Towards a Petascale Research Grid Infrastructure in 2010

Satoshi Matsuoka
Professor, Global Scientific Information and Computing Center
Tokyo Institute of Technology
&
Sub(Deputy) Leader, National Research Grid Initiative (NAREGI)
National Institute of Informatics
&
Steering Group Member, Global Grid Forum

Thai Grid Keynote, Bangkok, Thailand, Aug 29, 2006
Petascale Grid Infrastructure R&D for Future Deployment
- $45 mil (US) + $16 mil x 5 (2003-2010) = $125 mil total
- Hosted by National Institute of Informatics (NII) and Institute of Molecular Science (IMS)
- PL: Ken Miura (Fujitsu→NII)
  - Sekiguchi(AIST), Matsuoka(Titech), Shimojo(Osaka-U), Aoyagi (Kyushu-U)...
- Participation by multiple (>= 3) vendors, Fujitsu, NEC, Hitachi, NTT, etc.
- Follow and contribute to GGF Standardization, esp. OGSA
Japanese CyberScience Infrastructure Project

Cyber-Science Infrastructure (CSI)

NAREGI Output

Deployment of NAREGI Middleware (GOC)

UPKI: National Research PKI Infrastructure

Management Body / Education & Training

SuperSINET and Beyond: Lambda-based Academic Networking Backbone

Restructuring Univ. IT Research Resources
Extensive On-Line Publications of Results

NII-REO (Repository of Electronic Journals and Online Publications)

GeNii (Global Environment for Networked Intellectual Information)

VOs Live Collaborations

Industry/Societal Feedback

International Infrastructural Collaboration

GeNii (Global Environment for Networked Intellectual Information)

Hokkaido-U

Tohoku-U

Kyoto-U

Osaka-U

Nagoya-U

Kyushu-U

Titech, Waseda-U, KEK, etc.

NAREGI

Output

Supervision
What are the plausible "utility" of petascale grids?

Peta

Bunch of distributed PCs pretending to be a Supercomputer?
Super SINET provides lambda 10 Gbps Backbone
University Computer Centers (excl. National Labs) circa Spring 2006

10Gbps SuperSINET Interconnecting the Centers

~60 SC Centers in Japan

- 10 Petaflop center by 2011
The Titech TSUBAME Production Supercomputing Cluster, Spring 2006

- **ClearSpeed CSX600 SIMD accelerator**
  - 360 boards, 35TeraFlops (Current)

- **Storage**
  - 1 Petabyte (Sun “Thumper”)
  - 0.1 Petabyte (NEC iStore)
  - Lustre FS, NFS (v4?)

- **Voltaire ISR9288 Infiniband**
  - 10Gbps x2 (xDDR) x ~700 Ports

- **Unified IB network**

- **Sun Galaxy 4 (Opteron Dual core 8-Way)**
  - 10480 core/655 Nodes
  - 50.4 TeraFlops
  - OS Linux (SuSE 9, 10)
  - NAREGI Grid MW

- **NEC SX-8 Small Vector Nodes (under plan)**

- **7th on June 2006 Top500, 38.18 TFlops**
Titech TSUBAME
～80+
350m² floor area
1.2 MW (peak)
**TSUBAME Physical Installation**

- 3 rooms (600m²), 350m² service area
- 76 racks incl. network & storage, 46.3 tons
  - 10 storage racks
- 32 AC units, 12.2 tons
  - Total 58.5 tons (excl. rooftop AC heat exchangers)
- Max 1.2 MWatts
- ~3 weeks construction time
"Everybody’s Supercomputer"

Seamless integration of multiple petascale supercomputers with end-user machines

- Different usage env. from
- No HP sharing with client’s PC
- Special HW/SW, lack of ISV support
- Lack of common development env. (e.g. Visual Studio)
- Simple batch based, no interactive usage, good UI

Hmm, it’s like my personal machine

Seamless, Ubiquitous access and usage

=> Breakthrough Science through Commoditization of Supercomputing and Grid Technologies
TSUBAME HPL Results as of May 8, 2006 (Opteron Only)

- N : 1334160
- NB : 240
- PMAP : Row-major process map
- P : 36
- Q : 144
- PFACT : Right
- NBMIN : 4
- NDIV : 2
- RFACT : Right
- BCAST : 1ring
- DEPTH : 1
- SWAP : Mix (threshold = 240)
- L1 : transposed form
- U : transposed form
- EQUIL : yes
- ALIGN : 8 DP words

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site/Institute</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>Q</th>
<th>Time (s)</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOE/NNSA/LNL</td>
<td>1334160</td>
<td>240</td>
<td>36</td>
<td>144</td>
<td>41462.22</td>
<td>3.818e+04</td>
</tr>
</tbody>
</table>

38.18 TeraFlops@648 nodes
~14 Terabytes memory
76.56% Efficiency
7th on the June 2006 list
Grand Challenge Supercomputing @ Titech
100 Teraflops-scale computing with Petascale Storage

CFD
EMF Simulation
Nanotech
Civil Engineering
Environmental
Weather Prediction

Bio-simulation + Bioinformatics

Bioinformatics

ミンナのスパコン
Incubating the Next-Generation HPC Users

- General Purpose x86 Supercomputing Resources
- Specialized HPC resources

Education

"Expert" HPC Users Evolution

All users, incl. education

General-purpose usage for education, administration

Provision High C/P
Supercomputing in All Educational Activities

- High-End education using supercomputers in undergrad labs
  - High end simulations to supplement “physical” lab courses
- Seamless integration of lab resources to SCs w/grid technologies
- Portal-based application usage

WebMO
Computational Chemistry Web Portal for a variety of Apps
(Gaussian, NWChem, GAMESS, MOPAC, Molpro)
(Prof. Takeshi Nishikawa @ GSIC)

My desktop scaled to 1000 CPUs!😊
TSUBAME General Purpose DataCenter Hosting

• Campus-wide AAA System (April 2006)
  - 50TB (for email), 9 Galaxy1 nodes

• WebDAV Campus-wide Storage Service (July)
  - 10s GBs per everyone on campus
    PC mountable, but accessible directly from TSUBAME
  - Research Repository
    (Research DBs, simulation result archives)

• CAI, On-line Courses
  (OCW = Open CourseWare)

• Administrative Hosting (VM)
Titech University
SSO/AAA & IT Services
Titech Supercomputer Contest
“The 12th SuperCon”

High-school students (18 teams 2006)
Since 1995: Cray => Origin => TSUBAME
Thai teams in 2004 & 5 (again in 2007?)

Titech (TOKYO)

OSAKA Univ.
TSUBAME in Production
June 17-24 2006 (phase 2) ~4700 CPUs
(last week 85% load)
What are the plausible “utility” of petascale grids?

Not really feasible IF ONLY COMPUTE CYCLES IS KEY

Even the Internet is now a datacenter-centric infrastructure

Bunch of distributed PCs pretending to be a Supercomputer?
So what is a Petascale Grid useful for in Research?

