Programming on the Grid using GridRPC

Yoshio Tanaka

Grid Technology Research Center, AIST

Outline

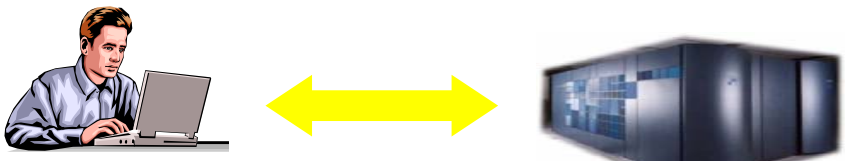
- **Tutorial Talk (~45 min)**
  - What is GridRPC?
    - Overview
    - v.s. MPI
    - Typical scenarios
  - Overview of Ninf-G and GridRPC API
    - Ninf-G: Overview and architecture
    - GridRPC API
    - Ninf-G API
  - How to develop Grid applications using Ninf-G
    - Build remote libraries
    - Develop a client program
    - Run
- **Practicals (30 min)**
- **Recent activities/achievements and summary (10 min)**

What is GridRPC?

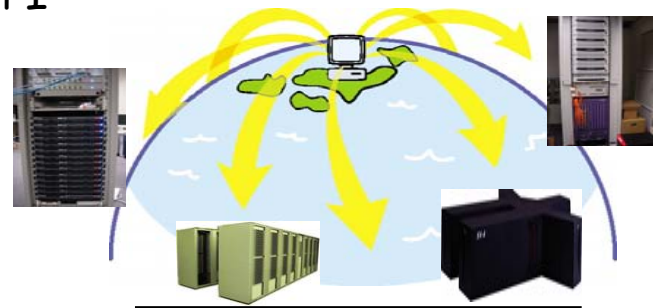
Programming model on Grid based on
Grid Remote Procedure Call (GridRPC)

GridRPC: A programming model based on RPC

- **What is GridRPC?**
  - Realizing remote procedure call (RPC) on the Grid
  - The GridRPC API is published as a GGF proposed recommendation (GFD-R.P 52)
- **Usage scenarios (suitable applications)**
  - Desktop Supercomputing
    - Executing compute-intensive tasks on a remote high performance computing resource
  - Task parallel applications
    - Executing large numbers of independent tasks on distributed computing resources
      - parameter survey
      - combinatorial optimization problem solver
- **Easy three steps to make your program Grid aware**
  - Write IDL file that specifies interface of your library
  - Compile it with an IDL compiler called ng_gen
  - Modify your client program to use GridRPC API



Desktop Supercomputing



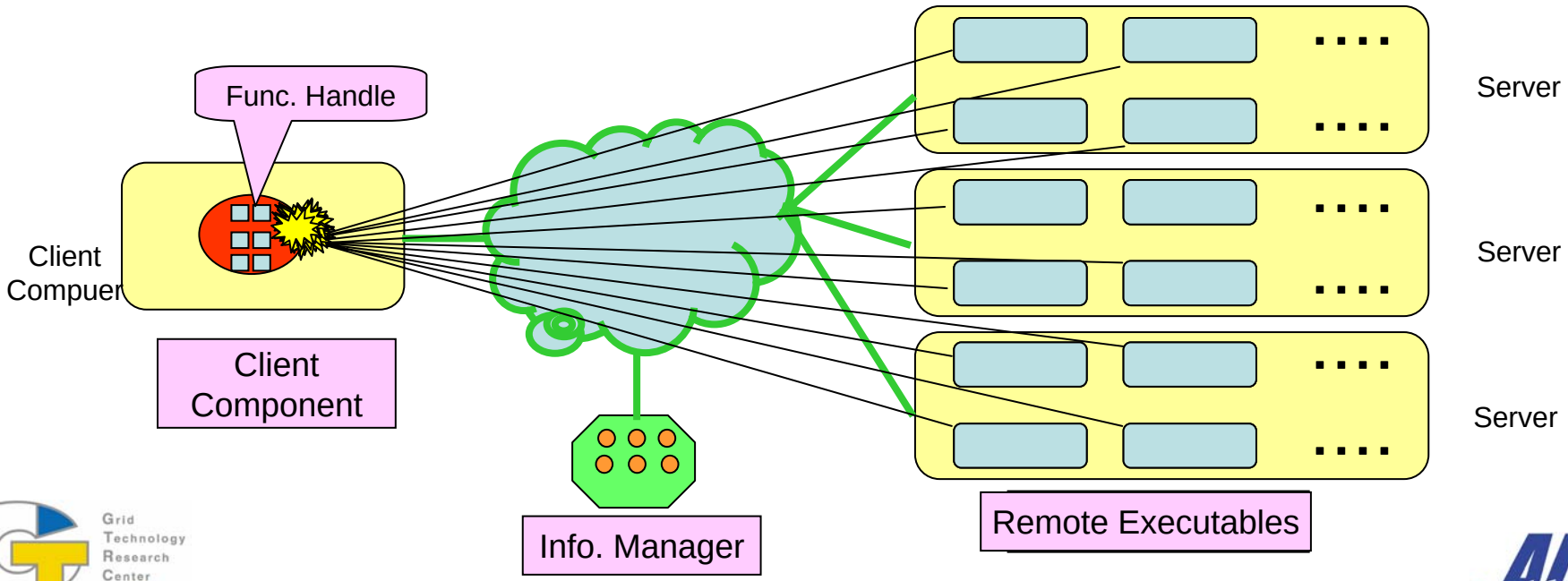
Task parallel processing

Grid Technology Research Center AIST

AIST

GridRPC Model

- **Client Component**
  - Caller of GridRPC.
  - Manages remote executables via function handles
- **Remote Executables**
  - Callee of GridRPC.
  - Dynamically generated on remote servers.
- **Information Manager**
  - Manages and provides interface information for remote executables.



GridRPC v.s. MPI

	GridRPC	MPI
parallelism	task parallel	data parallel
model	client/server	SPMD
API	GridRPC API	MPI
co-allocation	dispensable	indispensable
fault tolerance	good	poor (fatal)
private IP nodes	available	unavailable
resources	can be dynamic	static *
others	easy to gridify existing apps.	well known seamlessly move to Grid

* May be dynamic using process spawning

Ninf-G: A Reference Implementation of the GridRPC API

- **A software package which implements the GridRPC API.**
- **Ninf-G includes**
  - C/C++, Java APIs, libraries for software development
  - IDL compiler for stub generation
  - Shell scripts to
    - compile client program
    - build and publish remote libraries
  - sample programs and manual documents

