A Complete Scenario on Grid
- How to build, program, use a Grid -

Yoshio Tanaka

Grid Technology Research Center, AIST

Background

- **Grid is going to be practical/production level**
  - Key technologies have been standardized
    - GSI, OGSA, etc.
  - Grid middlewares become mature
    - Globus Toolkit, UNICORE, Condor, etc.
- **OK, Let's use Grid ! But...**
  - Do you have a Grid testbed?
  - Is your application Grid-enabled?
  - How do you develop Grid-enabled applications?
    - MPI is the only programming model?

Tutorial Goal

- **Enable attendees to understand**
  - how to build a Grid testbed using the Globus Toolkit version 2 (GT2)
    - sociological issues
    - design, implementation and management
  - how to develop and run Grid applications on Grid testbeds
    - development and execution of Grid applications using Globus-based GridRPC system
    - MPI is not the only programming model on the Grid !
- **Through the above two major topics...**
  - how each component of the GT2 works on a computational Grids.
    - introduction of the Globus Toolkit from a point of view of users

Outline

I. **Review of the GT2 (10min)**
   I. Common software infrastructure for both building and using Grids
II. **How to build a Grid (40min)**
   I. general issues
   II. use the ApGrid Testbed as an example
III. **How to program on a Grid (60min)**
   I. Programming on the Grid using GridRPC
   II. use Ninf-G as a sample GridRPC system
IV. **How to run a Grid application (20min)**
   I. Experiences on running climate simulation on the ApGrid Testbed
V. **Summary (10min)**
   I. What has been done? What hasn't?

PART I
Review of the GT2

several slides are by courtesy of the Globus Project

What is the Globus Toolkit?

- A Toolkit which makes it easier to develop computational Grids
- Developed by the Globus Project Developer Team (ANL, USC/ISI)
- Defacto standard as a low level Grid middleware
  - Most Grid testbeds are using the Globus Toolkit
- Latest version is 2.4
- Alpha version of the GT3 is available
  - Based on OGSA. Different architecture with GT2

Some notes on the Globus Toolkit (1/2)

- **Globus Toolkit is not providing a framework for anonymous computing and mega-computing**
  - Users are required
    - to have an account on servers to which the user would be mapped when accessing the servers
    - to have a user certificate issued by a trusted CA
    - to be allowed by the administrator of the server
  - Complete differences with mega-computing framework such as SETI@HOME

Some notes on the Globus Toolkit (2/2)

- **Do not think that the Globus Toolkit solves all problems on the Grid.**
  - The Globus Toolkit is a set of tools for the easy development of computational Grids and middleware
    - The Globus Toolkit includes low-level APIs and several UNIX commands
    - It is not easy to develop application programs using Globus APIs. High-level middleware helps application development.
  - Several necessary functions on the computational Grids are not supported by the Globus Toolkit.
    - Brokering, Co-scheduling, Fault Managements, etc.
  - Other supposed problems
    - using IP-unreachable resources (private IP addresses + MPICH-G2)
    - scalability (ldap, maintenance of grid-mapfiles, etc.)

GT2 components

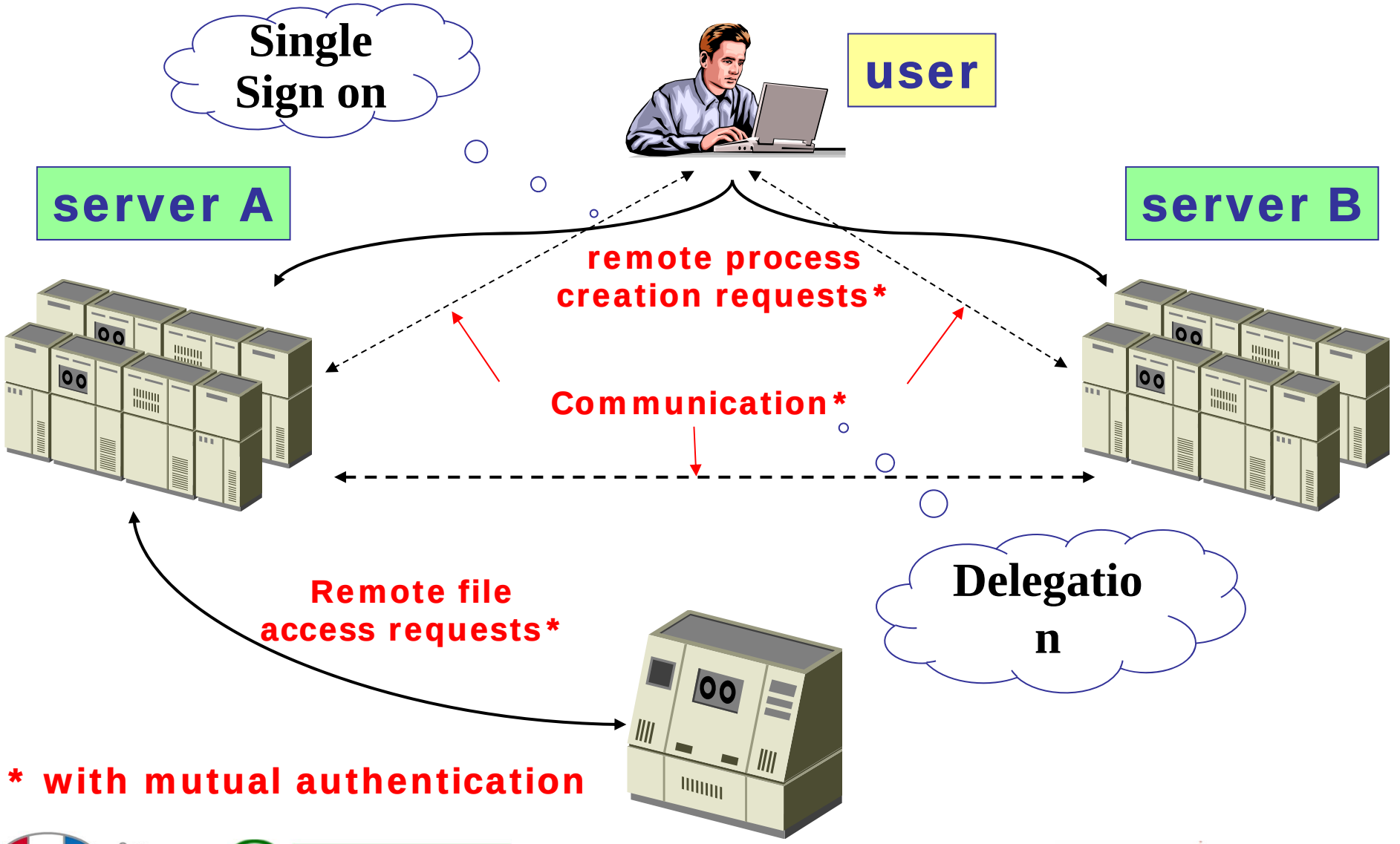
- **GSI: Single Sign On + delegation**
- **MDS: Information Retrieval**
  - Hierarchical Information Tree (GRIS+GIIS)
- **GRAM: Remote process invocation**
  - Three components:
    - Gatekeeper
    - Job Manager
    - Queuing System (pbs, sge, etc.)
- **Data Management:**
  - GridFTP
  - Replica management
  - GASS

Security
GSI

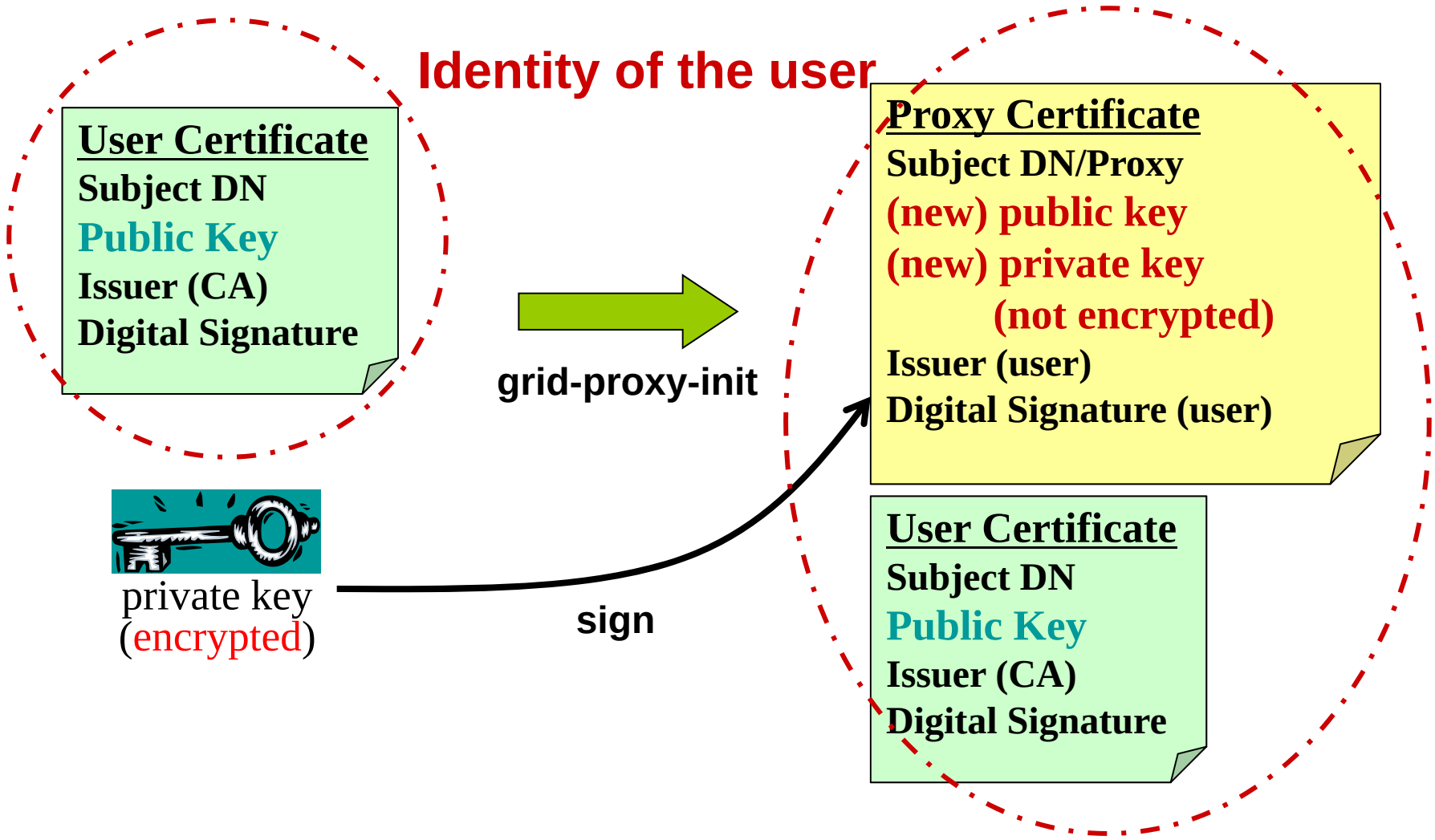
GSI: Grid Security Infrastructure

- **Authentication and authorization using standard protocols and their extensions.**
  - Authentication: Identify the entity
  - Authorization: Establishing rights
- **Standards**
  - PKI, X.509, SSL,...
- **Extensions: Single sign on and delegation**
  - Entering pass phrase is required only once
  - Implemented by proxy certificates

Requirements for security



Proxy Certificate



Requirements for users

- **Obtain a certificate issued by a trusted CA**
  - Globus CA can be used for tests
  - Run another CA for production run. The certificate and the signing policy file of the CA should be put on an appropriate directory (/etc/grid-security/certificates).
- **Run grid-proxy-init command in advance**
  - Will generate a proxy certificate. Enter PEM pass phrase for the decryption of a private key.
  - A proxy certificate will be generated /tmp directory

Requirements for system admins.

- **CA certificate and the signing policy file are used for verifying end entity's certificate.**
  - Those files must be placed in /etc/grid-security/certificates/ directory
  - example:
    - If the server certificate is issued by AIST GTRC CA, the certificate and the signing policy file of AIST GTRC CA must be put in /etc/grid-security/certificates/ on client machine.
    - If my certificate is issued by KISTI CA, the certificate and the signing policy file of KIST CA must be put in /etc/grid-security/certificates/ on all server machines.

PART II
How to build a Grid

many slides are by courtesy of Bill Johnston (NASA)

Building a Multi-site, Computational and Data Grid

- **Like networking, successful Grids involve almost as much sociology as technology.**
- **The first step is to establish the mechanisms for promoting cooperation and mutual technical support among those who will build and manage the Grid.**
- **Establish an Engineering Working Group that involves the Grid deployment teams at each site**
  - schedule regular meetings / telecons
  - involve Globus experts in these meetings
  - establish an EngWG archived email list

Grid Resources

- **Identify the computing and storage resources to be incorporated into your Grid**
  - be sensitive to the fact that opening up systems to Grid users may turn lightly or moderately loaded systems into heavily loaded systems
  - batch schedulers may have to be installed on systems that previously did not use them in order to manage the increased load
  - carefully consider the issue of co-scheduling!
    - many potential Grid applications need this
    - only a few available schedulers provide it (e.g. PBSPro)
    - this is an important issue for building distributed systems

Build the Initial Testbed

- **Plan for a Grid Information Service / Grid Information Index Server (GIS/GIIS) at each distinct site with significant resources**
  - this is important in order to avoid single points of failure
    - if you depend on an MDS/GIIS at some other site site, and it becomes un-available, you will not be able to examine your local resources
- **The initial testbed GIS/MDS model can be independent GIISs at each site**
  - in this model
    - Either cross-site searches require explicit knowledge of each of the GIISs, which have to searched independently, or
    - All resources cross-register in each GIIS

Build the Initial Testbed

- **Build Globus on test systems**
  - use PKI authentication and certificates from the Globus Certificate Authority, or some other CA, issued certificates for this test environment
    - Globus CA will expire on January 23, 2004.
    - can use the OpenSSL CA to issue your own certs manually
  - validate the access to, and operation of the GIS/GIISs at all sites

Preparing for the Transition to a Prototype-Production Grid

- **There are a number of significant issues that have to be addressed before going to even a pseudo production Grid**
  - Policy and mechanism must be established for the Grid X.509 identity certificates
  - the operational model for the Grid Information Service must be determined
    - who maintains the underlying data?
  - the model and mechanisms for user authorization must be established
    - how are the Grid mapfiles managed?
  - your Grid resource service model must be established (more later)
  - your Grid user support service model must be established
  - Documentation must be published

Trust Management

- **Trust results from clear, transparent, and negotiated policies associated with identity**
- **The nature of the policy associated with identity certificates depends a great deal on the nature of your Grid community**
  - It is relatively easy to establish policy for homogeneous communities as in a single organization
  - It is very hard to establish trust for large, heterogeneous virtual organizations involving people from multiple, international institutions

Trust Management (cont'd)

