The GRID-TLSE Project and the Nation-Wide Grid Experimental Platform GRID'5000

Michel Daydé

IRIT-ENSEEIHT,
2, rue Camichel, 31071 Toulouse Cedex
Michel.Dayde@enseeiht.fr
http://www.irit.enseeiht.fr/tlse
http://www.irit.enseeiht.fr/grid5000
http://www.grid5000.org

Goals of GRID'5000

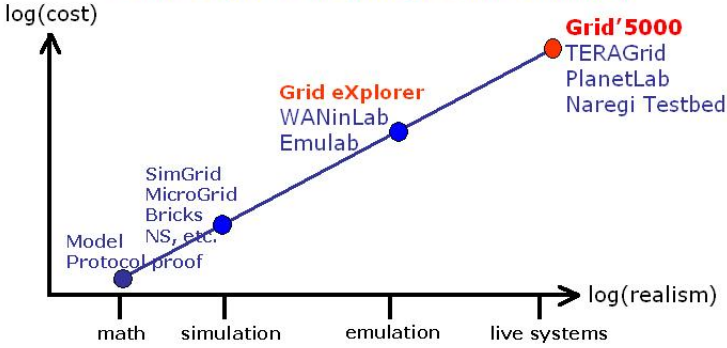
- Building a nation wide experimental plaform for Grid researches
  - UP to 5,000 PC in 10 sites (currently 9)
  - Connection using RENATER (french academic network, 10 GB in nov.)
  - With a flexible system / middleware /management allowing testing and repeated experiments safely
- Platform used for Grid experiments
  - Address critical issues of Grid system / middleware (programming, scalability, fault tolerance, scheduling, . . .)
  - Address critcial issues in Grid Networking (high perf. protocols, QoS, . . .)
  - Gridification of appplications
  - Investigate new approaches: P2P resource discovery, Desktop grids, . . .

Grid'5000

We need a Grid experimental platform

According to the current knowledge:
There is no large scale testbed dedicated to Grid experiments

→ Grid'5000 as a live system
→ Grid eXplorer as a large scale emulator

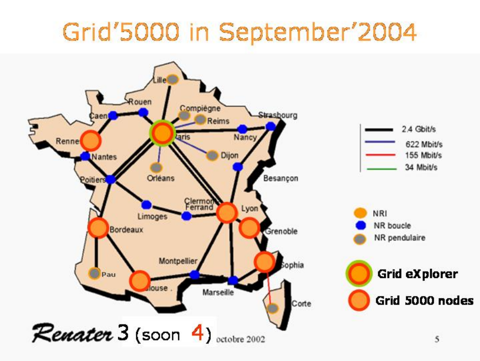


From Franck Cappello

Requirements (from Franck Cappello)

- Nodes geographically distributed that can be controlled remotely
- Network between the grid nodes that can be controlled and monitored
- Middleware insuring at least some security (identification, isolating traffic, . . .)
- Toolkit for deploying, managing, running experiments and collecting results

