Grid Applications – What are They?

Satoshi Matsuoka

Professor, Global Scientific Information and Computing Center,

Deputy Director, NAREGI Project

Tokyo Institute of Technology / NII

Preface Lecture For Grid School 2005

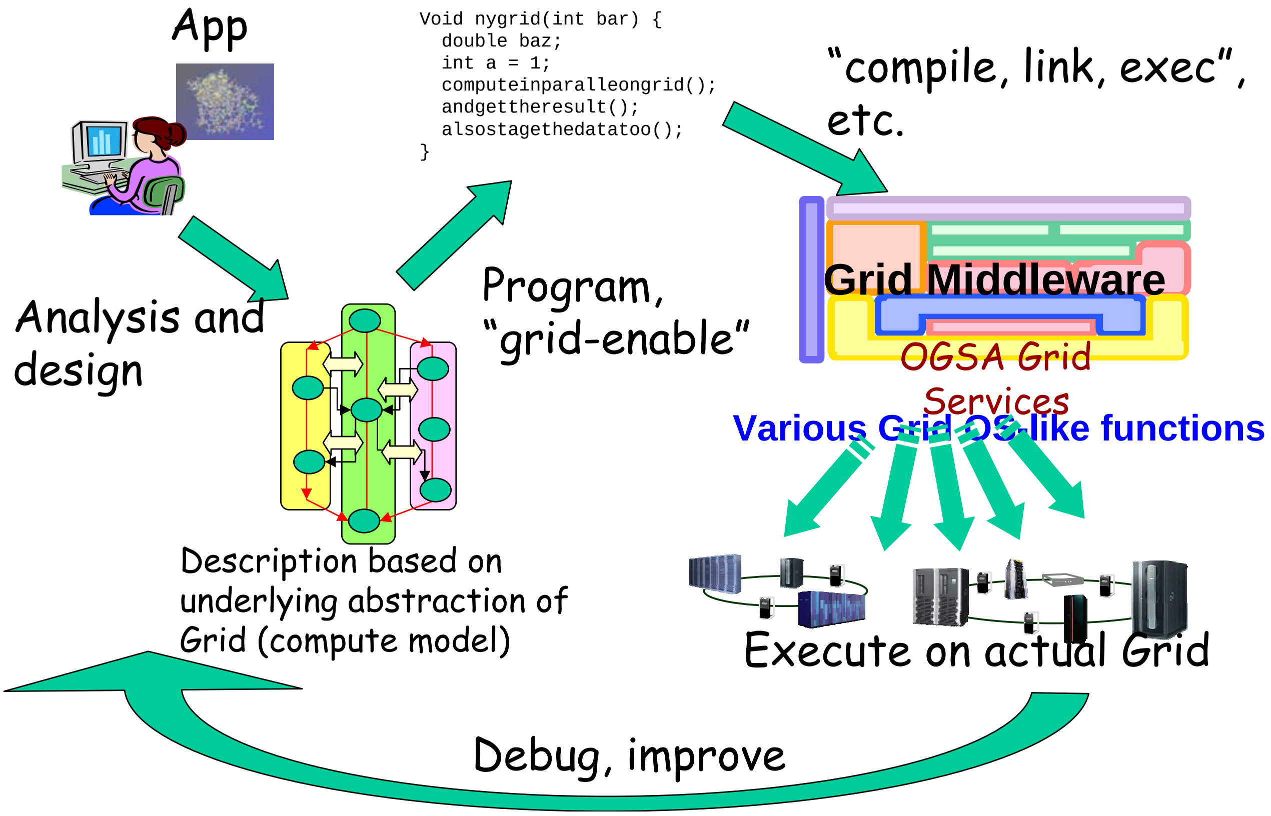
First, the Fundamental Questions

- Question1: What is a "Grid Application"?
- Question2: How do you "program" (and run) a "Grid Application"?

Slightly different version of the questions…

- Question1: What is a "Parallel Application"?
  - A user-level program that involves multiple (possibly communicating) independent threads of computation that run in parallel to output a consolidated result
- Question2: How do you "program" (and run) a "Parallel Application"?
  - Map some (implicitly or explicitly) parallel application algorithm onto some parallel computational model describable in terms of some parallel programming system (e.g., languages, frameworks, components, and/or libraries) embodying some higher-level parallel semantics/abstractions, and the underlying computer system will perform the appropriate compilation/translation into lower-level machine entities across multiple CPUs and their mutual communications

Grid Application Lifecycle



App

```
Void nygrid(int bar) {
  double baz;
  int a = 1;
  computeinparalleongrid();
  andgettheresult();
  alsostagethedatatoo();
}
```

"compile, link, exec", etc.

Analysis and design

Program, "grid-enable"

Grid Middleware

OGSA Grid Services

Various Grid OS-like functions

Description based on underlying abstraction of Grid (compute model)

Execute on actual Grid

Debug, improve

So again, the Fundamental Questions

- Question1: What is a “Grid Application”?
  - An application designed to effectively run in a heterogeneous, distributed, resource-sharing environment a.k.a. the Grid
- Question2: How do you “program” (and run) a “Grid Application”?
  - No Microsoft Visual Studio Grid Edition…
  - What is the underlying pinnings as a computing model?
  - What are the tools and methodologies available?

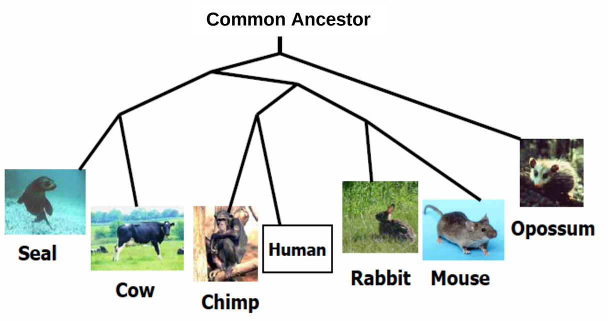
Desirable Property of Programming Apps on a Grid

- Programming System will always embody some model of computation and some higher level abstraction of the underlying system in the model, and a rigorous notion of how they interact
  - => A Grid programming system will present the programmer/user with some higher-level abstraction of the Grid…
  - But what are they?
  - C.f. MPI parallel programming, Internet XML programming…

Taxonomy of current "Grid Programming" Methodologies and their Grid Abstractions

- Complete Grid Transparency
  - Directly utilize (non grid-ware) existing apps
  - Grid is completely abstracted away
  - Middleware does the hard job, akin to parallelizing compilers
  - Parameter-Sweep, Workflow + staging services, etc.
- Component (Wrapper) Based
  - Wrap existing applications as independent Grid components
  - Expose their functionalities as Grid (Web) services
  - Assemble apps by tying together these services
  - Abstractions in terms of various Grid components and how their services are tied together and interact
- Grid Programming Languages, Libraries, etc.
  - Expose Grid Abstractions directly to application programs
  - Most powerful, but steeper learning curve
  - Must modify or recreate existing programs
  - Still a minority, but may become dominant?