- **Assuming a PKI Based Grid Security Infrastructure (GSI)**
- **Set up, or identify, a Certification Authority to issue Grid X.509 identity certificates to users and hosts**
- **Make sure that you understand the issues associated the Certificate Policy / Certificate Practices ("CP") of the CA**
  - one thing governed by CP is the "nature" of identity verification needed to issue a certificate (this is a primary factor in determining who will be willing to accept your certificates as adequate authentication for resource access)
  - changing this aspect of the CP could well mean not just re-issuing all certificates, but requiring all users to re-apply for certificates

Trust Management (cont'd)

- **Do not try and invent your own CP**
- **The GGF is working on a standard set of CPs**
- **We are trying to establish international collaborations for Policy Management Authority at the GGF.**
  - DOE Science Grid, NASA IPG, EU Data Grid, ApGrid, etc...
  - First BOF will be held at GGF9 (Chicago, Oct.)
- **Establish and publish your Grid CP**

PKI Based Grid Security Infrastructure (GSI)

- **Pay very careful attention to the subject namespace**
  - the X.509 Distinguished Name (the full form of the certificate subject name) is based on an X.500 style hierarchical namespace
  - if you put institutional names in certificates, don't use colloquial names for institutions - consider their full organizational hierarchy in defining the naming hierarchy
  - find out if anyone else in your institution, agency, university, etc., is working on PKI (most likely in the administrative or business units) - make sure that your names do not conflict with theirs, and if possible follow the same name hierarchy conventions
  - CAs set up by the business units of your organization frequently do not have the right policies to accommodate Grid users

PKI Based Grid Security Infrastructure (GSI)

- **Think carefully about the space of entities for which you will have to issue certificates**
  - Humans
  - Hosts (systems)
  - Services (e.g. GridFTP)
  - Security domain gateways (e.g. PKI to Kerberos)
- **Each must have a clear policy and procedure described in your *CA's CP/CPS***

Preparing for the Transition to a Prototype-Production Grid

- Issue host certificates for all the resources and establish procedures for installing them
- Count on revoking and re-issuing all of the certificates at least once before going operational
- Using certificates issued by your CA, validate correct operation of the GSI/GSS libraries, GSI ssh, and GSIftp / Gridftp at all sites

The Model for the Grid Information System

- **Index servers**
  - resources are typically named using the components of their DNS name
    - advantage is that of using an established and managed name space
  - must use separate "index" servers to define different relationships among GIISs, virtual organization, data collections, etc.
    - on the other hand, you can establish "arbitrary" relationships within the collection of indexed objects
  - this is the approach favored by the Globus R&D team

Local Authorization

- **Establish the conventions for the Globus mapfile**
  - maps user Grid identities to system UIDs – this is the basic local authorization mechanism for each individual platform, e.g. compute and storage
  - establish the connection between user accounts on individual platforms and requests for Globus access on those systems
  - if your Grid users are to be automatically given accounts on a lot of different systems, it may make sense to centrally manage the mapfile and periodically distribute it to all systems
    - however, unless the systems are administratively homogeneous, a non-intrusive mechanism such as email to the responsible sys admins to modify the mapfile is best
  - Community Authorization Service (CAS)

Site Security Issues

- **Establish agreements on firewall issues**
  - Globus can be configured to use a restricted range of ports, but it still needs several tens, or so (depending on the level of usage of the resources behind the firewall), in the mid 700s
  - A Globus "port catalogue" is available to tell what each Globus port is used for
    - this lets you provide information that you site security folks will likely want
    - should let you estimate how many ports have to be opened (how many per process, per resource, etc.)
  - GIS/MDS also needs some ports open
  - CA typically uses a secure Web interface (port 443)
- **Develop tools/procedures to periodically check that the ports remain open**

Preparing for Users

- **Build and test your Grid incrementally**
  - very early on, identify a test case distributed application that requires reasonable bandwidth, and run it across as many widely separated systems in your Grid as possible
    - try and find problems before your users do
  - design test and validation suites that exercise your Grid in the same way that applications do
- **Establish user help mechanisms**
  - Grid user email list and / or trouble ticket system
  - Web pages with pointers to documentation
  - a Globus "Quick Start Guide" that is modified to be specific to your Grid, with examples that will work in your environment (starting with a Grid "hello world" example)

The End of the Testbed Phase

- At this point Globus, the GIS/MDS, and the security infrastructure should all be operational on the testbed system(s). The Globus deployment team should be familiar with the install and operation issues, and the sys admins of the target resources should be engaged.
- Next step is to build a prototype-production environment.

Moving from Testbed to Prototype Production Grid

- Deploy and build Globus on at least two production computing platforms at two different sites.Establish the relationship between Globus job submission and the local batch schedulers (one queue, several queues, a Globus queue, etc.)
- Validate operation of this configuration

Take Good Care of the Users as Early as Possible

- ***Establish a Grid/Globus application specialist group***
  - they should be running sample jobs as soon as the testbed is stable, and certainly as soon as the prototype-production system is operational
  - they should serve as the interface between users and the Globus system administrators to solve Globus related application problems
- ***Identify early users and have the Grid/Globus application specialists assist them in getting jobs running on the Grid***
  - On of the scaling / impediment-to-use issues currently is that the Grid services are relatively primitive (I.e., at a low level). The Grid Services and Web Grid Services work currently in progress is trying to address this.

Case Study

ApGrid Testbed

Outline

- Brief introduction of ApGrid and the ApGrid Testbed
- Software architecture of the ApGrid Testbed
- Lessons learned

What is ApGrid?

- Asia-Pacific Partnership for Grid Computing.
- **ApGrid focuses on**
  - Sharing resources, knowledge, technologies
  - Developing Grid technologies
  - Helping the use of our technologies in create new applications
  - Collaboration on each others work
- **Not only a Testbed**
- **Not restricted to just a few developed countries, neither to a specific network nor its related group of researchers**
- **Not a single source funded project**

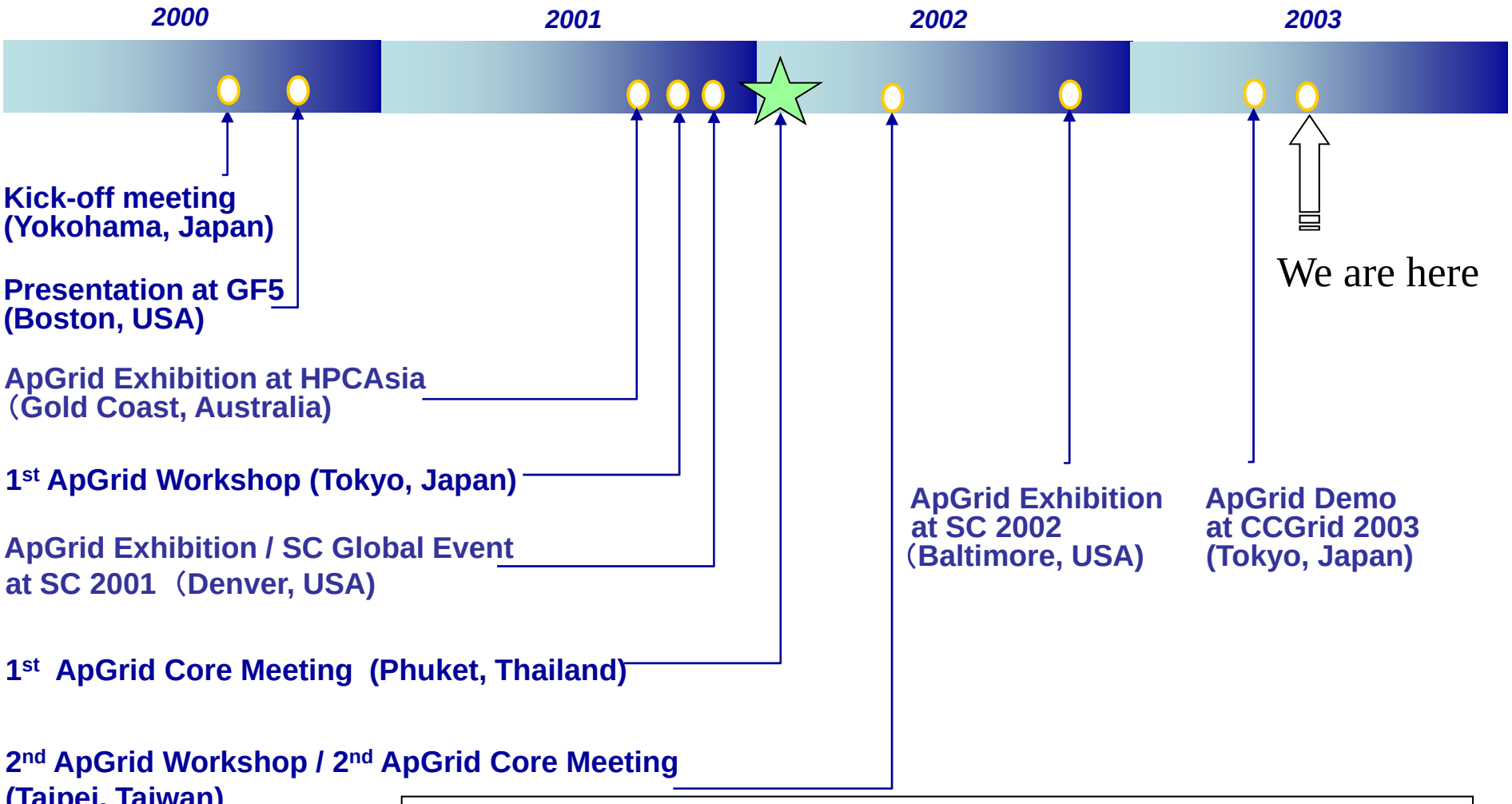
Grid Technology Research Center AIST

APGrid Asia-Pacific Grid

Nimf

AIST

History of ApGrid



15 countries, 41 organizations (as of May, 2003)

Grid Technology Research Center AIST

APGrid Asia-Pacific Grid

Nimf

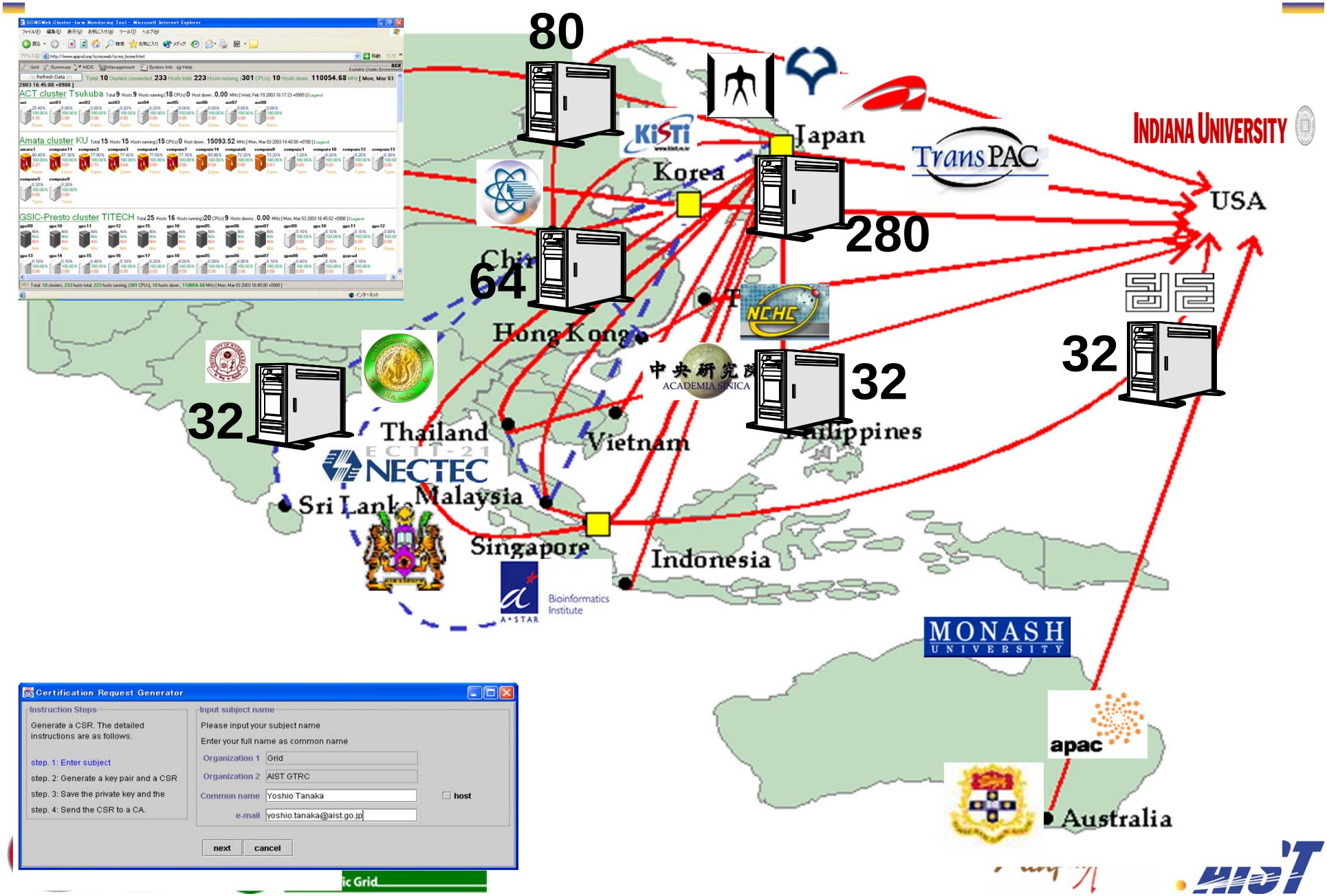
AIST

ApGrid Testbed – features -

- **Truly multi national/political/institutional VO**
  - not an application-driven testbed
  - differences in languages, culture, policy, interests, ...
- **Donation (Contribution) based**
  - Not a single source funded for the development
  - Each institution contributes his own share
  - bottom-up approach
- **We can**
  - have experiences on running international VO
  - verify the feasibility of this approach for the testbed development

ApGrid Testbed: Status

http://www.apgrid.org/



ApGrid Testbed – status and plans -

- **Resources**
  - 500 CPUs from more than 10 institution
    - Most resources are not dedicated to the ApGrid Testbed.
  - many AG nodes, 1 virtual venue server
  - Special devices (MEG, Ultra High Voltage Microscope, etc.)
- **Going to be a production Grid**
  - Most current participants are developers of Grid middleware rather than application people
  - Should be used for running REAL applications
    - increase CPUs
    - keep it stable
    - provide documents

