NAREGI WP1 Presentation

The NAREGI Server Grid
Resource Management Framework

Satoshi Matsuoka
Tokyo Institute of Technology/NII
July 21, 2004
NAREGI WP1 Objectives

• Mid-range Project Target
  - R&D on scalable middleware infrastructure for server grids and metacomputing resource management, deployable at large computing centers, based on Unicore, Globus, Condor

• Final Project Target
  - R&D Build resource management framework and middleware for VO hosting by the centers based on the OGSA standard
  - Distribute result as high-quality open source software
Server Grids and Metacomputing

Various (existing) applications and their workflow

Automated Allocation of resources and Scheduling of jobs

Distributed Servers

Metascheduler/Broker & Workflow Engine

Metacomputing

National Research Grid Initiative
The Future: VO Hosting by Grid Centers

“Hosting” of Various VOs for Research Areas, Research Groups, etc. by a federation of centers

Dynamic provisioning of large resources to VOs

Various dedicated project machines

NanoGrid VO

Lab X Univ A

Lab Y Univ A

Grid Center Univ A

Research Lab B

Grid Center

Univ A

Lab V Univ C

Company D Division U

Project X VO
NAREGI Software Stack

- Grid Enabled Nano-Applications
  - Grid Visualization
  - Grid PSE
  - Grid Workflow
  - Super Scheduler
  - Distributed Information Service

- Grid Programming
  - Grid RPC
  - Grid MPI

- Grid VM

- High-Performance & Secure Grid Networking

- SuperSINET

- NII
- IMS
- Research Organizations
- etc

Computing Resources

National Research Grid Initiative
R&D in Grid Software and Networking Area (Work Packages)

- WP-1: Lower and Middle-Tier Middleware for Resource Management:
  - Matsuoka(Titech), Kohno(ECU), Aida(Titech)
- WP-2: Grid Programming Middleware:
  - Sekiguchi(AIST), Ishikawa(AIST)
- WP-3: User-Level Grid Tools & PSE:
  - Miura(NII), Sato(Tsukuba-u), Kawata(Utsunomiya-u)
- WP-4: Packaging and Configuration Management:
  - Miura(NII)
- WP-5: Networking, Security & User Management
  - Shimojo(Osaka-u), Oie(Kyushu Tech.)
- WP-6: Grid-enabling tools for Nanoscience Applications:
  - Aoyagi(Kyushu-u)
WP-1: Lower and Middle-Tier Middleware for Resource Management

• **Unicore ↔ Condor ↔ Globus Interoperability** (UniCondore)

• **(Scalable) MetaScheduler**
  – Schedule large metacomputing jobs
  – “Scalable”, Agreement-based scheduling
  – Assume preemptive metascheduled jobs

• **(Scalable) Grid Information Service**
  – Support multiple monitoring systems
  – User and job auditing, accounting
  – CIM-based node information schema

• **GridVM (Lightweight Grid Virtual Machine)**
  – Metacomputing Support
  – Enforcing Authorization Policies, Sandbox
  – Checkpointing/FT/Preemption
  – Cluster Node virtualization
WP1 SuperScheduler (Fujitsu)

- Meta-computing
- MPI Job
- Site A
  - Gateway
  - NJS
    - GridVM sched
    - GridVM Engine
  - JobA-1 64CPU
  - JobA-2 64CPU

- Site B
  - Gateway
  - NJS
    - GridVM sched
    - GridVM Engine
    - Condor

- Accounting Info
- CIMM Service
- Information Service
- ORDB
- Provider/Monitor

- JobA-1...SiteA
- JobA-2...SiteB

- GridMPI Comm
- GridVM Comm

- MPI Jobs

- Meta-computing
- MPI Job
- Site A

- Gateway
- NJS
  - GridVM sched
  - GridVM Engine
  - JobA-1 64CPU
  - JobA-2 64CPU

- Site B
  - Gateway
  - NJS
    - GridVM sched
    - GridVM Engine
    - Condor

- Accounting Info
- CIMM Service
- Information Service
- ORDB
- Provider/Monitor

- JobA-1...SiteA
- JobA-2...SiteB

- GridMPI Comm
- GridVM Comm

- MPI Jobs
NAREGI SuperScheduler Architecture

End User MPI Job A

JobA-1 3CPU
JobA-2 2CPU
Network BW :100Mbps

Client

CMM

Information Service

Gateway

Delegation

Gateway

-1 Co-allocation (WS-Agreement reservation)

-2 Co-allocation (reserve)

-3 100Mbps BW Reserv.