- Grid resources managed as grid/datacenters
- Collaborative VO Research Activities using
  - Data and Information Explosion
    - Expensive / insecure to move massive data
    - But data access needs are ubiquitous (unlike computing)
  - Sensor Networks / Sensor Grid
    - Sensors also follow Moore’s law
  - Seamless integration with desktop environment
    “Everybody Supercomputes”
    - Commoditization of petascale usage
    - Thousands usage by millions of users
    - VO Components and Services ALWAYS accessible

- Incubating and managing VOs is the KEY
Information-explosion Era
Peta-Exa-Zeta-Yotta... Bytes
(courtesy Prof. Masaru Kitsuregawa, Univ. of Tokyo)

http://itkaken.ex.nii.ac.jp/i-explosion/

Next Generation Search on PetaScale Grid

Time-based searches facilitate evaluation of social changes and transitions

Exact minority searches

Interactive Searches

Information management by 10,000 processors

© 特定領域研究「情報爆発 IT 基盤」
Explosive World of Sensors on the Petascale Grid

- Sensor data exponential growth
  - Moore’s law
  - Wireless NW
  - Ubiquity
  - More applications

- Essential for Science and Social Infrastructures

- Petabytes that need to be gathered, stored, processed, archived...

Source: Intel
Nano-Science: coupled simulations on the Grid as the sole future for true scalability... between Continuum & Quanta.

Material physics
- (Infinite system)
- Fluid dynamics
- Statistical physics
- Condensed matter theory

Molecular Science
- Quantum chemistry
- Molecular Orbital method
- Molecular Dynamics

Multi-Physics

Limit of Idealization

Limit of Computing Capability

Old HPC environment:
- decoupled resources,
- limited users,
- special software, ...

Coordinates decoupled resources;
Meta-computing,
High throughput computing,
Multi-Physics simulation w/ components and data from different groups within VO composed in real-time

The only way to achieve true scalability!
Distributed Servers

LifeCycle of Grid Apps and Infrastructure

MetaComputing

Workflows and Coupled Apps / User

Many VO Users

VO Application Developers&Mgrs

Place & register data on the Grid

Assign metadata to data

SuperScheduler

Dist. Grid Info Service

Application Contents Service

HL Workflow NAREGI WFML

GridRPC/Grid MPI

User Apps

GridVM

Distributed Servers

GridFS, Metadata, Staging, etc.

Meta-data

Data 1

Data 2

Data n

Grid-wide Data Management Service

User Apps

User Apps

User Apps

Data 1

Data 2

Data n
Grid for VO Collaborative Computing

To realize heterogeneous large scale computational environment
To share Large and expensive devices and databases
NAREGI is/has/will...

- Is THE National Research Grid in Japan
  - Part of CSI and future Petascale initiatives
  - METI “Business Grid” counterpart 2003-2005
- Has extensive commitment to WS/GGF-OGSA
  - Entirely WS/Service Oriented Architecture
  - Set industry standards e.g. 1st impl. of OGSA-EMS
- Will work with EU/US/AP counterparts to realize a “global research grid”
  - Various talks have started, incl. SC05 interoperability meeting
- Delivered first OS public beta 1 in May 2006
  - Distributed @ GGF17/GridWorld in Tokyo
NAREGI is not/doesn’t/won’t…

• Is NOT an academic research project
  - All professional developers from Fujitsu, NEC, Hitachi, (No students)
  - *IMPORTANT for Vendor buy-in and tech transfer*

• Will NOT develop all software by itself
  - Will rely on external components in some cases
  - Must be WS and other industry standards compliant
  - *IMPORTANT for Vendor buy-in and tech transfer*

• Will NOT hinder industry adoption at all costs
  - Intricate open source copyright and IP policies
  - We want companies to save/make money using NAREGI MW
  - *IMPORTANT for Vendor buy-in and tech transfer*
# NAREGI Grid Middleware Roadmap

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UNICORE-based R&amp;D Framework</td>
<td>OGSA/WSRF-based R&amp;D Framework</td>
<td>Utilization of NAREGI NII-IMS Testbed</td>
<td>Utilization of NAREGI-Wide Area Testbed</td>
<td></td>
</tr>
</tbody>
</table>

- **Prototyping NAREGI Middleware Components**
- **Apply Component Technologies to Nano Apps and Evaluation**
- **Development and Integration of α release**
- **β Release (public)**
- **Evaluation of α release in NII-IMS Testbed**
- **Development and Integration of β release**
- **Evaluation on NAREGI Wide-area Testbed**
- **Deployment of β**
- **Series of β Releases (public)**
- **Evaluation of β Release By IMS and other Collaborating Institutes**
- **Development of OGSA-based Middleware**
- **Version 1.0 Release**
- **Verification & Evaluation Of Ver. 1**

**Midpoint Evaluation**
NAREGI Software Stack (beta 1 2006)
- WS(RF) based (OGSA) SW Stack -

Packaging

Grid-Enabled Nano-Applications (WP6)

Grid Programming (WP2)
- Grid RPC
- Grid MPI

Grid Visualization

Grid PSE

Grid Workflow (WFML (Unicore+ WF))

Super Scheduler

Distributed Information Service (CIM)

Grid VM (WP1)

WSRF (GT4+Fujitsu WP1) + GT4 and other services

Grid Security and High-Performance Grid Networking (WP5)

SuperSINET

NII
IMS
Research Organizations
Major University Computing Centers

Computing Resources and Virtual Organizations
Standards: Setting & Adoption
Critical to Grid Adoption & Longevity

- Project lifetimes limited
- Most production centers, commercial entities, and users won’t commit their time & money unless longevity guaranteed
  - A few brave ones... (high-risk, high-return)
  - Success legacies overtold
- **Responsible Action** for major research grid infrastructural projects (e.g. NAREGI) to guarantee longevity beyond project lifetime
- Commercial Adoption by IT vendors: ISPs, ISVs, ASPs, etc. key for such longevity
  - But how do we achieve that?
List of NAREGI “Standards”
(beta 1 and beyond)

- GGF Standards and Pseudo-standard Activities set/employed by NAREGI
  - GGF “OGSA CIM profile”
  - GGF AuthZ
  - GGF DAIS
  - GGF GFS (Grid Filesystems)
  - GGF Grid CP (GGF CAOPs)
  - GGF GridFTP
  - GGF GridRPC API (as Ninf-G2/G4)
  - GGF JSDL
  - GGF OGSA-BES
  - GGF OGSA-Byte-IO
  - GGF OGSA-DAI
  - GGF OGSA-EMS
  - GGF OGSA-RSS
  - GGF RUS
  - GGF SRM (planned for beta 2)
  - GGF UR
  - GGF WS-I RUS
  - GGF ACS
  - GGF CDDLM

- Other Industry Standards Employed by NAREGI
  - ANSI/ISO SQL
  - DMTF CIM
  - IETF OCSP/XKMS
  - MPI 2.0
  - OASIS SAML2.0
  - OASIS WS-Agreement
  - OASIS WS-BPEL
  - OASIS WSRF2.0
  - OASIS XACML

- De Facto Standards / Commonly Used Software Platforms Employed by NAREGI
  - Ganglia
  - GFarm 1.1
  - Globus 4 GRAM
  - Globus 4 GSI
  - Globus 4 WSRF (Also Fujitsu WSRF for C binding)
  - IMPI (as GridMPI)
  - Linux (RH8/9 etc.), Solaris (8/9/10), AIX, ...
  - MyProxy
  - OpenMPI
  - Tomcat (and associated WS/XML standards)
  - Unicore WF (as NAREGI WFML)
  - VOMS

Implement “Specs” early even if nascent if seemingly viable

Necessary for Longevity and Vendor Buy-In

Metric of WP Evaluation
NAREGI Middleware

Demonstrates process flow and several functionalities working together as Grid Services.

