Presentation of GRID-TLSE

http://www.enseeiht.fr/lima/tlse

ACI GDS Meeting, May 20th, 2005

Outline

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.

Expertise for Sparse Matrices: Motivations

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

Easy access to:

- Software and tools: public... as well as commercial, sequential... as well as parallel;
- A wide range of computer architectures;
- Matrix collections.

Goal (bis): Provide a testbed for sparse linear algebra software developers.

Scope of TLSE: focus on direct methods for sparse matrices

Why Using a Grid ?

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

Why Using a Grid ?

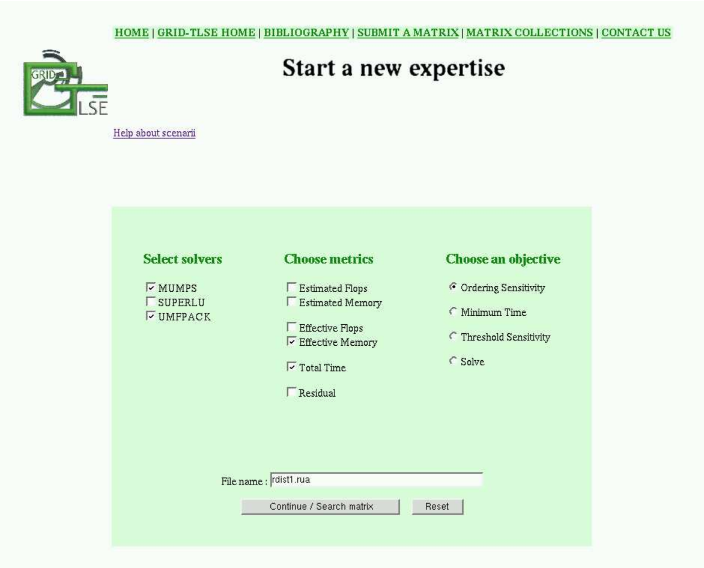
- Sparse linear algebra software makes use of sophisticated algorithms for (pre-/post-) processing/solving a sparse system $Ax = b$.
- Multiple parameters interfere for efficient execution of a sparse solver:
  - Ordering;
  - Amount of memory;
  - Architecture of computer;
  - Libraries available.
-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

Main Components of the Site

- Sparse matrix software: direct solvers.
- Database: matrices, scenarios, bibliography, experimental results.
- 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.
- Use of tools developed within GRID-ASP project (LIP-ReMAP, LORIA-Résédas, LIFC-SDRP): DIET.

Examples of Requests (scenarios)

- Memory required to factor a matrix, with which algorithm/solver/input parameters ?
- Error analysis as a function of the threshold pivoting value.
- Minimum time on a given computer to factor a given unsymmetric matrix. (naive or more elaborated scenario)
- Which ordering heuristic is the best one for solving a given problem?

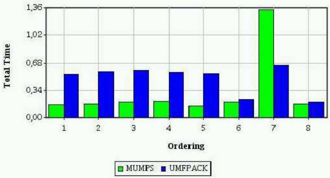

HOME | GRID-TLSE HOME | BIBLIOGRAPHY | SUBMIT A MATRIX | MATRIX COLLECTIONS | CONTACT US
GRID
LSE
Start a new expertise
Help about scenarii
Select solvers
MUMPS
SUPERLU
UMFPACK
Choose metrics
Estimated Flops
Estimated Memory
Effective Flops
Effective Memory
Total Time
Residual
Choose an objective
Ordering Sensitivity
Minimum Time
Threshold Sensitivity
Solve
File name :
rdist1.rua
Continue / Search matrix
Reset

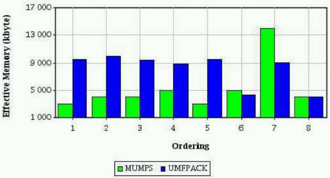
Experimental Results

New expertise

MATRIX NAME : rdist1.rua

	Ordering	Total Time	Effective Memory (kbyte)
MUMPS	(1) AMD	1,6E-1	3E3
	(2) AMF	1,7E-1	4E3
	(3) PORD	1,9E-1	4E3
	(4) METIS	2E-1	5E3
	(5) QAMD	1,4E-1	3E3
	(6) MMD x	1,9E-1	5E3
	(7) MMD +	1,34E0	1,4E4
	(8) COLAMD	1,7E-1	4E3
UMFPACK	(1) AMD	5,4E-1	9,528E3
	(2) AMF	5,7E-1	9,888E3
	(3) PORD	5,9E-1	9,456E3
	(4) METIS	5,6E-1	8,823E3
	(5) QAMD	5,5E-1	9,528E3
	(6) MMD x	2,2E-1	4,246E3
	(7) MMD +	6,5E-1	9,055E3
	(8) COLAMD	1,9E-1	4,009E3