Design Policy

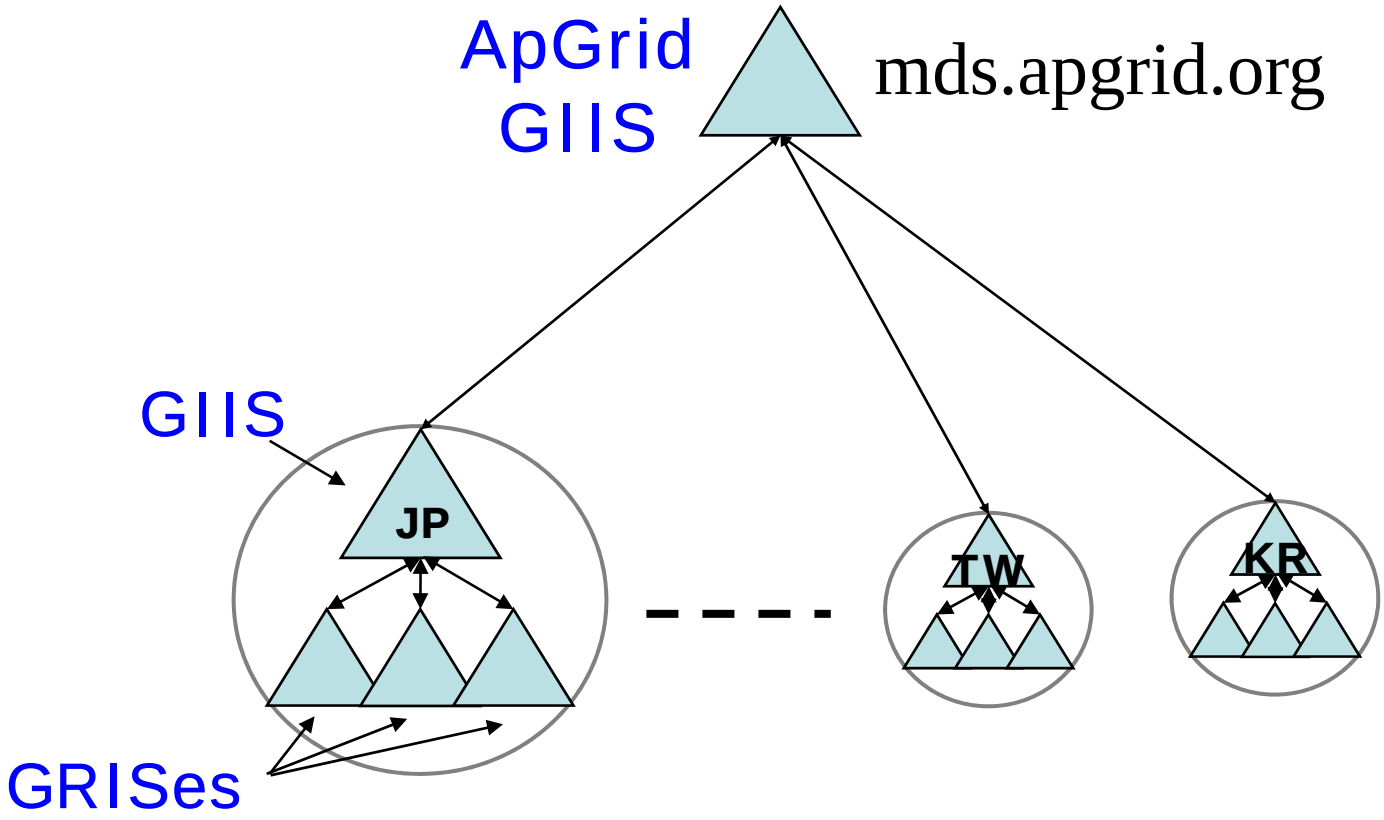
- **Security is based on GSI**
- **Information service is provided by MDS**
  - Use Globus Toolkit Ver.2 as a common software infrastructure

Testbed Developments – Security Infrastructure -

- **Certificates and CAs**
  - Users and resources have to have their certificates issued by a trusted CA.
  - The ApGrid Testbed runs CAs and issues certificates for users and resources.
- **ApGrid CA?**
  - The ApGrid Testbed allows multiple root CAs.
  - Each country/organization/project may run its own CA and these could be root CAs on the ApGrid Testbed.
  - Certificates, signing policy files of the ApGrid CAs are put on the ApGrid home page and can be downloaded via https access.
  - Planning to establish ApGrid PMA and collaborate with other communities.

Testbed Developments – Information Services -

- **Based on MDS (GRIS/GIIS)**



Requirements for users

- **obtain a user certificate**
- **be permitted accesses to resources by the resource providers**
  - need to have an account and an entry to grid-mapfile on each server
- **Put certificates of all CAs by which server certificates are issued.**

Requirements for resource providers

- **Install GT2 on every server**
- **Decide your policy**
  - which CA will be trusted?
  - to whom is your resource opened?
  - make limitations such as max job running time, etc.?
  - ...
- **Give appropriate accounts and add entries to grid-mapfile for the users**
  - Possible policies:
    - Give accounts for all individuals
    - Give a common account for each institution
- **Accept job requests via the Globus Gatekeeper**
- **Provide information via GRIS/GIIS**
- **Push the sites' GIIS to the ApGrid GIIS**

How to contribute to the ApGrid Testbed

1. **Install ApGrid Recommended Software**
    1. Configure GRIS/GIIS
    2. Put trusted CA's cert. and policy files
2. **Provide Users' Guide for ApGrid users**
    1. Resource information
    2. How to get an account
    3. Contact information
    4. etc.
3. **Administrative work**
    1. Create accounts
    2. Add entries to grid-mapfile
    3. etc.

ApGrid Testbed – Software Infrastructure -

- **Minimum Software: Globus Toolkit 2.2 (or later)**
  - Security is based on GSI
  - Information Service is based on MDS

- **The ApGrid Recommended Package will include**
  - GPT 2.2.5
  - Globus Toolkit 2.4.2
  - MPICH-G2 (MPICH 1.2.5.1)
  - Ninf-G 1.1.1
  - Iperf 1.6.5
  - SCMSWeb 2.1
  - + installation tool

Configuration of GIIS
- Define name of the VO -

Add the following contents to $GLOBUS_LOCATION/etc/grid-info-slapd.conf

```
database        giis
suffix          "Mds-Vo-name=AIST, o=Grid"
conf            /usr/local/gt2/etc/grid-info-site-giis.conf
policyfile      /usr/local/gt2/etc/grid-info-site-policy.conf
anonymousbind     yes
access to * by * write
```

Need to change $GLOBUS_LOCATION/etc/grid-info-site-policy.conf so that the GIIS can accept registration from GRISes

Configuration of GRIS
- Example: Register to the ApGrid MDS -

Add the following contents to $GLOBUS_LOCATION/etc/grid-info-resource-register.conf

```
dn: Mds-Vo-Op-name=register,   Mds-Vo-name=ApGrid, o=Grid
regtype: mdsreg2
reghn: mds.apgrid.org
regport: 2135
regperiod: 600
type: ldap
hn: koume.hpcc.jp
port: 2135
rootdn: Mds-Vo-name=AIST, o=Grid
…
```

Lessons Learned

- **Difficulties caused by the bottom-up approach and the problems on the installation of the Globus Toolkit.**
  - Most resources are not dedicated to the ApGrid Testbed.
  - Site's policy should be respected.
  - There were some requirements on modifying software configuration, environments, etc.
    - Version up of the Globus Toolkit (GT1.1.4 -> GT2.0 -> GT2.2)
    - Apply patches, install additional packages
    - Build bundles using other flavors
  - Different requirements for the Globus Toolkit between users.
    - Middleware developers needs the newest one.
    - Application developers satisfy with using the stable (older) one.
    - It is not easy to catch up frequent version up of the Globus Toolkit.
  - ApGrid software package should solve some of these problems

Lessons Learned (cont'd)

- **Scalability problems in LDAP**
  - sizelimit should be specified in grid-info-slapd.conf (default is 500)
  - GIIS lookup takes several ten seconds
- **Well known problem ☺**
  - Firewall, private IP addresses…
- **Human interaction is very important**
  - have timely meetings/workshops as well as regular VTCs.
  - understand and respect each other's culture, interests, policy, etc.

For more info

Home Page	http://www.apgrid.org/
Mailing Lists	
Core Member ML	core@apgrid.org
Tech. Contacts ML (approved members)	tech-contacts@apgrid.org
ML for discussion (open for anyone)	discuss@apgrid.org

Grid Technology Research Center AIST

Asia-Pacific Grid

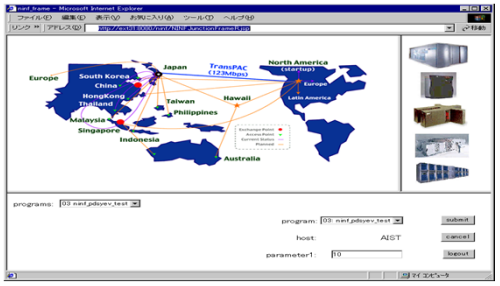
AIST

PART III
How to program on a Grid

many slides are by courtesy of Bill Johnston (NASA)

Layered Programming Model/Method

Portal / PSE
GridPort, HotPage,
GPDK, Grid PSE Builder,
etc...



Easy but inflexible

High-level Grid Middleware
MPI (MPICH-G2, PACX-MPI, ...)
GridRPC (Ninf-G, NetSolve, ...)

Ninf

MPI

Low-level Grid Middleware
Globus Toolkit

the globus project™
www.globus.org

Primitives
Socket, system calls, ...

Difficult but flexible

Grid Technology Research Center AIST

ApGrid Asia-Pacific Grid

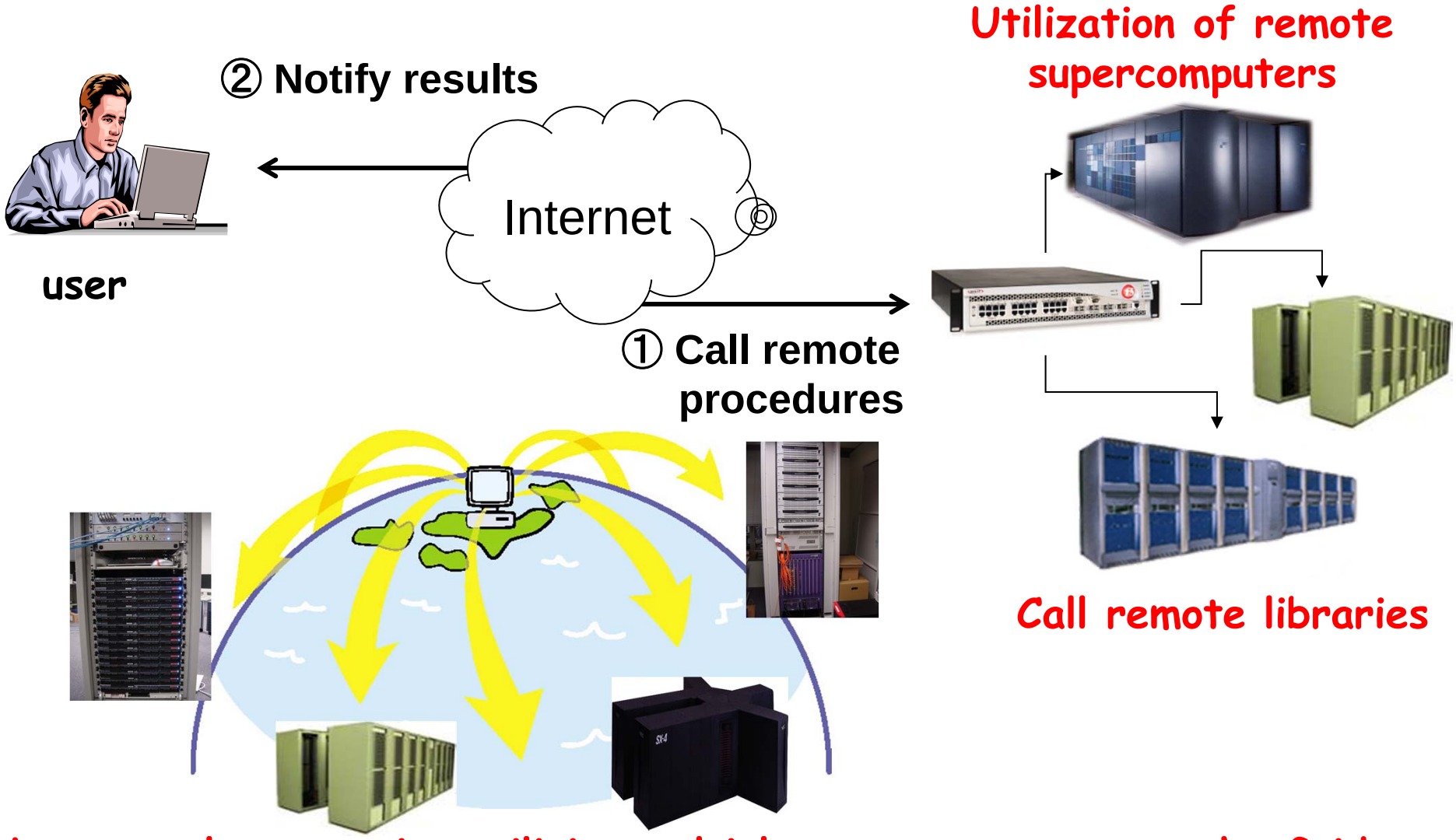
Ninf

AIST

Some Significant Grid Programming Models/Systems

- **Data Parallel**
  - MPI - MPICH-G2, Stampi, PACX-MPI, MagPie
- **Task Parallel**
  - GridRPC – Ninf, Netsolve, Punch…
- **Distributed Objects**
  - CORBA, Java/RMI, …
- **Data Intensive Processing**
  - DataCutter, Gfarm, …
- **Peer-To-Peer**
  - Various Research and Commercial Systems
    - UD, Entropia, Parabon, JXTA, …
- **Others…**

GridRPC: RPC based Programming model



GridRPC: RPC "tailored" for the Grid

- Medium to Coarse-grained calls
  - Call Duration < 1 sec to > week
- Task-Parallel Programming on the Grid
  - Asynchronous calls, 1000s of scalable parallel calls
- Large Matrix Data & File Transfer
  - Call-by-reference, shared-memory matrix arguments
- Grid-level Security (e.g., Ninf-G with GSI)
- Simple Client-side Programming & Management
  - No client-side stub programming or IDL management
- Other features…

GridRPC (cont'd)

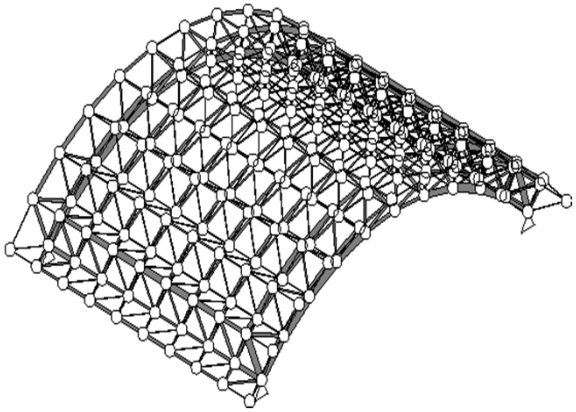
- **v.s. MPI**
  - Client-server programming is suitable for task-parallel applications.
  - Does not need co-allocation
  - Can use private IP address resources if NAT is available (at least when using Ninf-G)
  - Better fault tolerancy
- **Activities at the GGF GridPRC WG**
  - Define standard GridRPC API; later deal with protocol
  - Standardize only minimal set of features; higher-level features can be built on top
  - Provide several reference implementations
    - Ninf-G, NetSolve, …

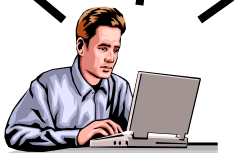
Typical Scenario: Optimization Problems and Parameter Study on Cluster of Clusters