Network Mgmt Service (WP5)

MPI Comm

MPI Job (metacomputing)

Site A(NII)

Super SINET

Site B(IMS)

National Research Grid Initiative
FY 03-04 SuperScheduler R&D Results

- UNICORE Client
  - Workflow job execution
  - UNICORE Server (NJS)
  - UNICORE TSI

- WF Exec. Svc.
- Job Exec. Svc.
- PBS/Maui (modified)

- Brokering Svc.
- NAREGI Info Service
  - NAREGI Resource Schema
  - NAREGI Resource Broker
  - Grid Resource DB

- National Research Grid Initiative
Inter-site job execution using NAREGI super scheduler
WP1 Unicondore (Fujitsu)

Meta-computing
MPI Job
Site A

Client

Gateway

NJS

GridVM sched
Local Scheduler
GridVM Engine

JobA-1 64CPU
JobA-2 64CPU

Accounting Info

JobA-1...SiteA
JobA-2...SiteB

CIMM Service
Information Service

ORDB
Provider/ Monitor

Site B

NJS

GridVM sched
Local Scheduler
GridVM Engine

MPI Jobs

GridMPI Comm

GridVM Comm
What is “UNICONDORE”? 

- Seamless bridging of UNICORE and Condor for parameter-sweep-like metacomputing jobs

- **UNICORE-C**
  - Use Condor as a batch sub system of UNICORE
  - This development is mainly about TSI for Condor

- **Condor-U**
  - Use UNICORE for grid job submission of Condor
  - Extend Condor-G (Globus) function to UNICORE
UNICONDORE Overview

UNICORE-C

UNICORE(client)

UNICORE(pro Client)

UNICORE(server)

UNICORE Gateway

UNICORE NJS

UNICORE IDB/Condor

UNICORE TSI/Condor

Condor

Condor-U

Condor

condor_submit

condor_q

condor_rm

Condor-G

Condor-G Grid Manager

Ga4p Protocol for UNICORE

Ga4p Server for UNICORE

UNICORE

National Research Grid Initiative
Demonstration (UNICORE-C)

Submit Condor Jobs from UNICORE Client, which will appear as a Condor Job.

(1) Submit Job

(2) Confirm status on Condor Pool
Condor-U Overview

Condor Submit Machine (Grid Universe)

- User
- Grid Manager

UNI CORE GAHP

- UNI CORE GAHP Server

UNI CORE WORLD

UNI CORE Protocol

Gateway

- NJS
- TSI
  - Batch Subsystem
  - Job

Vsite

Usite

National Research Grid Initiative
WP1 Information Service (Hitachi)

Meta-computing MPI Job
Site A

JobA-1 64CPU
JobA-2 64CPU

Client

Gateway

NJS

GridVM sched
Local Scheduler
GridVM Engine

MPI Jobs

JobA-1…SiteA
JobA-2…SiteB

Accounting Info

CIMM Service

Information Service

ORDB

Provider/ Monitor

Condor

Site B

NJS

GridVM sched
Local Scheduler
GridVM Engine

National Research Grid Initiative
Overview of NAREGI Grid Information Service

- Viewer
- Client (Resource Broker etc.)
- Sample Client
- GridVM (Chargeable Service)
- Information Service
- CIM Query Service
- RDB
- DAI
- Decode Service
- Aggregate Service
- Light-weight CIMOM Service
- CIM Providers
- Distributed Query ...
- … Hierarchical filtered aggregation
- … Redundancy for FT