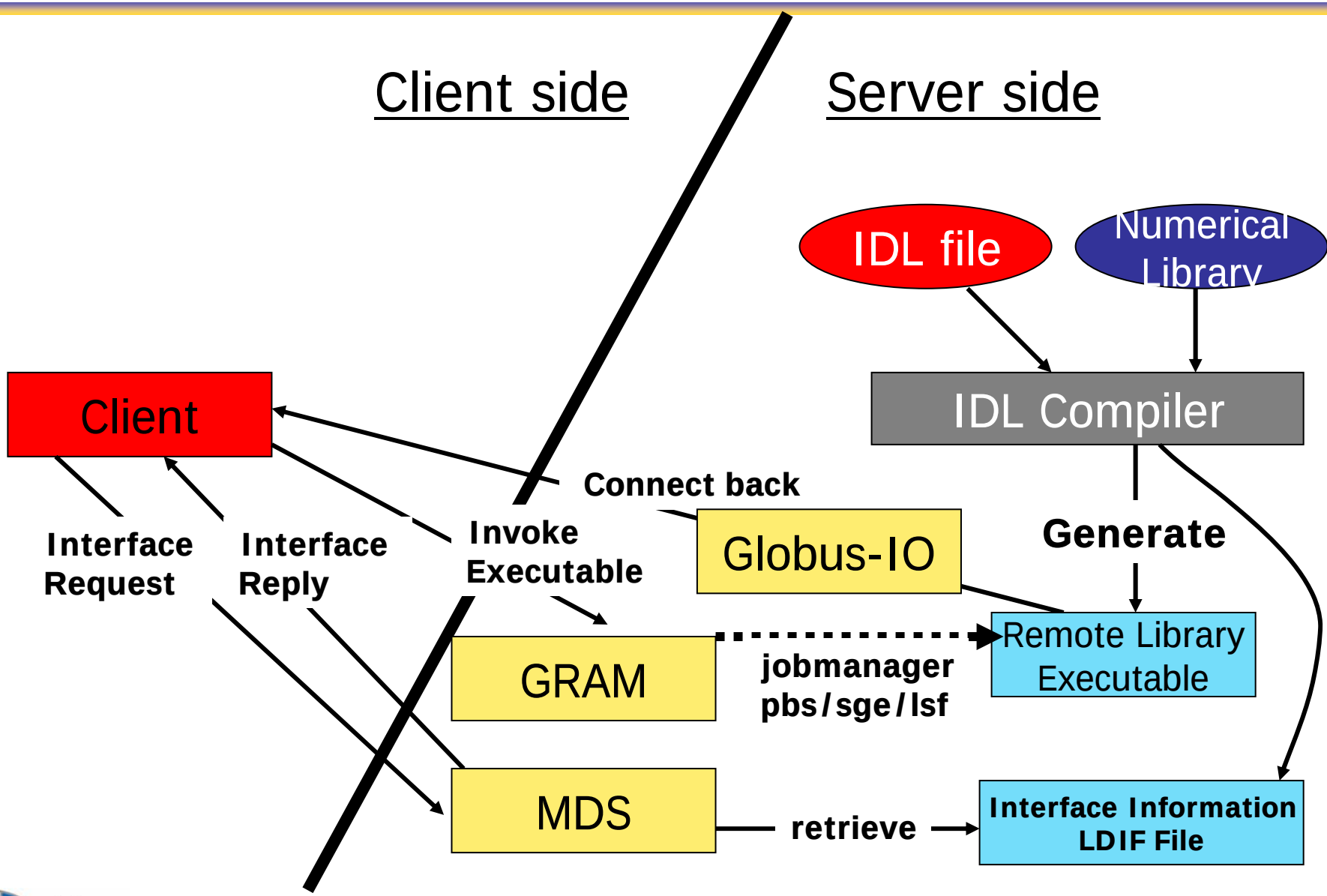
Ninf-G: A Reference Implementation of the GridRPC API (cont'd)

- **Ninf-G is developed using Globus C and Java APIs**
- **Two major versions**
  - Version 2 (Ninf-G2) (not yet supported)
    - Works with GT2 and pre-WS GRAM in GT3, GT4
    - Latest version is 2.4.0a
  - Version 4 (Ninf-G4)
    - Works with GT4 WS GRAM and Pre-WS GRAM
    - Has an interface for working with other Grid middleware for remote process invocation
      - UNICORE
      - Condor and ssh will be available on Mid. Sep.
    - Latest version is 4.1.0
- **Ninf-G is included in NMI (NSF Middleware Initiative)**
  - Ninf-G 4.1.0 is in Release 9.

How to use Ninf-G

- **Build remote libraries on server machines**
  - Write IDL files
  - Compile the IDL files
  - Build and install remote executables
- **Develop a client program**
  - Programming using GridRPC API
  - Compile
- **Run**
  - Create a client configuration file
  - Generate a proxy certificate
  - Run

Architecture of Ninf-G



Rough steps for RPC (client-side programming)

- **Initialization**

```
grpc_initialize(config_file);
```

- **Create a function handle**
  - abstraction of a connection to a remote executable

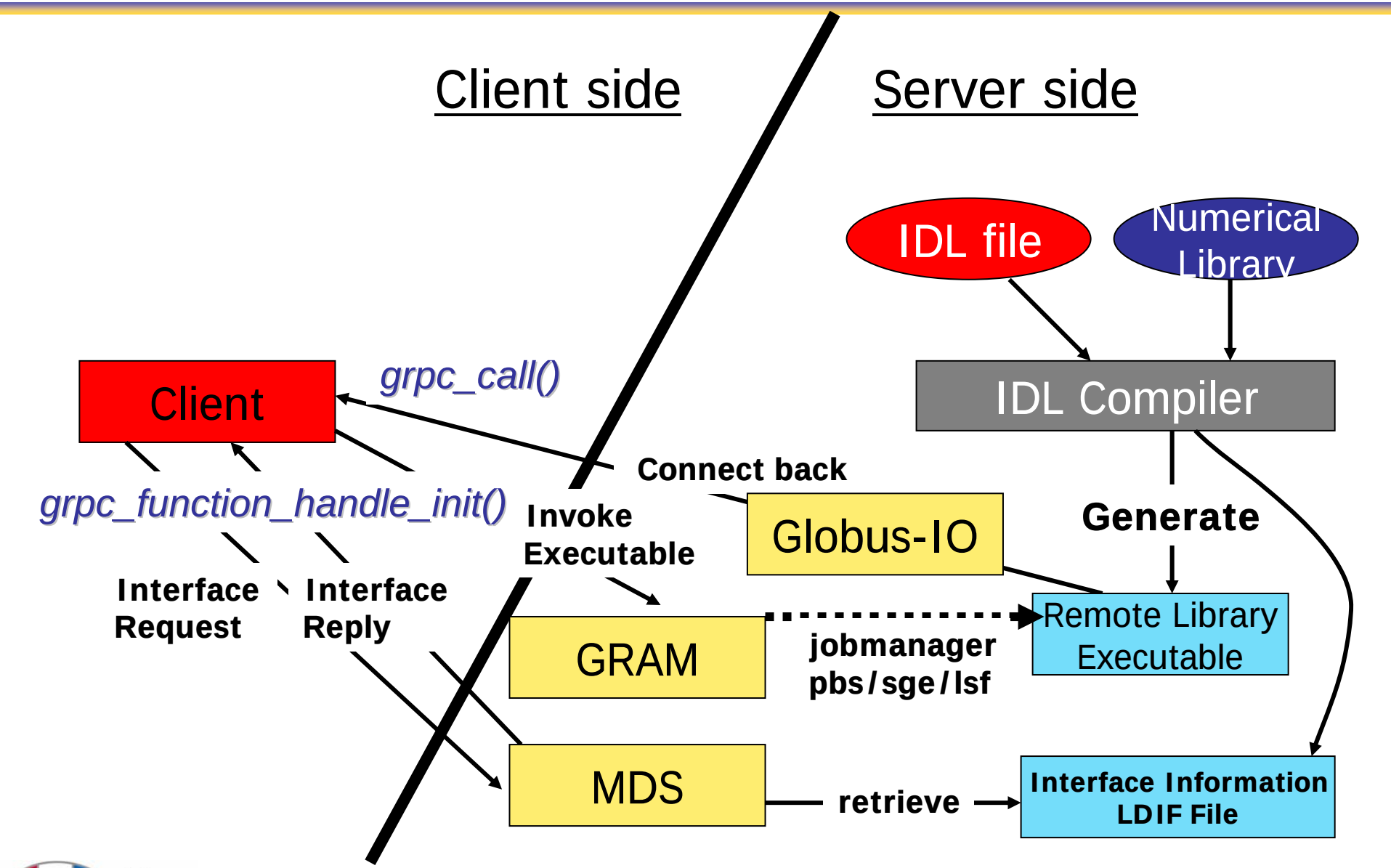
```
grpc_function_handle_t  handle;

grpc_function_handle_init(
  &handle, host, port, “lib_name”);
```