- Users can execute huge jobs such as coupled simulations using appropriate resources across multiple computer centers,
- VO members across organizations can share the resources.
- NAREGI widely adopts grid standards, and will contribute standardization and multi-grid interoperation.
- Open Source package self-contained

⇒ Sustainable Research Grid Infrastructure.
NAREGI R&D Assumptions and Goals

• Future Research Grid Metrics for Petascale
  - 10s of Grid Centers, various Project VOs
  - > 100,000 users, > 100,000~1,000,000 CPUs
    • Machines are very heterogeneous
      CPUs (super computers, clusters, desktops), OSes, local
      schedulers => Datacenter model
    - 24/7 usage, production deployment
    - Server Grid, Data Grid, Metacomputing …

• High Emphasis on Standards
  - WS and GT4 and other common Grid MW as basis
  - GGF contributions, esp. OGSA™ reference implementation

• Win support of users
  - R&D for production quality (free) software
  - Nano-science involvement
  - Medium-sized dedicated testbed (~3000 CPUs) of varying
types
\[\beta_1\] Release Features

- **Resource and Execution Management**
  - GGF OGSA-EMS based architecture
  - Automatic resource brokering and job scheduling
  - Reservation based co-allocation
  - VO and local policy based access control
  - Network traffic measurement and control
  - GGF JSDL based job description
  - DMTF-CIM based resource information model
  - GGF OGSA-RUS based accounting

- **Data Grid (New from FY2005)**
  - WSRF based grid-wide data sharing
Release Features continued

• User-Level Tools
  – Web based Portal
  – Workflow tool w/NAREGI-WFML
  – Application contents and deployment service
  – Large-Scale Interactive Grid Visualization

• Grid Ready Programming Tools & Libraries
  – Standards compliant GridMPI and GridRPC

• Bridge Method for Different Type Applications

• VO support
  – Production-quality CA, APGrid PMA
  – VOMS based VO user management

and more …
NAREGI Programming Models

- **High-Throughput Computing**
  - But with complex data exchange inbetween
  - NAREGI Workflow or GridRPC

- **Metacomputing (cross-machine parallel)**
  - Workflow (w/co-scheduling) + GridMPI
  - GridRPC (for task-parallel or task/data-parallel)

- **Coupled Multi-Resolution Simulation**
  - Workflow (w/co-scheduling) + GridMPI + Coupling Components
    - Mediator (coupled simulation framework)
    - GIANT (coupled simulation data exchange framework)
NAREGI Application Mediator (WP6) for Coupled Applications

Mediator Components
Support data exchange between coupled simulation

Data transfer management
- Synchronized file transfer
- Multiple protocol GridFTP/MPI

Data transformation management
- Semantic transformation libraries for different simulations
- Coupled accelerator

* SBC: Storage-based communication
**RISM-FMO Coupled Simulation**

Electronic structure of Nano-scale molecules in solvent is calculated self-consistent by exchanging solvent charge distribution and partial charge of solute molecules.

*Original RISM and FMO codes are developed by Institute of Molecular Science and National Institute of Advanced Industrial Science and Technology, respectively.*
RISM-FMO Application
A coupled simulation of the electronic structure of a molecule in solution

Requirements

<table>
<thead>
<tr>
<th>System</th>
<th>CPU</th>
<th>OS</th>
<th>#CPUs</th>
<th>#nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISM</td>
<td>Power4</td>
<td>AIX</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>FMO</td>
<td>Xeon</td>
<td>Linux</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Mediator</td>
<td>Xeon</td>
<td>Linux</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>IMPI</td>
<td>Xeon</td>
<td>Linux</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
a – c, 1: Job Submission: Client Side

- a: Sign-on
  - User cert.
  - VOMS
  - Proxy cert. (User DN, VO)
  - MyProxy

- b: Deployment
  - Portal
  - PSE
  - WFT

- c: Edit
  - Super Scheduler
  - GridVM
  - Local Scheduler
  - IMPI Server
  - GridMPI
  - RISM Job
  - Network monitor
  - DataGrid

- 1: Submission
- 2: Resource discovery
- 3: Negotiation
- 4: Reservation
- 5: IMPI starts
- 6: MPI job starts
- 7: MPI init.
- 8: Visualization
- 9: Accounting

Note) There is one more job: Mediator Job
a: Single Sign-On

Users access to portal, get VOMS-proxy certificates (/w their VO attrib) deposited beforehand and submit their jobs to NAREGI M/W using Grid tools.
b-1: Registration & Deployment of Applications

Application sharing in Research communities

1. Register Application
2. Select Compiling Host
3. Compile
4. Send back Compiled Application Environment
5. Select Deployment Host
6. Deploy
7. Register Deployment Info.

Server#1
- Compiling: OK!
- Test Run: OK!

Server#2
- Test Run: NG!

Server#3
- Test Run: OK!

PSE Server
ACS (Application Contents Service)

Resource Info.
Information Service
b-2: Data Import to Data Sharing Service

Data Grid Components

Data Access Management
Metadata Management
Data Resource Management

Import data into workflow
Place & register data on the Grid
Assign metadata to data
Store data into distributed file nodes

Grid-wide Data Sharing Service

Grid-wide File System

Job 1
Data 1
Job 2
Data 2
Job n
Data n

Job 1
Job 2
Job n
c: Description of Workflow and Job Submission Requirements

Data icon
/gfarm/..

Program icon

Appli-A
JSDL

Appli-B
JSDL

Global file information

Application Information

DataGrid

PSE

Web server (apache)

Workflow Servlet

tomcat

Workflow Description
By NAREGI-WFML

NAREGI JM I/F module

Super Scheduler

Server

GridFTP
(Stdout Stderr)

BPEL+JSDL

BPEL
<invoke name=EPS-jobA>
↓
JSDL -A

<invoke name=BES-jobA>
↓
JSDL -A

http(s)

applet

Appi-A
JSDL

JSDL

Appi-B
JSDL

Application Information

Information Service

Global file information

c: Description of Workflow and Job Submission Requirements
2 - 6: Resource Discovery, Co-Allocation, Advanced Reservation and Submission

Application requirement definition

1: Submission

2: Resource discovery

3: Negotiation

4: Reservation

5: IMPI starts

6: MPI job starts

7: MPI init.

8: Visualization

9: Accounting

2: Monitoring
Distributed Info. Services maintain various kind of information; CPU, Memory, OS, Job Queue, Account, Usage Record, etc. etc. across multiple administrative domains,

- Abstract heterogeneous resources (CIM schema) → RDF
- Retrieve resource DB through Grid Service (OGSA-DAI)
- Access resource info. according to the users’ right.

2: Resource Discovery

WFT

Sub job requirements

Super Scheduler

OGSA-DAI

Distributed Information Service

... Associated tables based on CIM schema.
NAREGI Resource Management and OGSA-EMS
Open Grid Service Architecture - Execution Management Service will be standardized by OGSA EMS-WG

- NAREGI will contribute our middleware as a reference implementation of OGSA-EMS.
3, 4: Co-allocation and Reservation

to allocate and execute jobs on multiple sites simultaneously.

