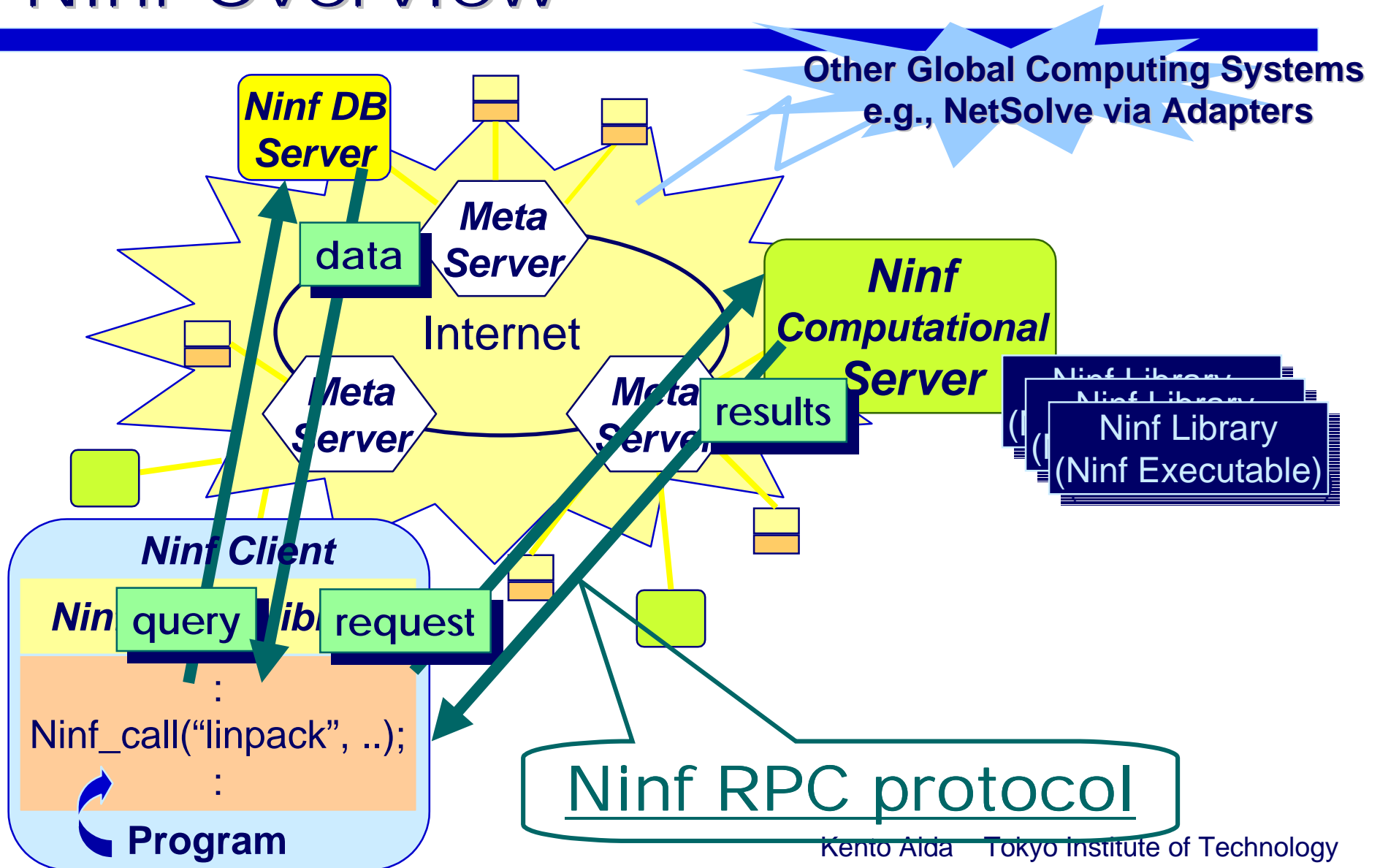


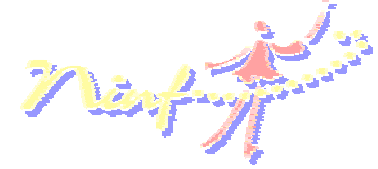
Outline

- **overview of Ninf system**
 - **Ninf RPC protocol**
 - **Client API**
 - **Providing Ninf library program**
- **applications**
- **scheduling**
 - **Metaserver**
 - **Bricks**
- **conclusions**



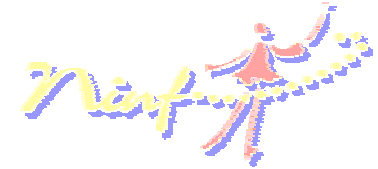
Ninf Overview





Ninf Overview (cont'd.)

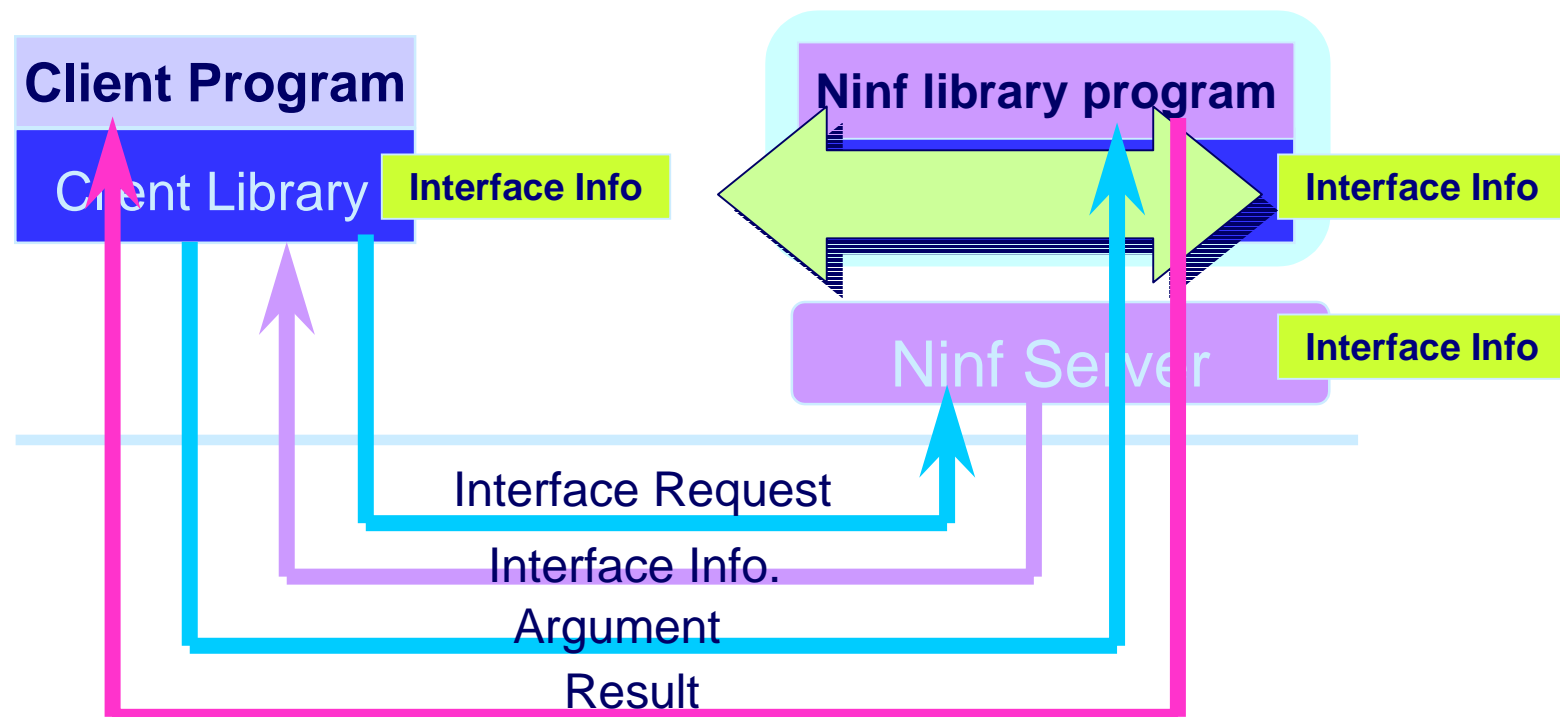
- **Ninf server + Ninf RPC protocol**
transparent execution of Ninf library program on computational server
- **Ninf database server**
direct query on accurate constant database
- **Ninf Client Interface**
easy-to-use programming interface
- **Ninf Metaserver**
scheduling of computation, asynchronous and automatic parallel computation,