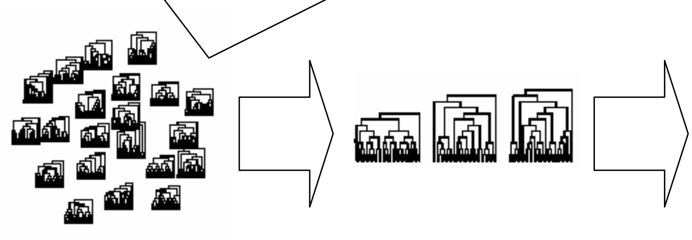
Complete Grid Transparency App: Phylogenetic Tree Inference on Condor/DAGMAN/our Steering Portal

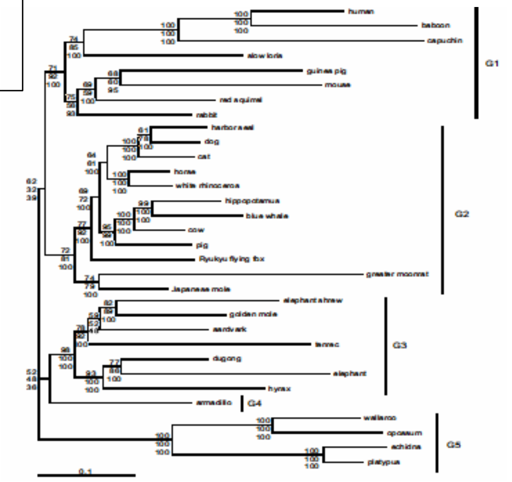


- Infer phylogenetic relationships between different species from their genomic sequences [Hasegawa&Shimodaira04]
- App Characteristics
  - Basically execute multiple parallel jobs in sequence
    => Workflow of batch jobs
  - But difficult to judge the termination condition of the application phases
    => Need human steering

Phylogenetic Tree Inference Breakdown

Narrow down on the candidate phylogenetic trees:
Hard to automate=>batch jobs difficult





Compute Posterior Probability

“MrBayes”

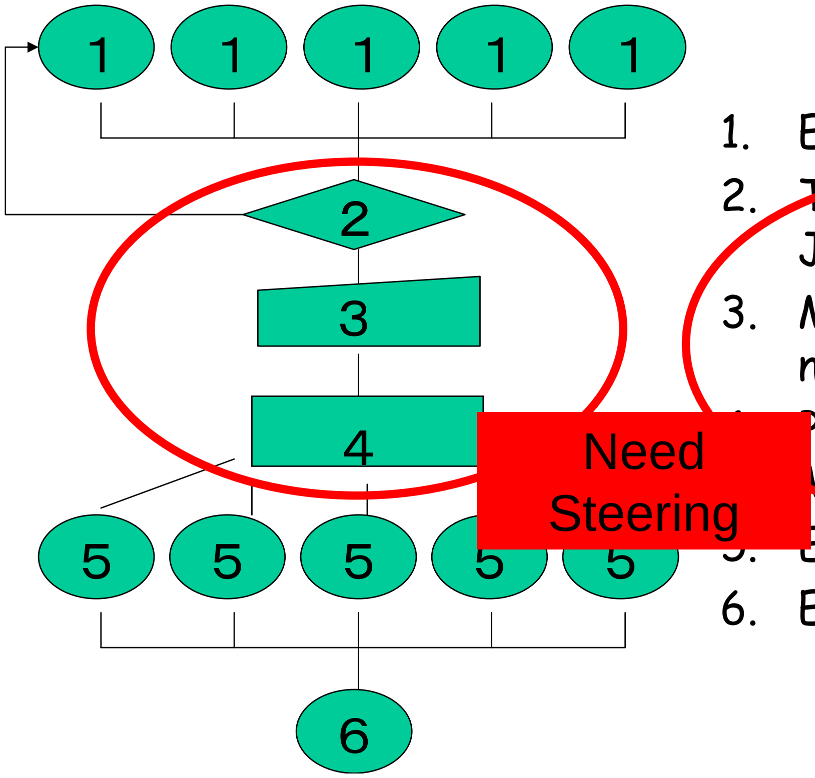
Compute likelihood value

“PAML”

Test
“CONSEL”