Abbreviation
SS: Super Scheduler
JSDL: Job Submission Description Document
JM: Job Manager
EPS: Execution Planning Service
CSG: Candidate Set Generator
RS: Reservation Service
IS: Information Service
SC: Service Container
AGG-SC: Aggregate SC
GVM-SC: GridVM SC
FTS-SC: File Transfer Service SC
PBS, LoadLeveler: Basic Execution Service I/F
PBS, LoadLeveler: Basic Execution Service I/F (BES+)
CIM: Common Information Model
GNIS: Grid Network Information Service
5, 6: Advanced Reservation and Job Execution (GridMPI job)

WorkFlow Tool → Abstract JSDL → Super Scheduler

Co-allocation

Concrete JSDL
Reserved resources for the parallel jobs

Concrete JSDL

Start the jobs at the reserved time

GridVM
Site A

GridVM
Site B

Generate a shell script for the local MPI job according to JSDL and run the script.
Configure the IMPI server and the execution nodes, etc…
7: GridMPI Communication and Network Monitoring

Application requirement definition

1: Submission
2: Resource discovery
3: Negotiation
4: Reservation
5: IMPI starts
6: MPI job starts
7: MPI init.
8: Visualization
9: Accounting

Portal

a: Sign-on
b: Deployment
c: Edit

MPI source

Work-flow

Super Scheduler

PSE

WFT

IMPI

Input files

RISM

FMO

GridMPI

GVS

Output files

GridMPI

SMP machine 64 CPUs
PC cluster 128 CPUs

Network monitoring

Co-Allocation

Agreement

Co-Allocation

Resource discovery

Reservation

Visualization

Accounting

Information Service

Sign-on

Portal

VOMS

MyProxy

Submission

MPI job starts

MPI init.
GridMPI

MPI applications run on the Grid environment

- **Metropolitan area, high-bandwidth environment:** $\geq 10$ Gpbs, $\leq 500$ miles (smaller than 10ms one-way latency)
  - Parallel Computation
- **Larger than metropolitan area**
  - MPI-IO

```
computing resource
site A
```
```
computing resource
site B
```

Wide-area Network

Single (monolithic) MPI application over the Grid environment
GridMPI is a library which enables MPI communication between parallel systems in the grid environment. (in Metropolitan Area: ≥ 10Gbps, ≤ 10ms)

- Huge data size jobs which cannot be executed in a single cluster system
- Multi-Physics jobs in the heterogeneous CPU architecture environment

1. Inter-operability:
   - IMPI (Interoperable MPI) compliance communication protocol
   - Strict implementation of MPI 2.0 standard (test suite compliant)

2. High performance:
   - Simple implementation
Ninf-G: A Reference Implementation of the GGF GridRPC API

- **What is GridRPC?**
  - Programming model using RPCs on a Grid
  - Provide easy and simple programming interface
  - The GridRPC API is published as a proposed recommendation (GFD-R.P 52)
- **What is Ninf-G?**
  - A reference implementation of the standard GridRPC API
  - Built on the Globus Toolkit
  - Now in NMI Release 8 (first non-US software in NMI)
- **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

**Utilization of remote supercomputers**

**Large scale computing across supercomputers on the Grid**
7': Network Monitoring

Monitor, control and manage network resources with Network Middleboxes.

GUI for NW admin (requests of route control, visualization of topology & measuring result)

Active measurement

IP Tunnel (for Route Control)
8, 9: Grid Visualization and Accounting

1: Submission
2: Resource discovery
3: Negotiation
4: Reservation
5: IMPI starts
6: MPI job starts
7: MPI init.
8: Visualization
9: Accounting
Integrated WSRF-based remote visualization environment
- Post-Processing Visualization
- Loosely Coupled Concurrent Visualization

8: Visualization Service

Portal

Client

Service Provider

MPI Parallel Visualizer

Service Provider

MPI Parallel Visualizer

Front-end

Computing node

PC cluster

Front-end

Back-end

Super computer
9: Job Accounting

Collects and maintains Job/WF Usage Records (RUS, UR) in the form of the GGF proposed standard.

RUS: Resource Usage Service
UR: Usage Record

Users can search and summarize their URs by:
- Global User ID
- Global Job ID
- Global Resource ID
- Time range...

Diagram:
- "Cell domain" Info.Service
- GridVM_Scheduler
- GridVM_Engine
- Process
- Computing node
- Cluster / Site
VO and Resources in Beta 2

Full Decoupling of Grid MW ---- Resources vs. VOs

Center1

vo-APP1

vo-Center1

Center2

vo-APP2

Research1

Client

Client

VO-Center1

GridVM

IS

Policy
• VO-C1
• VO-APP1

GridVM

IS

Policy
• VO-C1
• VO-APP1

GridVM

IS

Policy
• VO-C1
• VO-APP1
• VO-APP2

GridVM

IS

Policy
• VO-C2
• VO-APL1
• VO-APP2

GridVM

IS

Policy
• VO-C2
• VO-APP2

GridVM

IS

Policy
• VO-C2

Full Decoupling of Grid MW ---- Resources vs. VOs
NAREGI Phase 1 Testbed

Dedicated Testbed
No “ballooning” w/production resources
NOT just Linux/x86 boxes

~3000 CPUs
~17 Tflops
~100TBytes

Super-SINET
(10Gbps MPLS)

TiTech Campus Grid

AIST SuperCluster

Tohoku Univ. Small Test App Clusters

AIST Small Test App Clusters

KEK Small Test App Clusters

ISSP Small Test App Clusters

Kyushu Univ. Small Test App Clusters

Kyoto Univ. Small Test App Clusters

Osaka Univ. BioGrid

Computational Nano-science Center (IMS)
~10 Tflops

Center for GRID R&D (NII)
~5 Tflops
Titech Supercomputing Grid 2006

~13,000 CPUs, 90 TeraFlops, ~26 TeraBytes Mem, ~1.1 Petabytes Disk

- CPU Cores: x86: TSUBAME (~10600), Campus Grid Cluster (~1000), COE-LKR cluster (~260), WinCCS (~300)
Scaling To Petaflops...

2010 TSUBAME2
=> Interim 200TeraFlops @ 2008
=> Sustained Petaflop @ 2010
Sustain leadership in Japan

US Petascale
(Peak)
(2007~8)

US HPCS
(2010)

US 10P
(2011~12?)

Japanese
“Keisoku”
>10PF(2011-12)

Titech
Supercomputing
Campus Grid (incl
TSUBAME )~90TF
(2006)

KEK 59TF
BG/L+SR11100

Titech
Campus Grid
40TF (2002)

Earth Simulator
40TF (2002)

BlueGene/L
360TF(2005)

U-Tokyo,
Kyoto-U,
etc.
>200TF
(2008-1H)

Titech

Titech

Scaling To Petaflops…
TSUBAME 2: Achieving 1 Petaflop Linpack Sustained in 2010-2011

• Details forthcoming
  (Can’t tell you yet 😊)
## Performance/Watt of TSUBAME Comparisons

<table>
<thead>
<tr>
<th>Machine</th>
<th>CPU Cores</th>
<th>Watts</th>
<th>Peak GFLOPS</th>
<th>Peak MFLOPS/Watt</th>
<th>Watts/CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSUBAME (Opteron)</td>
<td>10480</td>
<td>800,000</td>
<td>50,400</td>
<td>63</td>
<td>76.336</td>
</tr>
<tr>
<td>TSUBAME (w/CS)</td>
<td>11,200</td>
<td>810,000</td>
<td>85,000</td>
<td>104.94</td>
<td>72.321</td>
</tr>
<tr>
<td>Earth Simulator</td>
<td>5120</td>
<td>8,000,000</td>
<td>40,000</td>
<td>5</td>
<td>1562.5</td>
</tr>
<tr>
<td>ASCI Purple</td>
<td>12240</td>
<td>7,000,000</td>
<td>77,824</td>
<td>11.18</td>
<td>571.9</td>
</tr>
<tr>
<td>Orion (DS-96)</td>
<td>96</td>
<td>1400</td>
<td>268.8</td>
<td>192</td>
<td>14.583</td>
</tr>
<tr>
<td>Low power Cluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BG/L (rack)</td>
<td>2048</td>
<td>25,000</td>
<td>5734.4</td>
<td>229.38</td>
<td>12.207</td>
</tr>
<tr>
<td>BG/P? (rack)</td>
<td>4096</td>
<td>30,000</td>
<td>16384</td>
<td>546.13</td>
<td>7.3242</td>
</tr>
<tr>
<td>Next Gen (2010)</td>
<td>40000</td>
<td>800,000</td>
<td>10,000,000</td>
<td>1250</td>
<td>20</td>
</tr>
</tbody>
</table>
The Colossal Goal of 10 PetaScale