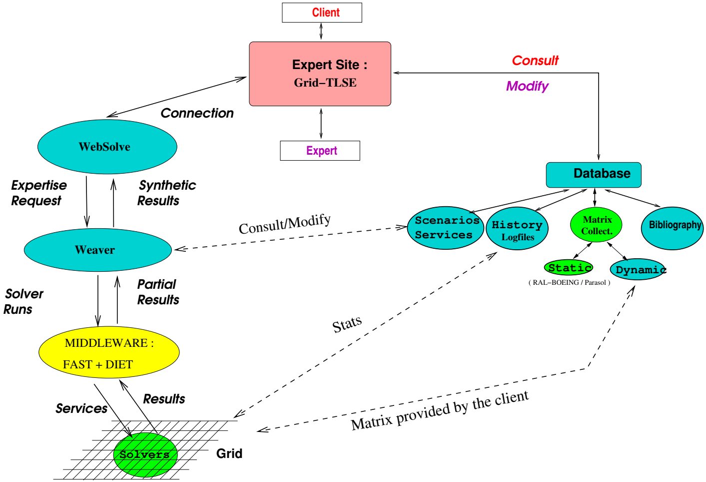
Scenario examples: Ordering sensitivity

- **Phase 1:** Get orderings (permutations):
  - one solver: get all of its internal orderings.
  - more than one solver: get all possible orderings from all solvers.
- **Phase 2:** Obtain value of required metrics for each ordering:
  - for metrics of type estimation, the analysis is performed for each required solver.
  - for metrics of type effective, the factorization is also performed.
- **Phase 3:** Report metrics for all combinations of solvers/orderings

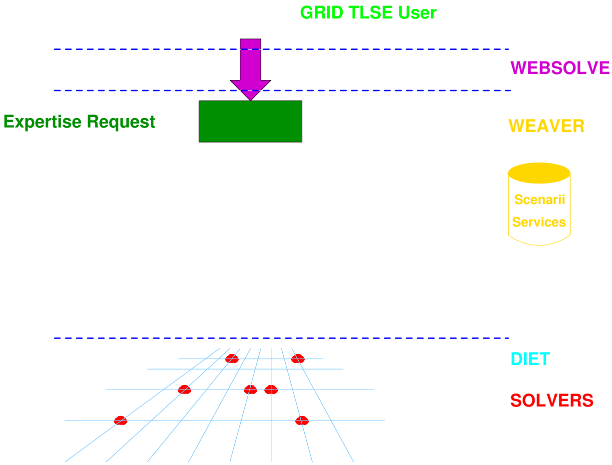
Scenario examples: Minimum time

- **Phase 1:** Get orderings from all solvers.
- **Phase 2:** For each ordering and requested solver
  - perform Flops estimation
  - keep best ordering per solver.
- **Phase 3:** For each solver:
  - factorize with BOTH selected ordering and internal default ordering
  - report statistics with minimum time.