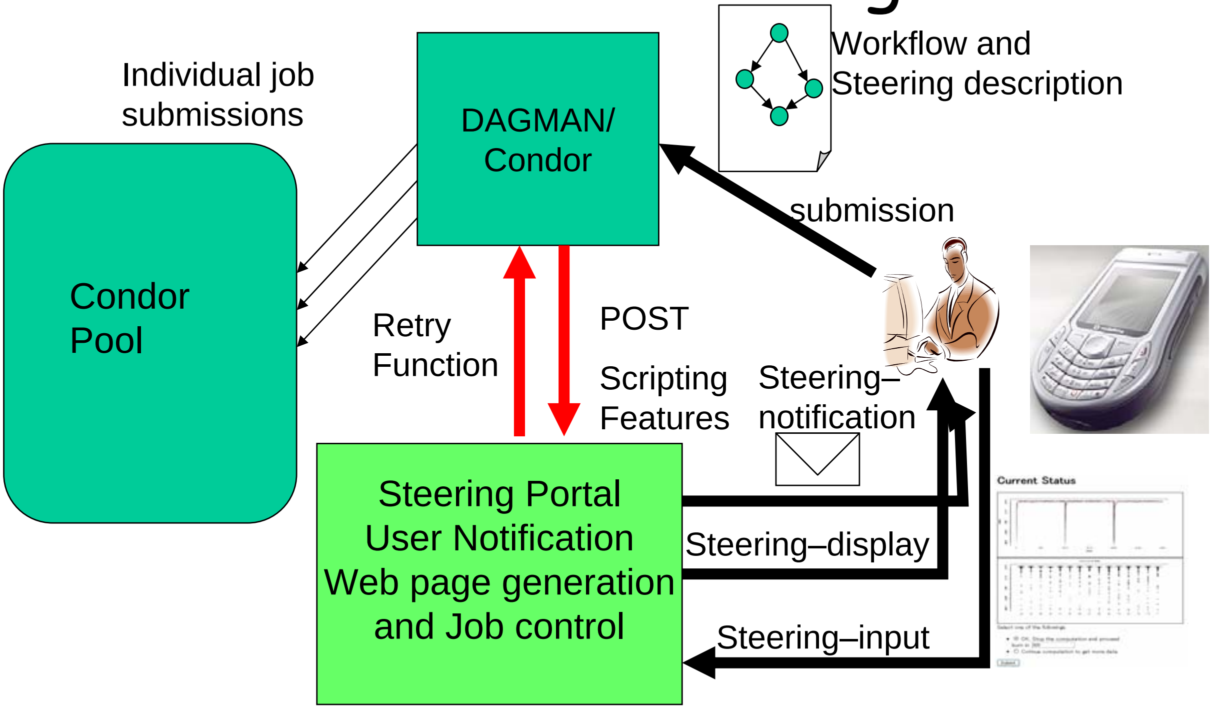
(These are all exisiting, grid unware apps)

The Actual Workflow



1. Exec. MrBayes
2. Termination Judgement
3. Manutal input of new parameters
4. Post-Process MrBayes
5. Execute PAML
6. Execute CONSEL

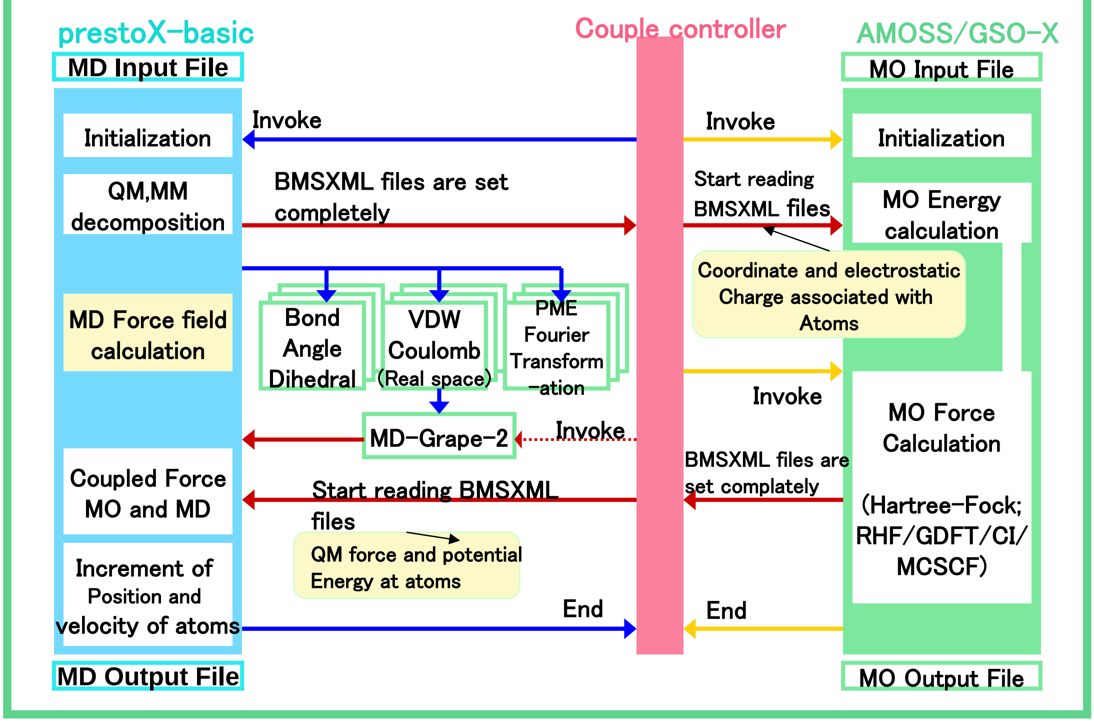
Overview of our Condor/DAGMAN Steering Portal