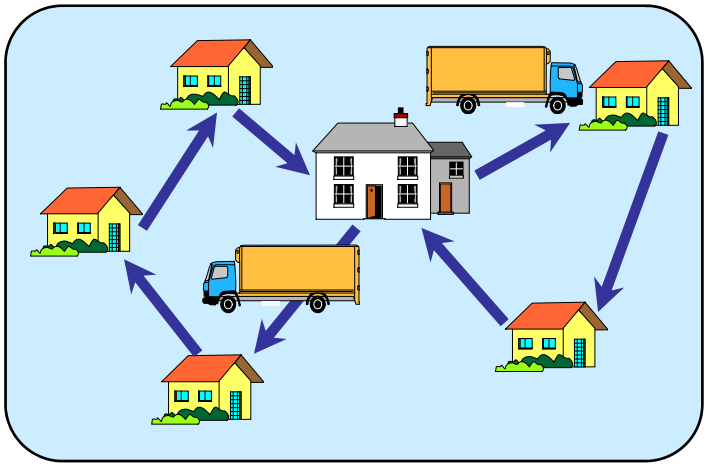
rpc

rpc

rpc







Structural Optimization

Vehicle Routing Problem

Slide by courtesy of Prof. Fujisawa

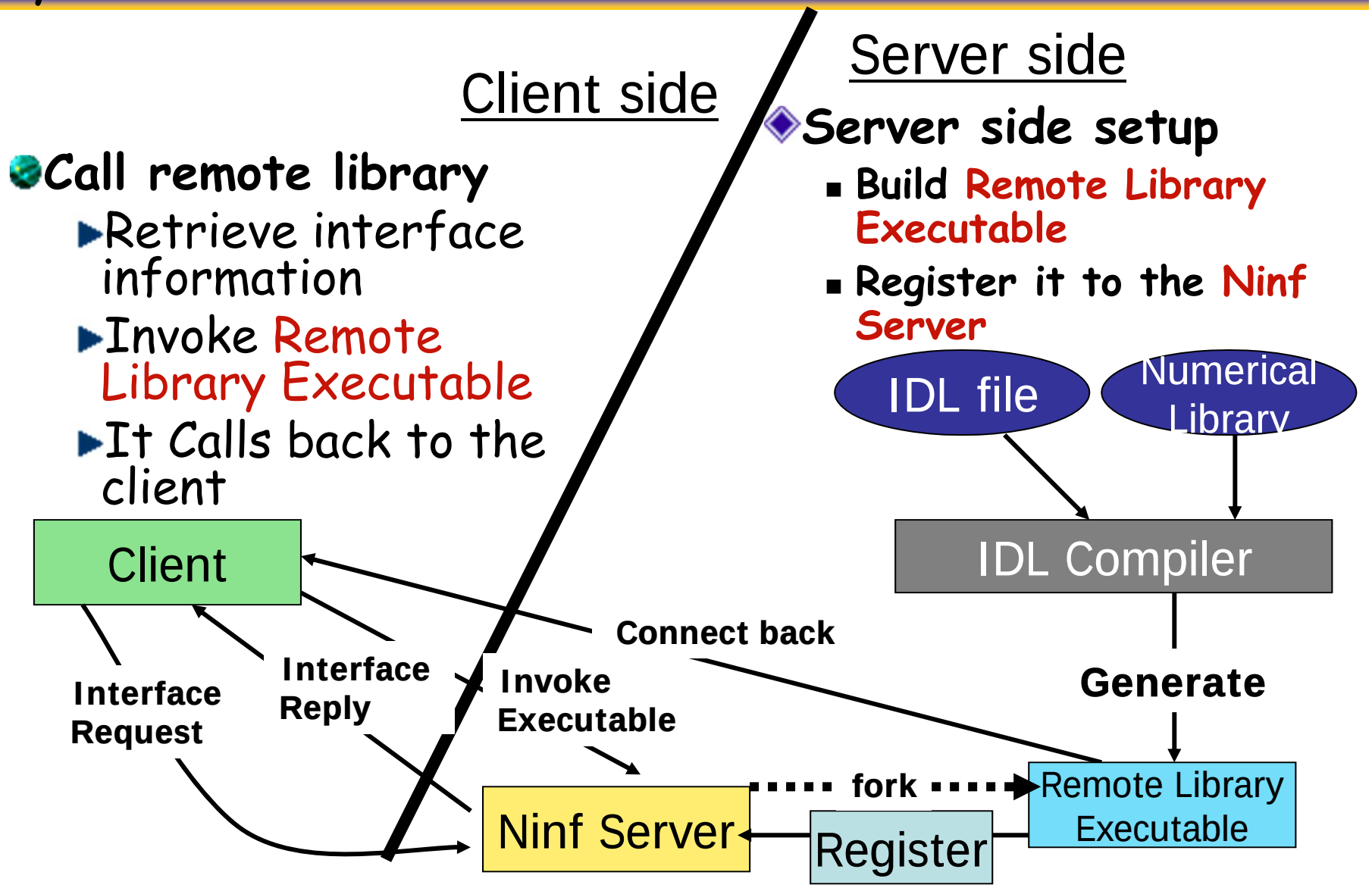
Grid Technology Research Center AIST

Asia-Pacific Grid

Niaf

AIST

Sample Architecture and Protocol of GridRPC System – Ninf -



GridRPC: based on Client/Server model

- **Server-side setup**
  - Remote libraries must be installed in advance
    - Write IDL files to describe interface to the library
    - Build remote libraries
  - Syntax of IDL depends on GridRPC systems
    - e.g. Ninf-G and NetSolve have different IDL
- **Client-side setup**
  - Write a client program using GridRPC API
  - Write a client configuration file
  - Run the program

GridRPC API

API for client programming

Grid Technology Research Center AIST
ApGrid Asia-Pacific Grid
Ninf
AIST

The GridRPC API

- **Provide standardized, portable, and simple programming interface for Remote Procedure Call**
- **Attempt to unify client access to existing grid computing systems (such as NetSolve and Ninf-G)**
- **Working towards standardization through the GGF GridRPC WG**
  - Initially standardize API; later deal with protocol
  - Standardize only minimal set of features; higher-level features can be built on top
  - Provide several reference implementations
    - Not attempting to dictate any implementation details

Rough steps for RPC

- **Initialize**

```
grpc_initialize(config_file);
```

- **Create a function handle**
  - Abstraction to a remote library

```
grpc_function_handle_t  handle;

grpc_function_handle_init(
  &handle, host, port, "lib_name");
```

- **RPC**
  - Call remote procedure

```
grpc_call(&handle, args…);
          とか
grpc_call_async(&handle, args…);
```

The GridRPC API - Fundamentals

- **Function handle – *grpc_function_handle_t***
  - Represents a mapping from a function name to an instance of that function on a particular server
  - Once created, calls using a function handle always go to that server
- **Session ID – *grpc_sessionid_t***
  - Identifier representing a previously issued non-blocking call
  - Allows checking status, canceling, waiting for, or getting the error code of a non-blocking call
- **Error and Status code – *grpc_error_t***
  - Represents all error and return status codes from GridRPC functions

Initializing and Finalizing

- **grpc_error_t grpc_initialize(char *config_file_name)**
  - Reads *config_file_name* and initializes the system
  - Initialization is system dependent
  - Must be called before any other GridRPC calls
  - Return value:
    - GRPC_OK if successful
    - GRPC_ERROR if not successful
- **grpc_error_t grpc_finalize()**
  - Releases any resources being used by GridRPC
  - Return value:
    - GRPC_OK if successful
    - GRPC_ERROR if not successful

Function Handle Management

- **grpc_error_t grpc_function_handle_default(**
  grpc_function_handle_t *handle,
  char *func_name)
  - Creates a function handle for function *func_name* using the default server
  - Server selection is implementation-dependent
- **grpc_error_t grpc_function_handle_init(**
  grpc_function_handle_t *handle,
  char *host_port_str,
  char *func_name)
  - Allows explicitly specifying the server in *host_name* and *port*

Function Handle Management (cont.)

- **grpc_error_t grpc_function_handle_destruct(**
  grpc_function_handle_t *handle)
  - Release the memory allocated for *handle*
- **grpc_error_t grpc_function_handle_t**
  ***grpc_get_handle(**
  grpc_function_handle_t * handle,
  grpc_sessionid_t sessionId)
  - Returns the function handle which corresponds to *sessionId*

GridRPC Call Functions

- **grpc_error_t grpc_call(**
  grpc_function_handle_t *handle, ...)
  - Blocking remote procedure call
- **grpc_error_t grpc_call_async(**
  grpc_function_handle_t *handle,
  grpc_sessionid_t *sessionID,
  ...)
  - Non-blocking remote procedure call
  - session ID (positive integer) is stored in *sessionID*
  - session ID can be checked for completion later

GridRPC Call Functions Using ArgStack

- **grpc_error_t grpc_call_argstack(**
  grpc_function_handle_t *handle,
  grpc_arg_stack *stack)
  - Blocking call using argument stack
  - Returns GRPC_OK on success, GRPC_ERROR on failure
- **grpc_error_t grpc_call_argstack_async(**
  grpc_function_handle_t *handle,
  grpc_sessionid_t *sessionID,
  grpc_arg_stack *stack)
  - Non-blocking call using argument stack
  - session ID (positive integer) is stored in *sessionID*
  - Session ID can be checked for completion later

Asynchronous Session Control Functions

- **grpc_error_t grpc_probe(**
  grpc_sessionid_t sessionID)
  - Checks whether call specified by *sessionID* has completed
  - Returns "session done" or "session not done"
- **grpc_error_t grpc_probe_or(**
  grpc_sessionid_t *idArray,
  size_t length,
  grpc_sessionid_t *idPtr)
  - Checks the array of sessionIDs for any GridRPC calls that have completed
  - Returns exactly one session ID in idPtr if any calls have completed
- **grpc_error_t grpc_cancel(grpc_sessionid_t sessionID)**
  - Cancels a previous call specified by *sessionID*
- **grpc_error_t grpc_cancel_all()**
  - Cancels all outstanding sessions

Asynchronous Wait Functions

- **grpc_error_t grpc_wait(**
  grpc_sessionid_t sessionID)
  - Wait for the specified non-blocking requests to complete
  - Blocks until the specified non-blocking requests to complete
- **grpc_error_t grpc_wait_and(**
  grpc_sessionid_t *idArray, size_t length)
  - Wait for all of the specified non-blocking requests in a given set (*idArray*) have
  - *length* is the number of elements in *idArray*

Asynchronous Wait Functions (cont.)

- **grpc_error_t grpc_wait_or(**
  grpc_sessionid_t *idArray,
  size_t length,
  grpc_sessionid_t *idPtr)
  - Wait for any of the specified non-blocking requests in a given set (*idArray)* have completed
  - *length* is the number of elements in *idArray*
  - On return, *idPtr* contains the session ID of the call that completed
- **grpc_error_t grpc_wait_all()**
  - Wait for all previously issued non-blocking requests have completed.

Asynchronous Wait Functions (cont.)

- **grpc_error_t grpc_wait_any(**
  grpc_sessionid_t *idPtr)
  - Wait for any previously issued non-blocking request has completed
  - On return, *idPtr* contains the session ID of the call that completed
  - Returns GRPC_OK if the call (returned in *idPtr*) succeeded, otherwise returns GRPC_ERROR
  - Use grpc_get_error() to get the error value for a given session ID

Error Reporting Functions

- **char * grpc_error_string(**
  grpc_error_t error_code)
  - Gets the error string given a numeric error code
  - For *error_code* we typically pass in the global error value *grpc_errno*

Argument Stack Functions

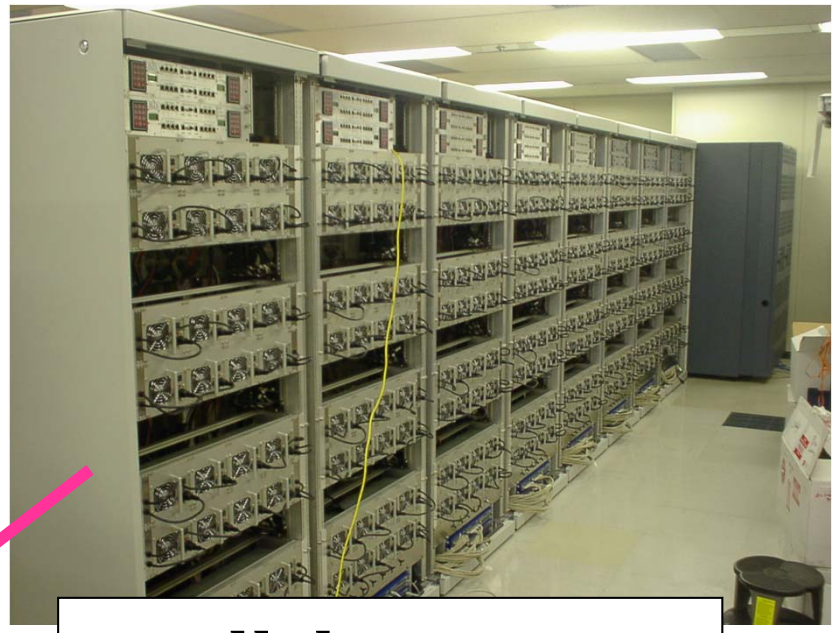
- **grpc_error_t grpc_arg_stack_create(**
  grpc_arg_stack_t *stack,
  size_t maxsize)
  - Creates a new argument stack with at most *maxsize* entries
- **grpc_error_t grpc_arg_stack_destruct(**
  grpc_arg_stack_t *stack)
  - Frees resources associated with the argument stack

Argument Stack Functions (cont.)