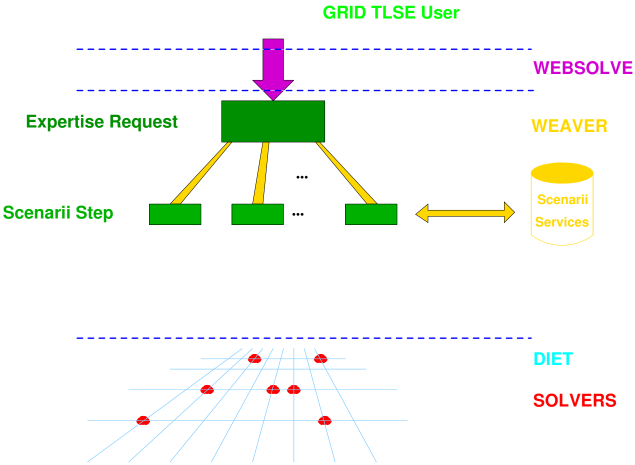
Software Architecture



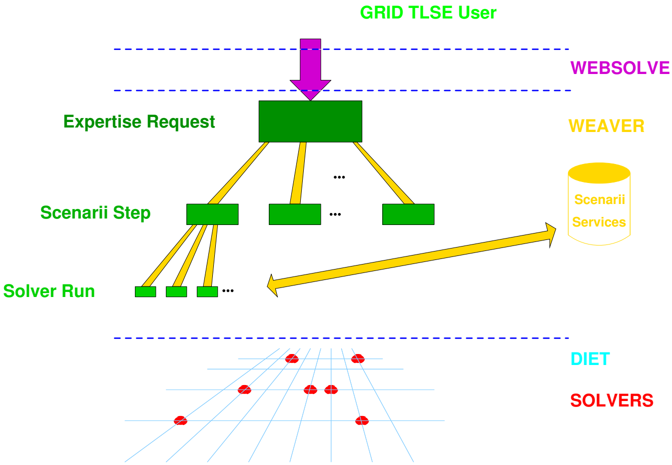
Expertise Run



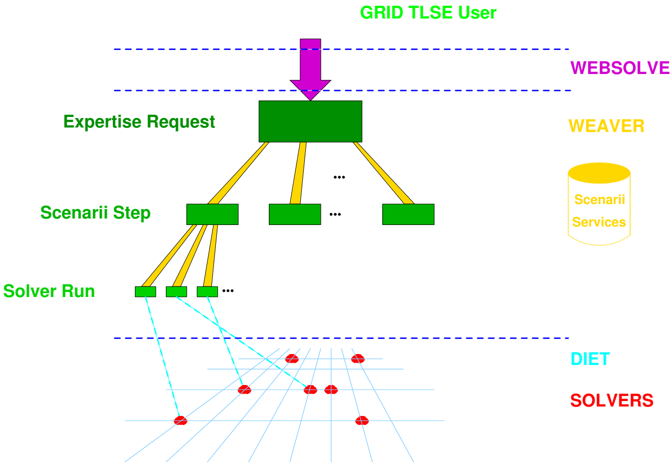
Expertise Run



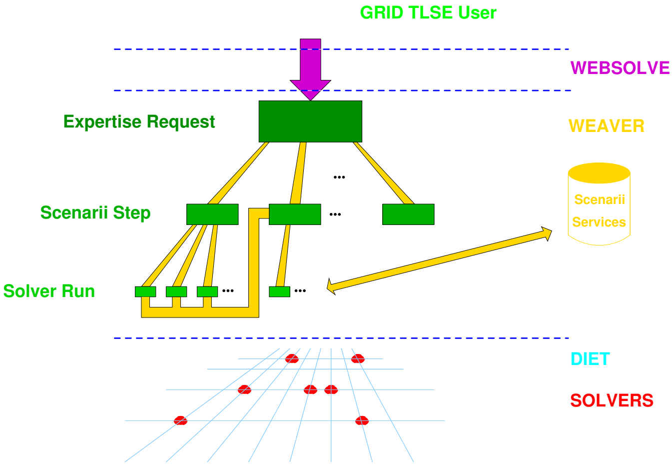
Expertise Run



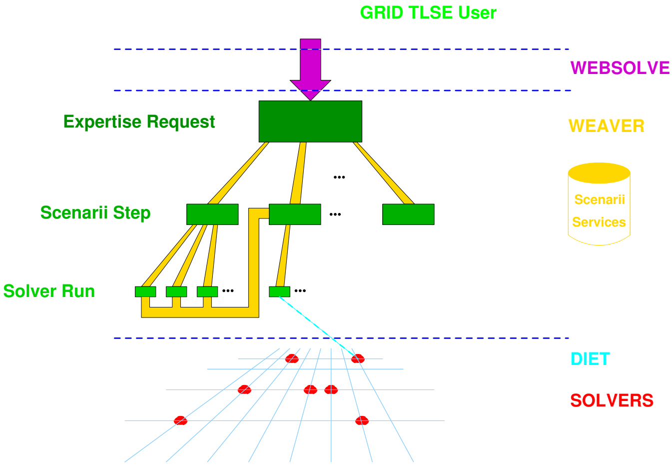
Expertise Run



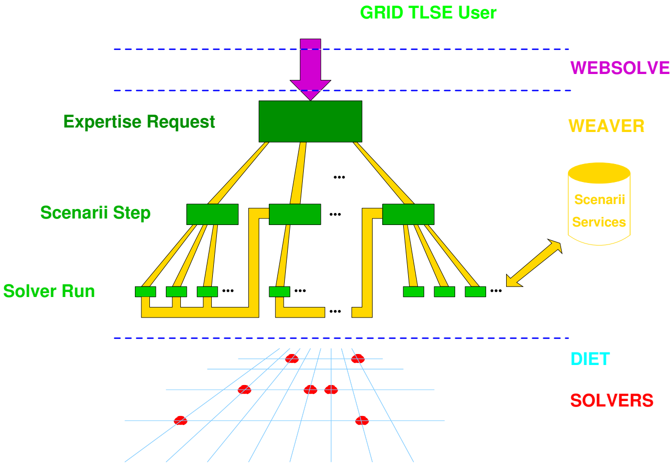
Expertise Run



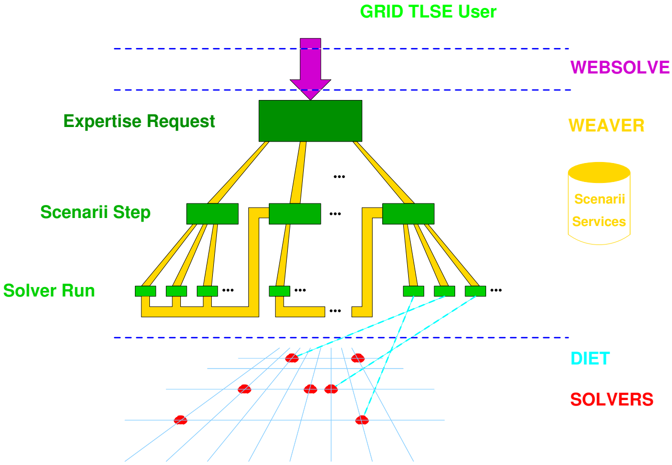
Expertise Run



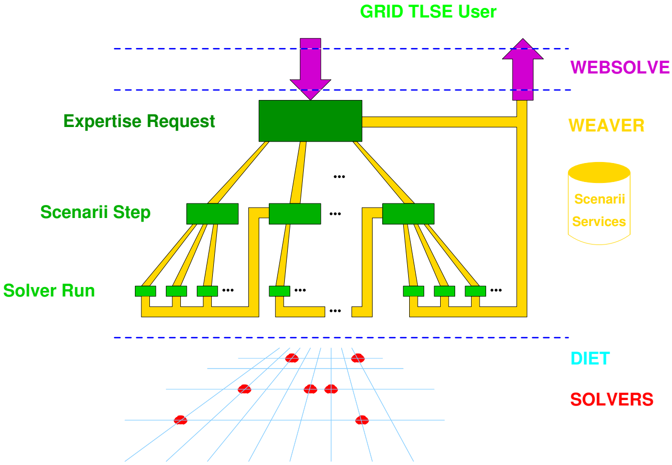
Expertise Run



Expertise Run



Expertise Run



Main Software Difficulties

A Web interface provides the users with access to

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

Main Software Difficulties

A Web interface provides the users with access to

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

Experts provide expertise scenarios which

- reduce the combinatorial complexity;
- produce useful synthetic comparisons.

Main Software Difficulties

A Web interface provides the users with access to

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

Experts provide expertise scenarios which

- reduce the combinatorial complexity;
- produce useful synthetic comparisons.

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.

Main Software Bottleneck

Synthesis:

- Many possible algorithms for solving a linear system;
- Many possible control parameters;
- Many values for each parameter;
- Many metrics to evaluate/compute numerical results;
- Many metrics to evaluate/compute software runs.