Ninf RPC Protocol

○ Exchange interface information at run-time

- No need to generate client stub routines (cf. SunRPC)
- No need to modify a client program when server's libraries are updated.



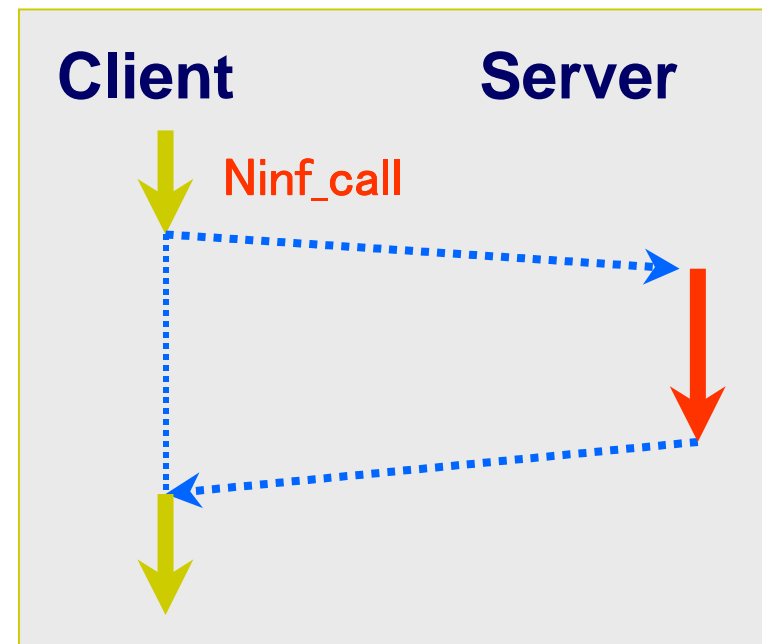


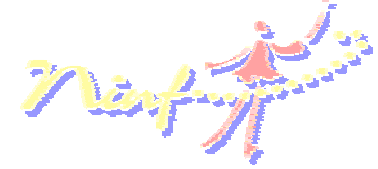
Ninf Client API

- A client user only needs to specify the name of libraries to invoke.

```
Ninf_call("ninf://HOST:PORT/ENTRY_NAME", arg,...)
```

- Implemented API:
C, C++, Fortran, Java,
Lisp ..., Mathematica,
Excel, ...





Ninf Client API (cont'd.)

○ Example

```
double A[n][n],B[n][n],C[n][n]; /* Data Decl.*/  
      ·  
dmmul(n,A,B,C); /* Call local function*/
```

Ninfy

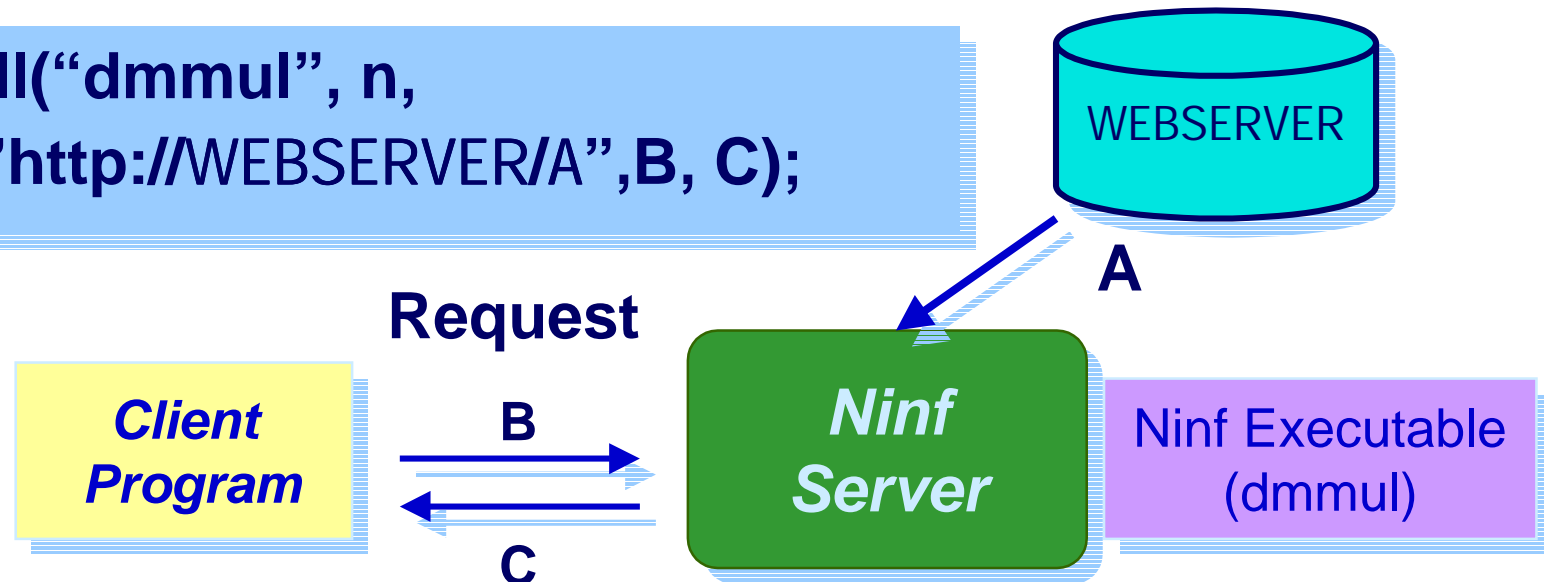
```
double A[n][n],B[n][n],C[n][n]; /* Data Decl.*/  
      ·  
Ninf_call("dmmul",n,A,B,C); /* Call Ninf Executable*/
```