Java-API
APP
(GridVM (G)WSDL (RUS::insert)
(GridVM (G)WSDL (SQL query)

User Admin.
R&D in 2003

Super Scheduler
Admin viewer

Search, Set, Notify

CIM Operation, SQL Query

CIM2GIS

GT3 IndexService

PostgreSQL

Budget Mgmt

CIMOM (Pegasus)

Service Data Provider

CIM Providers

Resource
User
Account
Policy

Performance
Budget
Log

NaReGI Schema

Account Mapping Service

Account application

CIM World
OGSI World

CIM World

OGSI World
Information Model based on CIM Schema

Example of Association class:
Schema for JobQueue
Schema for Usage Record
(1) Resource information

A Grid user obtains information regarding Grid nodes where he has the access rights.

(2) Accounting information

A Grid user obtains Resource Usage information on Grid nodes where he has the access rights.

(3) Log analysis

A System administrator searches Logs under some specific conditions; {Category, Severity, Source, Time stamp, Message}.
WP1 GridVM (NEC/Titech)

Meta-computing
MPI Job
Site A

Site B

JobA-1 64CPU
JobA-2 64CPU

Client

Gateway

NJS

GridVM sched
Local Scheduler
GridVM Engine

MPI Jobs

GridMPI Comm

JobA-1…SiteA
JobA-2…SiteB

Accounting Info

CIMM Service

Information Service

ORDB

Provider/ Monitor

Condor

GridVM Comm

GridVM sched
Local Scheduler
GridVM Engine

Meta-computing
MPI Job
Site A

Site B

JobA-1 64CPU
JobA-2 64CPU

Client

Gateway

NJS

SS ブローカー

NJS

Gateway

GridVM sched
Local Scheduler
GridVM Engine

MPI Jobs

GridMPI Comm

JobA-1…SiteA
JobA-2…SiteB

Accounting Info

CIMM Service

Information Service

ORDB

Provider/ Monitor

Condor

GridVM Comm

GridVM sched
Local Scheduler
GridVM Engine

National Research Grid Initiative
Lack of Enforcement in Grid Middleware

- Problems in Current Grid middleware
  - Only authentication enforced
    - E.g. CAS authorizes resource usage but enforcement is up to each application
  - Lack of central user administration and unified execution environment
    - Anonymous user e.g., desktop Grid
    - Burden on administration
  - Lack of Grid-aware resource control by the underlying OS
    - Many resources i.e. are either unmanaged and/or quota on machine basis
    - No enforced I/O resource control at Grid level
GridVM Overview

- **Virtual Machine Layer Tailored for the Grid**
- Provides unified and secure execution environment for Grid applications
- Copes with underlying OS/HW heterogeneity

**Upper-level Grid Middleware**

**Grid Service**

**Metacomputing**
Co-scheduling, Gang Scheduling and other inter-node synch. Coallocation reservation support

**Security**
Fine-Grain Access Control
Authorization Enforcement
Sandboxed Execution

**Fault Tolerance**
Checkpointing, Fault Detection/Prevention/Injection

**Virtualizing underlying OS/HW**

**Cluster Software**
Local Scheduler

Node OS, e.g., Linux

CPU NW Mem HDD

National Research Grid Initiative
Prototype GridVM

- Auth. Enforcement and Sandboxing
- Resource Control and Virtualization
  - Virtual File System
  - Network/Disk Bandwidth
  - CPU/Memory Utilization
- Able to Select from multiple virtualization scheme/policies
  - Grid UID, site, etc.
  - Prototype on Linux 2.4
- Virtualization of multiple parallel processes
  - Forked process
  - MPI jobs across nodes
Prototype VM : Virtualization Schemes

- **Ptrace (System Call Virtualization)**
  - A system call to trap system calls
  - Traps *ALL* system calls, not just necessary ones
  - (Linux) can only modify 1 word per call

- **mod_janus (System Call Virtualization)**
  - Kernel module in janus [Goldberg 96]
  - Can selectively trap system calls
  - Need admin privilege on load