- **Call a remote library**

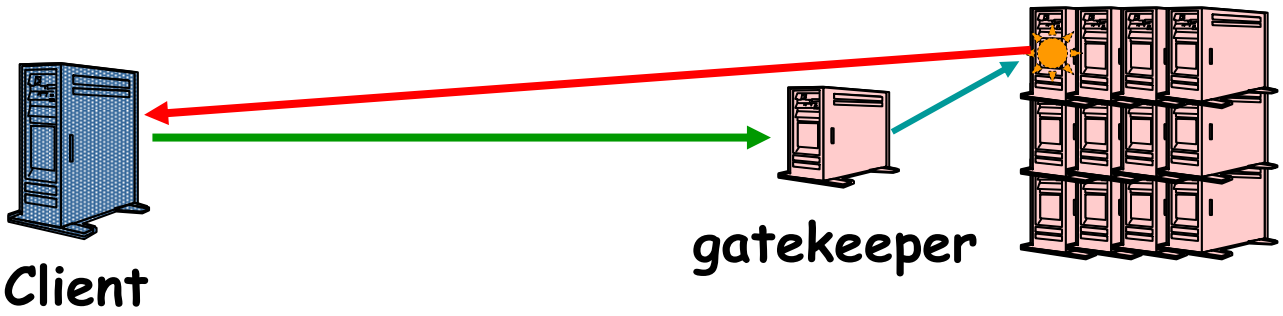
```
grpc_call(&handle, args…);
          or
grpc_call_async(&handle, args…);
grpc_wait( );
```

Architecture of Ninf-G

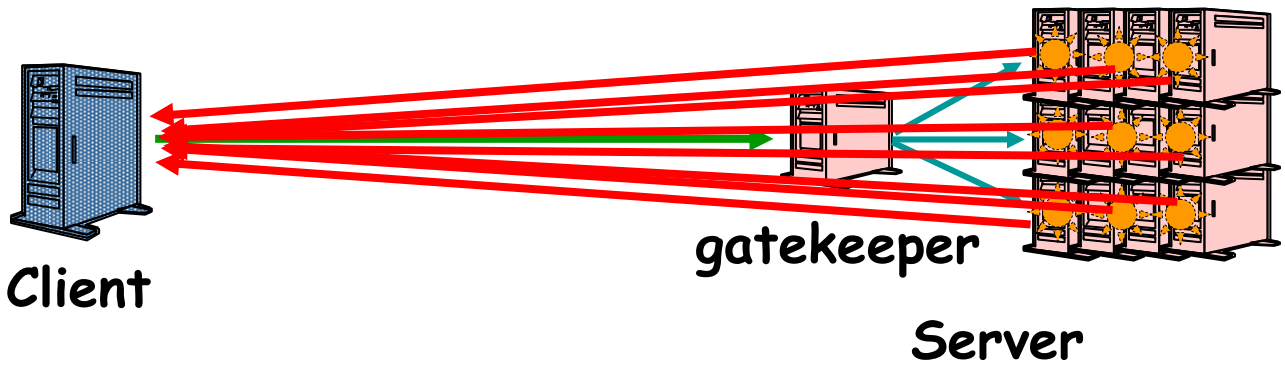


Three types of remote process invocation

- **Single (jobtype=single)**
  - grpc_function_handle_init(...);

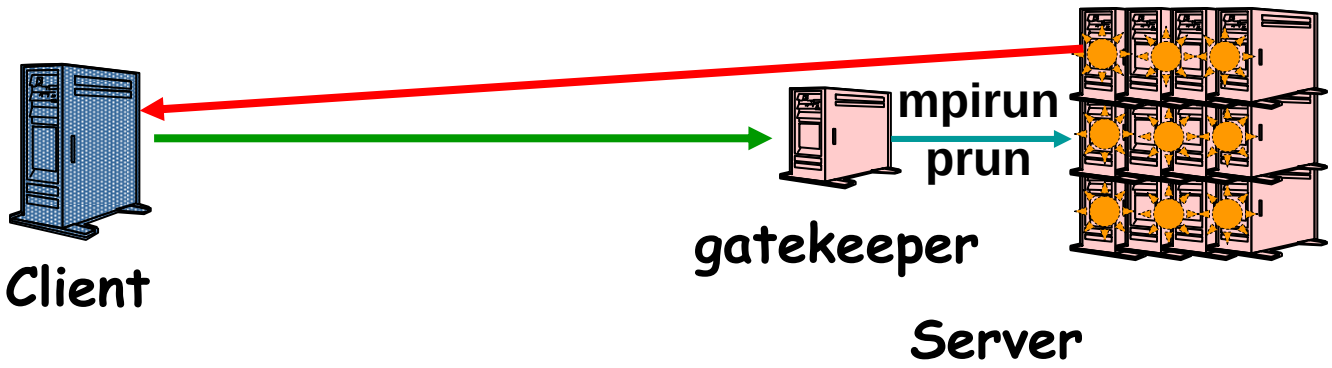


- **Multiple (jobtype=multiple)**
  - grpc_function_handle_array_init_np(...);



Three types of remote process invocation (cont'd)

- **MPI (jobtype=MPI)**
  - grpc_function_handle_init(...);
  - "Backend=MPI" is specified in the IDL



Ninf-G fnctions for fault detection

- **Ninf-G Client API returns an error if it detects explicit errors**
  - server processes died
  - network disconnection
- **Ninf-G provides functions for implicit error detection**
  - timeout
    - for initialization (e.g. stacked in the queue, etc.)
    - for execution (something unexpected would happen)
    - heatbeat

Ninf-G

Overview and Architecture

Terminology

- **Ninf-G Client**
  - This is a program written by a user for the purpose of controlling the execution of computation.
- **Ninf-G IDL**
  - Ninf-G IDL (Interface Description Language) is a language for describing interfaces for functions and objects those are expected to be called by Ninf-G client.
- **Ninf-G Stub**
  - Ninf-G stub is a wrapper function of a remote function/object. It is generated by the stub generator according to the interface description for user-defined functions and methods.

Terminloogy (cont'd)

- **Ninf-G Executable**
  - Ninf-G executable is an executable file that will be invoked by Ninf-G systems. It is obtained by linking a user-written function with the stub code, Ninf-G and the Globus Toolkit libraries.
- **Session**
  - A session corresponds to an individual RPC and it is identified by a non-negative integer called Session ID.
- **GridRPC API**
  - Application Programming Interface for GridRPC. The GridRPC API is going to be standardized at the GGF GridRPC WG.