Direct Web Access

- URL can be specified as an argument.
- computation of Ninf library using data on a specified Web server
 - ➡ (ex.) important constant for physics or chemistry
- storing interim results to a Web Server

```
Ninf_call("dmmul", n,  
"http://WEBSERVER/A", B, C);
```



Ninf Client API(2)

- asynchronous call -

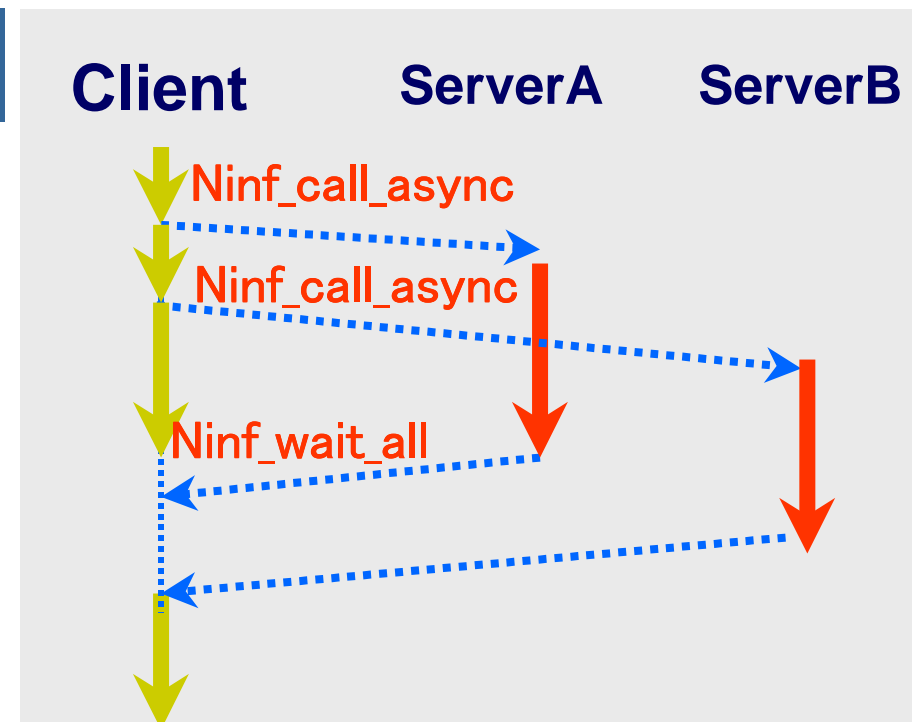


○ Asynchronous Call

```
Ninf_call_async("FUNC", ...);
```

○ Waiting the reply

```
Ninf_wait(ID);  
Ninf_wait_all();  
Ninf_wait_and(IDList, len);  
Ninf_wait_or(IDList, len);  
Ninf_cancel(ID);
```



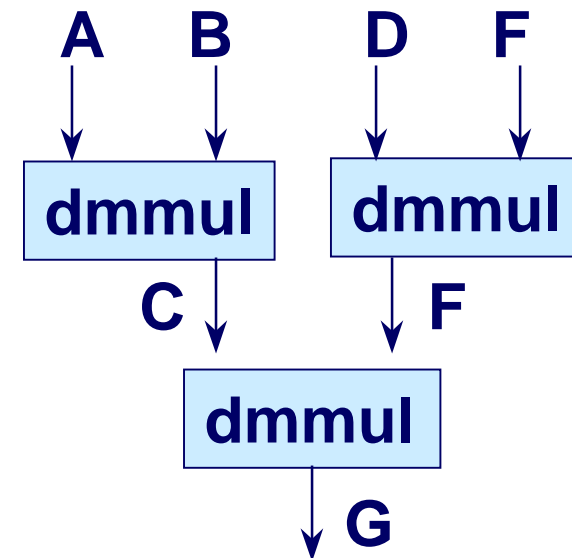
Ninf Client API(3)

- Transaction-



- Aggregate invocation of user specified code region
- Macrodataflow execution by Metaserver

```
Ninf_transaction_start();  
Ninf_call("dmmul",n,A,B,C);  
Ninf_call("dmmul",n,D,E,F);  
Ninf_call("dmmul",n,C,F,G);  
Ninf_transaction_end();
```

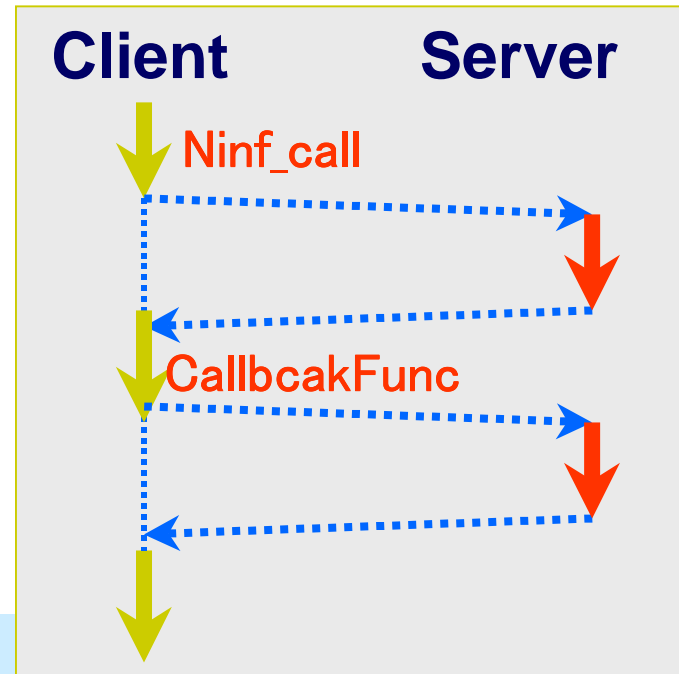


Ninf Client API(4)

- Callback -



- Server side routine can callback client side routine.
- (ex.) Display of interim results of computation on servers to a client machine



```
void CallbackFunc(...){
    .... /* define callback routine */
}
Ninf_call("Func", arg .., CallbackFunc);
/* call with pointer to the function */
```