<table>
<thead>
<tr>
<th></th>
<th>Titech Super-computing Grid</th>
<th>Keisoku machine</th>
<th>Multiplication Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak Perf.</td>
<td>&gt;100 TeraFlops</td>
<td>&gt;10 PetaFlops</td>
<td>&gt;x100</td>
</tr>
<tr>
<td>Power (W)</td>
<td>~1 MW</td>
<td>Over 100 MW</td>
<td>Over $50 mil elec. bill</td>
</tr>
<tr>
<td>Cost (US$)</td>
<td>$10~20 mil /y</td>
<td>$1~2 bil/y</td>
<td>x 10 Earth Simulator</td>
</tr>
<tr>
<td>Area (m²)</td>
<td>200~300 m²</td>
<td>20,000~30,000 m²</td>
<td>Requires a Tokyo Dome for housing</td>
</tr>
<tr>
<td>CPU Cores</td>
<td>~11,000</td>
<td>&gt;1 million</td>
<td>20% all PCs sold in Japan (units/y)</td>
</tr>
<tr>
<td>Memory</td>
<td>21.4 TeraByte</td>
<td>&gt;2 PetaByte</td>
<td>2 million 1GB DIMMs</td>
</tr>
<tr>
<td>Secondary Storage</td>
<td>1.1 PetaByte</td>
<td>110 PetaByte</td>
<td>250,000 400GB HDD 5% all HDD/DVD recorders in Japan</td>
</tr>
</tbody>
</table>

Need R&D (and Money)  
(and motivation, too!)
Upscaling the Resources to a Petascale Grid

- #Users: ~1000
- Capacity: ~10TF

- #Users: ~1,000,000
- Capacity: ~10PF

- Capacity: ~1GF
- Users: > x1000

- Users: ~1000
- Capacity: ~1GF

- Users: ~1,000,000
- Capacity: ~1GF

- Users: x10^6
- Capacity: x10^7
The Ideal World: Ubiquitous VO & user management for international e-Science

Europe: EGEE, UK e-Science, ...

US: TeraGrid, OSG,

Japan: NII CyberScience (w/NAREGI), ...
Other Asian Efforts (China Grid, Thai Grid)...

HEP Grid VO
NEES-ED Grid VO
Astro IVO

Standardization, and Grid Interoperations NOW! (GIN)

Different software stacks but interoperable

Grid Regional Infrastructural Efforts
Collaborative talks on PMA, etc.
More Details About NAREGI Execution Architecture
New NAREGI Super-Scheduler

GGF OGSA-EMS implementation written in C.

Support Resource Discovery

- Discover available resources for a job and select the appropriate system(s) and submit the job.

Support Complex Job Execution - Co-allocation

- Enable to run a job on more than one resources for MPI job.
- If a MPI job doesn’t match appropriate resources, SuperScheduler divide into the job into multiple fragment jobs and run on multiple resources.

Full Feature Toolkit

- HTTP(s), SOAP, WS-Addressing, WS-ResourceFramework, WS-Notification, BPEL compiler/Interpreter, WSDL Compiler, JSDL, OGSA-EMS(CSG,EPS,RS, SC,JM)
Naregi WSRF Kit (NWK)

- Base software for SuperScheduler.
- Features
  - WSRF, WS-Notification, WS-Addressing
  - Wsdl Compiler (generate stub/skeleton code)
  - Interoperability with Globus4
  - Developed by C and libxml2
Abbreviation

SS: Super Scheduler
JSDL: Job Submission Description Document
JM: Job Manager
EPS: Execution Planning Service
CSG: Candidate Set Generator
RS: Reservation Service
IS: Information Service
SC: Service Container
AGG-SC: Aggregate SC
GVM-SC: GridVM SC
FTS-SC: File Transfer Service SC
BES: Basic Execution Service I/F
CES: Co-allocation Execution Service I/F (BES+)
CIM: Common Information Model
GNIS: Grid Network Information Service

Co-allocation FileTransfer

globus-url-copy
uber-ftp
globus-url-copy
GFarm server
FileTransfer

MakeReservation
CancelReservation

SelectResource
FromJSDL

Fork/Exec
is-query

Generate Candidate Set
Generate SQL Query From JSDL

GetGroups-OfNodes

MakeReservation CancelReservation

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSIDL)
GetActivityStatus RequestActivityStateChanges

MakeReservation CancelReservation

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromBPEL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
GetActivityStatus RequestActivityStateChanges

CreateActivity(FromJSDL)
Meta computing scheduler is required to allocate and to execute jobs on multiple sites simultaneously.

The super scheduler negotiates with local RSs on job execution time and reserves resources which can execute the jobs simultaneously.
The NAREGI SSS Architecture (2007/3)

GRID MIDDLEWARE

PETABUS (Peta Application services Bus)

WESBUS (Workflow Execution Services Bus)

CESBUS (Coallocation Execution Services Bus; a.k.a. BES+ Bus)

BESBUS (Basic Execution Services Bus)

GRID RESOURCE

Application Specific Service

Application Specific Service

Application Specific Service

JM

EPS

CSG

BPEL Interpreter Service

FTS-SC

GRAM-SC

UniGridS-SC

AGG-SC with RS (Aggregate SCs)

Globus

WS-GRAM I/F (with reservation)

UniGridS Atomic Services (with reservation)

GridVM
NAREGI Info Service (beta) Architecture

- CIMOM Service classifies info according to CIM based schema.
- The info is aggregated and accumulated in RDBs hierarchically.

Client library utilizes OGSA-DAI client toolkit.

Accounting info is accessed through RUS.
NAREGI IS: Standards Employed in the Architecture

- Client (OGSA-RSS etc.)
- Java-API
- Client library
- OGSA-DAI Client toolkit
- GridVM (Chargeable Service)
- RUS::insertURs
- GGF/UR
- Cell Domain Information Service
- Grid VM
- Job Queue
- OS
- File System
- Processor
- Performance
- Ganglia
- Node A
- Node B
- Node C
- GT4.0.1
- Tomcat 5.0.28
- Viewer
- Information Service Node
- Information Service Node
- OGSA-DAI WSRF 2.1
- WS-I-RUS
- RDB
- Aggregator Service
- CIM spec. CIM/XML
- Lightweight CIMOM Service
- Performance
- Client (OGSA-BES etc.)
GridVM Features

✓ Platform independence as OGSA-EMS SC
  • WSRF OGSA-EMS Service Container interface for heterogeneous platforms and local schedulers
  • “Extends” Globus4 WS-GRAM
  • Job submission using JSDL
  • Job accounting using UR/RUS
  • CIM provider for resource information

✓ Meta-computing and Coupled Applications
  • Advanced reservation for co-Allocation

✓ Site Autonomy
  • WS-Agreement based job execution (beta 2)
  • XACML-based access control of resource usage

✓ Virtual Organization (VO) Management
  • Access control and job accounting based on VOs (VOMS & GGF-UR)
NAREGI GridVM (beta) Architecture