- **DyninstAPI [Buck et.al. 00] (Library Call Virtualization)**
  - Insert library interception calls dynamically into binaries
  - May have trouble with optimized binaries
## Guidelines for Virtualization Scheme Selection

<table>
<thead>
<tr>
<th></th>
<th>Functional</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Within User Previdged</strong></td>
<td><strong>&quot;Complete&quot; Virtualization</strong></td>
<td><strong>Suppress System Calls</strong></td>
</tr>
<tr>
<td>ptrace</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>mod_janus</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>dyninst</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Virtualizing MPI Jobs

• A (HPC) Grid job is typically MPI
  - Typically Forks remote process with rsh/ssh (e.g., MPICH)
  
  We must virtualized the remote MPI processes

• Remote process virtualization schemes
  - Create VM shell and make it default
  - Modify rsh parameters on execve
  - Modify rsh parameters within MPI library

• MPI Library mod chosen for Prototype GridVM
Performance Evaluation: MPI Job Under GridVM (With Bandwidth Control)

- Evaluation Env.
  - NPB MG Benchmark
  - 4 nodes 4 processes
  - Control Bandwidth
- Bandwidth Control does (properly) affect performance
- Per system call penalty affects performance

mod_janus > ptrace > DyninstAPI
(System Call Virtualization) (Library Virtualization)
Connecting NJS and GridVM

- Replace TSI with GridVM
  (For the time being, TSI will be used for file staging ...)
- GridVM Java API for local NJS job management
GridVM Resource Reservation

- **Making reservations** ("makeReservation" method)
  - Makes a reservation with "JSDL" as input
  - NJS assigns "SubjobID" in JSDL used for subsequent operations
  - Properties that are reserved are:
    - Start time, End time, Number of nodes, Node names (if necessary)

- **Cancel Reservation** ("cancelReservation" method)
  - Cancels a specified reservation

- **Query Reservation** ("queryReservation" method)
  - Queries information about specified reservation
  - Returns a XML including the following information
    - Start time, End time, Number of nodes, Node names (if necessary)
## Sample NAREGI JSDL Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Value</th>
<th>Note about JSDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>/naregi-jsdl:SubJobID</td>
<td>Job ID assigned by NJS</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>/naregi-jsdl:CPUCount</td>
<td>Number of needed CPUs</td>
<td>8</td>
<td>Deleted from 0.4.3</td>
</tr>
<tr>
<td>/naregi-jsdl:TasksPerHost</td>
<td>Number of Tasks per host</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>/naregi-jsdl:TotalTasks</td>
<td>Number of MPI tasks</td>
<td>8</td>
<td>N/A</td>
</tr>
<tr>
<td>/naregi-jsdl:NumberOfNodes</td>
<td>Number of Nodes used</td>
<td>8</td>
<td>N/A</td>
</tr>
<tr>
<td>/naregi-jsdl:CheckpointablePeriod</td>
<td>Interval for checkpoint</td>
<td>100</td>
<td>N/A</td>
</tr>
<tr>
<td>/jsdl:PhysicalMemory</td>
<td>Amount of needed Physical Memory</td>
<td>1GB</td>
<td>JSDL 0.4.3</td>
</tr>
<tr>
<td>/jsdl:ProcessVirtualMemoryLimit</td>
<td>Amount of needed Virtual Memory</td>
<td>100GB</td>
<td>JSDL 0.4.3</td>
</tr>
<tr>
<td>/naregi-jsdl:JobStartTrigger</td>
<td>Reserved job start time</td>
<td></td>
<td>Deleted from 0.4.3</td>
</tr>
<tr>
<td>/jsdl:WallTimeLimit</td>
<td>Max amount of wall time required</td>
<td>100h</td>
<td>JSDL 0.4.3</td>
</tr>
<tr>
<td>/jsdl:CPUTimeLimit</td>
<td>Max amount of cpu time required</td>
<td>200h</td>
<td>JSDL 0.4.3</td>
</tr>
<tr>
<td>/jsdl:FileSizeLimit</td>
<td>Max file size created</td>
<td>10GB</td>
<td>JSDL 0.4.3</td>
</tr>
<tr>
<td>/jsdl:Queue</td>
<td>Name of queue to which job is submitted</td>
<td>grid_01</td>
<td>JSDL 0.4.3</td>
</tr>
</tbody>
</table>
co-scheduling with synchronous startup and suspension of a job running across multiple sites

**Synchronized start**

- While **Job 1** is running on site A, **Job 2**, which runs across site A and B, is submitted.