Ninf-G

How to build Remote Libraries
- server side operations -

Ninf-G remote libraries

- **Ninf-G remote libraries are implemented as executable programs (Ninf-G executables) which**
  - contain stub routine and the main routine
  - will be spawned off by GRAM or other middleware
- **The stub routine handles**
  - communication with clients and Ninf-G system itself
  - argument marshalling
- **Underlying executable (main routine) can be written in C, C++, Fortran, etc.**

Ninf-G remote libraries (cont'd)

- **Ninf-G provides two kinds of Ninf-G remote executables:**
  - Function
    - Stateless
    - Defined in standard GridRPC API
  - Ninf-G object
    - stateful
    - enables to avoid redundant data transfers
    - multiple methods can be defined
      - initialization
      - computation

How to build Ninf-G remote libraries (1/3)

- Write an interface information using Ninf-G Interface Description Language (Ninf-G IDL).
  Example:

```
Module mmul;
Define dmmul (IN int n,
              IN double A[n][n],
              IN double B[n][n],
              OUT double C[n][n])
Require "libmmul.o"
Calls "C" dmmul(n, A, B, C);
```

- Compile the Ninf-G IDL with Ninf-G IDL compiler

  % ng_gen <IDL_FILE>

  ns_gen generates stub source files and a makefile (<module_name>.mak)

How to build Ninf-G remote libraries (2/3)

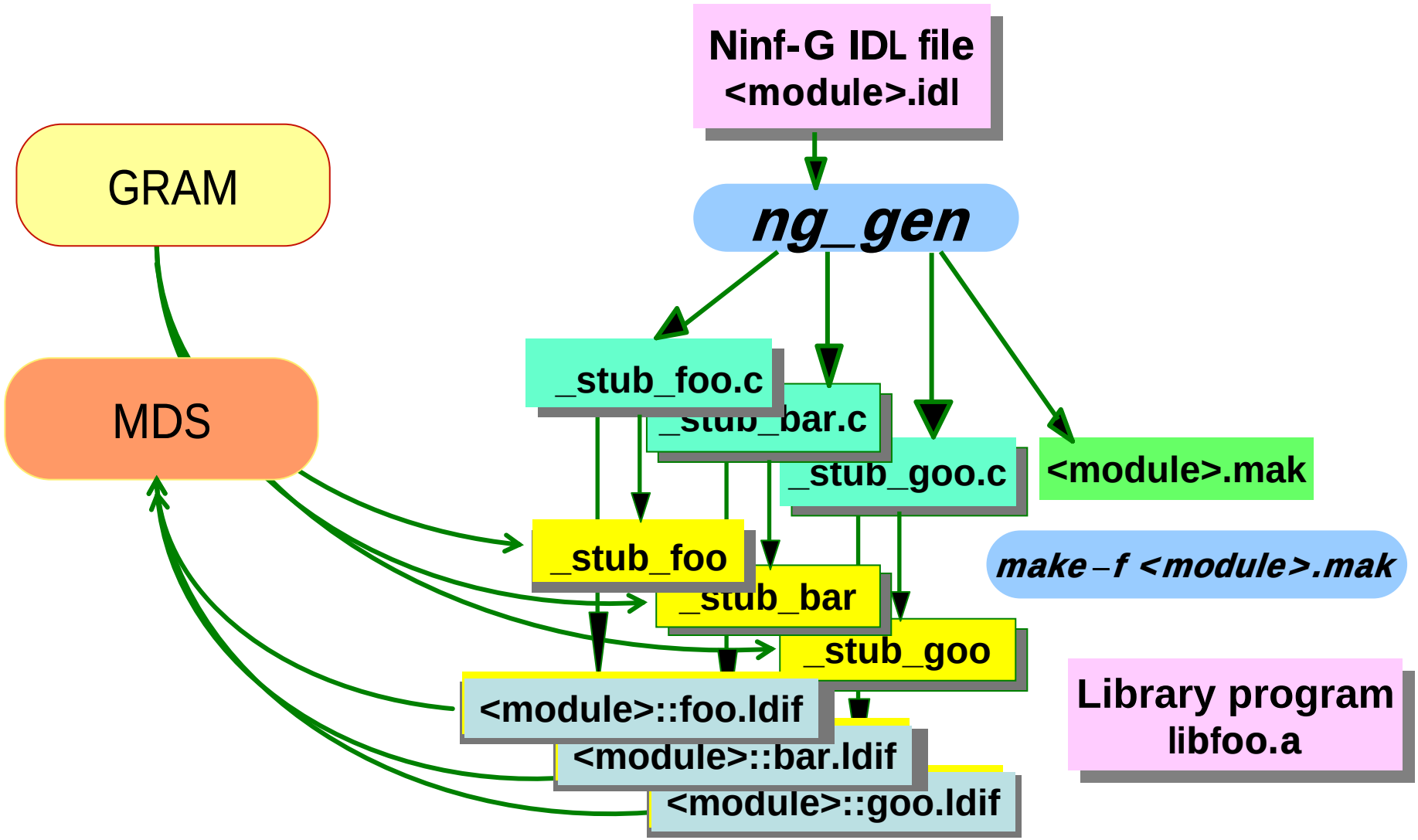
- **Compile stub source files and generate Ninf-G executables and LDIF files (used to register Ninf-G remote libs information to GRIS).**

  % make –f <module_name>.mak

- **Publish the Ninf-G remote libraries (optional)**

  % make –f <module_name>.mak install

How to build Ninf-G remote libraries (3/3)



Ninf-G

How to call Remote Libraries
- client side APIs and operations -

(Client) User's Scenario

- Write client programs in C/C++/Java using APIs provided by Ninf-G
- Compile and link with the supplied Ninf-G client compile driver (*ng_cc*)
- Write a client configuration file in which runtime environments can be described
- Run *grid-proxy-init* command
- Run the program

GridRPC API / Ninf-G API

APIs for programming client applications

The GridRPC API and Ninf-G API

- **GridRPC API**
  - Standard C API defined by the GGF GridRPC WG.
  - Provides portable and simple programming interface.
  - Enable interoperability between implementations suchas Ninf-G and NetSolve.
- **Ninf-G API**
  - Non-standard API (Ninf-G specific)
  - complement to the GridRPC API
  - provided for high performance, usability, etc.
  - ended by _np
    - eg: grpc_function_handle_array_init_np(…)

Rough steps for RPC

- **Initialization**

```
grpc_initialize(config_file);
```

- **Create a function handle**
  - abstraction of a connection to a remote executable

```
grpc_function_handle_t  handle;

grpc_function_handle_init(
  &handle, host, port, “func_name”);
```

- **Call a remote library**
  - synchronous or asynchronous call

```
grpc_call(&handle, args…);
          or
grpc_call_async(&handle, args…);
grpc_wait( );
```

Data types

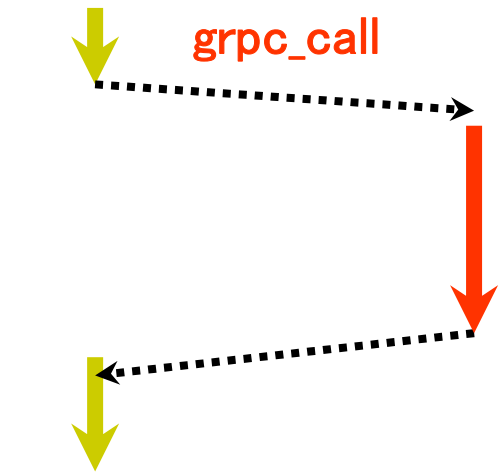
- **Function handle – *grpc_function_handle_t***
  - A structure that contains a mapping between a client and an instance of a remote function
- **Object handle – *grpc_object_handle_t_np***
  - A structure that contains a mapping between a client and an instance of a remote object
- **Session ID – *grpc_sessionid_t***
  - Non-nevative integer that identifies a session
  - Session ID can be used for status check, cancellation, etc. of outstanding RPCs.
- **Error and status code – *grpc_error_t***
  - Integer that describes error and status of GridRPC APIs.
  - All GridRPC APIs return error code or status code.