- Virtual execution environment on each site
  - Virtualization of heterogeneous resources
  - Resource and job management services with unified I/F

Diagram:
- Super Scheduler
- Information Service
- Advance reservation, Monitoring, Control
- Resource Info.
- Accounting
- GridMPI
- Sandbox
- Job Execution
- GridVM Engine
- Local Scheduler
- GridVM Scheduler
- GRAM4
- WSRF I/F
- AIX/LoadLeveler
- Linux/PBSPro
- Site Policy
NAREGI GridVM: Standards Employed in the Architecture

- Super Scheduler
- Information Service
  - GT4 GRAM-integration and WSRF-based extension services
  - Job submission based on JSDL and NAREGI extensions
- GridVM Scheduler
- Local Scheduler
- GridVM Engine
- GridMPI
- site Policy
- site Policy
- xacml-like access control policy
- CIM-based resource info. provider
- UR/RUS-based job accounting
GT4 GRAM-GridVM Integration

- Integrated as an extension module to GT4 GRAM
- Aim to make the both functionalities available

SS

globusrun RSL+JSDL'

Basic job management + Authentication, Authorization

Delegate

Delegation

RFT File Transfer

GRAM services

GridVMJobFactory

GridVMJob

GridVM JobFactory

Scheduler Event Generator

SUDO

GRAM Adapter

GridVM scheduler

Local scheduler

GridVM Engine

PBS-Pro

LoadLeveler
VO and Resources in Beta 2

Decoupling of WP1 components for pragmatic VO deployment

"Peter Arzberger" <parzberg@sdsce.edu>
VO and User Management Service

- Adoption of VOMS for VO management
  - Using proxy certificate with VO attributes for the interoperability with EGEE
  - GridVM is used instead of LCAS/LCMAPS
- Integration of MyProxy and VOMS servers into NAREGI
  - with UMS (User Management Server) to realize one-stop service at the NAREGI Grid Portal
  - using gLite implemented at UMS to connect VOMS server
- MyProxy+ for SuperScheduler
  - Special-purpose certificate repository to realize safety delegation between the NAREGI Grid Portal and the Super Scheduler
  - Super Scheduler receives jobs with user’s signature just like UNICORE, and submits them with GSI interface.
## Security Requirements in AAA

### Authentication
- Developed
  - PKI based user authentication
  - Compatible with GSI standards
  - Trust federation between CA's (IGTF)

### Authorization
- Current Issues to be solved
  - VO management for
    - Inter-organizational collaboration
  - Interoperable with other Grid projects - Common Policies and Schemas (Vocabularies)

### Accounting & Identity
- Future Issues
  - ID federation for authorization & traceability
  - With privacy protection!
  - Across entire Japan and the World (millions!)
A virtual organization (VO) is a dynamic collection of resources and users unified by a common goal and potentially spanning multiple administrative domains.

Definition of VO on GGF

- CAS (Community Authorization Service)
- VOMS (Virtual Organization Membership Service)
VOMS-type VO Management developed in EGEE

CA/RA → CRL

DN, VO, Group, roll, capability

VOMS

User Cert

Proxy Cert + VO

Grid map file

MK-gridmapfile

DN > pseudo accounts

GACL

LCAS

GRAM

EGEE Grid site

User

Grid Job Submission
VOMS-type VO Management adopted in NAREGI

- CA/RA
- CRL
- DN, VO info
- Information Service
- Account Mapping
- Gridmap file
- Policy file
- Grid VM
- NAREGI Grid site

- User
- User Cert
- Proxy Cert + VO
- Certificates handling is too hard for users
- Grid Job Submission
- Managed by the Super Scheduler
- GRAM
- Managed by the Super Scheduler

VOMS-type VO Management adopted in NAREGI
Job Submission mechanism in NAREGI Middleware β version

Integrated and easy handling of VOMS and MyProxy

WF Credential is a user proxy cert passed through to the SS with the delegation protocol

The SS receives WF and deploys Grid jobs
AuthN/AuthZ Services in the future

- CA/RA
- CRL
- LDAP
- MyProxy
- VO Management
- Proxy Cert of User
- User CRL
- User Cert
- Web Server
- Grid Job Submission
- SAML+XACML
- Policy Information Point
- Policy Decision Point
- Authentication & Authorization Service
- Policy Enforcement Point
- Super Scheduler GRAM (Grid VM)
- Grid Job Submission
- Log in
Computational Resource Allocation based on VO

- Resource Configuration
  - VO1
    - pbg1042
      - 4 CPU
    - png2040
      - 2 CPU
    - png2041
      - 4 CPU
  - VO2
    - pbg2039
      - 8 CPU

- Workflow
  - App-4a
    - 4 CPU
  - App-4b
    - 4 CPU
  - App-1
    - 1 CPU
  - App-2
    - 2 CPU

Different resource mapping for different VOs
Local-File Access Control (GridVM)

- Provide VO-based access control functionality that does not use gridmap files.
- Control file-access based on the policy specified by a tuple of **Subject**, **Resource**, and **Action**.
- Subject is a grid user ID or VO name.

```
Policy
Permit: Subject=X, Resource=R, Action=read,write
Deny: Subject=Y, Resource=R, Action=read
```
Structure of Local-File Access Control Policy

<GridVMPolicyConfig>
  1
  1
</GridVMPolicyConfig>

<AccessControl>
  1
  1
</AccessControl>

<AccessProtection>
  1
  1
</AccessProtection>

<AccessRule>
  1
  1..*
</AccessRule>

<Effect>
  1
  0..1
</Effect>

<Resources>
  1
  1
</Resources>

<Actions>
  1
  0..1
</Actions>

<Subjects>
  1
  0..1
</Subjects>

<AppliedTo>
  0..1
  +TargetUnit
</AppliedTo>

<Subject>
  0..1
</Subject>

<Resource>
  0..1
</Resource>

<Action>
  0..1
</Action>

Who
user / VO

What Resource
file / directory

Access Type
read / write / execute

Control
permit / deny
Policy Example (1)

```xml
<gvmcf:AccessProtection gvmac:Default="Permit"
  gvmac:RuleCombiningAlgorithm="Permit-overrides">

<!-- Access Rule 1: for all user -->
<gvmcf:AccessRule gvmac:Effect="Deny">
  <gvmcf:AppliedTo> <gvmac:Subjects> ... </gvmac:Subjects>
  <gvmac:Resources>
    <gvmac:Resource>/etc/passwd</gvmac:Resource>
  </gvmac:Resources>
  <gvmac:Actions> ... </gvmac:Actions>
</gvmcf:AccessRule>

<!-- Access Rule 2: for a specific user -->
<gvmcf:AccessRule gvmac:Effect="Permit">
  <gvmcf:AppliedTo gvmcf:TargetUnit="user">
    <gvmcf:Subjects> <gvmcf:Subject>User1</gvmcf:Subject> ... </gvmcf:Subjects>
  </gvmcf:AppliedTo>
  <gvmac:Resources>
    <gvmac:Resource>/etc/passwd</gvmac:Resource>
  </gvmac:Resources>
  <gvmac:Actions>
    <gvmac:Action>read</gvmac:Action>
  </gvmac:Actions>
</gvmcf:AccessRule>
```