GRID'5000 Map and Funding

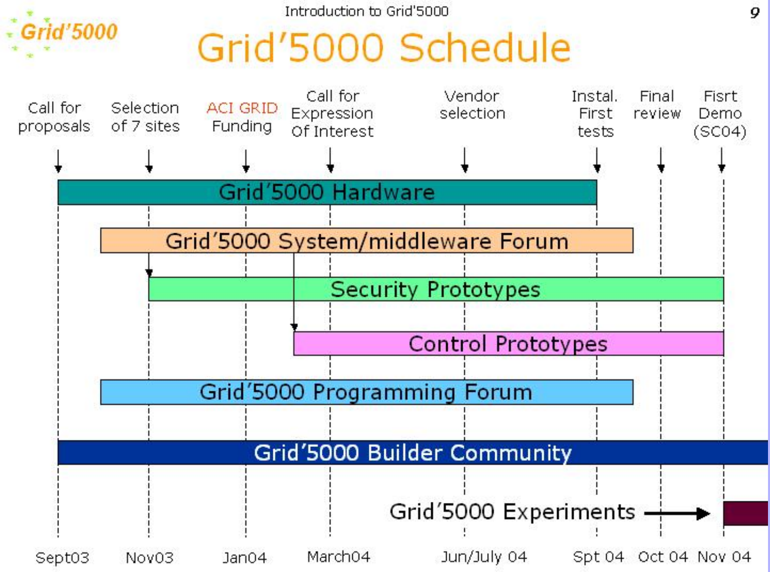


- In 2003 :
  - 2 M€ from ACI GRID and MD (Ministery of Research)
  - Plus 3 M€ from: Local and Regional Councils, Universities, CNRS, INRIA
- In 2004 - 2005 :
  - 1 M€ from ACI GRID (2004)
  - 1.5 M€ from Local and Regional Councils, Universities, CNRS, INRIA
- Total ≈ 7.5 M€

Main issues

- Native system image on each cluster (Linux-based : Fedora, RedHat, . . .)
- Node reservation system (use of experimental software named OAR)
- Users have ability to deploy their own system image (by resetting the nodes they have acquired)
- Security model
- Grid'5000 approach is quite new

Schedule of GRID'5000 (from Franck Cappello)



Research topics on Grid'5000 (from Franck Cappello)

- Networking
  - End host communication layer
  - High performance long distance protocols
  - High speed network emulation
  - Grid networking layer
- Middleware / OS
  - Grid'5000 control / Access / experiment automation
  - Scheduling / data distribution on the Grid
  - Fault tolerance
  - Resource management
  - Computational steering
  - Grid SSI OS and Grid I/O
  - Desktop Grid/P2P systems

Research topics on Grid'5000 con't (from Franck Cappello)

- Programming
  - Use of software components (Java, Corba, ...)
  - Grid-RPC
  - Grid-MPI
  - Code coupling
- Applications
  - Multi-parametric application (climate modelling, functional genomic, sparse linear algebra, ...)
  - Large scale distributed applications (electromagnetism, multi-material fluid mechanics, parallel optimization, astrphysics, ...)
  - Medical images, collaborative tools in virtual 3D environment

Cluster in Toulouse

- End Nov. 2004 : 32 bi-pro (64 procs) AMD Opteron 2.2 GHZ, 2 GB mem. / node, 73 GB disk, 2 frontals and 700 GB disk
- Switch 1 GB
- 28 additional nodes currently installed
- 120 processors end of november

GRID-TLSE Project
Tests for **L**arge **S**ystems of **E**quations

Main purpose: Sparse linear algebra Web expert site.

Funding: ACI GRID, 01/03 – 01/06.

Partners:

- Academic partners: CERFACS, ENSEEIHT-IRIT, LaBRI, LIP-ENSL;
- Industrial partners: CNES, CEA, EADS, EDF, IFP;
- International links: LBNL-Berkeley, Parallab-Bergen, Univ. of Florida, RAL, Old Dominion Univ., Univ. of Minnesota, Univ. of Tennessee, Univ. of San Diego, Indiana Univ., Tel-Aviv Univ.

Grid issues in the Project

- Application server oriented
- Use of GridRPC type of mechanism (available in Globus, NetSolve, DIET,...)
- Use of tools developed within GRID-ASP project (LIP-ReMAP, LORIA-Résédas, LIFC-SDRP) : DIET
- High-level administrator interface for the definition, the deployment, and the exploitation of services over a grid : Weaver
- Interactive Web interface with the Grid: WebSolve
- We currently investigate use JUXMEM developed by Paris Project at IRISA for management of data over a grid

Sparse Matrices Expert Site ?

Expert site: Help users in choosing the right solvers and its parameters for a given problem

Chosen approach: Expert scenarios which answer common user requests

Main goal: Provide a friendly test environment for expert and non-expert users of sparse linear algebra software.

Easy access to:

- Software and tools;
- A wide range of computer architectures;
- Matrix collections;
- Expert Scenarios.

Also : Provide a testbed for sparse linear algebra software

Examples of user request

- Memory required to factor a given matrix.
- Error analysis as a function of the threshold pivoting value.
- Minimum time on a given computer to factor a given unsymmetric matrix.
- Which ordering heuristic is the best one for solving a given problem?

Why do we use a Grid ?

- Sparse linear algebra software makes use of sophisticated algorithms for (pre-/post-) processing the matrix.
- Multiple parameters interfere for efficient execution of a sparse direct solver:
  - Ordering;
  - Amount of memory;
  - Architecture of computer;
  - Libraries available.
- Determining the best combination of parameter values is a multi-parametric problem.