- **Job 2** is invoked simultaneously on site A and B after **Job 1** is done.

- If **Job 3** is submitted onto site B during the execution of **Job 1** (and small enough), then it gets started prior to **Job 2**.
GridVM Demo System Environment

- GlobusToolkit3
- GridVM
- Scheduler
- SCore (Server Host)
- Job program
- SCore (Execution Host)

Client

Site A
- Cluster Server
- Execution Node
- Execution Node

Site B
- Cluster Server
- Execution Node
- Execution Node

NATIONAL RESEARCH GRID INITIATIVE
The diagram illustrates a GridVM Demo Screenshot. It contains information on:

- **Host name**: kiwi, grigri
- **Job name**: job1, job2, job3
- **Elapsed time of Job**: 25 sec

The diagram also indicates that the elapsed time is displayed only when the `-showElapsed` option is specified. The time frame is specified by the `-length` option.
Autonomous Configuration of Grid Monitoring Systems

- **Grid Monitoring Systems**
  - Necessary for Effective Management and resource provisioning on the Grid
    - Subject: CPU load, networks, middleware & application status, user accounts, ...
  - Common Characteristics (e.g. GGF-GMA)
    - Multiple, interdependent components
    - Distributed all around the Grid
  - Requirements for Large-scale, production Grid Monitoring System deployment
    - (Automated) Distributed Component setup, dynamic reconfiguration, automated recovery
    - Efficient and scalable component management (# of components could be very large)

Need Grid Monitoring Systems with Autonomous Management
Autonomous Configuration of Grid Monitoring System (continued)

1. Overview of System

- A central management System for Network Weather Service (NWS) [Wolski et al.]
- Input: Node List (with attribute info)
- Forming node groups and configuring NWS component based on RTT
- Reconfiguration reflecting component dependencies

**Four Steps for management**
- Forecast network topology and check status of nodes and components
- Form node groups
- Decide configuration
- Start up components on assigned nodes

**DEMO**
- Configure NWS on the Campus Grid (Titech Grid) @ Tokyo Inst. Technology
- Node groups formation & assignment of the NWS component to the nodes

**Application example for the Grid environment**

- Data management components (Receive data from Sensor component & make them available)
- Initial configuration for all nodes Reconfiguration for node faults
- Reconfiguration reflecting component dependencies
- Directory Service component

**Representation of network performance measurement** (proposed by NWS)

**Commands**
Details of the Experiment

Evaluation of NWS-based prototype on a real production Grid (Titech Campus Grid) that consists of 16 cluster nodes and 800 CPUs.

Oo-okayama Campus

Suzukakedai Campus

Network Measurement between representing nodes

Component placement, groupings are performed in a “sensible” fashion.
Results of Evaluation

Setup time

Most time spent on RTT measurement and NWS invocation

Execution time is $O(N)(N: \# \text{ clusters})$ => need to parallelize measurement, setup
WP1 Towards VO-based Resource Mgmt.

Real World

VO-based Resource Model

Virtual Org.

Resource

User

Budget

Log

Performance

Account

Policy

Map & Translate

Resource

User

Budget

Log

Performance

Account

Policy

Authorized Operation

Resource Mgmt.

Service

Ex. Account Mapping

Change St.

V.O.

Management

Admin.

People, Networks, Computers, Storage, ...

User

Local Management System

Local Management System

Information provider

Monitor

Req

e.g. VO Hosting

Grid Initiative