- **grpc_error_t grpc_stack_push(**
  grpc_arg_stack_t *stack,
  void *arg)
  - Pushes *arg* onto *stack*
- **grpc_error_t grpc_stack_pop(**
  grpc_arg_stack_t *stack)
  - Returns the top element of *stack* or NULL if the stack is empty
- **Arguments are passed in the order they were pushed onto the stack. For example, for the call F(a,b,c), the order would be:**
  - Push(a); Push(b); Push(c);

Data Parallel Application

- Call parallel libraries (e.g. MPI apps).
- Backend "MPI" or Backend "BLACS" should be specified in the IDL

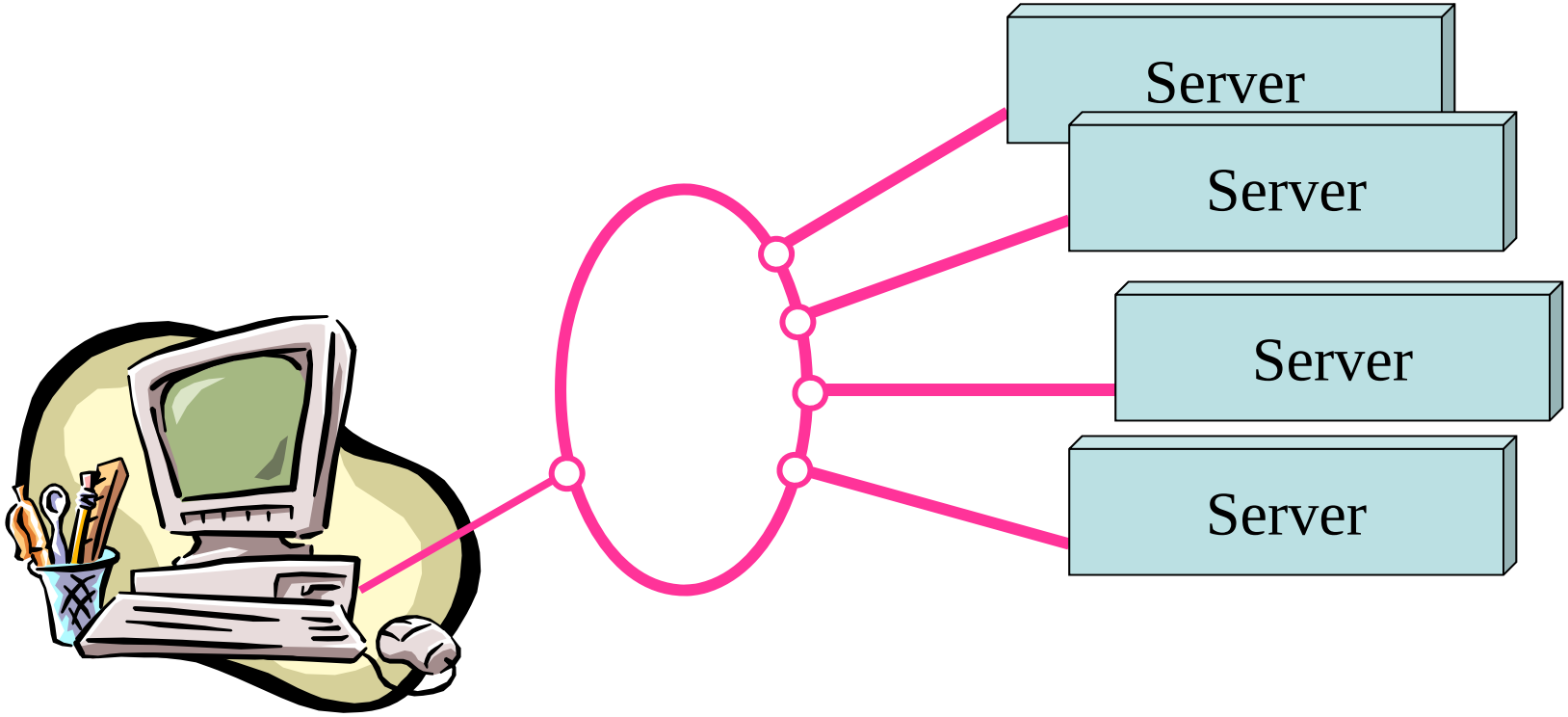


Parallel Computer

Parallel Numerical Libraries
Parallel Applications

Task Parallel Application

- **Parallel RPCs using asynchronous call.**



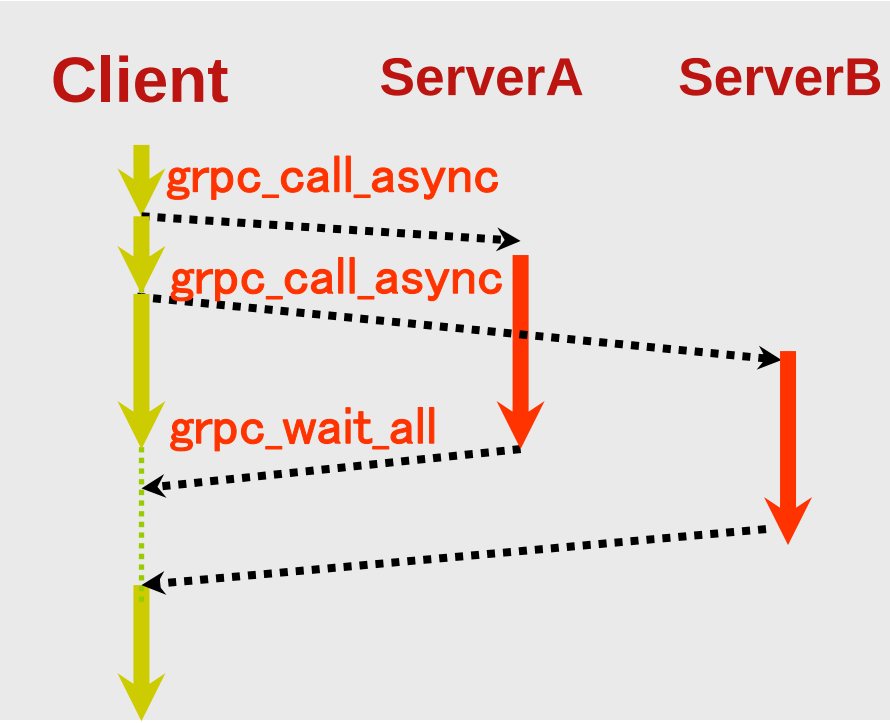
Task Parallel Application

- **Asynchronous Call**

```
grpc_call_async(...);
```

- **Waiting for reply**

```
grpc_wait(sessionID);
grpc_wait_all();
grpc_wait_any(idPtr);
grpc_wait_and(idArray, len);
grpc_wait_or(idArray, len, idPtr);
grpc_cancel(sessionID);
```



Various task parallel programs spanning clusters are easy to write

Ninf-G

Overview and Architecture

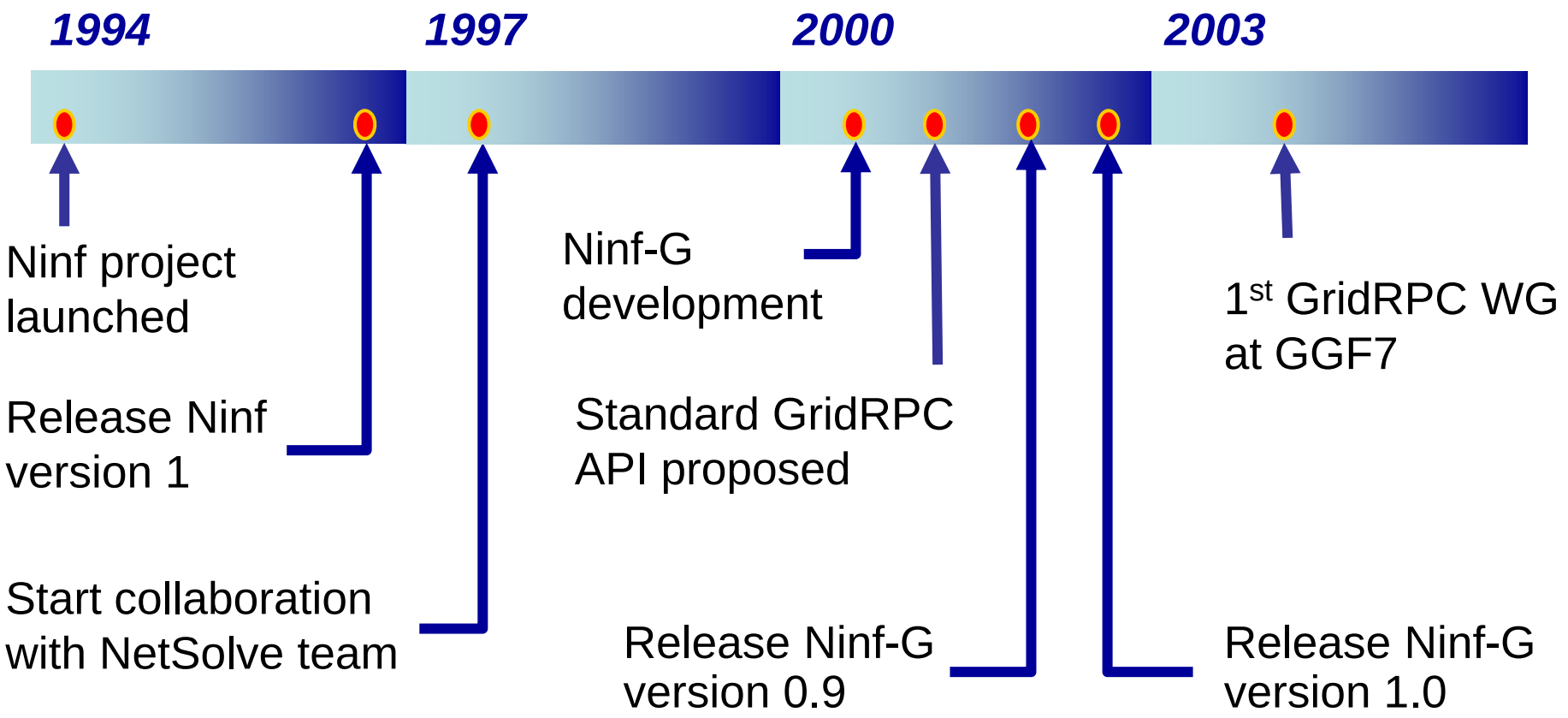
Ninf Project

- **Started in 1994**
- **Collaborators from various organizations**
  - AIST
    - Satoshi Sekiguchi, Umpei Nagashima, Hidemoto Nakada, Hiromitsu Takagi, Osamu Tatebe, Yoshio Tanaka, Kazuyuki Shudo
  - University of Tsukuba
    - Mitsuhisa Sato, Taisuke Boku
  - Tokyo Institute of Technology
    - Satoshi Matsuoka, Kento Aida, Hirotaka Ogawa
  - Tokyo Electronic University
    - Katsuki Fujisawa
  - Ochanomizu University
    - Atsuko Takefusa
  - Kyoto University
    - Masaaki Shimasaki

Grid Technology Research Center AIST

Asia-Pacific Grid

Brief History of Ninf/Ninf-G



What is Ninf-G?

- **A software package which supports programming and execution of Grid applications using GridRPC.**
- **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
  - manual documents

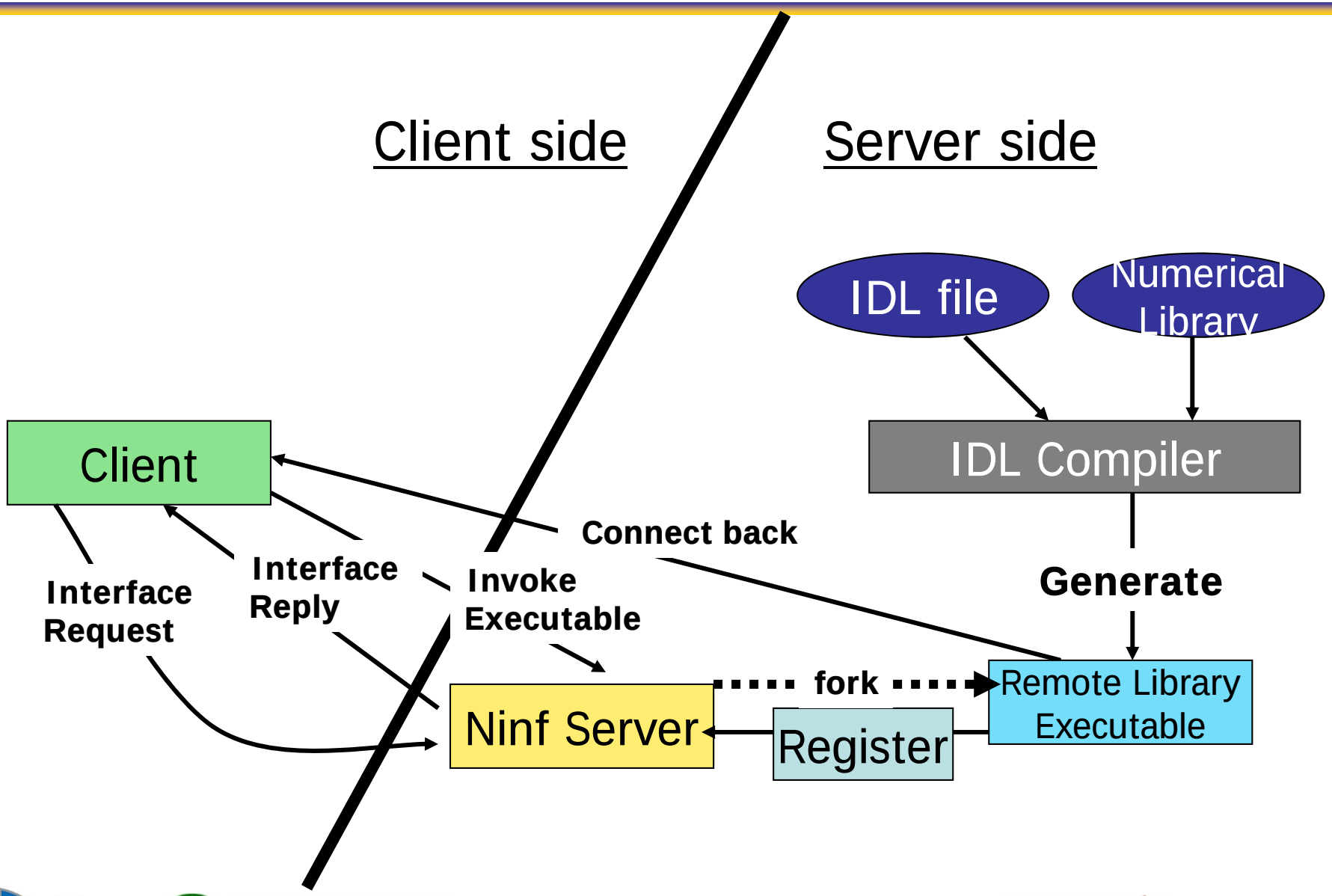
Ninf-G: Features At-a-Glance

- **Ease-of-use, client-server, Numerical-oriented RPC system**
- **No stub information at the client side**
- **User's view: ordinary software library**
  - Asymmetric client vs. server
- **Built on top of the Globus Toolkit**
  - Uses GSI, GRAM, MDS, GASS, and Globus-IO
- **Supports various platforms**
  - Ninf-G is available on Globus-enabled platforms
- **Client APIs: C/C++, Java**