Synchronous RPC v.s. Asynchronous RPC

Synchronous RPC

- Blocking Call
- Same semantics with a local function call.

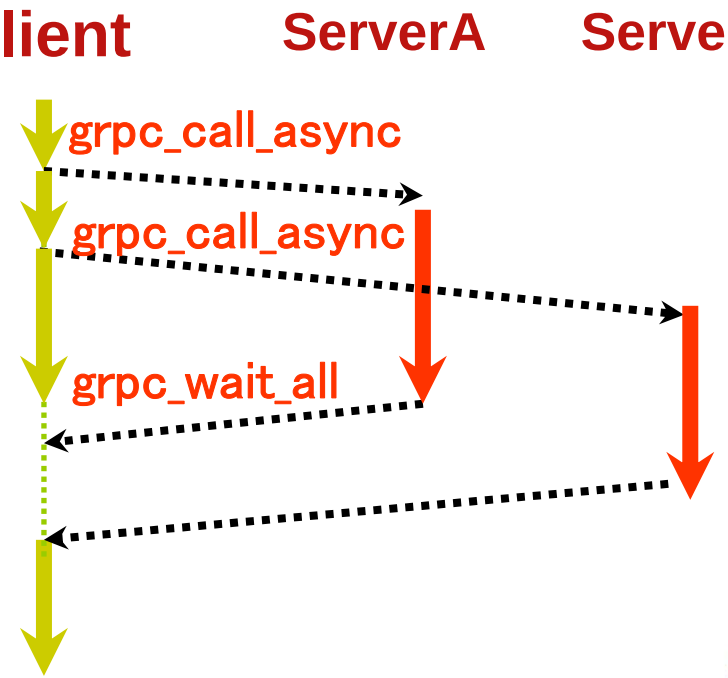
```
grpc_call(...);
```



Asynchronous RPC

- Non-blocking Call
- Useful for task-parallel applications

```
grpc_call_async(...);
grpc_wait_*(...);
```



Grid Technology Research Center AIST

AIST

Ninf-G

Compile and run

Prerequisite

- **Environment variables**
  - GLOBUS_LOCATION
  - NG_DIR
- **PATH**
  - ${GLOBUS_LOCATION}/etc/globus-user-env.{csh,sh}
  - ${NG_DIR}/etc/ninfg-user-env.{csh,sh}
- **Globus-level settings**
  - User certificate, CA certificate, grid-mapfile
  - test
    % grid-proxy-init
    % globus-job-run server.foo.org /bin/hostname

Compile and run

- Compile the client application using *ngcc* command
  % ng_cc -o myapp app.c

- Create a proxy certificate
  % grid-proxy-init

- Prepare a client configuration file

- Run
  % ./myapp config.cl [args...]

Client configuration file

- **Specifies runtime environments**
- **Available attributes are categorized to sections:**
  - INCLUDE section
  - CLIENT section
  - LOCAL_LDIF section
  - FUNCTION_INFO section
  - MDS_SERVER section
  - SERVER section
  - SERVER_DEFAULT section
  - INVOKE_SERVER section

Frequently used attributes

- **<CLIENT> </CLIENT> section**
  - loglevel
  - refresh_credential
- **<SERVER> </SERVER> section**
  - hostname
  - mpi_runNoOfCPUs
  - jobmanager
  - job_startTimeout
  - job_queue
  - heatbeat / heatbeat_timeoutCount
  - redirect_outerr
- **<FUNCTION_INFO> </FUNCTION_INFO> section**
  - session_timeout
- **<LOCAL_LDIF> </LOCAL_LDIF> section**
  - filename

Ninf-G

Summary

How to use Ninf-G (again)

- **Build remote libraries on server machines**
  - Write IDL files
  - Compile the IDL files
  - Build and install remote executables
- **Develop a client program**
  - Programming using GridRPC API
  - Compile
- **Run**
  - Create a client configuration file
  - Generate a proxy certificate
  - Run

Ninf-G tips

- **How the server can be specified?**
  - Server is determined when the function handle is initialized.
    - *grpc_function_handle_init();*
      - hostname is given as the second argument
    - *grpc_function_handle_default();*
      - hostname is specified in the client configuration file which must be passed as the first argument of the client program.
  - Ninf-G does not provide broker/scheduler/meta-server.
- **Should use LOCAL LDIF rather than MDS.**
  - easy, efficient and stable
- **How should I deploy Ninf-G executables?**
  - Deploy Ninf-G executables manually
  - Ninf-G provides automatic staging of executables
- **Other functionalities?**
  - heatbeating
  - timeout
  - client callbacks
  - attaching to debugger
  - ...

Ninf-G

practicals

http://ninf.apgrid.org/NBCR/

username: ninf

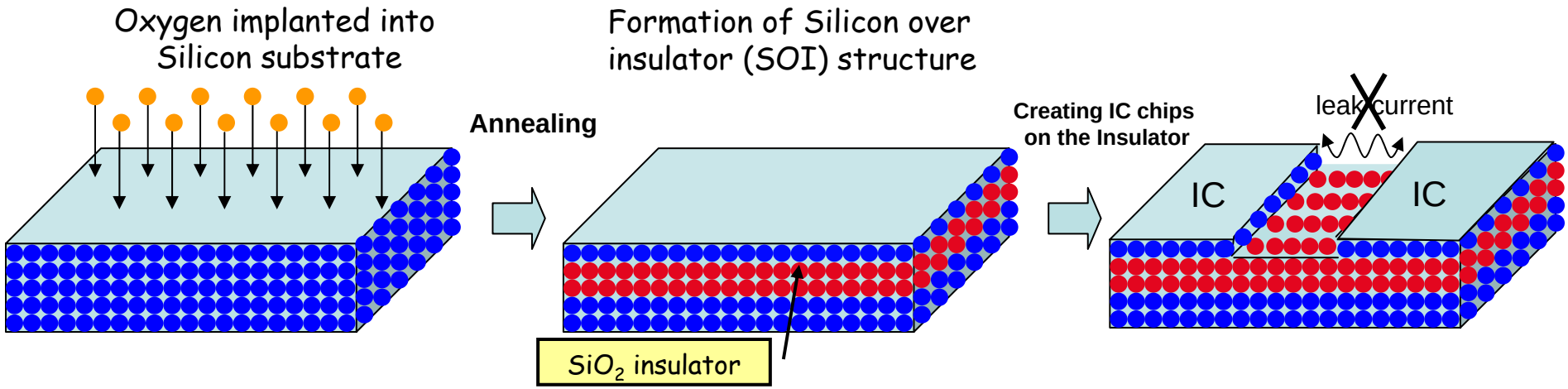
password: aist-ninfg

Ninf-G

Recent achievements

SIMOX (Separation by Implanted Oxygen)