Many solver packages provide different combinations:

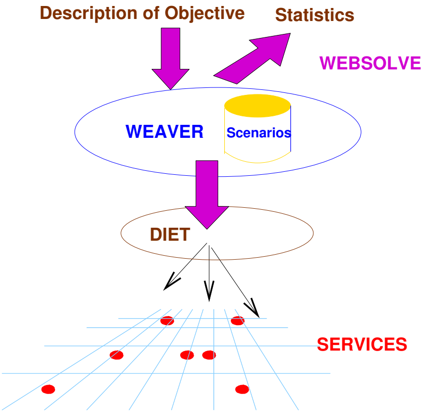
- Currently in TLSE: MUMPS, SuperLU, UMFpack;
- Being integrated: TAUCS, PaStiX;
- Future: HSL MAxx, SPOOLES, OBLIO, PARDISO, ...

Rationale: Rather than providing a common API for all these packages with the union of all possible parameters from all solvers, use higher-level "classes" of parameters (meta-data, also called abstract parameter) that can be instantiated for each solver.

Expert Site: Main Resources

1. Matrices :
   - from existing collections,
   - private to a user or a group of users.
2. Software :
   - public or commercial packages,
   - different types, approaches, languages.
3. Computers
4. Users : 2 main types
   - standard users: can upload a matrix, experiment with matrices and software
   - "super users": can add new scenarios, new software, new computers, validate/decontaminate resources (matrix, software, computer)

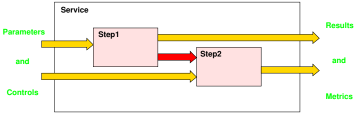
Managing Scenarios



Scenarios :
Describe the sequence of services to reach an objective

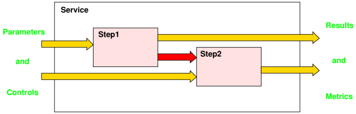
Some Services

1. **Solution**: solve $Ax = b$.
2. **Matrix transformation**: format conversion. (standard format if a matrix is made publically available in the TLSE collection)
3. **Matrix validation/decontamination**.
4. **Matrix generators**.
5. **Tools to help an expert user validate a resource (matrix/solver/computer)**



Some Services

1. **Solution**: solve $Ax = b$.
2. **Matrix transformation**: format conversion. (standard format if a matrix is made publically available in the TLSE collection)
3. **Matrix validation/decontamination**.
4. **Matrix generators**.
5. **Tools to help an expert user validate a resource (matrix/solver/computer)**



Focus on **1. Solution**.

Remarks on Service *Solution*

To solve $Ax = b$, A unsym., with LU factorisation, we often need to:

- Improve the numerical properties of A
  - Equilibrate the matrix (D_r, D_c): Scaling
  - Permute large entries to the diagonal (Q_r, Q_c): Unsym. Permutation

$$A \Longrightarrow Q_r D_r A D_c Q_c$$

- Reduce fill-in
  - Compute symmetric permutation (P): Symmetric Ordering

$$A \Longrightarrow PAP^\top$$

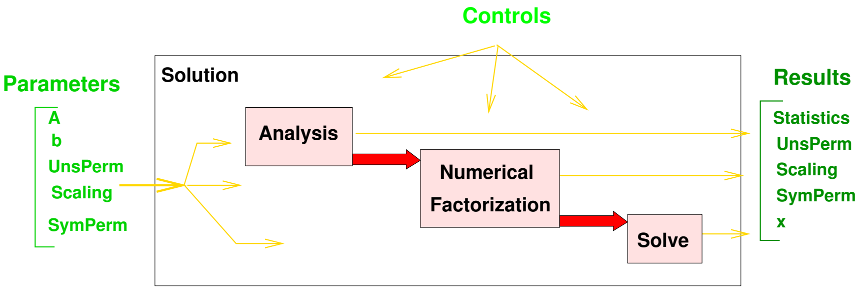
Signature of the Service *Solution*

So what we solve is:

$$(PQ_rD_r)A(D_cQ_cP^\top)(PQ_c{}^\top D_c{}^{-1})x = (PQ_rD_r)b$$

where

- D_r and D_c are **scaling** matrices;
- Q_r, Q_c hold the unsymmetric permutations: **UnsPerm**;
- P holds the symmetric permutation: **SymPerm**.



Using Abstract Parameters

From the Web interface (to define the objective and parameters of the scenarios) up to the service description, it is critical using a common abstract parameter.