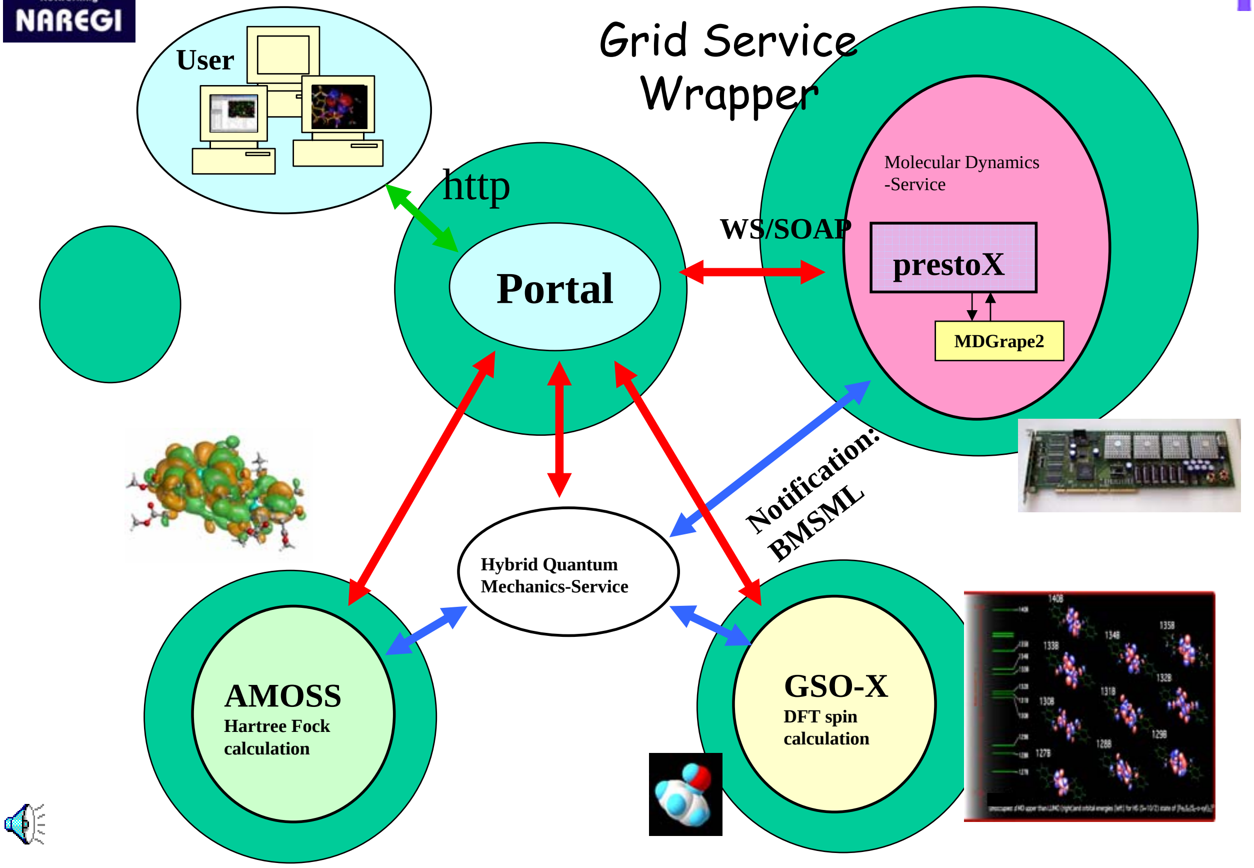


Component/Wrapper/Service based

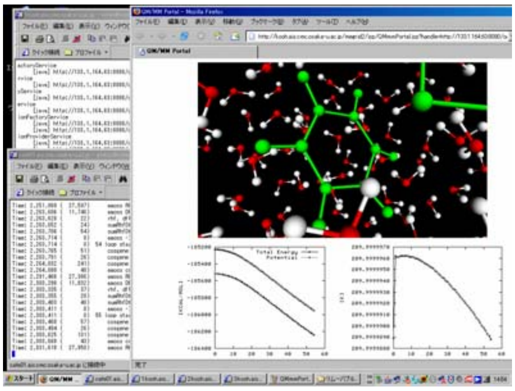
- (Example courtesy of Osaka University BioGrid Project)
- Want to perform multi-scale simulation based on interacting components
  - Hybrid QM/MM calculation
- Wrap exisiting apps as Grid components, and have their services interact to implement the necessary application interaction pattern

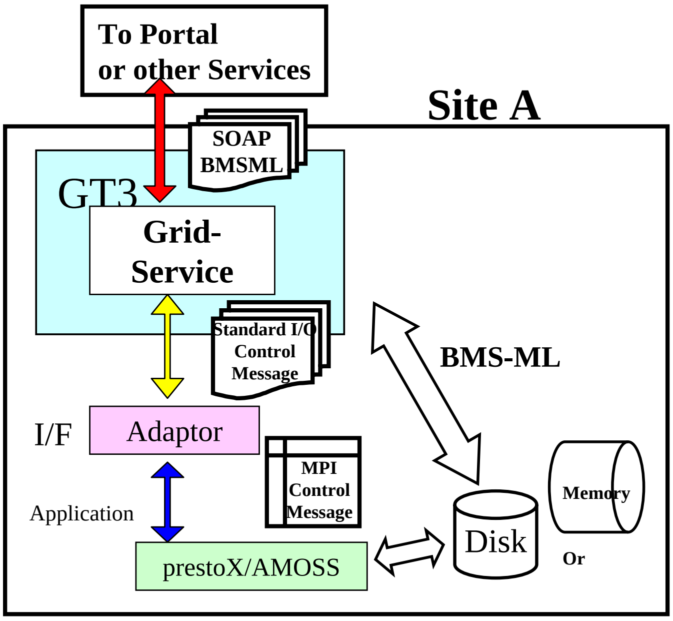
Interaction pattern of BioPfuga for prestoX-basic (MM) and AMOSS/GSO-X (QM)


prestoX-basic
MD Input File
Initialization
QM,MM decomposition
MD Force field calculation
Coupled Force MO and MD
Increment of Position and velocity of atoms
MD Output File
Invoke
BMSXML files are set completely
Bond Angle Dihedral
VDW Coulomb (Real space)
PME Fourier Transform -ation
MD-Grape-2
Invoke
Start reading BMSXML files
QM force and potential Energy at atoms
End
Couple controller
Invoke
Start reading BMSXML files
Coordinate and electrostatic Charge associated with Atoms
Invoke
BMSXML files are set complately
End
AMOSS/GSO-X
MO Input File
Initialization
MO Energy calculation
MO Force Calculation
(Hartree-Fock; RHF/GDFT/CI/ MCSCF)
MO Output File



Grid-Service wrapper details of QM/MM Calculation on *BioPfuga*





Grid Programming Languages, Libraries, etc.

- NAREGI
  - GridMPI (data parallel)
  - GridRPC (task parallel)
  - Mediator (coupled apps)
- EU Projects
  - GAT API (GAT, set of APIs of grid abstractions)
  - SAGA API (GGF-RG, similar to GAT)
  - ProActive (INRIA, concurrent objecgts)
- (Domain-specific) Programming Frameworks
  - Cactus
  - GridSAT
  - MW
  - …
- But no Visual Studio Grid Edition (yet).

Simple API for Grid Applications Research Group (GGF SAGA-RG)

- Chairs: Tom Goodale (CCT), Keith Jackson (LBL)
- Goals
  - The SAGA group is developing a high-level API which abstracts the underlying Grid middleware and allows grid application developers to grid-enable their applications easily, allowing them to make use of the distributed resources available to them without such a steep learning curve.
  - The SAGA API aims to be as easy and natural to use as possible, and available in a range of languages, such as Fortran, C, C++, Java, etc.
  - For details, refer to the website:
    - https://forge.gridforum.org/projects/saga-rg/

SAGA API

- Chose to represent abstract API design using an object oriented model
  - allows consistent concrete APIs across languages
  - easier to go from object-oriented description to a procedural language than from a procedural description to an object-based language
- Represented API in Scientific IDL (SIDL) to provide a language-neutral description.