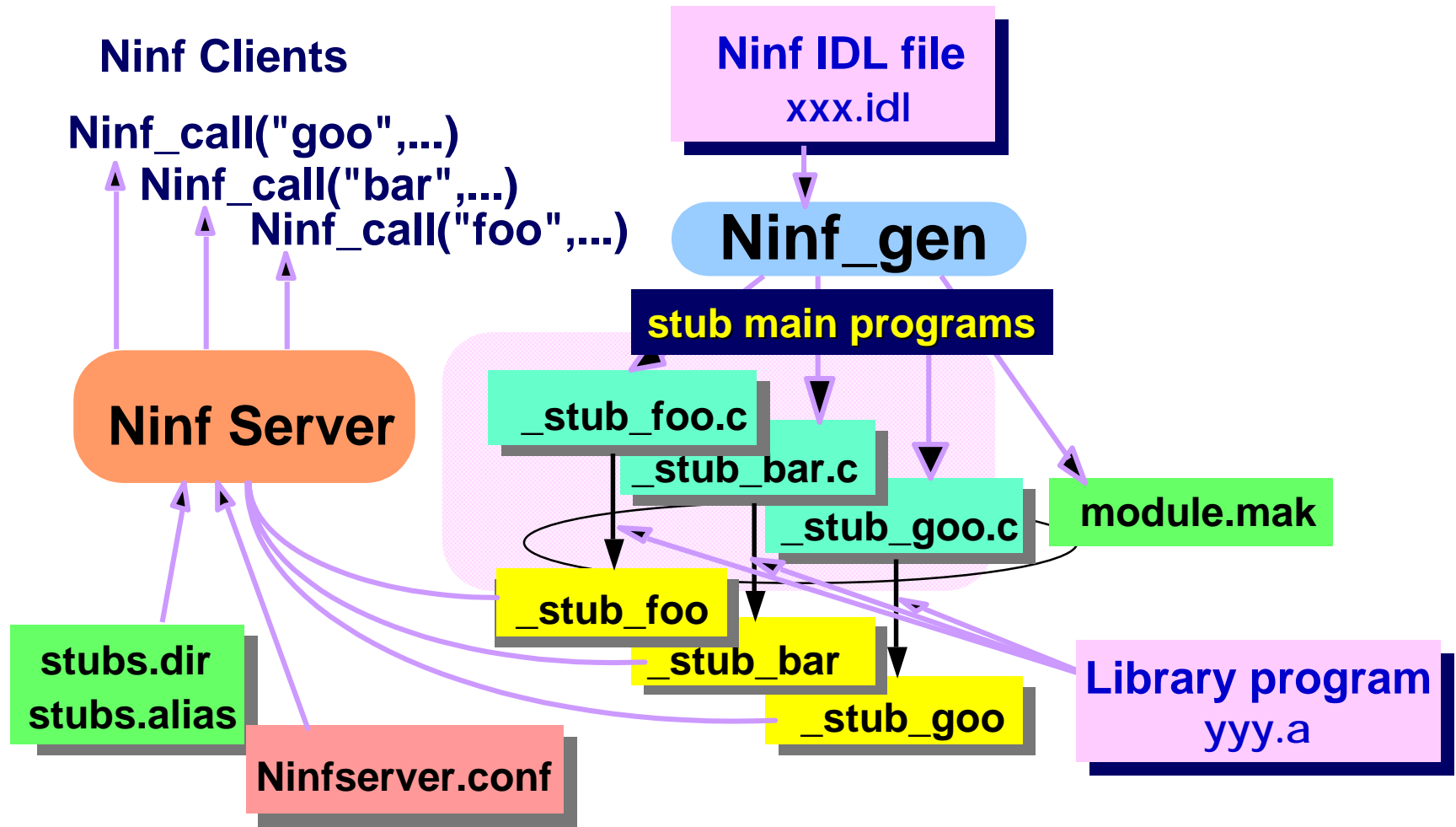


Providing Ninf Library

- (1) Writing interface description for a library (computational) program in Ninf IDL**
 - ➔ Ninf IDL file
- (2) Running Ninf interface generator**
 - ➔ stub programs and Makefile
- (3) Compiling the library program and linking with the stub programs**
 - ➔ Ninf executables
- (4) Registering the Ninf executables with Ninf server**



Generating Ninf executables



Ninf Interface Description (Ninf IDL)

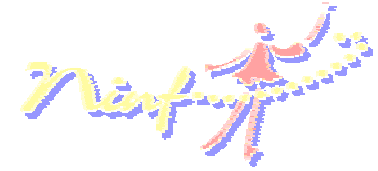


```
Define dmmul(long mode_in int n,  
  mode_in double A[n][n], mode_in double B[n][n],  
  mode_out double C[n][n])  
" description "  
Required "libXXX.o"  
CalcOrder n^3  
Calls "C" dmmul(n,A,B,C);
```

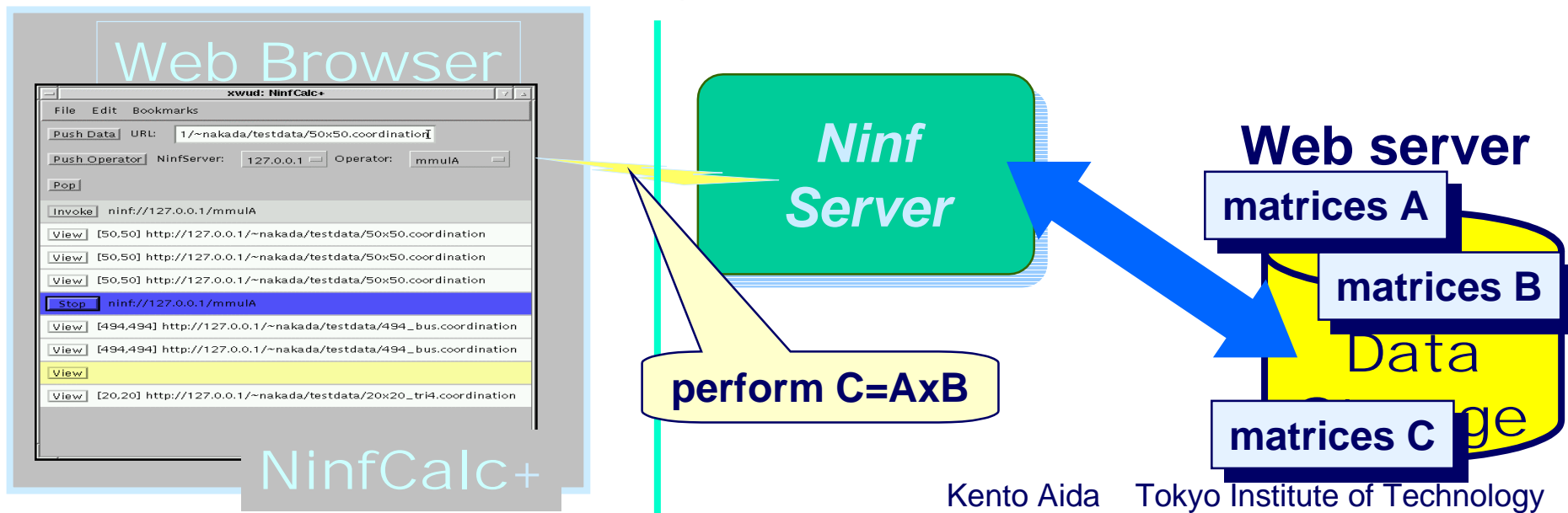
○ IDL information:

- library function's name, and its alias (**Define**)
- arguments' access mode, data type (**mode_in, out, inout, ...**)
- required library for the routine (**Required**)
- computation order (**CalcOrder**)
- source language (**Calls**)

Application on Ninf: NinfCalc+



- **Matrix Calculator implemented as an Applet in browser**
 - linear system solver, eigen-value problem, ...
 - click and quick operation
- **direct Web access**
 - Huge matrices are stored to Web servers.
 - interactive control of huge matrix calculation via thin network



Application on Ninf : Excel Ninf



- Arguments for Ninf_call are specified on the Excel worksheet

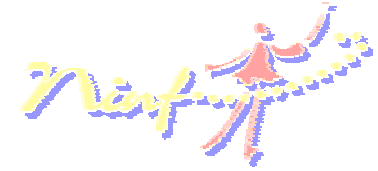
	A	B	C	D	E	F
1						
2	2	1	2		1	0
3		3	4		0	1
4						
5					1	2
6					3	4

`Ninf_call("dmmul", 2, A, B, C)`

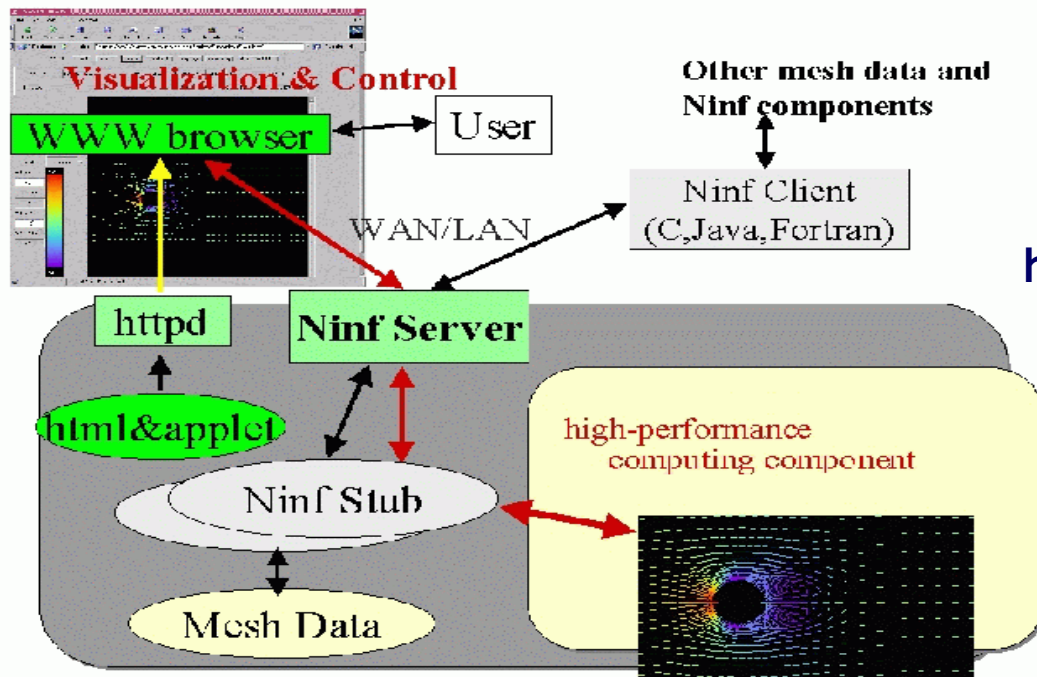
Ninf Server

$C = A \times B$

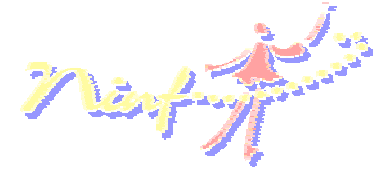
Application on Ninf : NetCFD



- Ninf computational component for CFD
 - Parallelized CFD program is “ninfied” on Ninf server.
 - providing an interface to a parallel CFD program running on MPP, PC cluster,...