Why do we use a Grid ?

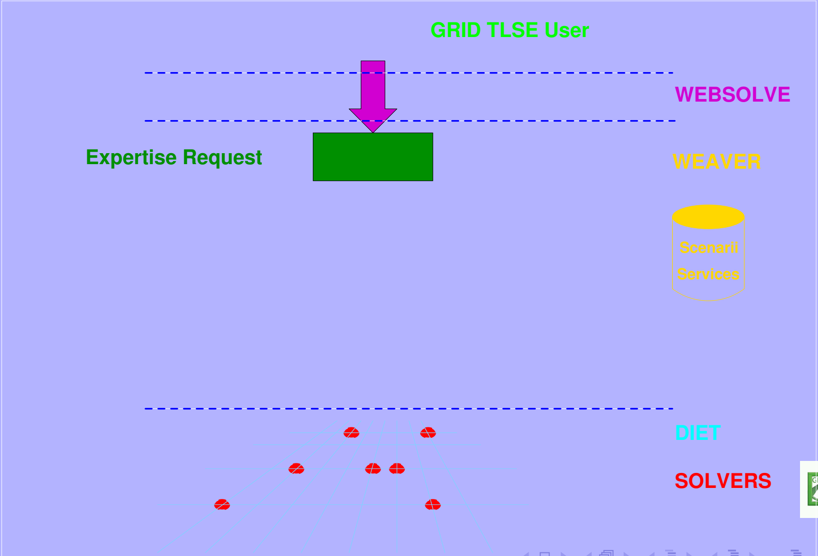
- Sparse linear algebra software makes use of sophisticated algorithms for (pre-/post-) processing the matrix.
- Multiple parameters interfere for efficient execution of a sparse direct solver:
  - Ordering;
  - Amount of memory;
  - Architecture of computer;
  - Libraries available.
- Determining the best combination of parameter values is a multi-parametric problem.
- Well-suited for execution over a Grid.

Additional Benefits of Using a Computational Grid

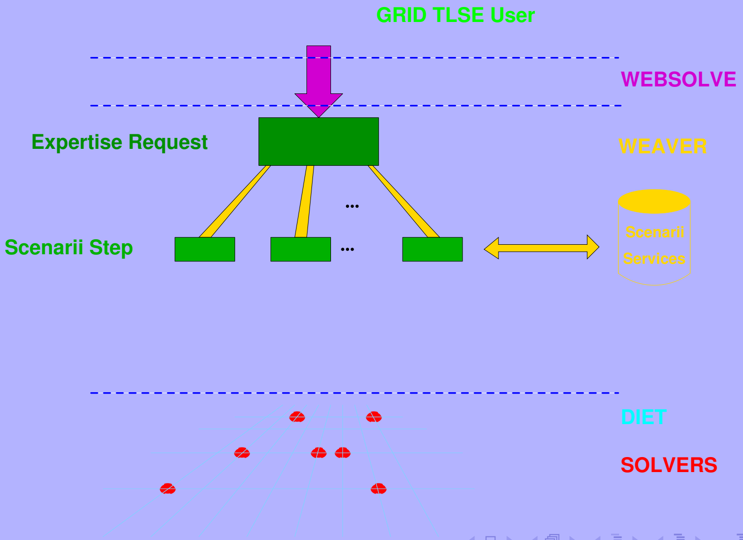
Provides access to:

- Large range of software and tools (academic or industrial);
- Wide range of architectures;
- Computational resources.