Example: Jobs

- Jobs: Information/Status objects
  - JobDefinition
    - encapsulates all the attributes which define a job to be run.
  - JobInfo
    - encapsulates the state of an existing job.
  - JobExitStatus
    - holds the exit status of a finished job.
- Jobs: Interfaces
  - Job
    - provides the manageability interface to a job submitted to a resource manager
  - JobService
    - provides an interface for job creation and discovery
- Jobs: Usage
  - User constructs a JobDefinition
  - User passes JobDefinition to JobService to submit a job; is returned a Job object
  - User uses Job object to get information about the running/finished job and to manipulate it.

Jobs: SIDL

```
package SAGA version 0.1 {
package JobManagement {
   class JobDefinition implements-all SAGA.Attribute {
   /* This object encapsulates all the attributes which
    * define a job to be run. (Controlled by attributes interface.)
    */
   }
   class JobInfo implements-all SAGA.Attribute {
   /* This object encapsulates the state of an existing job.
    * (Controlled by attributes interface.)
    */
      getStdinStream(out opaque stdin);
      getStdoutStream(out opaque stdout);
      getStderrStream(out opaque stderr);
   }
    class JobExitStatus implements-all SAGA.Attribute {
   /* This object holds the state of a finished job.
    * (Controlled by attributes interface.)
    */
   }
```

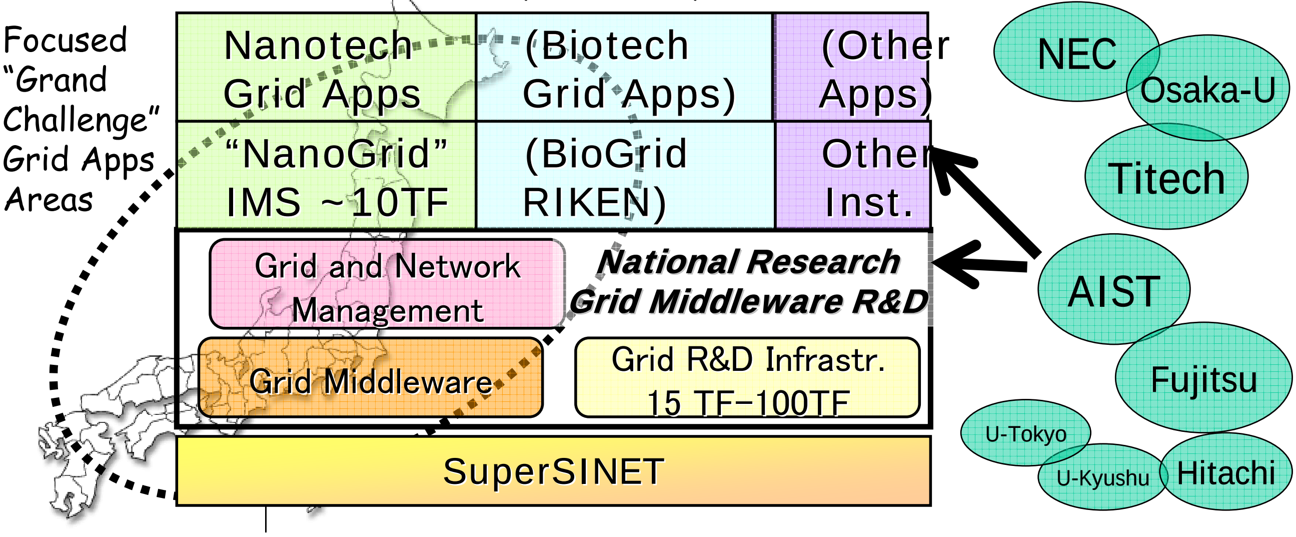
Jobs: Example

```
JobDefinition jobdef = new JobDefinition();
jobdef.setAttribute("SAGA_JobCmd", "myjob.sh");
jobdef.setAttribute("SAGA_NumCpus", "16");
jobdef.setVectorAttribute("SAGA_FileTransfer",
            { "infile > infile",
              "gridftp://somehost/some/path/outputfile << outfile"
    });

JobService myjs = SomeJobServiceFactory(...);
Job myjob = new Job();

myjs.submitJob(jobdef, myjob);
```

The Japanese National Research Grid Infrastructure (NAREGI) 2003-2007

- Petascale Grid Infrastructure R&D for Future Deployment
  - $45 mil (US) + $16 mil x 5 (2003-2007) = $125 mil total
  - Hosted by National Institute of Informatics (NII) and Institute of Molecular Science (IMS)
  - PL: Ken Miura (Fujitsu➔NII)
    - SLs Sekiguchi(AIST), Matsuoka(Titech), Shimojo(Osaka-U), Hirata(IMS)…
  - Participation by multiple (>= 3) vendors
  - Resource Contributions by University Centers as well

Various Partners



Focused "Grand Challenge" Grid Apps Areas

Nanotech Grid Apps

(Biotech Grid Apps)

(Other Apps)

"NanoGrid" IMS ~10TF

(BioGrid RIKEN)

Other Inst.