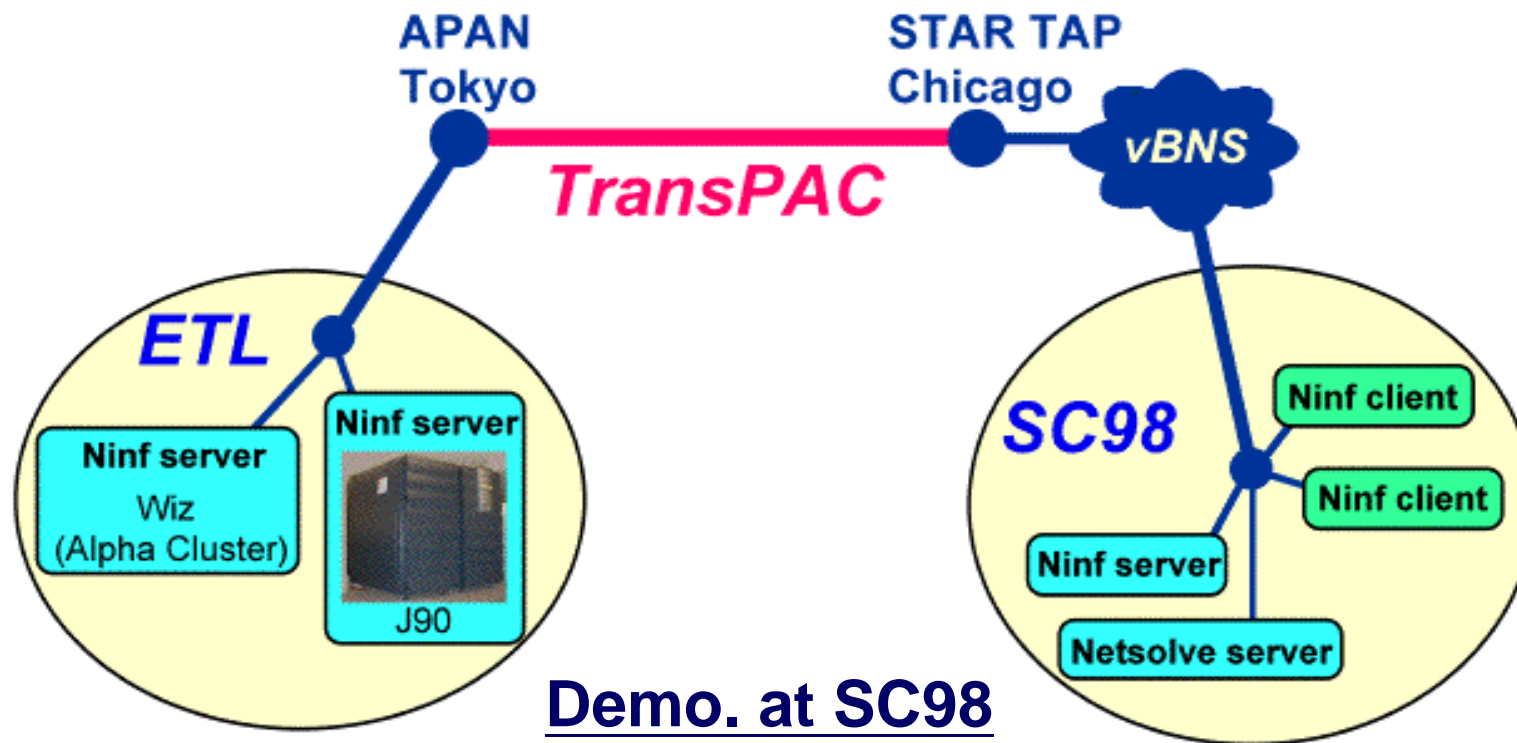


Callback function
<http://pdplab.trc.rwcp.or.jp/netCFD/>



Ninf over TransPAC

- Demonstration of Ninf system over TransPAC
 - SC98, SC99



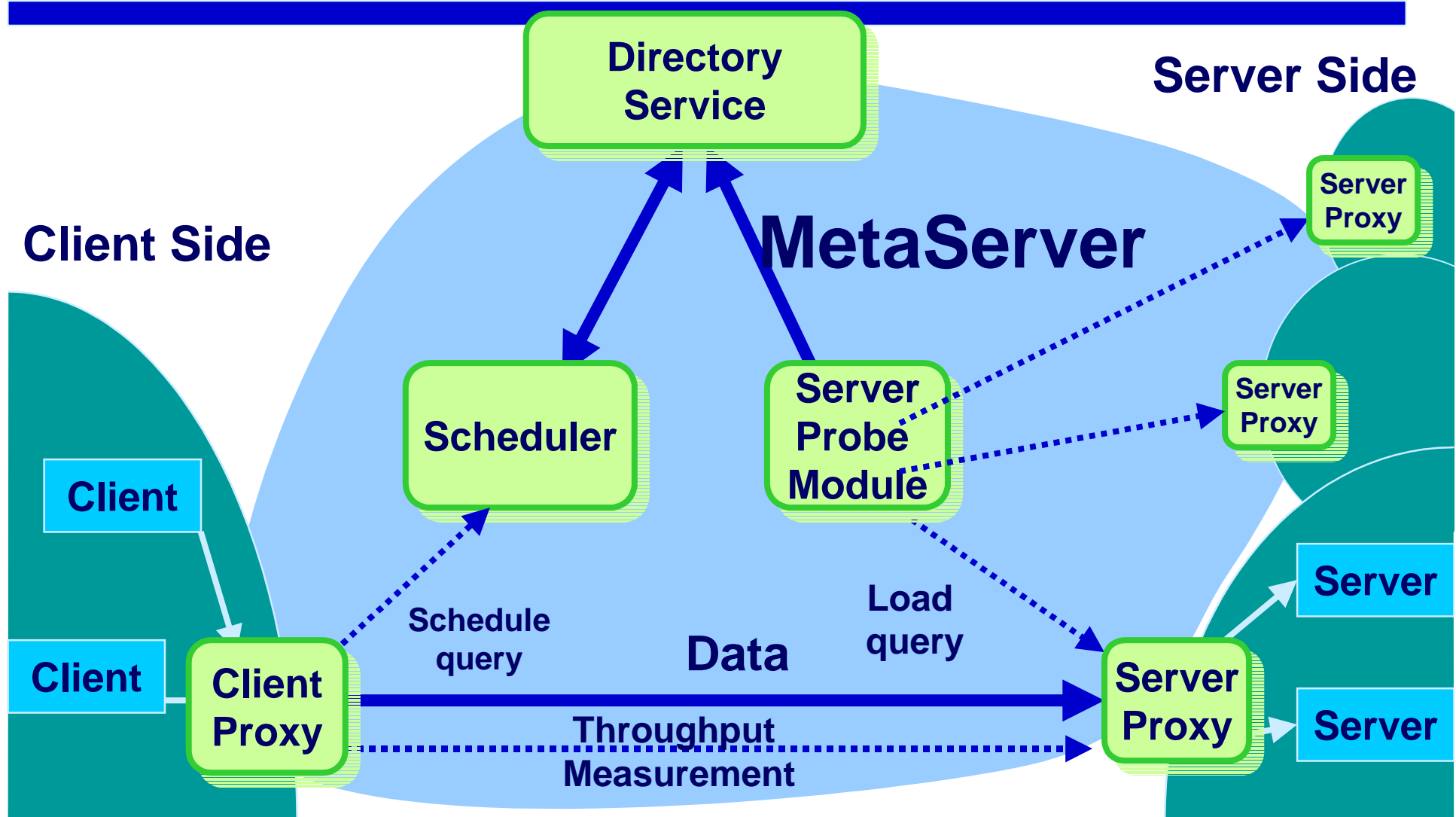


Scheduling

- **An effective scheduling is required!**
 - **Server/Network status dynamically change.**
 - **Status information is distributed in WAN.**
 - **Servers may be located inside Firewall.**
- **Our approaches**
 - **Metaserver**
 - **monitoring server/network status, performing scheduling**
 - **development of effective scheduling systems**
 - **Bricks**
 - **simulating scheduling in Grid**
 - **modeling computation in Grid, discussion of scheduling algorithm by simulation**



MetaServer Architecture





Scheduling Algorithm

- **Metaserver needs to select the most suitable server for client jobs using effective scheduling algorithm.**
- **Performance evaluation/estimation for scheduling algorithm**
 - **measurement on real “Grid” system**
 - ➡ **practical measurement**
 - ➡ **difficult to perform large-scale experiments**
 - ➡ **difficult to have reproducible results**