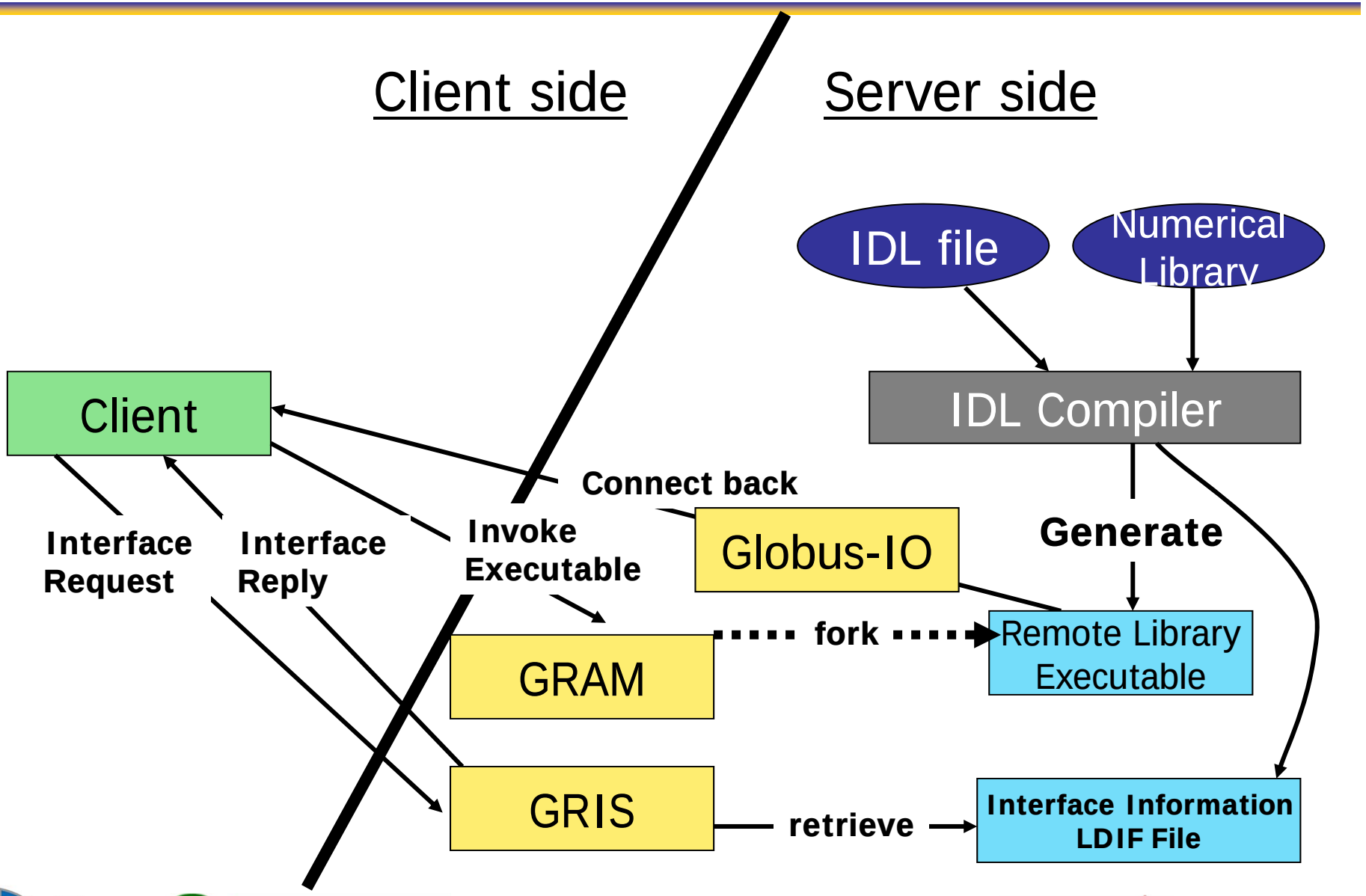
Sample Architecture Review

- **Client API**
  - Provides users easy to use API
- **Remote Library Executable**
  - Execute numerical operation
- **Ninf Server**
  - Provides library interface info.
  - Invokes remote library executable
- **IDL compiler**
  - Compiles Interface description
  - Generates 'stub main' for remote library executable
  - Helps to link the executable
- **Ninf Register driver**
  - Registers remote library executable into the Server

Architecture of Ninf



Architecture of Ninf-G



Grid Technology Research Center AIST
Asia-Pacific Grid
Ninf
AIST

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**
  - contains stub routine and the main routine
  - will be spawned off by GRAM
- **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.**

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**

  % ns_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).

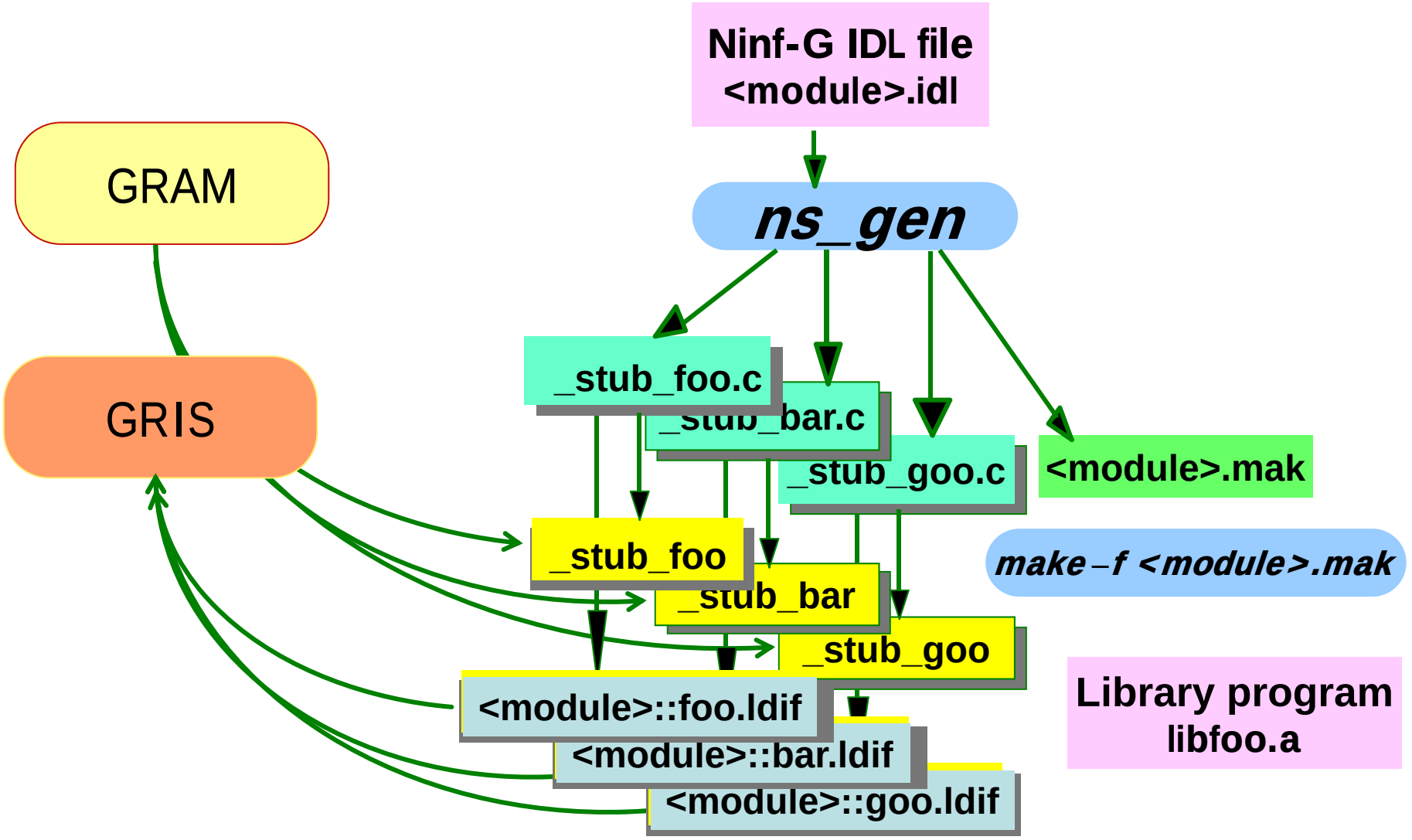
  % make –f <module_name>.mak

- Publish the Ninf-G remote libraries

  % make –f <module_name>.mak install

  This copies the LDIF files to ${GLOBUS_LOCATION}/var/gridrpc

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



Ninf-G IDL Statements (1/2)

- **Module** ***module_name***
  - specifies the module name.
- **CompileOptions** ***"options"***
  - specifies compile options which should be used in the resulting makefile
- **Library** ***"object files and libraries"***
  - specifies object files and libraries
- **FortranFormat** ***"format"***
  - provides translation format from C to Fortran.
  - Following two specifiers can be used:
    - %s: original function name
    - %l: capitalized original function name
  - Example:
    FortranFormat "_%l_";
    Calls "Fortran" fft(n, x, y);
    will generate function call
    FFT(n, x, y);
    in C.

Ninf-G IDL Statements (2/2)

- **Globals** ***{ ... C descriptions }***
  - declares global variables shared by all functions
- **Define** ***routine_name*** **(*parameters*...)**
  [*"description"*]
  [Required *"object files or libraries"*]
  [Backend "MPI"|"BLACS"]
  [Shrink "yes"|"no"]
  {*{C descriptions}* |
  Calls "C"|"Fortran" *calling sequence*}
  - declares function interface, required libraries and the main routine.
  - Syntax of parameter description:
    [mode-spec] [type-spec] formal_parameter
    [[dimension [:range]]+]+

Syntax of parameter description (detailed)

- **mode-spec: one of the following**
  - IN: parameter will be transferred from client to server
  - OUT: parameter will be transferred from server to client
  - INOUT: at the beginning of RPC, parameter will be transferred from client to server. at the end of RPC, parameter will be transferred from server to client
  - WORK: no transfers will be occurred. Specified memory will be allocated at the server side.
- **type-spec should be either *char, short, int, float, long, longlong, double, complex, or filename*.**
- **For arrays, you can specify the size of the array. The size can be specified using scalar IN parameters.**
  - Example:
    IN int n, IN double a[n]

Sample Ninf-G IDL (1/2)

- **Matrix Multiply**

```
Module matrix;

Define dmmul (IN int n,
                    IN double A[n][n],
                    IN double B[n][n],
                    OUT double C[n][n])
“Matrix multiply: C = A x B“
Required “libmmul.o”
Calls “C” dmmul(n, A, B, C);
```

Sample Ninf-G IDL (2/2)

ScaLAPACK (pdgesv)

```
Module SCALAPACK;

CompileOptions “NS_COMPILER = cc”;
CompileOptions “NS_LINKER = f77”;
CompileOptions “CFLAGS = -DAdd_ -O2 –64 –mips4 –r10000”;
CompileOptions “FFLAGS = -O2 -64 –mips4 –r10000”;
Library “scalapack.a pblas.a redist.a tools.a libmpiblacs.a –lblas –lmpi –lm”;

Define pdgesv (IN int n, IN int nrhs, INOUT double global_a[n][lda:n], IN int lda,
                INOUT double global_b[nrhs][ldb:n], IN int ldb, OUT int info[1])
Backend “BLACS”
Shrink “yes”
Required “procmap.o pdgesv_ninf.o ninf_make_grid.of Cnumroc.o descinit.o”
Calls “C” ninf_pdgesv(n, nrhs, global_a, lda, global_b, ldb, info);
```

Ninf-G

How to call Remote Libraries
- client side operations -

(Client) User's Scenario

- Write client programs using GridRPC API
- Compile and link with the supplied Ninf-G client compile driver (*ns_client_gen*)
- Write a configuration file in which runtime environments can be described
- Run *grid-proxy-init* command
- Run the program

Compile and run

- Compile the program using *ns_client_gen* command.

  % ns_client_gen -o myapp app.c

- Before running the application, generate a proxy certificate.

  % grid-proxy-init

- When running the application, client configuration file must be passed as the first argument.

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

Client Configuration File (1/2)

- **Specifies runtime environments.**
- **Available attributes:**
  - host
    - specifies client's hostname (callback contact)
  - port
    - specifies client's port number (callback contact)
  - serverhost
    - specifies default server's hostname
  - ldaphost
    - specifies hostname of GRIS/GIIS
  - ldapport
    - specifies port number of GRIS/GIIS (default: 2135)
  - vo_name
    - specifies Mds-Vo-Name for querying GIIS (default: local)
  - jobmanager
    - specifies jobmanager (default: jobmanager)

Client Configuration File (2/2)

- **Available attributes (cont'd):**
  - loglevel
    - specifies log leve (0-3, 3 is the most detail)
  - redirect_outerr
    - specifies whether stdout/stderr are redirect to the client side (yes or no, default: no)
  - forkgdb, debug_exe
    - enables debugging Ninf-G executables using gdb at server side (TRUE or FALSE, default: FALSE)
  - debug_display
    - specifies DISPLAY on which xterm will be opened.
  - debug_xterm
    - specifies absolute path of xterm command
  - debug_gdb
    - specifies absolute path of gdb command

Sample Configuration File

```
# call remote library on UME cluster
serverhost = ume.hpcc.jp

# grd jobmanager is used to launch jobs
jobmanager = jobmanager-grd

# query to ApGrid GIIS
ldaphost = mds.apgrid.org
ldapport = 2135
vo_name = ApGrid

# get detailed log
loglevel = 3
```

Examples

- **Ninfy the existing library**
  - Matrix multiply
- **Ninfy task-parallel program**
  - Calculate PI using a simple Monte-Carlo Method

Matrix Multiply

- **Server side**
  - Write an IDL file
  - Generate stubs
  - Register stub information to GRIS
- **Client side**
  - Change local function call to remote library call
  - Compile by ns_client_gen
  - write a client configuration file
  - run the application

Matrix Multiply - Sample Code -

```
void  mmul(int n, double * a,
      double * b, double * c) {
  double t;
  int i, j, k;
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      t = 0;
      for (k = 0; k < n; k++){
        t += a[i * n + k] * b[k * n + j]; }
      c[i*N+j] = t;
}}}
```

- The matrix do not itself embody size as type info.