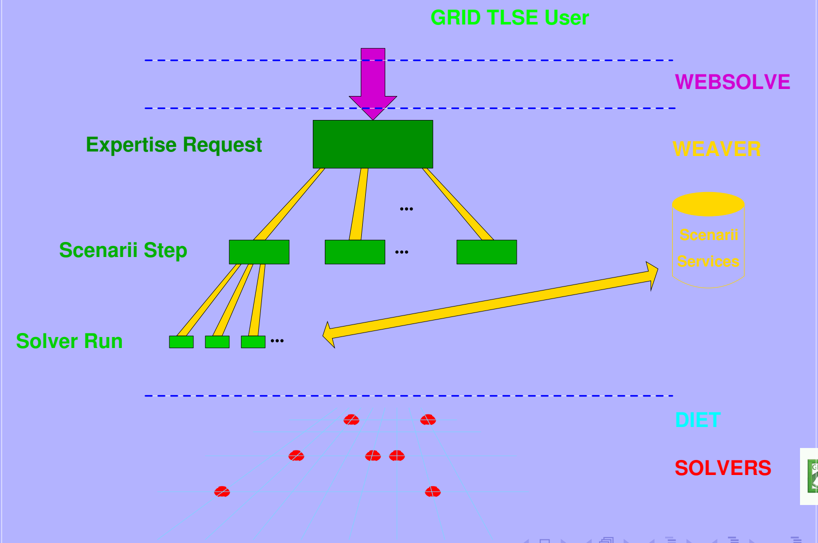
Processing a user request



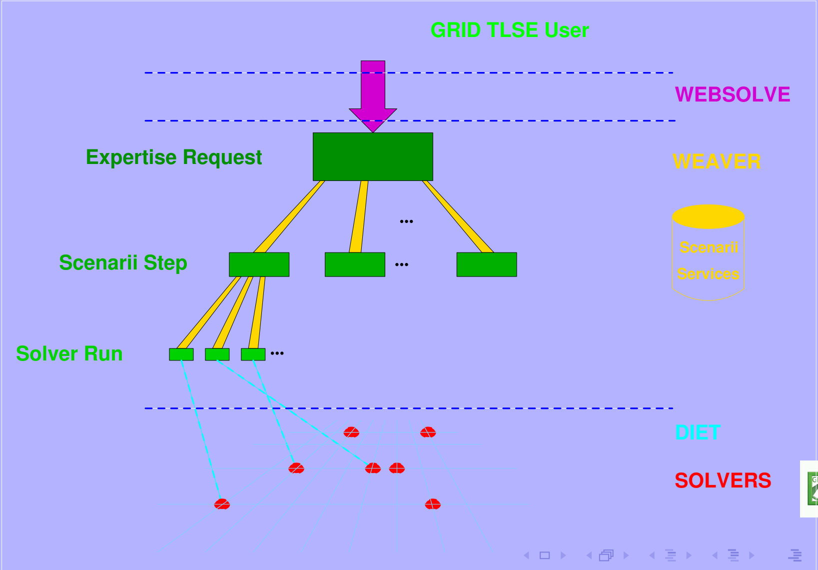
Processing a user request



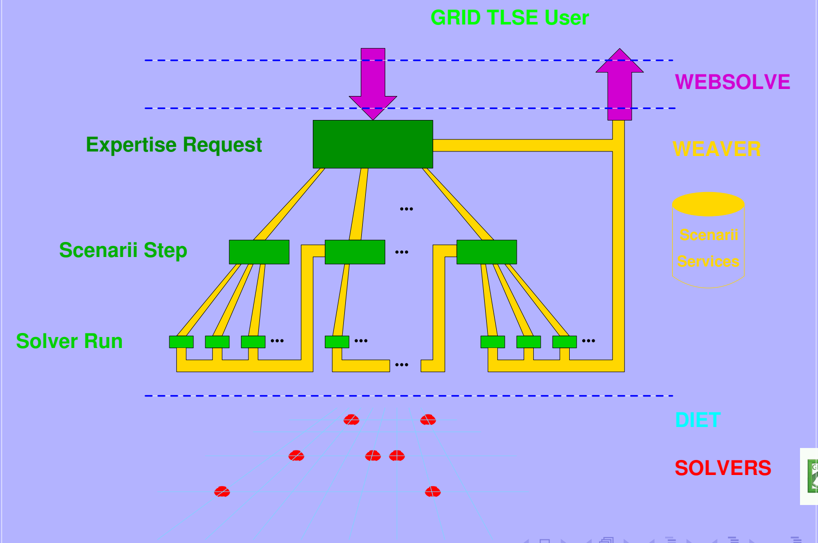
Processing a user request



Processing a user request



Processing a user request



Main Software Bottleneck

The same interface provides the users with access to

- several expertise scenarios;
- several solvers and their parameters (using middleware to access the GRID).