Grid and Network Management

National Research Grid Middleware R&D

Grid Middleware

Grid R&D Infrastr. 15 TF–100TF

SuperSINET

NEC

Osaka-U

Titech

AIST

Fujitsu

U-Tokyo

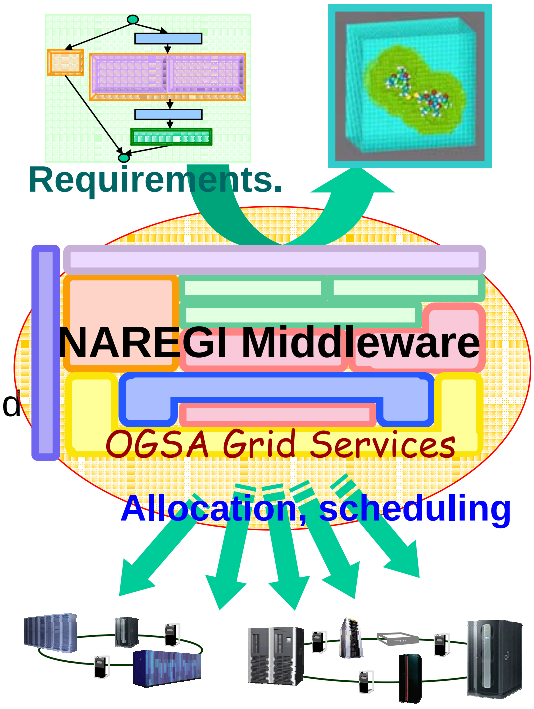
U-Kyushu

Hitachi

NAREGI Middleware Objectives

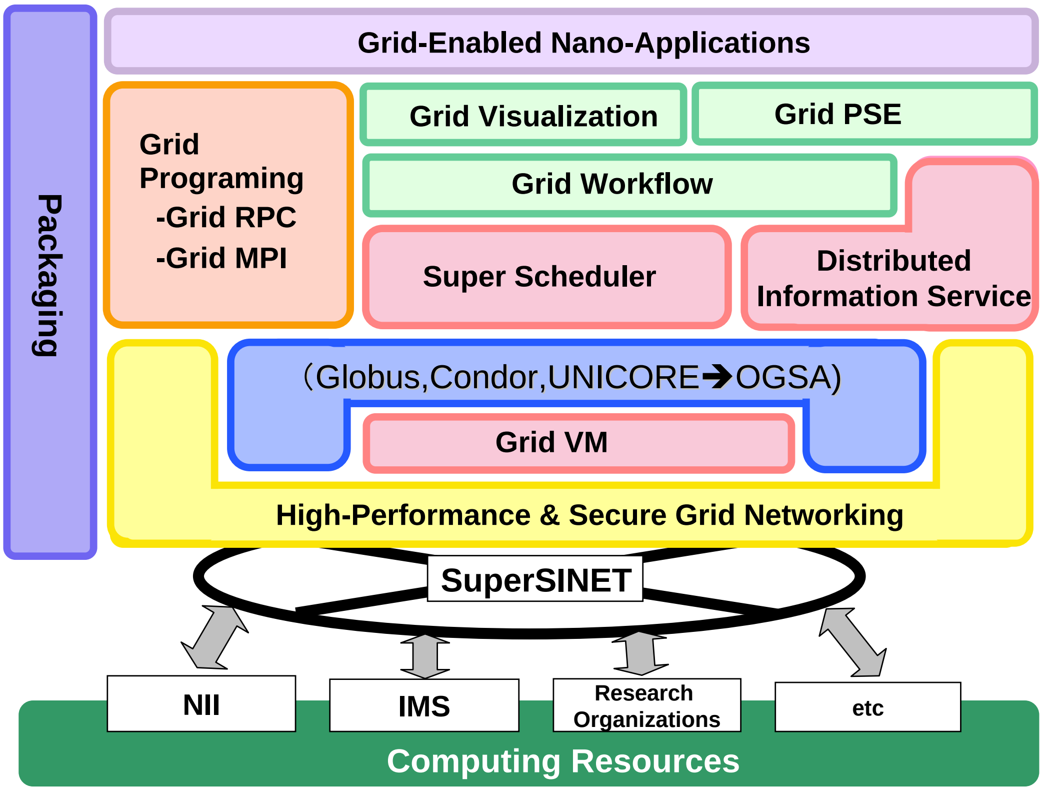
Different Work Package deliverables implement Grid Services that combine to provide the followings:

- Allow users to execute complex jobs with various interactions on resources across multiple sites on the Grid
  - E.g., nano-science multi-physics coupled simulations w/execution components & data spread across multiple groups within the nano-VO
- Stable set of middleware to allow scalable and sustainable operations of centers as resource and VO hosting providers
- Widely adopt and contribute to grid standards, and provide open-source reference implementations, esp. OGSA.



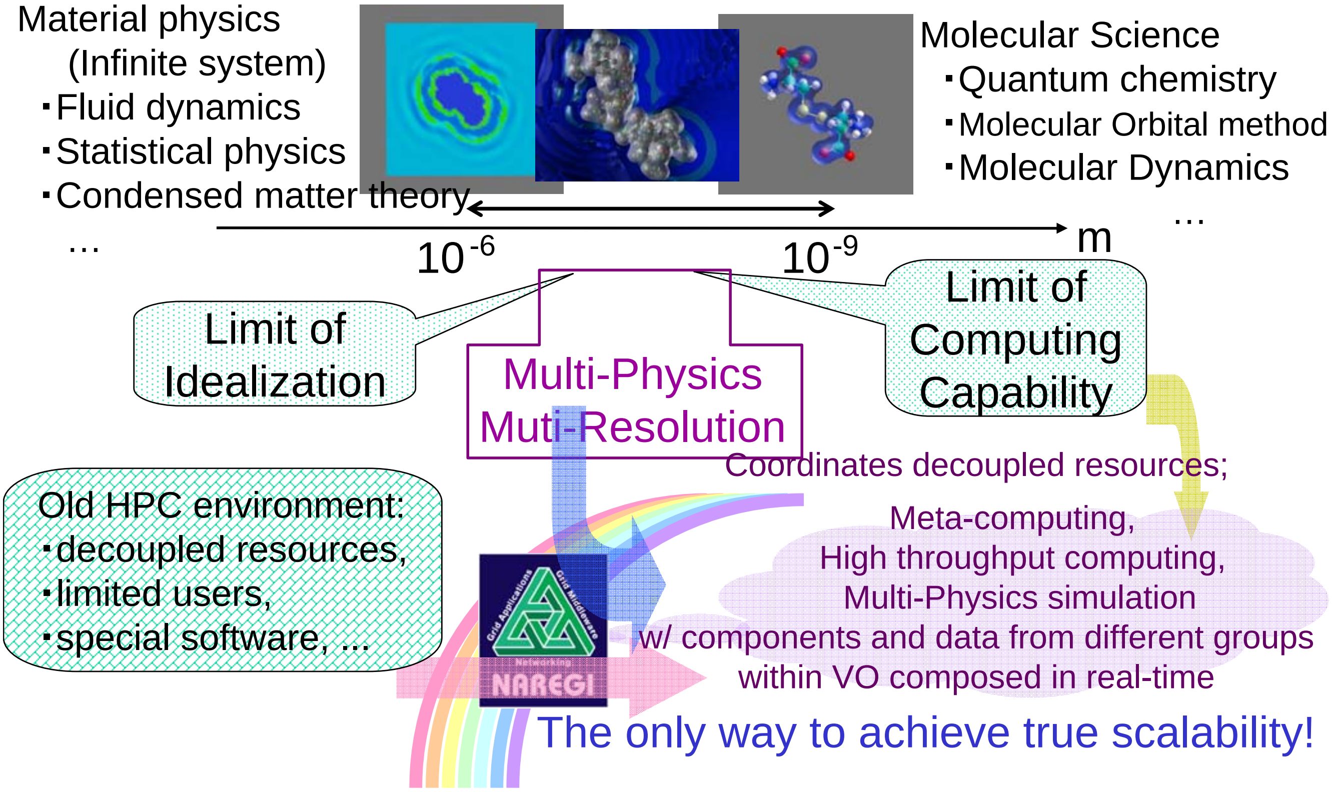
⇒ Sustainable Research Grid Infrastructure.