Matrix Multiply- Server Side (1/3) -

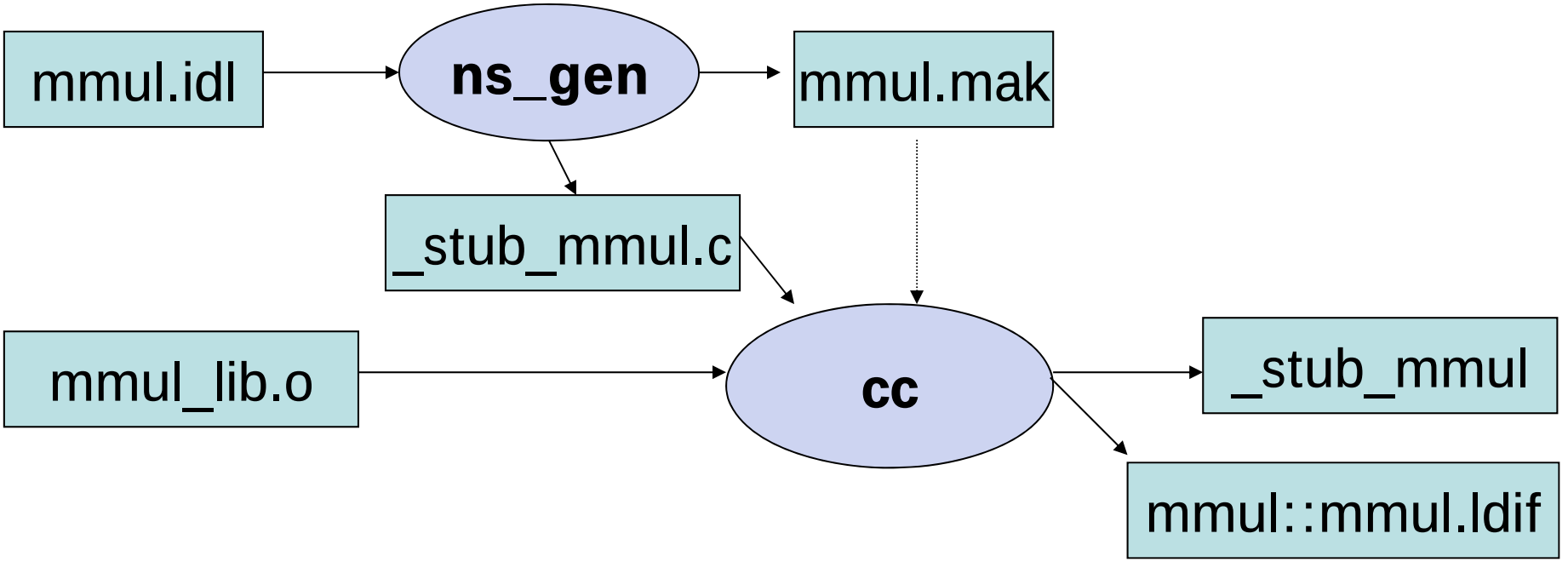
- **Write IDL file describing the interface information (mmul.idl)**

```
Module mmul;
Define mmul(IN int N,
            IN double A[N*N],
            IN double B[N*N],
            OUT double C[N*N])
“Matrix Multiply: C = A x B"
Required "mmul_lib.o"
Calls "C" mmul(N, A, B, C);
```

Matrix Multiply – Server Side (2/3) –

- **Generate stub source and compile it**

```
> ns_gen mmul.idl
> make -f mmul.mak
```



Matrix Multiply - Server Side (3/3) -

Regisgter stub information to GRIS

```
dn: GridRPC-Funcname=mmul/mmul, Mds-Software-deployment=GridRPC-Ninf-G, __ROO
T_DN__
objectClass: GlobusSoftware
objectClass: MdsSoftware
objectClass: GridRPCEntry
Mds-Software-deployment: GridRPC-Ninf-G
GridRPC-Funcname: mmul/mmul
GridRPC-Module: mmul
GridRPC-Entry: mmul
GridRPC-Path: /usr/users/yoshio/work/Ninf-G/test/_stub_mmul
GridRPC-Stub:: PGZ1bmN0aW9uICB2ZXJzaW9uPSIyMjEuMDAwMDAwIiA+PGZ1bmN0aW9
 PSJtbXVsIiBlbnRyeT0ibW11bCIgLz4gPGFyZyBkYXRhX3R5cGU9ImludCIgbW9kZV90eXBl
 PSJpbiIgPgogPC9hcmc+CiA8YXJnIGRhdGFfdHlwZT0iZG91YmxlIiBtb2RlX3R5cGU9Imlu
 IiA+CiA8c3Vic2NyaXB0PjxzaXplPjxleHByZXNzaW9uPjxiaV9hcml0aG1ldGljIG5hbWU9
```

```
> make -f mmul.mak install
```

Grid Technology Research Center AIST

Asia-Pacific Grid

Ninf

AIST

Matrix Multiply - Client Side (1/3) -

- **Modify source code**

```
main(int argc, char ** argv){
  grpc_function_handle_t handle;
  …
  grpc_initialize(argv[1]);
  …
  grpc_function_handle_default(&handle, “mmul/mmul”);
  …
  if (grpc_call(&handle, n, A, B, C) == GRPC_ERROR) {
    …
  }
  …
  grpc_function_handle_destruct(&handle);
  grpc_finalize();
}
```

Matrix Multiply - Client Side (2/3) -

- **Compile the program by *ns_client_gen***

```
> ns_client_gen -o mmul_ninf mmul_ninf.c
```

- **Write a client configuration file**

```
serverhost = ume.hpcc.jp
ldaphost = ume.hpcc.jp
ldapport = 2135
jobmanager = jobmanager-grd
loglevel = 3
redirect_outerr = no
```

Matrix Multiply - Client Side (3/3) -

- **Generate a proxy certificate**

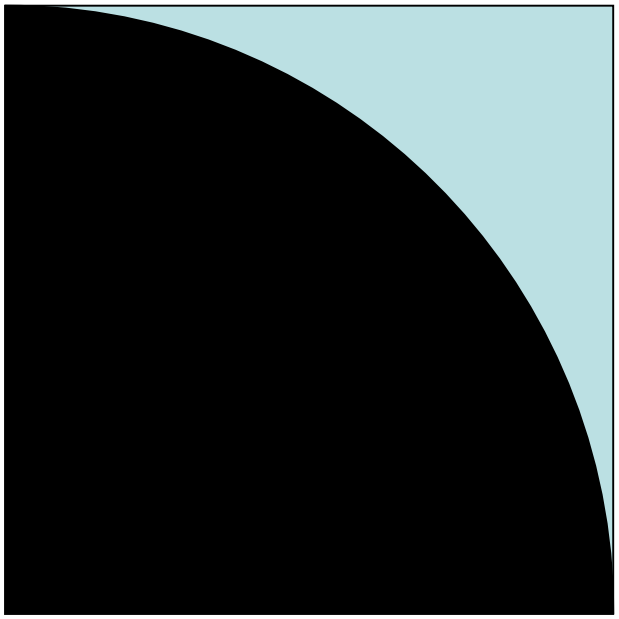
```
> grid-proxy-init
```

- **Run**

```
> ./mmul_ninf config.cl
```

Task Parallel Programs (Compute PI using Monte-Carlo Method)

- **Generate a large number of random points within the square region that exactly encloses a unit circle (1/4 of a circle)**
  - PI = 4 *p*



Compute PI - Server Side -

pi.idl

```
Module pi;

Define pi_trial (
  IN int seed,
  IN long times,
  OUT long * count)
"monte carlo pi computation"
Required "pi_trial.o"
{
  long counter;
  counter = pi_trial(seed, times);
  *count = counter;
}
```

pi_trial.c

```
long pi_trial (int seed, long times) {
  long l, counter = 0;

  srandom(seed);
  for (l = 0; l < times; l++) {
    double x =
      (double)random() / RAND_MAX;
    double y =
      (double)random() / RAND_MAX;

    if (x * x + y * y < 1.0)
      counter++;
  }
  return counter;
}
```

Compute PI - Client Side-

```
#include "grpc.h"
#define NUM_HOSTS 8
char * hosts[] =
  {"host00", "host01", "host02", "host03"
   "host04", "host05", "host06", "host07"
grpc_function_handle_t handles[NUM_HOSTS]

main(int argc, char ** argv){
  double pi;
  long times, count[NUM_HOSTS], sum;
  char * config_file;
  int i;
  if (argc < 3){
    fprintf(stderr,
    "USAGE: %s CONFIG_FILE TIMES \n",
    argv[0]);
    exit(2);
  }
  config_file = argv[1];
  times = atol(argv[2]) / NUM_HOSTS;

  /* Initialize */
  if (grpc_initialize(config_file)
      != GRPC_OK){
    grpc_perror("grpc_initialize");
    exit(2);
  }
  /* Initialize Function Handles */
  for (i = 0; i < NUM_HOSTS; i++)
    grpc_function_handle_init(&handles[i],
         hosts[i], port, "pi/pi_trial");

  for (i = 0; i < NUM_HOSTS; i++)
    /* Asynchronous RPC */
    if (grpc_call_async(&handles[i], i,
          times, &count[i]) ==
GRPC_ERROR){
      grpc_perror("pi_trial");
      exit(2);
    }
  /* Wait all outstanding RPCs */
  if (grpc_wait_all() == GRPC_ERROR){
    grpc_perror("wait_all");
    exit(2);
  }
  /* Display result */
  for (i = 0, sum = 0; i < NUM_HOSTS; i++)
    sum += count[i];
  pi = 4.0 *
    ( sum / ((double) times * NUM_HOSTS));
  printf("PI = %f\n", pi);
  /* Finalize */
  grpc_finalize();
}
```

PART IV
How to run a Grid application

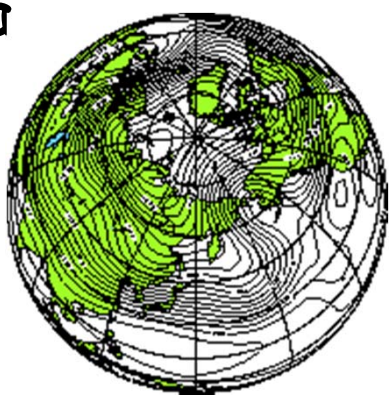
Experiences on running climate simulation on the ApGrid Testbed

What I did

1. **Develop a Grid application**
    1. climate simulation using Ninf-G
    2. giridified a legacy Fortran code using Ninf-G
2. **Test accessibility to each site at Globus level**
    1. Test using globus-job-run
3. **Install Ninf-G at each site**
4. **Test Ninf-G using a sample program**
5. **Install remote library for the climate simulation at each site**
6. **Run the climate simulation (client program)**
7. **Increase resources and improve performance**

Climate Simulation System

- **Forcasting short to middle term climate change**
  - Windings of jet streams
  - Blocking phenomenon of high atmospheric pressure
- **Barotropic S-model proposed by Prof. Tanaka**
  - Legacy FORTRAN program
- **Simple and precise**
  - Treating vertically averaged quantities
  - 150 sec for 100 days prediction/1 simulation
- **Keep high precision over long period**
  - Introducing perturbation for each simulation
  - Taking a statistical ensemble mean
- **Requires100 ~ 1000 simulations**



1989/1/30-2/12

Gridifying the program enables quick response

Grid Technology Research Center AIST

Asia-Pacific Grid

Ninf

AIST

Gridify the original (seq.) climate simulation

- **Dividing a program into two parts as a client-server system**
  - Client:
    - Pre-processing: reading input data
    - Post-processing: averaging results of ensembles
  - Server
    - climate simulation, visualize

Ninf-g

Ninf-g

Ninf-g

S-model Program

Reading data

Solving Equations

Averaging results

VIsualize

Ninf-g

Ninf-g

Grid Technology Research Center AIST

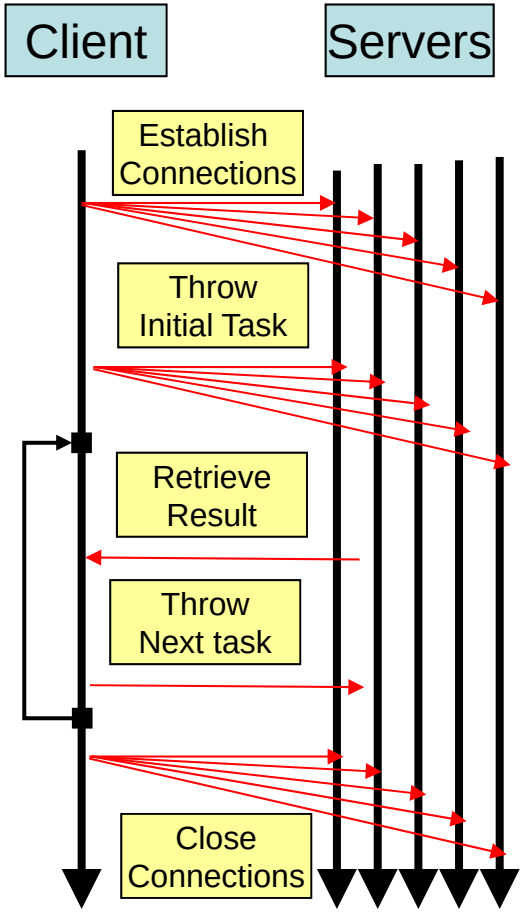
Asia-Pacific Grid

Ninf

AIST

Gridify the climate simulation (cont'd)

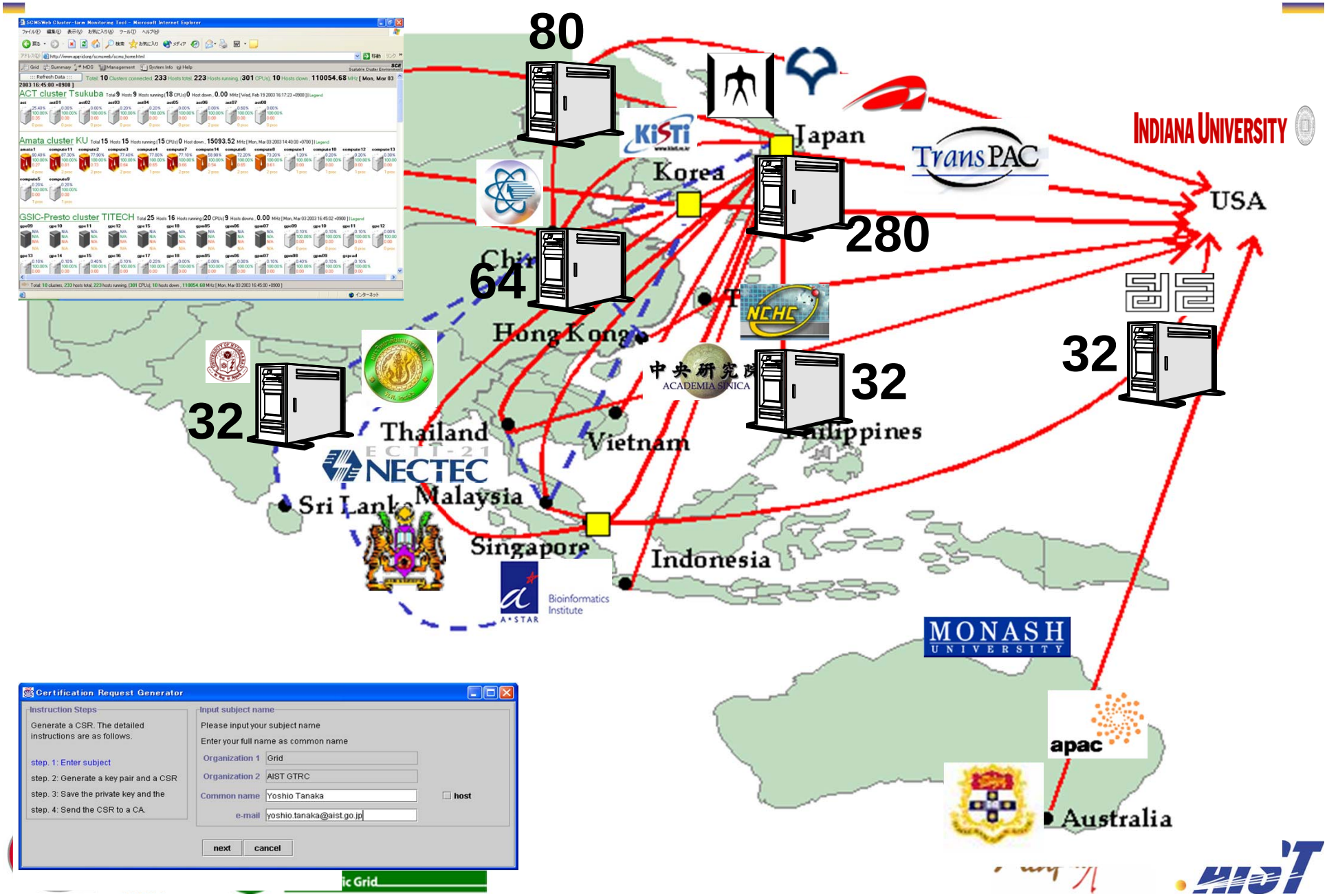
- **Behavior of the Program**
  - Typical to task parallel applications
    - Establish connections to all nodes
    - Distribute a task to all nodes
    - Retrieve a result
    - Throw a next task
- **Cost for gridifying the program**
  - Performed on a single computer
    - Eliminating common variables
    - Eliminating data dependence among server processes
      - Seed for random number generation
  - Performed on a grid environment
    - Inserting Ninf-g functions
    - Creating self scheduling routine