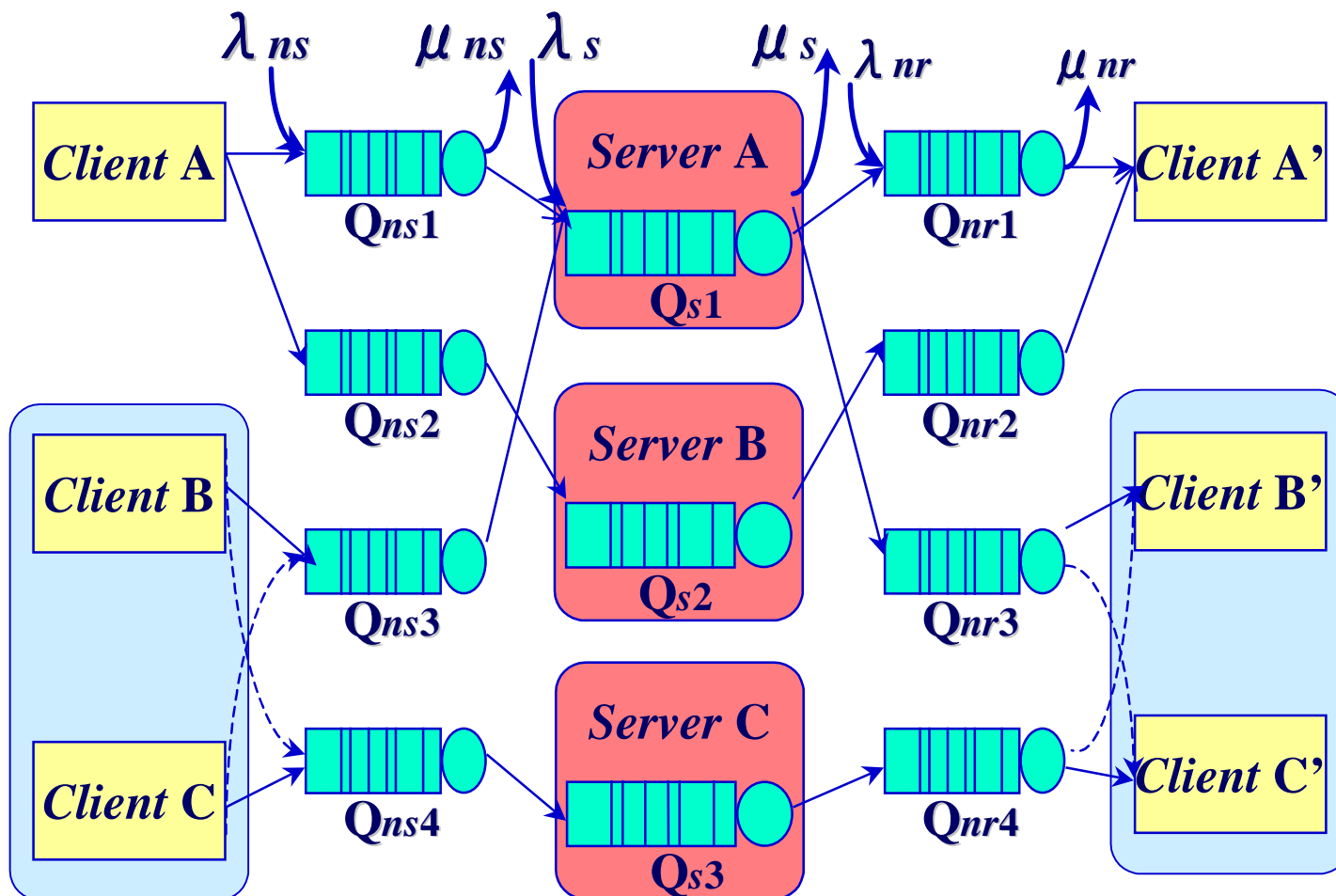
partial solution



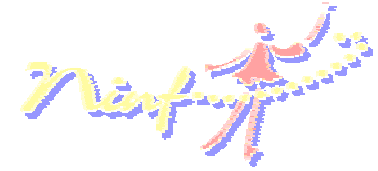
Bricks

- **Simulation system for scheduling in “Grid”**
- **Queuing Model**
 - **A global computing system is modeled by queuing network**
- **Features**
 - **evaluation/estimation of scheduling performance by simulation**
 - **verification of existing scheduling modules for actual Grid environment**

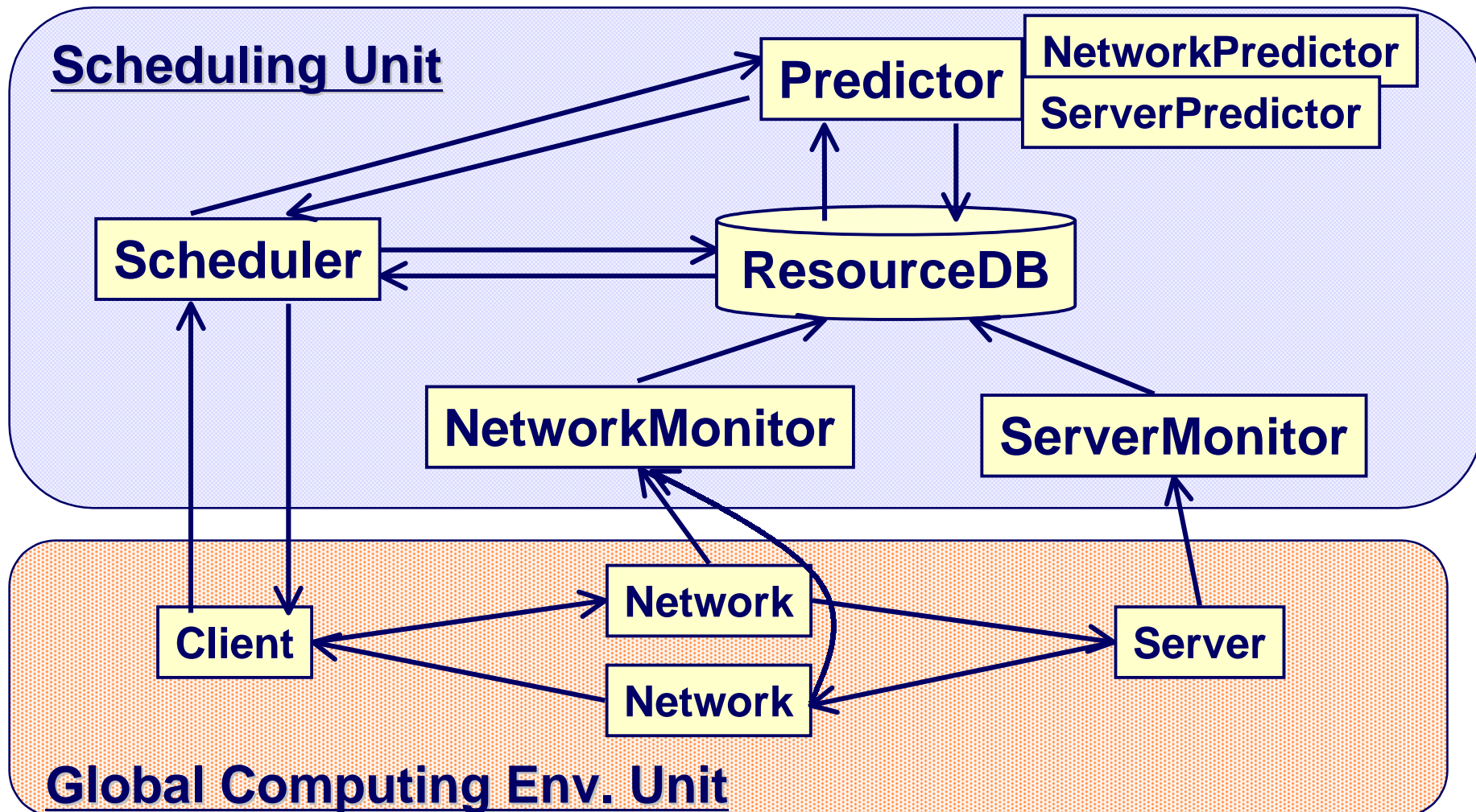
The Model in Bricks (Queuing System)