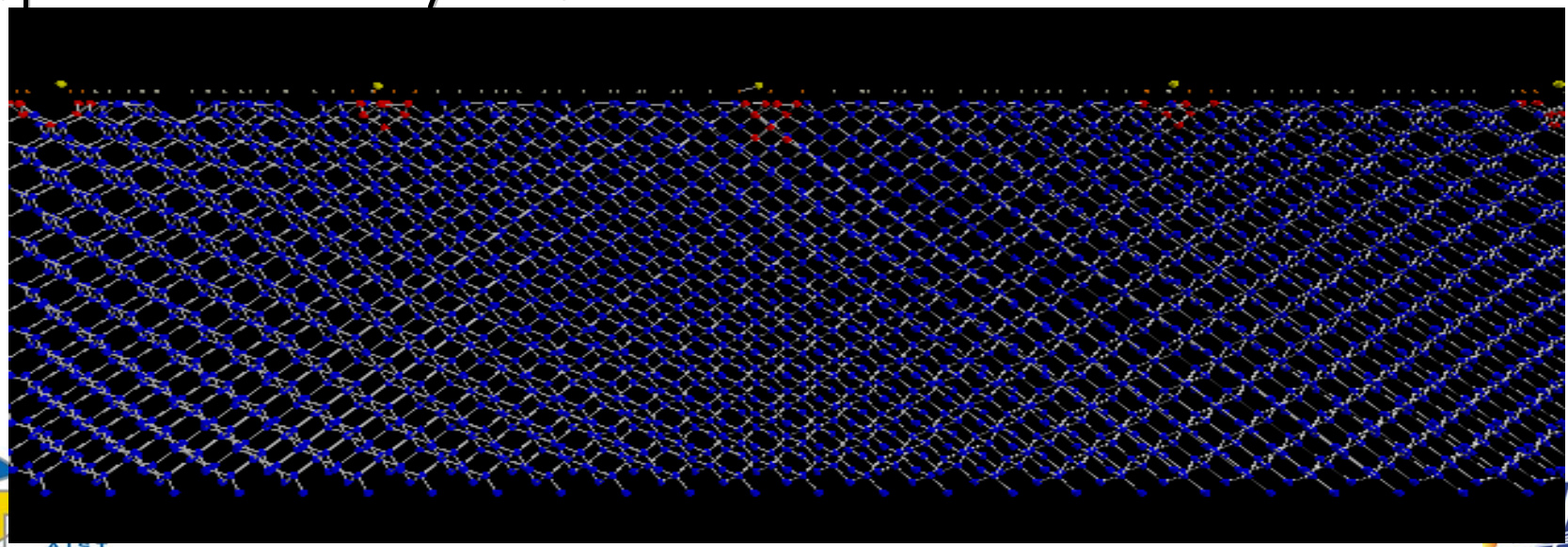
- A technique to fabricate a micro structure consisting of Si surface on the thin SiO_2 insulator
- Allows to create higher speed with lower power consumption device



- This technology has advantages for portable products, such as laptops, hand-held devices, and other applications that depend on battery power.
- Further advancement of the SIMOX technology to fabricate ultra-fine scale SOI structures in future, requires to understand the effects of the initial velocity and incident position of the implanted oxygen on the oxidation processes.

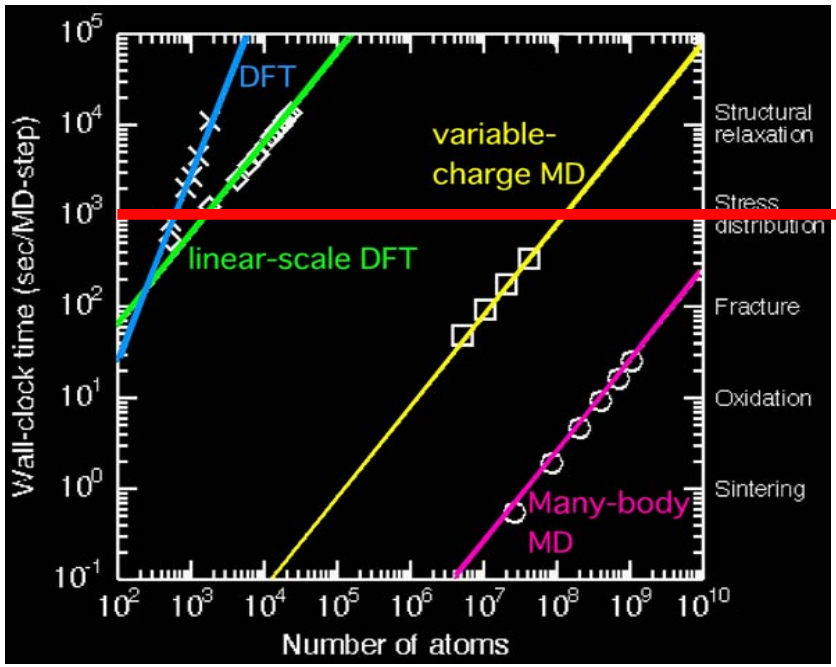
SIMOX simulation on the Grid

- Simulate SIMOX by implanting five oxygen atoms with their initial velocities much smaller than the usual values.
- The incident positions of the oxygen atoms relative to the surface crystalline structure of Si differ.
- 5 QM regions are initially defined
  - **Size and No. of QM regions are changed during the simulation**
- 0.11million atoms in total
- Results of the experiments will demonstrate the sensitivity of the process on the incident position of the oxygen atom when its implantation velocity is small.



Current Atomistic Simulations

- **Accuracy and computation cost are trade-off**
  - **QM Simulation**: large computation cost
    - Treat only small scale atomistic processes (< 10^{-8} m, 10^{-12} sec, 10^{3} atoms)
    - Much accurate
  - **Classical MD Simulation**: small computation cost
    - Treat large scale atomistic processes (~10^{-6} m, 10^{-9} sec, 10^{8} atoms)
    - Less accurate

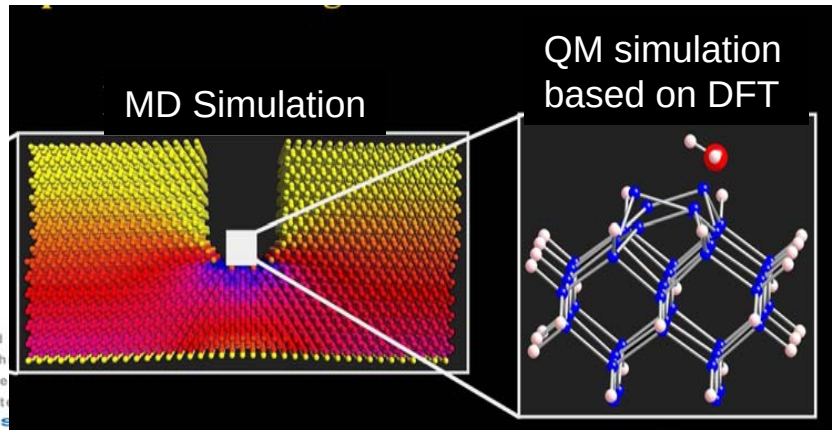


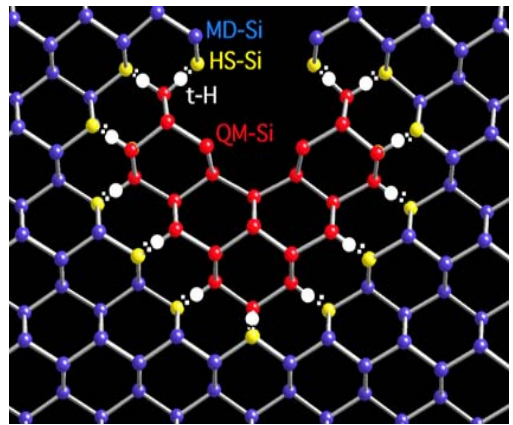
Reasonable Computation time (~10^{3} sec)

Benchmark tests on 1024-node Cray T3E

Hybrid QM/CL Simulation

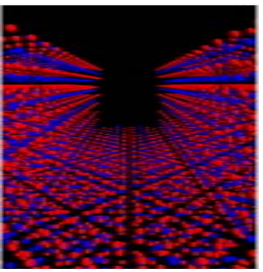
- **Enabling large scale simulation with quantum accuracy**
  - Combining classical MD Simulation with quantum simulation
    - CL simulation
      - Simulating the behavior of atoms in the entire region
      - Based on the classical MD using an empirical inter-atomic potential
    - QM simulation
      - Modifying energy calculated by MD simulation only in the interesting regions
      - Based on the density functional theory (DFT)