Adding totally ~100 lines (< 10 % of the original program)
Finished in a few days

Testbed: ApGrid Testbed

http://www.apgrid.org/



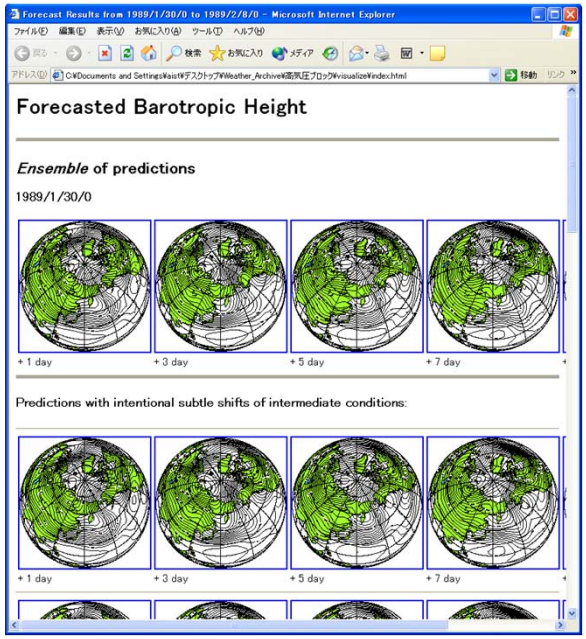
Resources used in the experiment

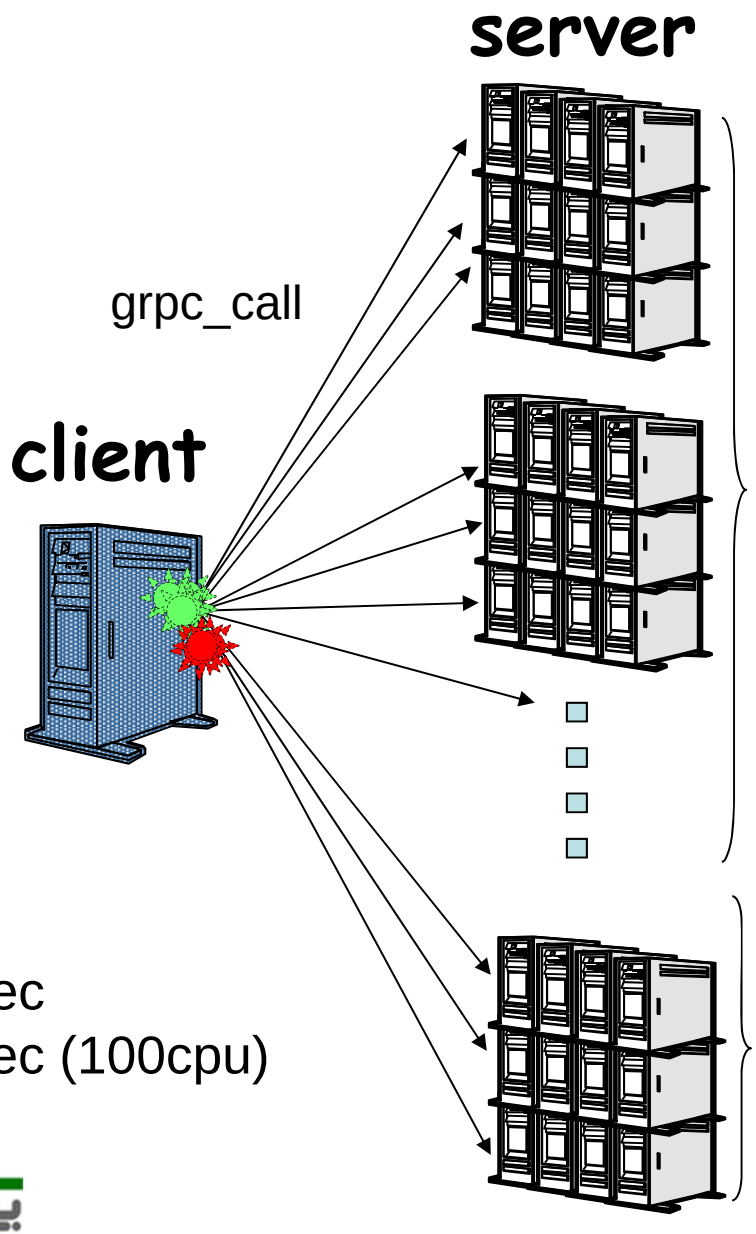
- **KOUME Cluster (AIST)**
  - Client
- **UME Cluster (AIST)**
  - jobmanager-grd, (40cpu + 20cpu)
  - AIST GTRC CA
- **AMATA Cluster (KU)**
  - jobmanager-sqms, 6cpu
  - AIST GTRC CA
- **Galley Cluster (Doshisha U.)**
  - jobmanager-pbs, 10cpu
  - Globus CA
- **Gideon Cluster (HKU)**
  - jobmanager-pbs, 15cpu
  - HKU CA
- **PRESTO Cluster (TITECH)**
  - jobmanager-pbs, 4cpu
  - TITECH CA
- **VENUS Cluster (KISTI)**
  - jobmanager-pbs, 60cpu
  - KISTI CA
- **ASE Cluster (NCHC)**
  - jobmanager-pbs, 8cpu
  - NCHC CA
- **Handai Cluster (Osaka U)**
  - jobmanager-pbs, 20cpu
  - Osaka CA
- **Total: 183**

Grid Technology Research Center AIST

Asia-Pacific Grid

Illustration of Climate Simulation





Sequential Run: 8000 sec
Execution on Grid: 300 sec (100cpu)

Grid Technology Research Center AIST

Asia-Pacific Grid

Ninf

AIST

Lessons Learned

- **We have to pay much efforts for initiation**
  - Problems on installation of GT2/PBS/jobmanger-pbs,grd
    - Failed in lookup service of hostname/IP addresses
      - Both for internet and intranet
      - Add host entries in /etc/hosts in our resources
    - failed in rsh/ssh server to/from backend nodes
      - .rhosts, ssh key, mismatch of hostname
    - pbs_rcp was located in NFS mounted (nosuid) volume
    - bugs in jobmanager scripts (jobmanager-grd is not formally released)
  - GT2 has poor interface with queuing system

Lessons Learned (cont'd)

- **We have to pay much efforts for initiation (cont'd)**
  - What I asked
    - Open firewall/TCP Wrapper
    - Additionally build Info SDK bundle with gcc32dbg
    - Add ${GLOBUS_LOCATION}/lib to /etc/ld.so.conf and run ldconfig (this can be avoided by specifying link option)
    - change configuration of xinetd/inetd
    - Enable NAT

Lessons Learned (cont'd)

- **Difficulties caused by the bottom-up approach for building ApGrid Testbed and the problems on the installation of the Globus Toolkit.**
  - Most resources are not dedicated to the ApGrid Testbed.
    - There may be busy resources
    - Need grid level scheduler, fancy Grid reservation system?
  - Incompatibility between different version of GT2

Lessons Learned (cont'd)

- **Performance Problems**
  - Overhead caused by MDS lookup
    - it takes several 10 seconds
    - Added a new feature to Ninf-G so as to bypass MDS lookup
  - Default polling interval of the Globus jobmanager (30 seconds) is not appropriate for running fine-grain applications.
    - AIST and Doshisha U. have changed the interval to 5 seconds (need to re-compile jobmanager)

Lessons Learned (cont'd)

- **Performance Problems (cont'd)**
  - Time for initialization of function handles cannot be negligible
    - Overhead caused by not only by MDS lookup but also hitting gatekeeper (GSI authentication) and a jobmanager invocation
    - Current Ninf-G implementation needs to hit gatekeeper for initialization of function handles one-by-one
      - Although Globus GRAM enables to invoke multiple jobs at one contact to gatekeeper, GRAM API is not sufficient to control each jobs.
    - Used multithreading for initialization to improve performance
    - Ninf-G2 will provide a new feature which supports efficient initialization of multiple function handles.

Lessons Learned (cont'd)

- **We observed that Ninf-G apps did not work correctly due to un-expected configuration of clusters**
  - Failed in GSI auth. for establishing connection for file transfers using GASS.
    - Backend nodes do not have host certs.
    - Added a new feature to Ninf-G which allows to use non-secure connection
  - Due to the configuration of local scheduler (PBS), Ninf-G executables were not activated.
    - Example:
      - PBS jobmanager on a 16 nodes cluster
      - Call grpc_call 16 times on the cluster. App. developer expected to invoke 16 Ninf-G executables simultaneously.
      - Configuration of PBS Queue Manager set the max number of simultaneous job invocation for each user a 9
      - 9 Ninf-G executables were launched, however 7 were not activated
    - Added a new feature to Ninf-G so as to set timeout for initialization of a function handle.

Lessons Learned (cont'd)

- **Some resources are not stable**
  - example: If I call many (more than 20) RPCs, some of them fails (but sometimes all will done)
  - not yet resolved
  - GT2? Ninf-G? OS? Hardware?
- **Other instability**
  - Version up of software (gt2, pbs, etc.) without notification
    - realized when the application would fail.
    - it worked well yesterday, but I'm not sure whether it works or not today
- **We could adapt for these instability by dynamic task allocation.**

Planned Additional Features for Ninf-G2

- **Get stub information without using LDAP**
  - To avoid unstableness of MDS
  - Enables to install Ninf-G without 'globus' privilege
- **Initialize multiple function handles with one GRAM call**
  - Reduce invocation cost
- **Callback from remote executable to the client**
  - Heatbeat monitoring, visualization, debugging
- **revise the structure of client configuration file**
  - multiple servers
  - jobmanager for each server
  - multiple ldapserver
  - ...

Planned Additional Features (2)

- **Stateful Stubs**
  - Keep state on the server-side
    - Reduce communication
  - Enable remote-object like operation

```
DefClass mvmul
Required "mvmul.o"
{
  DefState {
     double * tmpStorage;
  }
  Define init(IN int N, double  A[N][N]){
    tmpStorage = malloc(sizeof(double) * N * N);
    memcpy(tmpMat, A, sizeof(double) * N * N);
  }
  Define multiply(IN int N, IN double v_in[N],
                  OUT double vout[N])
  {
     mvmul(N, tmpMat, v_in, v_out);
  }
}
```

Planned Additional Features (3)

- **Stateful stubs ClientAPI**
  - Natural extension of the GridRPC API

```
double a[N][N];
double input_vectors[TIMES][N];
double output_vectors[TIMES][N];

grpc_handle_init_default(&handle,
                                    "sample/mvmul");
grpc_invoke(&handle, "init", N, a);
for (int i = 0; i < TIMES; i++){
        grpc_invoke(&handle, "multiply", N,
                                    input_vectors[i],
                                    output_vectors[i]);
}
grpc_handle_finalize(&handle);
```

High-level API development

- **Ninf-G itself does not provide Scheduling, Fault Tolerance and Farming capability**
  - These Capability will be implemented on top of the primitive GridRPC
- **Scheduling:**
  - automatically choose suitable server
- **Fault Tolerance:**
  - detect error and re-submit the failed computation request
- **Farming:**
  - Support massive data-parallel applications

For More Info

- **Ninf home page**
  - http://ninf.apgrid.org
- **GGF GridRPC WG home page**
  - http://www.globalgridforum.org/
- **Contacts**
  - ninf@apgrid.org

PART V
Summary
What has been done? What hasn't?

Summary: How to build a Grid Testbed

- **Difficulties are caused by not technical problems but sociological/political problems**
- **Each site has its own policy**
  - account management
  - firewalls
  - trusted CAs
  - …
- **Differences in interests**
  - Application, middleware, networking, etc.
- **Differences in culture, language, etc.**
  - Human interaction is very important

Summary: How to build a Grid Testbed (cont'd)

- **What has been done?**
  - Resource sharing between more than 10 sites (around 500cpus)
  - Use GT2 as a common software
  - Run Ninf-G applications
- **What hasn't?**
  - I could use, but it is difficult for others
    - I was given an account at each site by personal communication
  - Formalize "how to use the Grid Testbed"
  - Provide documentation
  - Keep the testbed stable
  - Tools for management
    - Browse information
    - CA/Cert. management

Summary: How to build a Grid Testbed (cont'd)

- **Activities at the GGF**
  - Production Grid Management RG
    - Draft a Case Study Document (ApGrid Testbed)
  - Groups in the Security Area
    - Policy Management Authority RG (not yet approved)
      - Discuss with representatives from DOE Science Grid, NASA IPG, EUDG, etc.
    - Federation/publishing of CAs (will kick off)
      - I'll be one of co-chairs

Summary: Programming using GridRPC

- **MPI is not the only programming model!**
  - Choose more appropriate programming model
- **GridRPC is suitable for task-parallel applications**
  - easy programming based on client/server model
- **GridRPC API is going to be standardized at the GGF GridRPC WG**

Summary: Programming using GridRPC (cont'd)

- **What has been done?**
  - Gridify legacy Fortran program (climate simulation) using GridRPC
  - Run the simulation on the ApGrid testbed
    - used approximately 200cpus
- **What hasn't?**
  - Performance improvements
  - Missing functions/capabilities for running large-scale applications
    - hearbeat, callback, high-capability client configuration file, etc.
  - Link to other modules
    - Scheduling/brokering
      - Climate simulation used self-scheduling
    - Fault tolerant
      - Fault simulation was discarded

Summary: Programming using GridRPC (cont'd)

- **Activities at the GGF**
  - GridRPC WG
    - Standardize GridRPC API
      - My colleague (Hide Nakada) is one of co-chairs
      - I'm a secretary
  - Applications and Testbeds RG

Special Thanks (for technical support) to:

- **Kasetsart University (Thailand)**
  - Sugree Phatanapherom
- **Doshisha University (Japan)**
  - Yusuke Tanimura
- **University of Hong Kong (Hong Kong)**
  - CHEN Lin, Elaine
- **KISTI (Korea)**
  - Gee-Bum Koo, Jae-Hyuck
- **Tokyo Institute of Technology (Japan)**
  - Ken'ichiro Shirose
- **NCHC (Taiwan)**
  - Julian Yu-Chung Chen
- **Osaka University (Japan)**
  - Susumu Date
- **AIST (Japan)**
  - Grid Support Team
- **APAN**
  - HK, TW, JP

Grid Technology Research Center AIST

APGrid Asia-Pacific Grid

Nimf AIST

For more info

- **ApGird**
  - http://www.apgrid.org/
  - discuss@apgrid.org
- **Ninf/Ninf-G**
  - http://ninf.apgrid.org/
  - ninf@apgrid.org
- **GGF**
  - http://www.globalgridforum.org/