Main Software Bottleneck

The same interface provides the users with access to

- several expertise scenarios;
- several solvers and their parameters (using middleware to access the GRID).

Experts provide scenarios which

- reduce the combinatorial nature;
- produce useful synthetic comparison.

Main Software Bottleneck

The same interface provides the users with access to

- several expertise scenarios;
- several solvers and their parameters (using middleware to access the GRID).

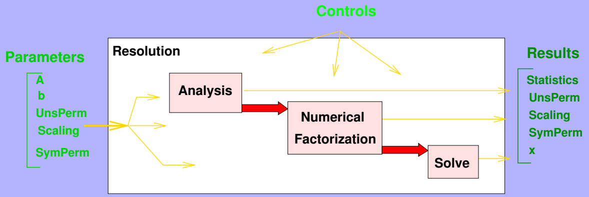
Experts provide scenarios which

- reduce the combinatorial nature;
- produce useful synthetic comparison.

It should be easy to

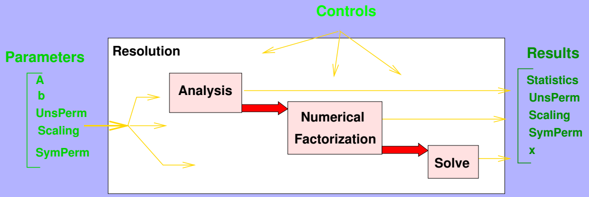
- add new solvers which can be used by old scenarios;
- add new scenarios which use old solvers;
- use the characteristics of new solvers in new scenarios.

Solver service API for solution of $Ax = b$



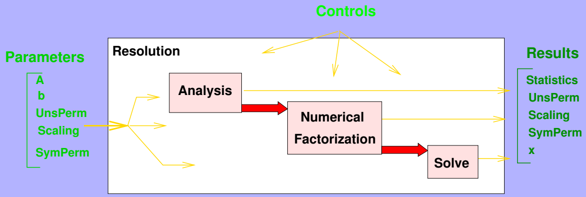
- Many possible algorithms / control parameters / metrics to evaluate efficiency (time, precision, memory, . . .)
- Many solver packages provide different combinations: MUMPS, SuperLU, UMFpack, TAUCS, HSL MAxx, PaStiX, SPOOLES, OBLIO, PARDISO, . . .

Solver service API for solution of $Ax = b$



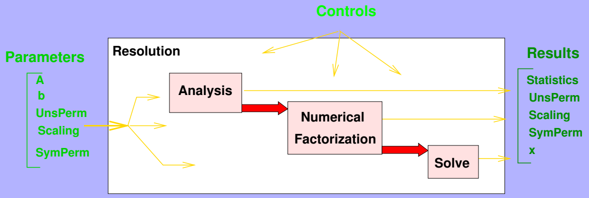
- Many possible algorithms / control parameters / metrics to evaluate efficiency (time, precision, memory, ...)
- Many solver packages provide different combinations: MUMPS, SuperLU, UMFpack, TAUCS, HSL MAxx, PaStiX, SPOOLES, OBLIO, PARDISO, ...
- Main difficulty: defining common API for all these packages.

Solver service API for solution of $Ax = b$



- Many possible algorithms / control parameters / metrics to evaluate efficiency (time, precision, memory, . . .)
- Many solver packages provide different combinations: MUMPS, SuperLU, UMFpack, TAUCS, HSL MAxx, PaStiX, SPOOLES, OBLIO, PARDISO, . . .
- Main difficulty: defining common API for all these packages.
- Classically: Write wrappers or adapters (design patterns)

Solver service API for solution of $Ax = b$



- Many possible algorithms / control parameters / metrics to evaluate efficiency (time, precision, memory, . . .)
- Many solver packages provide different combinations: MUMPS, SuperLU, UMFpack, TAUCS, HSL MAxx, PaStiX, SPOOLES, OBLIO, PARDISO, . . .
- Main difficulty: defining common API for all these packages.
- Classically: Write wrappers or adapters (design patterns)
- Major bottleneck: adding solvers and scenarios may require changes in all solver's wrappers and old scenarios

The Reflexive Way

Clients and providers of solvers should adapt dynamically to each other.