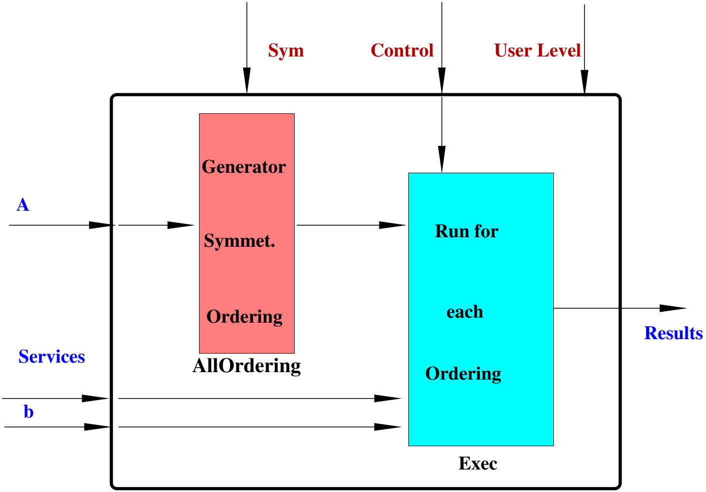
- To describe a service:
  - functionalities: assembled/elemental entries, type of factorisations (LU, LDL^T,QR), multiprocessor, multiple RHS;
  - algorithmic properties: unsymmetric/symmetric solver, multifrontal, left/right looking, pivoting strategy.
- To describe a scenario in addition to service parameters:
  - metrics: memory, numerical precision, time,
  - control: type of graphs for post-processing, level of user.

Abstract Parameters (continued)

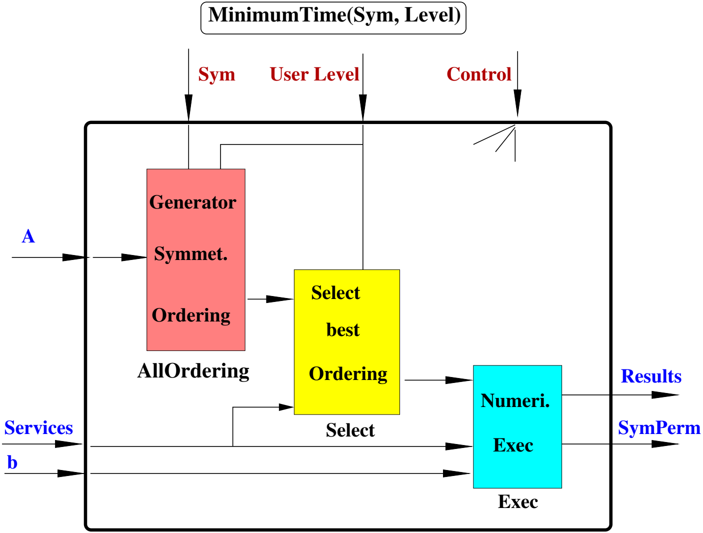
Abstract parameters are used to express constraints and/or relations.

- If A symmetric and standard user, then select only symmetric solver.
- Indicate that time and memory depend mostly on method and permutations but also on scaling and pivoting.
- Indicate that numerical accuracy depends mostly on pivoting but also on scaling and permutations.
- Advise orderings for QR based on $A^T A$.
- Indicate that multiple RHS option, although not available, can still be performed (simulated within SeD).
- Threshold for partial pivoting $\in [0, 1]$.

Building Scenarios (I): *Ordering sensitivity*



Building Scenarios (II): *Minimum time*



Building Scenarios: Remarks

The abstract parameter **SymPerm** *corresponds to an enumeration of large size.*

Building Scenarios: Remarks

The abstract parameter **SymPerm** *corresponds to an enumeration of large size.*

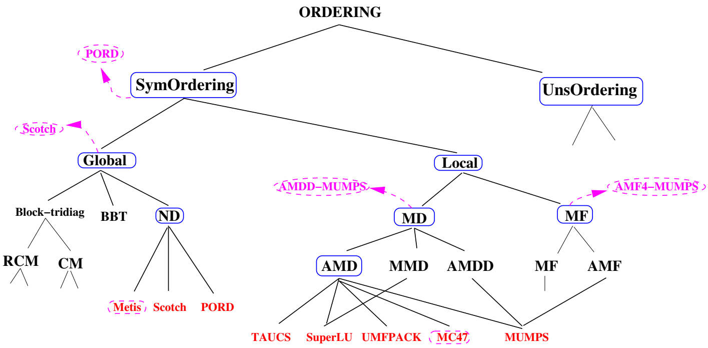
- Each software may have its own implementation of the AMD ordering.
  - One representative of this set might be enough in most cases.
  - How to define/select a representative ?
  - This representative might change from time to time.