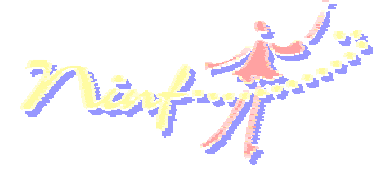
- Networks / Servers are represented as queues
- External network traffic / server jobs represent congestion



Software Architecture of Bricks



Verification of Scheduling Modules with Bricks



○ Replaceable component

- Components in **Scheduling Unit** are easily replaceable with other software modules (modules for actual Grid environment).

☞ (ex.) Monitor/Predictor ↔ NWS

○ Verification of modules with low cost

- Developers of scheduling module can verify/debug/evaluate their developed modules by **Bricks** with low cost.

The verification in actual Grid env. needs expensive cost!

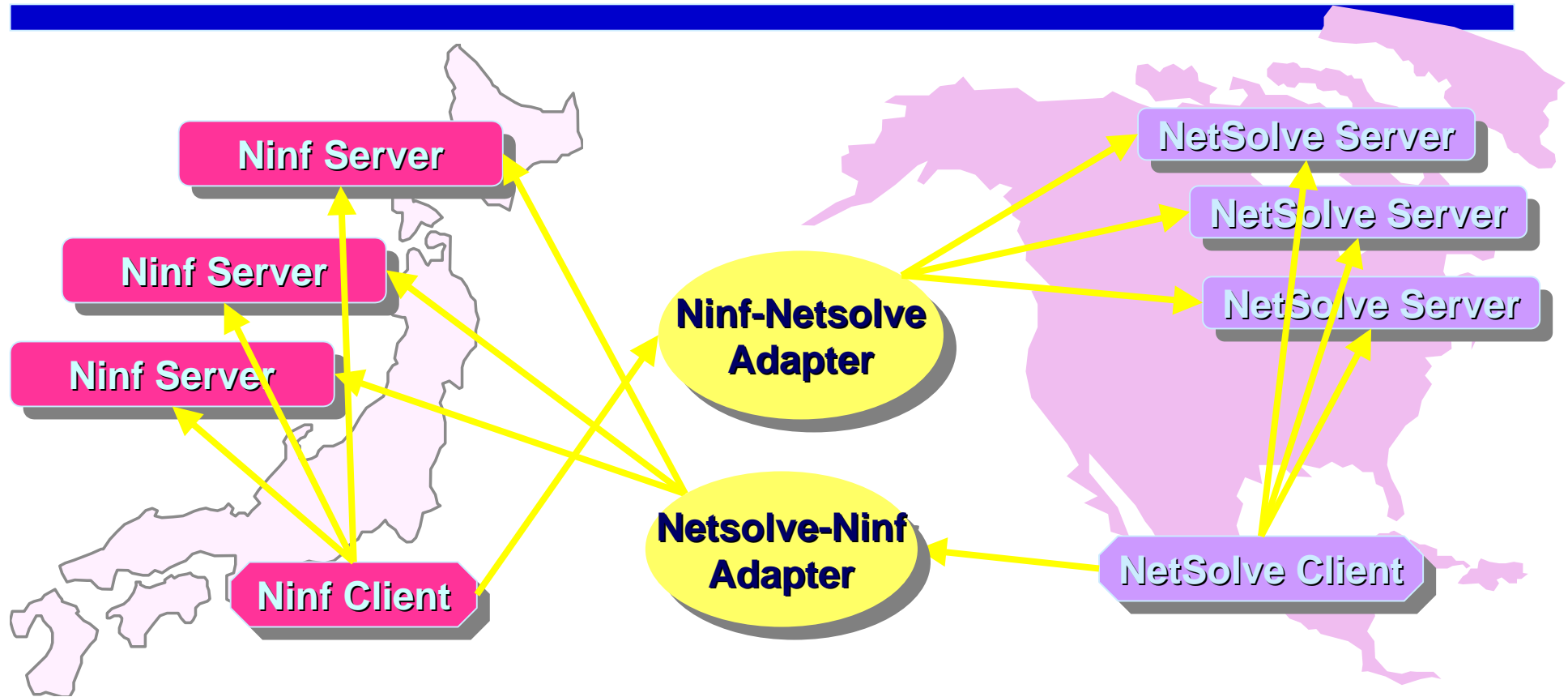


Related Work

- RPC based systems → use existing programming languages
 - **NetSolve** [Casanova and Dongarra, Univ. Tennessee]
 - ☞ The same basic API as Ninf_call (now interchangeable)
 - ☞ load-balancing with a daemon process called Agent.
 - **RCS** [Arbenz, ETH Zurich]
 - ☞ PVM-based
- systems using parallel distributed language etc.
 - **Legion** [Grimshaw, Univ. Virginia]
 - ☞ An user distributes his programs written with the parallel object-oriented language Mentat.
 - **Javelin** [Schauser et al., UCSB]
 - ☞ High portability due to using Java and WWW.



Ninf-NetSolve Collaboration



- Ninf client can use NetSolve server via adapter
- NetSolve client can use Ninf server via adapter



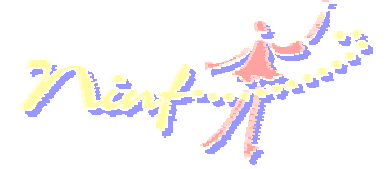
Related Work (cont'd.)

- systems for scheduling
 - **AppLes** [Berman, UCSD]
 - ☞ application level scheduler
 - **NWS** [Wolski, UCSD/NPACI]
 - ☞ monitoring and predicting congestion of server/network
- toolkits:
 - **Globus** [Argonne/USC]



Conclusions

- **Ninf is a RPC based global computing infrastructure.**
 - Ninf RPC protocol
 - easy-to-use client API
 - direct web access, transaction,...
- **Applications**
 - NinfCalc+, Excel Ninf, NetCFD,...
 - Demo. over TransPAC [SC98, SC99]
- **Scheduling**
 - Metaserver
 - Bricks



Conclusions (cont'd.)

○Ninf platforms

- Server: Solaris1,2, DEC, UNICOS, Linux, FreeBSD
- Client: server platforms + Win32

○Contact

<http://ninf.etl.go.jp/>