Algorithm and Implementation

Algorithm



MD part

initial set-up

Calculate MD forces of QM+MD regions

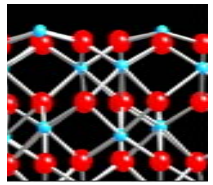
Data of QM atoms

Calculate MD forces of QM region

Calculate QM force of the QM region

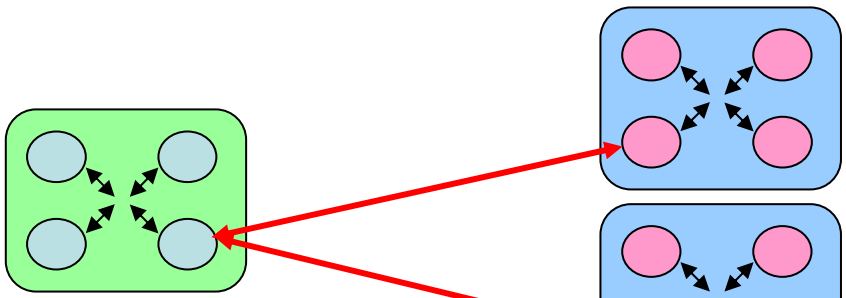
Update atomic positions and velocities

QM forces



QM part

Implementation



Grid Technology Research Center AIST

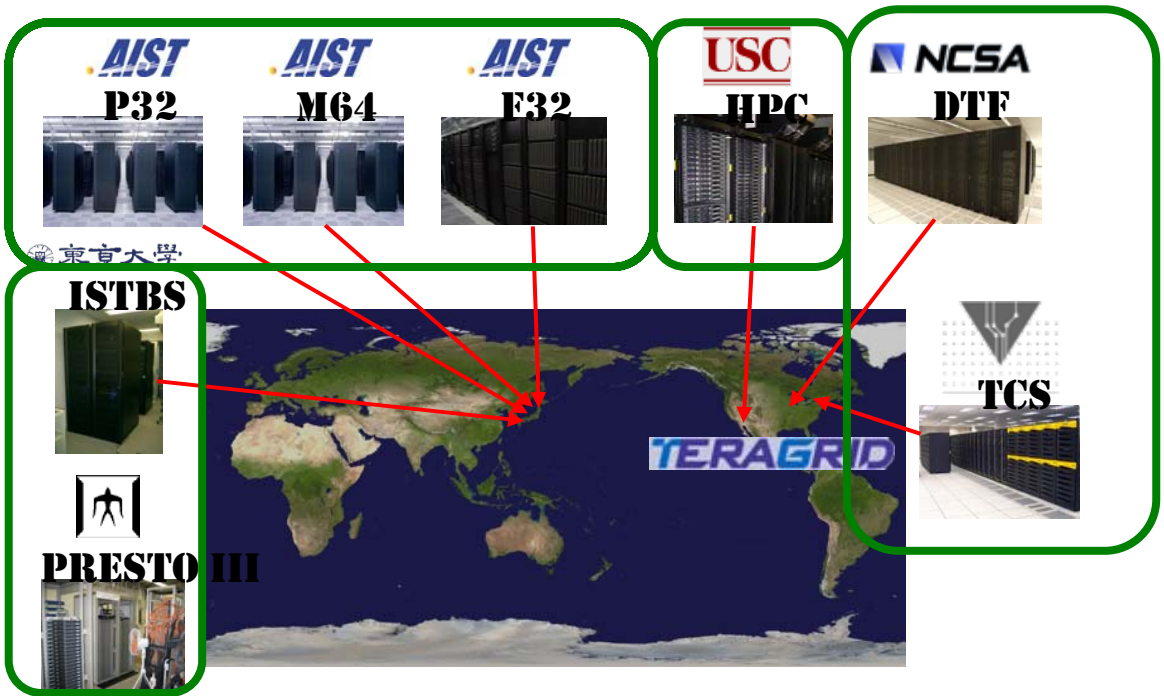
AIST

Testbed for the experiment

- AIST Super Clusters

QM1	P32	P32	P32	P32	P32	USC	USC	USC	ISTBS	ISTBS
QM2	P32	P32	NCSA	NCSA	NCSA	USC	USC	USC	Presto	Presto
QM3	M64	M64	M64	M64	M64	M64	M64	M64	M64	M64
QM4	P32	P32	TCS	TCS	TCS	USC	USC	USC	P32	P32
QM5	P32	P32	TCS	TCS	TCS	USC	USC	USC	P32	P32
Reserve	F32	F32	P32	P32	P32	P32	P32	P32	F32	F32
	Phase 1		Phase 2			Phase 3			Phase 4	

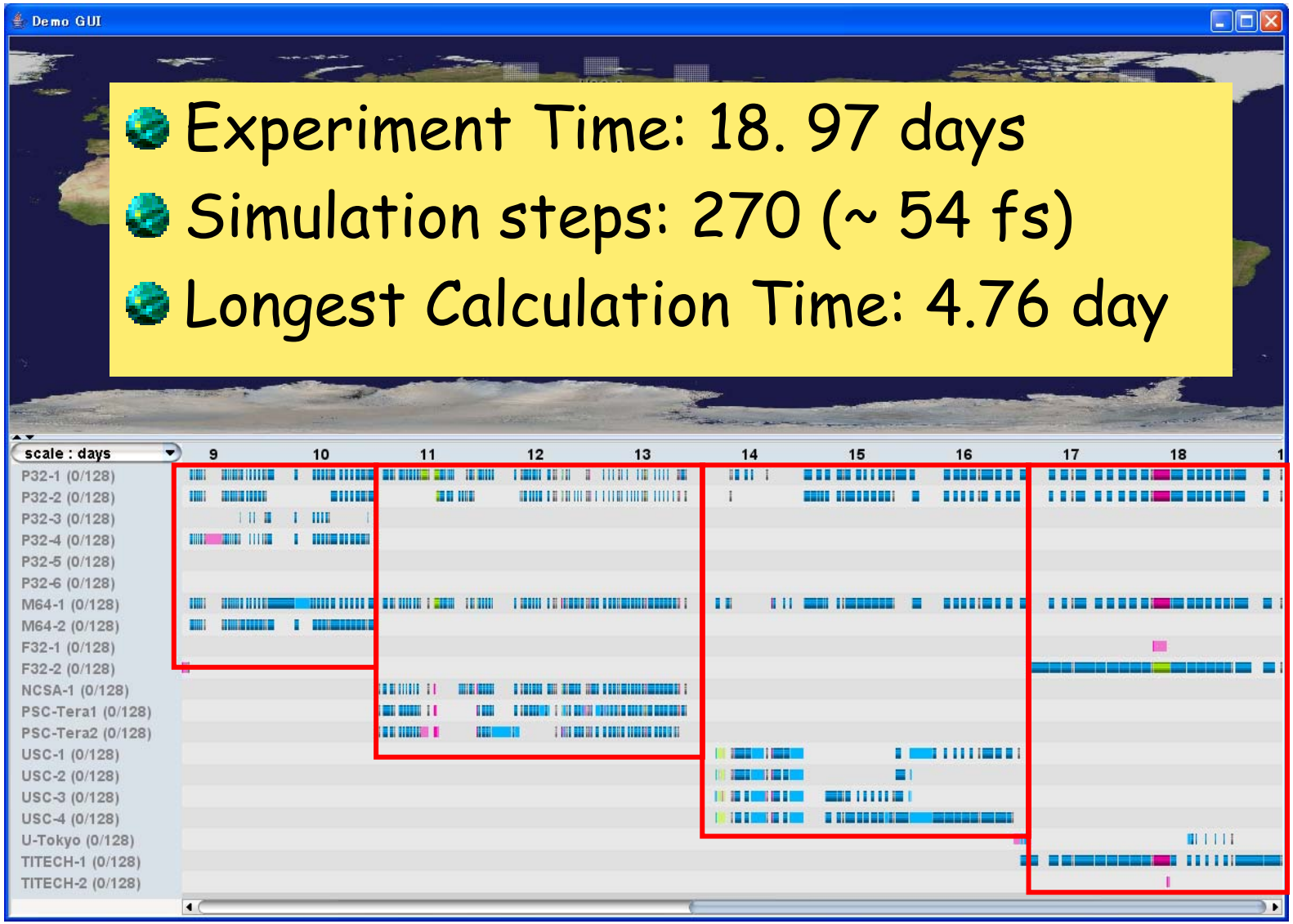
- Japan Clusters
  - U-Tokyo (386 CPUs), TITECH (512 CPUs)