Building Scenarios: Remarks

The abstract parameter **SymPerm** *corresponds to an enumeration of large size.*

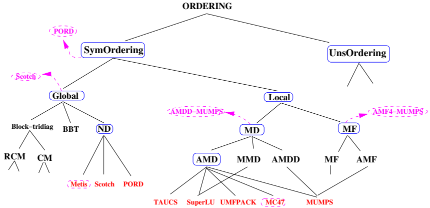
- Each software may have its own implementation of the AMD ordering.
  - One representative of this set might be enough in most cases.
  - How to define/select a representative ?
  - This representative might change from time to time.
- Furthermore: one might not want to test all possible values of the symmetric permutation.
  - On some matrices a subclass of orderings is known to be superior.
  - A (standard) user only wants to capture major differences between orderings.
  - Using a “good” representative of a subclass might be enough.

Structuring Abstract Parameters to Describe Scenarios and Services



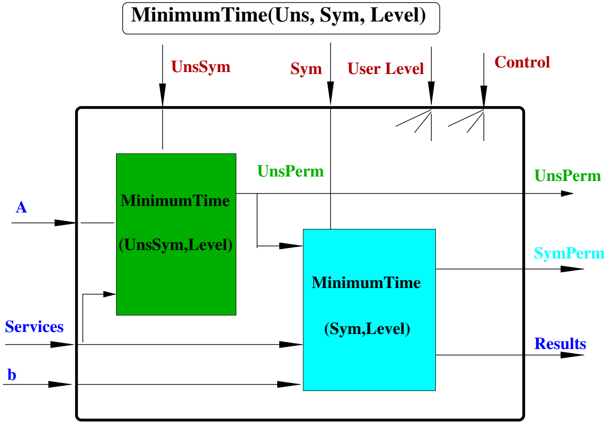
Use of Structured Abstract Parameters

- This structure for a parameter of type "enumeration":
  - defines a default representative at each level of the tree,
  - defines a default realization for each leaf of the tree.
- Application:
  - help to design even more dynamic server pages,
  - adapt to the level of the user (normal, expert, debugger),
  - limit cost of scenarios.

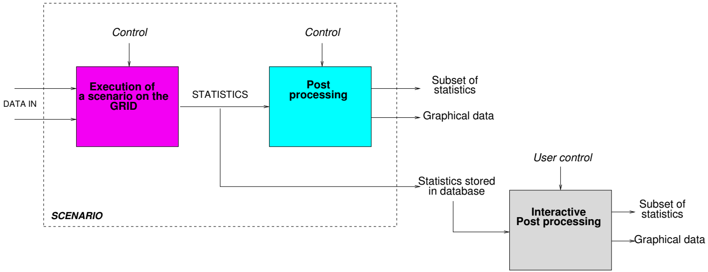


Building on a More Complex Scenario

For each selected solver, find best w.r.t. time **SymPerm** and **UnsPerm** to solve $(PQ_r)A(Q_cP^\top)(PQ_c{}^\top)x = (PQ_r)b$



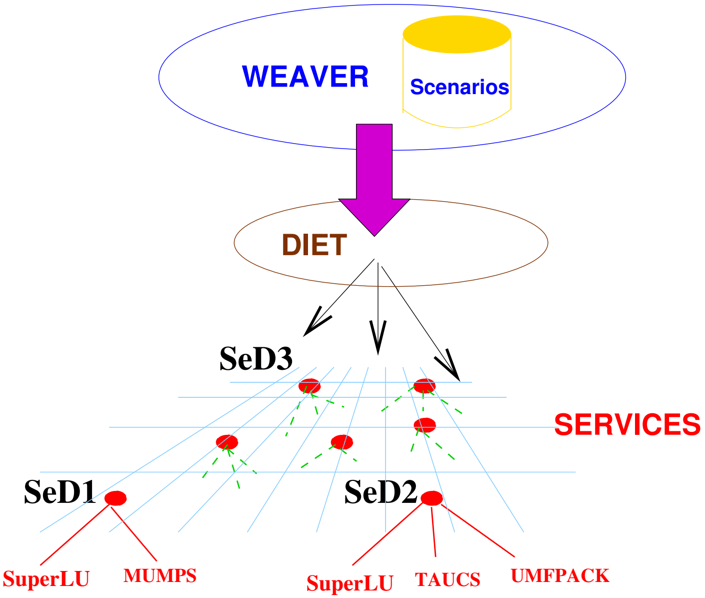
Post-Processing Facilities



Remarks on Post-Processing

- Both graphical and textual outputs may be provided.
- More statistics than requested are provided.
- Complete statistics produced by scenarios are stored in the database.
- Graphical navigation in the complete result set may be possible.