NAREGI Software Stack



Nano-Science : coupled simluations 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
- …

Limit of Idealization

Limit of Computing Capability

Multi-Physics
Muti-Resolution

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!

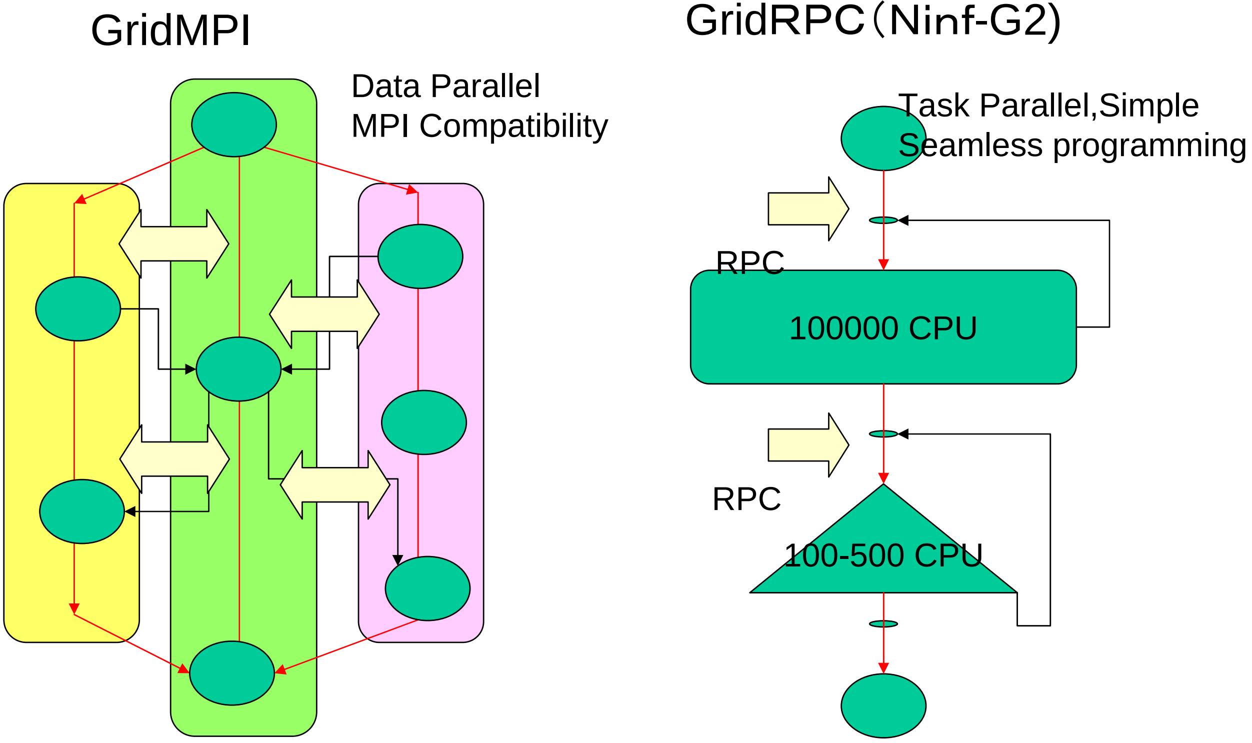
NAREGI Programming Models

- 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 + Mediator (coupled nanoscience simulation framework) / GIANT (coupled simulation data exchange framework)