Default

Applying rules
Policy Example (2)

```xml
<gvcmcf:AccessRule gvmcf:Effect="Permit">
  <gvcmcf:AppliedTo gvmcf:TargetUnit="vo">
    <gvcmcf:Subjects>
      <gvcmcf:Subject>bio</gvcmcf:Subject>
    </gvcmcf:Subjects>
  </gvcmcf:AppliedTo>
  <gvmac:Resources>
    <gvmac:Resource>/opt/bio/bin</gvmac:Resource>
    <gvmac:Resource>./apps</gvmac:Resource>
  </gvmac:Resources>
  <gvmac:Actions>
    <gvmac:Action>read</gvmac:Action>
    <gvmac:Action>execute</gvmac:Action>
  </gvmac:Actions>
</gvcmcf:AccessRule>
```
NAREGI Data Grid beta1 Architecture (WP4)

Grid Workflow

Job 1 → Job 2 → ... → Job n

Data Grid Components

- Data Access Management
- Metadata Management
- Data Resource Management

Import data into workflow → Data 1, Data 2, Data n →
Place & register data on the Grid

Assign metadata to data

Grid-wide Data Sharing Service

Data 1, Data 2, ..., Data n

Grid-wide File System

Currently GFarm v.1.x
VO-based Resource Mapping in Global File System (β2)

- Next release of Gfarm (version 2.0) will have access control functionality.
- We will extend Gfarm metadata server for the data-resource mapping based on VO.
Next Steps for WP1 - Beta2

- Stability, Robustness, Ease-of-install
- Standard-setting core OGSA-EMS: OGSA-RSS, OGSA-BES/ESI, etc.
- More supported platforms (VM)
  - SX series, Solaris 8-10, etc.
  - More batchQs - NQS, n1ge, Condor, Torque
- “Orthogonalization” of SS, VM, IS, WSRF components
  - Better, more orthogonal WSRF-APIs, minimize sharing of states
    - E.g., reservation APIs, event-based notification
  - Mix-and-match of multiple SS/VM/IS/external components, many benefits
    - Robustness
    - Better and realistic Center VO support
    - Better interoperability with external grid MW stack, e.g. Condor-C
Users describe
  • Workflow being executed with abstract requirement of sub jobs.

NAREGI M/W
  • discovers fitting grid resources,
  • allocates resources executes on virtualized environments
  • as a ref. impl. of OGSA-EMS.
  • Programming Model:
    GridMPI & Grid RPC
  • User Interface:
    Deployment, Visualization, WFT
  • Grid File System is supported.
  • VOMS based VO management.

under evaluation on NAREGI testbed.
  • \(~15TF\)
  • SuperSINET
From Interoperation to Interoperability
GGF16 “Grid Interoperations Now”

Charlie Catlett
Director, NSF TeraGrid
on Vacation

Satoshi Matsuoka
Sub Project Director, NAREGI Project
Tokyo Institute of Technology / NII
The Reality: Convergence/Divergence of Project Forces

(original slide by Stephen Pickles, edited by Satoshi Matsuoka)

NGS(UK)

TeraGrid(US)

OSG(US)

EGEE(EU)

OMII(UK)

CSI (JP)

AIST-GTRC

DEISA(EU)

Globus(US)

Unicore

UniGrids(EU)

APAC Grid

EU-China Grid (China)
gLite / GT2

LCP(EU)

GT4 WSRF

NMI(US)

GT4/Fujitsu WSRF

GFF

Condor(US)

TeraGrid(US)

WSRF & OGSA

common staff & procedures

NAREGI (JP)

interoperable infrastructure talks

common users

Condor(US)

NMI(US)

APAC Grid (Australia)

GT4 WSRF (OGSA?)
Interoperation Activities

• The GGF GIN (Grid Interoperations Now) effort
  - Real interoperation between major Grid projects
  - Four interoperation areas identified
    • Security, Data Mgmt, Information Service, Job Submission (not scheduling)
GGF Grid Interoperation Now

- Started Nov. 17 2005 @SC05 by Catlett and Matsuoka
  - Now participation by all major grid projects
- “Agreeing to Agree on what needs to be Agreed first”
- Identified 4 Essential Key Common Services
  - Authentication, Authorization, Identity Management
    - Individuals, communities (VO’s)
  - Jobs: submission, auditing, tracking
    - Job submission interface, job description language, etc.
  - Data Management
    - Data movement, remote access, filesystems, metadata mgmt
  - Resource discovery and Information Service
    - Resourche description schema, information services
“Interoperation” versus “Interoperability”

• **Interoperability**
  “The ability of software and hardware on multiple machines from multiple vendors to communicate”
  - Based on commonly agreed documented specifications and procedures

• **Interoperation**
  “Just make it work together”
  - Whatever it takes, could be ad-hoc, undocumented, fragile
  - Low hanging fruit, future interoperability
Interoperation Status

- GIN meetings GGF16 and GGF17
- 3-day meeting at CERN end of March
- Security
  - Common VOMS/GSI infrastructure
  - NAREGI more complicated use of GSI/Myproxy and proxy delegation but should be OK
- Data
  - SRM commonality and data catalog integration
  - GFarm and DCache consolidation
- Information Service
  - CIM vs. GLUE schema differences
  - Monitoring system differences fairly
  - Schema translation (see next slides)
- Job Submission
  - JDL vs. JSDL, Condor-C/CE vs. OGSA SS/SC-VM architectural differences, etc.
  - Simple job submission only (see next slides)
Information Service Characteristics

• Basic syntax:
  - Resource description schemas (e.g., GLUE, CIM)
  - Data representations (e.g., XML, LDIF)
  - Query languages (e.g., SQL, XPath)
  - Client query interfaces
    (e.g., WS Resource Properties queries, LDAP, OGSA-DAI)

• Semantics:
  - What pieces of data are needed by each Grid
    (various previous works & actual deployment experiences already)

• Implementation:
  - Information service software systems (e.g., MDS, BDII)
  - The ultimate sources of this information (e.g., PBS, Condor, Ganglia, WS-GRAM, GridVM, various grid monitoring systems, etc.).
• An implementation of the GGF Grid Monitoring Architecture (GMA)
• All data modelled as tables: a single schema gives the impression of one virtual database for VO
## Syntax Interoperability Matrix

<table>
<thead>
<tr>
<th>Grid</th>
<th>Schema</th>
<th>Data</th>
<th>Query Lang</th>
<th>Client IF</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tera-Grid</td>
<td>GLUE</td>
<td>XML</td>
<td>XPath</td>
<td>WSRF RP Queries</td>
<td>MDS4</td>
</tr>
<tr>
<td>OSG</td>
<td>GLUE</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>BDII</td>
</tr>
<tr>
<td>NAREGI</td>
<td>CIM 2.10+ext</td>
<td>Relational</td>
<td>SQL</td>
<td>OGSA-DAI WS-I RUS</td>
<td>CIMOM + OGSA-DAI</td>
</tr>
<tr>
<td>EGEE/LCG</td>
<td>GLUE</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>BDII</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NorduGrid</td>
<td>ARC</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>GIIS</td>
</tr>
</tbody>
</table>
Low Hanging Fruit

“Just make it work by GLUEing”

- Identify the minimum common set of information required for interoperation in the respective information service
- Employ GLUE and extended CIM as the base schema for respective grids
- Each Info service in grid acts as a information provider for the other
- Embed schema translator to perform schema conversion
- Present data in a common fashion on each grid; WebMDS, NAREGI CIM Viewer, SCMSWeb, ...


Minimal Common Attributes

- Define minimal common set of attributes required
- Each system components in the grid will only access the translated information
GLUE→CIM translation

- Development of information providers with translation from GLUE data model to CIM about selected common attributes such as up/down status of grid services.