Managing Services



Managing Services: Constraints and Difficulties

- Services written with different languages: C, C++, F77, F90.
- Hundreds of services of different types: solver, validation, matrix generators.
- Same service on different computers.
- Same computer required within a set of experiments: time measures.
- Multiprocessor and batch management.
- Matrix availability/matrix transfer on computers.

Service Naming

- One service corresponds to one solver / solver package.
- Service naming: on computer C1 of type SPX on which Serv1 is installed
  - Serv1: DIET is free to choose
  - Serv1_SPX: Computer type imposed
  - Serv1_C1: choice done by WEAVER

Prototype (old version for demos) and its limitations

- Use of DIET facilities to define each service profile (typed list of in/in-out/out parameters, in memory).
- Execution of services within the same UNIX process: $\rightarrow$ Pb link phase + robustness (solver failure, memory leaks).
- Need of a common interface for all solvers: $\rightarrow$ union of in/out parameters of services.
- How to manage optional in/out parameters (permutations, ...) ?

Modified Version

- Matrices, permutations, scalings . . . are files
- One UNIX process per service (robustness, batch systems).
- Main parameters of DIET:
  - an XML input file,
  - an XML output file,
- One generic UNIX process per language
  - Read/analyse XML input file, (filled with abstract parameter names and values).
  - Match abstract parameter with effective service parameter.
  - Get matrix file and read it.
  - Service realisation.
  - Fill XML output file and send it back (or not ?) to the TLSE server.

Comments on Data management (prospective)

1. Matrix files (described by an URL),
2. Temporary data (scenarios),
3. Solver internal data (eg, several solution steps with same factors) ?

Data management: matrices

- Characteristics:
  - Matrix files can be large (a few Gigabytes)
  - Required by all services
  - Never modified (or maybe only once when a private matrix becomes public)
  - Each server (DIET SeD) manages a cache mechanism
- Natural approach with DIET = cache mechanism
  - Use DIET plugin schedulers to give priority to servers where matrix has already been downloaded.

    if matrix file is not in cache (on disk) **then**

    server_adequacy = "bad" (the SeD would have to first download the file)

    else

    server_adequacy = "good" (the matrix file is available)

    endif
- Requires the name of the matrix (unique) to be passed to the SeDs, as a string, in the evaluation

Data Management: temporary files between elementary requests

- Characteristics:
  - Output from an expertise step
  - Input from another expertise step
  - Persistency needed
- Example: scenario "ORDERING SENSIBILITY"
  - A number of services (MUMPS, UMFPACK, ...) first compute permutation files
  - Permutation files are then applied to various solvers on various solvers in order to perform the actual computations.
  - Once all runs performed:
    - Present results to the user (Web interface).
    - Clean all permutation files related to the global request.
- Use DIET persistency mechanism or JUXMEM ?
- XML output file contains all aggregated data files (permutations, scalings, with XML tags), or identifiers to those data files

Data Management: Solver internal data

Idea: use functional decomposition analysis, factor, and solve steps.

- Same analysis step $\longrightarrow$ different parameters for factorization.
- Same factors $\longrightarrow$ parametric study on the solution step.

Data Management: Solver internal data

Idea: use functional decomposition analysis, factor, and solve steps.

- Same analysis step $\rightarrow$ different parameters for factorization.
- Same factors $\rightarrow$ parametric study on the solution step.

- Requires solvers to be able to "dump" their memory (possibly distributed on several processors) after one functional step.
- Not currently possible for any of the solvers we know.

Concluding remarks

- Final site still under development
- The abstract parameters and the SeDs are still being specified.

  Goal=open a first version of TLSE to users in summer 2005.
- Optimal data management may be long term work.
- Demo with Juxmem will be with the old (not further developed) prototype.