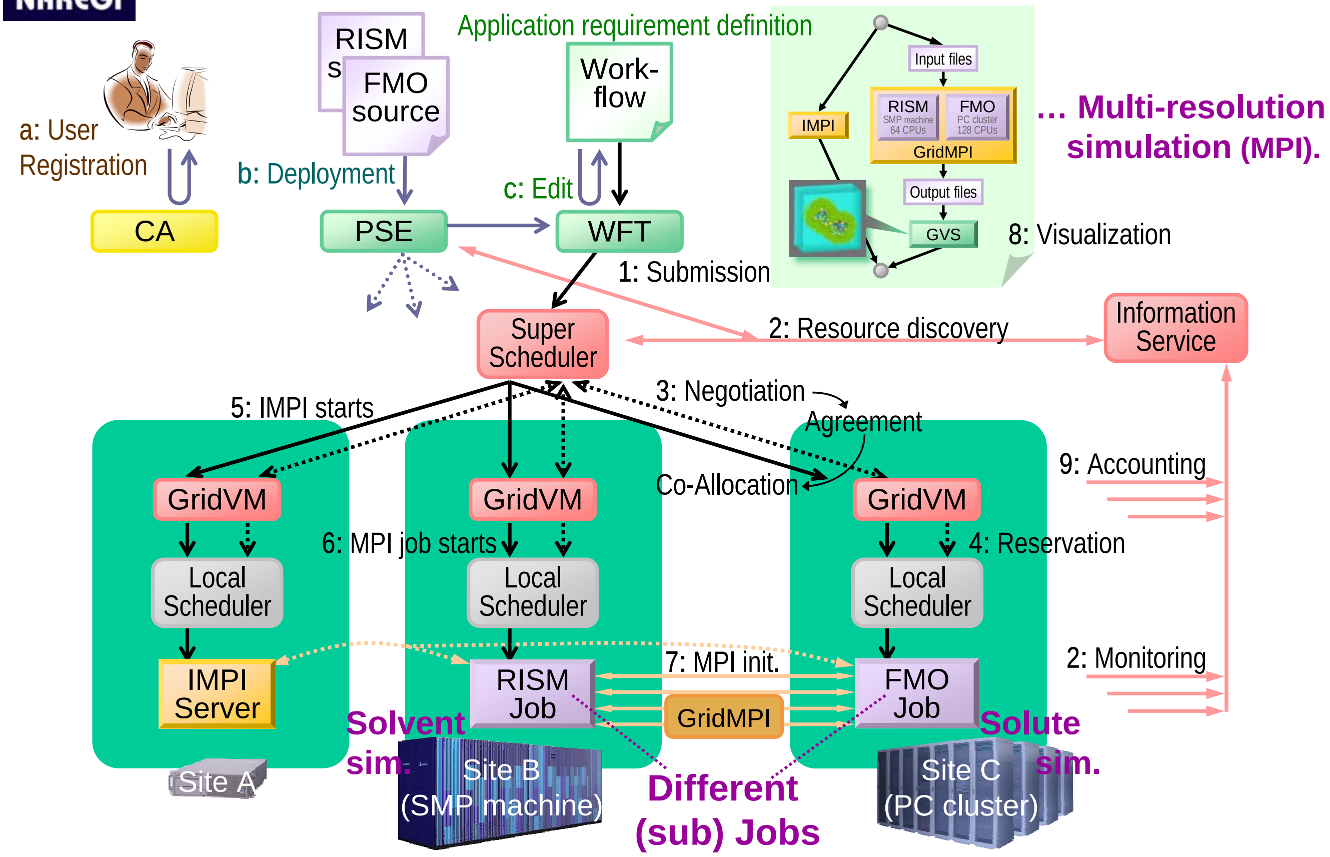
Task Parallel Grid RPC Programming with GridRPC on a Grid of Clusters

- GridRPC: A set of standard libraries that supports task-parallel programming and execution of Grid applications (GGF standard in the works)
- Allow easy "Grid Enabling", task parallelization, and federation of Cross-Cluster MPI Jobs
- To Task Parallel Programming what MPI is to data parallel programming for scientific apps.
- Details later by Dr. Yoshio Tanaka (AIST Grid Technology Research Center) incl. the practicals.

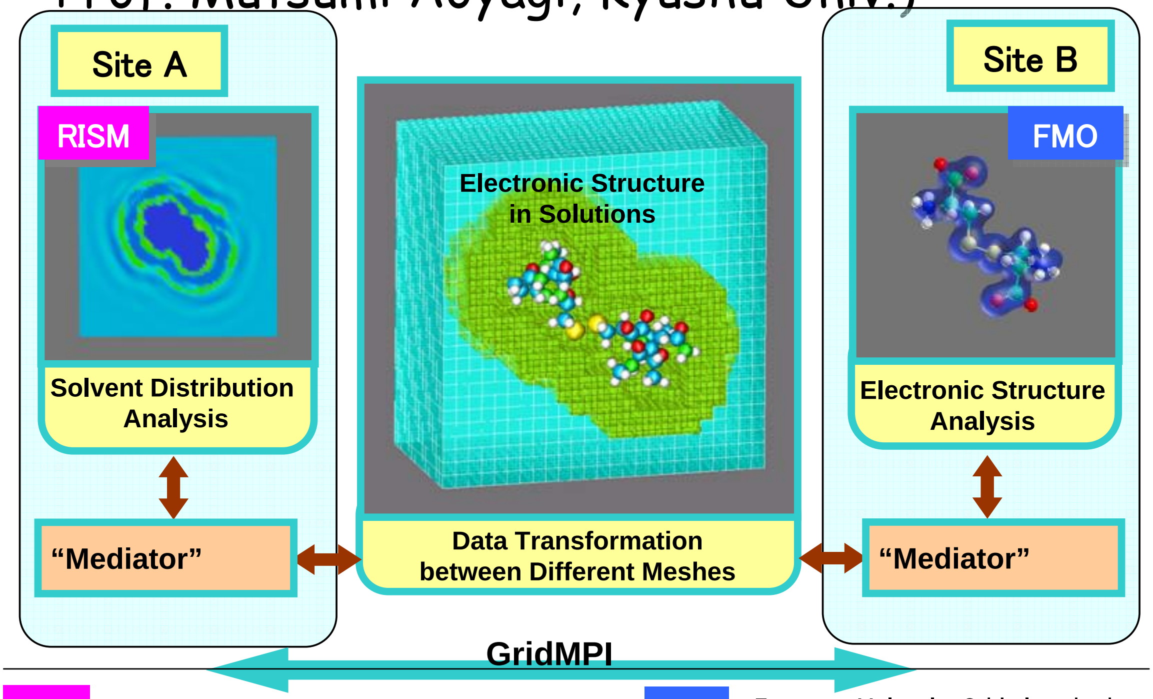
NAREGI WP2 Parallel Programming Models



NAREGI Workflow description example



Higher-level grid programming env for Nano-science Applications (Later lecture by Prof. Mutsumi Aoyagi, Kyushu Univ.)


Site A
RISM
Solvent Distribution Analysis
"Mediator"
Electronic Structure in Solutions
Data Transformation between Different Meshes
Site B
FMO
Electronic Structure Analysis
"Mediator"
GridMPI

RISM Reference Interaction Site Model

FMO Fragment Molecular Orbital method

Summary of Grid Applications and their Programming

- Programming on Grid has been either transparent or fairly ad-hoc
  - No real “rigourous” model yet
  - Due to complexity and infancy of the grid as a whole
  - Still need much R&D for quantitative as well as qualitative modeling to cope with heterogeneity, faults/dependability, portability, etc.
  - May be a great research topic! (as application/progrmming has been in general for the past 40 years :-)