NAREGI

Cell Domain Information Service for EGEE resources

OGSA-DAI

RDB

Aggregator Service

Lightweight CIMOM

System

CIM provider skeleton

GLUE→CIM translator

GLUE-CIM mapping; selected Minimal Attributes

OGSA-DAI

GLUE

→

CIM

producer

G-Lite / R-GMA

Publish Tuples

SQL “CREATE TABLE”

SQL “INSERT”

OGSA-DAI

CDIS for NAREGI

CDIS for NAREGI

Multi-Grid Information Service Node

Information Service Node

Schema

Registry Service

Mediator

Register

Query

Tuples

Locate

Tuples

Send Query

Receive Tuples

Producer Service

Consumer Service

Send Query

Receive Tuples

Tuples

GLUE

→

CIM

translator

CIM provider

→

GLUE

translator

Query

SQL “SELECT”

SQL “SELECT”
Interoperability: NAREGI Short Term Policy

- **gLite**
  - Simple/Single Job (up to SPMD)
  - Bi-Directional Submission
    - NAREGI \(\rightarrow\) gLite: GT2-GRAM
    - gLite \(\rightarrow\) NAREGI: Condor-C
  - Exchange Resource Information

- **GIN**
  - Simple/Single Job (up to SPMD)
  - NAREGI \(\rightarrow\) GIN Submission
  - WS-GRAM
  - Exchange Resource Information

- **BES/ESI**
  - TBD
    - Status somewhat Confusing
    - ESI middleware already developed?
      - Globus 4.X and/or UnicoreGS?
### Job Submission Standards

#### Comparison: Goals

<table>
<thead>
<tr>
<th>Feature</th>
<th>ESI</th>
<th>BES</th>
<th>NAREGI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>SS</td>
</tr>
<tr>
<td>Use JSDL</td>
<td>✔  *1</td>
<td>✔</td>
<td>✔  *1</td>
</tr>
<tr>
<td>WSRF OGSA Base Profile 1.0 Platform</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Job Management Service</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Extensible Support for Resource Models</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Reliability</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Use WS-RF modeling conventions</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Use WS-Agreement</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Advance reservation</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Bulk operations</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Generic management frameworks (WSDM)</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Define alternative renderings</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Server-side workflow</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

*1: Extended
### Job Factory Operations

<table>
<thead>
<tr>
<th></th>
<th>ESI (0.6)</th>
<th>BES (Draft v16)</th>
<th>NAREGI (β)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original</strong></td>
<td>CreateManagedJob (there is a subscribe option)</td>
<td>CreateActivityFromJSDL GetActivityStatus RequestActivityStateChanges StopAcceptingNewActivities StartAcceptingNewActivities IsAcceptingNewActivities GetActivityJSDLDocuments</td>
<td>MakeReservations CommitReservations</td>
</tr>
<tr>
<td><strong>WS-ResourceLifeTime</strong></td>
<td>ImmediateResourceDestruction ScheduledResourceDestruction</td>
<td></td>
<td>Destroy SetTerminationTime</td>
</tr>
<tr>
<td><strong>WS-BaseNotification</strong></td>
<td>NotificationProducer</td>
<td></td>
<td>Notify Subscribe</td>
</tr>
</tbody>
</table>

Now working with Dave Snelling et. al. to converge to ESI-API (which is similar to WS-GRAM), plus
* Advance Reservation
* Bulk submission
Interoperation: gLite-NAREGI

- gLite-WMS [JDL]
- gLite-CE
- GT2-GRAM
- Condor-C
- gLite-IS [GLUE]
- IS bridge
- NAREGI-Portal
- NAREGI-SC
- NAREGI-SS [JSDL]
- NAREGI-WF generation
- Job Submit./Ctrl.
- Status propagation
- Certification?
- NAREGI-IS [CIM]
- Interop-SC
- NAREGI GridVM
- NAREGI user
- gLite user
Interoperation: GIN (Short Term)

anotherGrid-IS
[CIM or GLUE]  IS bridge  NAREGI-IS
[CIM]

• GLUE ↔ CIM

anotherGrid
GT4

anotherGrid-IS

NAREGI-SS
[JSIDL]  Intergo-SC  NAREGI-SC

• JSDL → RSL
• Job Submit./Ctrl.
• Status propagation

WS-GRAM
anotherGrid-CE

WS-GRAM
NAREGI GridVM

Condor-G

NAREGI Portal

another grid user

NAREGI user
Approach: “Bait and Catch”
--- Typical Japanese National Project Strategy ---

- IT vendors involved (seriously) from Day 1
- Public entity project benefactors should *not* solely expend its grants to sustain itself
  - Most money going to IT vendors to have them commit
- *OTOH*, vendors are later pressured by the government not to make *direct* profit
- (Exclusive IPR acquisition or) standardization used as bait for commercial adoption
  - Thus, even a research grid infrastructural project *MUST* FOCUS on standards, above all other priorities
- Then, standards & supporting technologies are DIRECTLY transferred to IT vendors
  - *Productization and support* => *longevity* => *adoption*!
A Picture is Worth a 1000 Words...
TSUBAME as No.1 in Japan

>85 TeraFlops
1.1 Petabyte
4 year procurement cycle
Has beaten the Earth Simulator
Has beaten all the other Univ. centers combined

Total 45 TeraFlops,
350 Terabytes
<table>
<thead>
<tr>
<th>Grid</th>
<th>Schema</th>
<th>Data</th>
<th>Query Lang</th>
<th>Client IF</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tera-Grid</td>
<td>GLUE</td>
<td>XML</td>
<td>XPath</td>
<td>WSRF RP Queries</td>
<td>MDS4</td>
</tr>
<tr>
<td>OSG</td>
<td>GLUE</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>BDII</td>
</tr>
<tr>
<td>NAREGI</td>
<td>CIM 2.10+ext</td>
<td>Relational</td>
<td>SQL</td>
<td>OGSA-DAI WS-I RUS</td>
<td>CIMOM + OGSA-DAI</td>
</tr>
<tr>
<td>EGEE/LCG</td>
<td>GLUE</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>BDII</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relational</td>
<td>SQL</td>
<td>R-GMA i/f</td>
<td>R-GMA</td>
</tr>
<tr>
<td>Nordu Grid</td>
<td>ARC</td>
<td>LDIF</td>
<td>LDAP</td>
<td>LDAP</td>
<td>GIIS</td>
</tr>
</tbody>
</table>
GLUE → CIM translation

- Development of information providers with translation from GLUE data model to CIM about selected common attributes such as up/down status of grid services.

GLUE-CIM mapping; selected Minimal Attributes

- SQL "SELECT"
NAREGI beta on "VM Grid"

Create “Grid-on-Demand” environment using Xen and Globus Workspace

Vanilla personal virtual grid/cluster using our Titech Lab’s research results
"VM Grid" – Prototype “S”

- Request # of virtual grid nodes
- Fill in the necessary info in the form
- Confirmation page appears, follow instructions
- Ssh login to NMI stack (GT2+Condor) + selected NAREGI beta MW (Ninf-G, etc.)
- Entire Beta installation in the works
- Other “Instant Grid” research in the works in the lab
CyberScience Infrastructure for Advanced Science (by NII)

To Innovate Academia and Industry

CyberScience Infrastructure

Scientific Repository

Virtual Organization For science

NAREGI Middleware

UPKI

Human Resource Development and strong organization

Super-sinet: a next generation network infrastructure supported by NII and 7 National Computer Centers

Publication of scientific results from academina

Industry Liaison and Social Benefit

Global Contribution

Super-sinet: a next generation network infrastructure supported by NII and 7 National Computer Centers