Reflexivity: Dynamic discovery of a component characteristics.

Meta-data which describe for each package:

- Functional decomposition;
- Control parameters;
- Values for a given control parameter;
- Metrics;
- Values for a given metric;
- Qualitative and quantitative dependency between values of metrics and control parameters

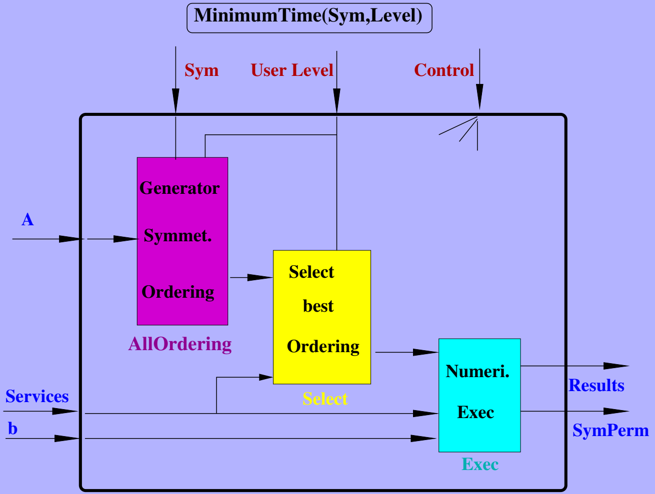
Adding new meta-data and possible values should be easy

Using Abstract Parameters

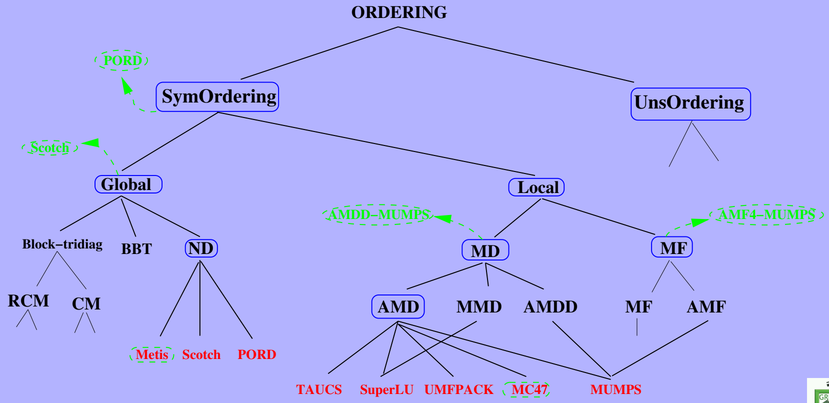
From Web interface (to define the objective and parameters of the scenarios) up to the service description one must use a common abstract parameter.

- To describe a service: functionalities (factorization, multiprocessor, multiple RHS , . . .), algorithmic properties (unsymmetric/symmetric solver, multifrontal, . . .)
- To describe a scenario in addition to service parameters: metrics (memory, numerical precision, time, . . .), control: type of graphs for post-processing, level of user.
- For expressing constraints and decrease combinatorial explosion e.g. if A symmetric for a standard user use only symmetric solvers.

Building Scenarios : *Minimum time*



Structuring Abstract Parameters to Describe Scenarios and Services



Trading service

- We want more than existing trading service :
  - Find me a service for computing efficiently the solution of a symmetric indefinite linear system with matrix of order 100,000 and sparse general structure
  - Give me the list of services that can solve a given system with 2GB of memory for the initial matrix in less than 10 hours

Semantic based component model and trading services (A. Hurault and M. Pantel)

Example:

$$C = A \times B \quad \text{or} \quad foo(A, B)$$

Finding composition of services capable of executing the request based on:

- High level description of the services
- Mathematic properties of the operation (associativity, commutativity,. . .)
- Properties of the operands (e.g. structure and numerical properties of matrices)
- Characteristics of machine hosting the service (load, performance, memory available,. . .)
- . . .

Conclusion

- Key points: high level description of scientific software and use of scenarios for generating dynamic workflows
- Practical consequences:
  - Adding / removing solvers does not require to update scenarios (it will be automatically discovered)
  - Introduction of new scenarios make use of deployed software
  - The approach described is intended to be generic: we explore the use of this approach in other areas