Grid Technology Research Center AIST

AIST

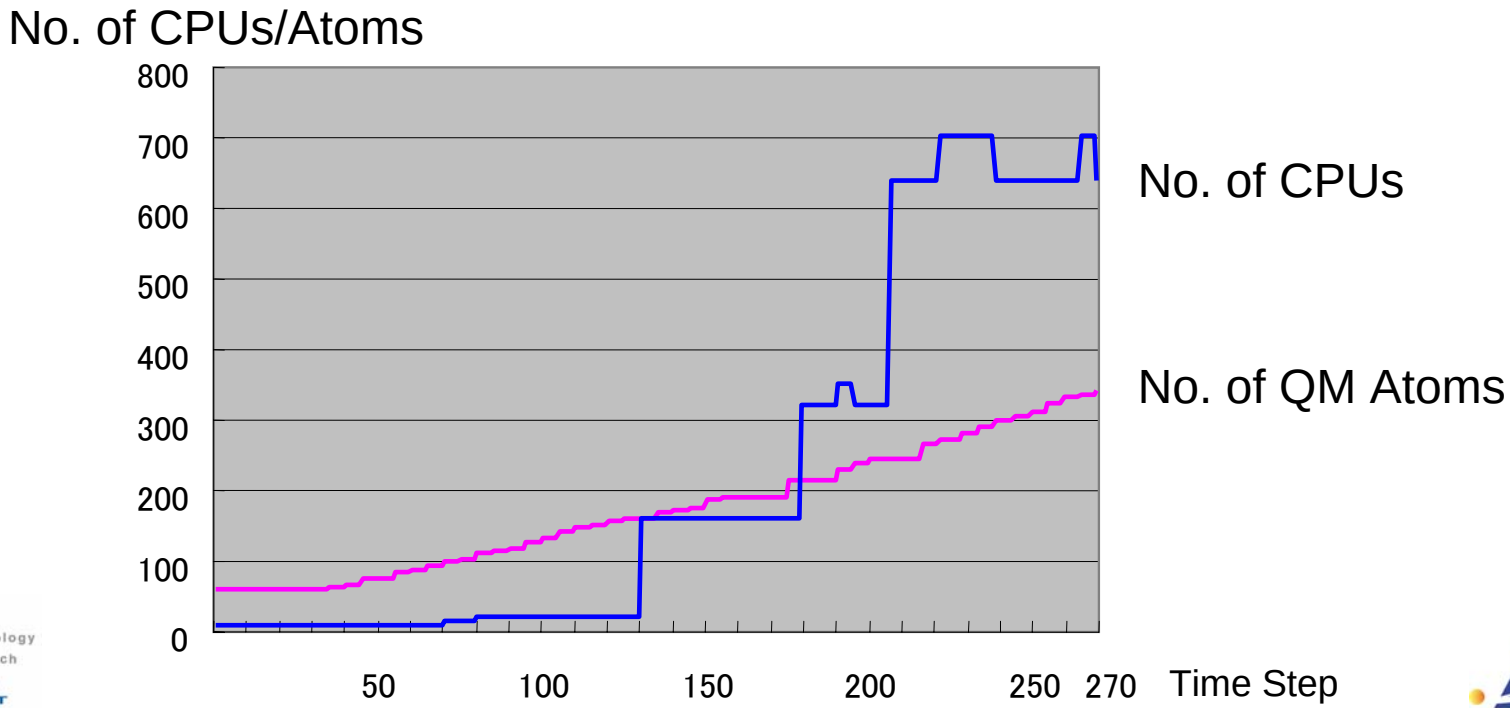
Result of the experiment



Flexibility

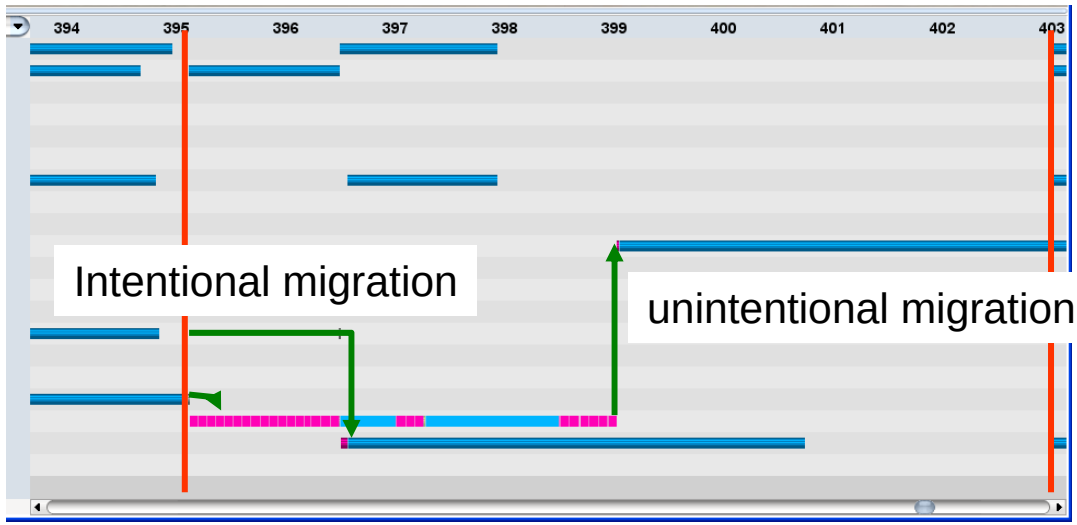
- **Expanding/Dividing QM regions at every 5 time steps**
  - **Expansion: 47 times**
  - **Division: 8 times**

Automatic reallocation of QM simulations to clusters
Automatic adjustment of MPI processes



Robustness

- **Many kinds of errors**
  - Queue was not activated
  - Failed to start MPI programs
  - Exceeding a quota limit
  - :
- **Our application succeeded in detecting errors and continuing simulation using other clusters**



For more info, related links

- **Ninf project ML**
  - ninf@apgrid.org
- **Ninf-G Users' ML (subscribed member's only)**
  - ninf-users@apgrid.org
- **Ninf project home page**
  - http://ninf.apgrid.org
- **Open Grid Forum**
  - http://www.ogf.org/
- **GGF GridRPC WG**
  - http://forge.gridforum.org/projects/gridrpc-wg/
- **Globus Alliance**
  - http://www.globus.org/