# Management of Services based on a Semantic Description within the GRID-TLSE Project

Patrick Amestoy, Michel Daydé, Christophe Hamerling, Marc Pantel and Chiara Puglisi

TLSE Project*, IRIT-ENSEEIHT, 2 rue Camichel, 31071 Toulouse CEDEX, France
surname.name@enseeiht.fr
http://www.irit.enseeiht.fr/tlse

**Abstract.** The goal of the GRID-TLSE Project is to design an expert site that provides an easy access to a number of tools allowing comparative analysis of sparse matrix packages on a user-submitted problem, as well as on particular matrices from the matrix collection also available on the site.
When making available a large amount of software over a computational Grid, facilitating its deployment and its exploitation become crucial. Within the GRID-TLSE Project, we use a software component approach based on a high level semantic description of the scientific computing services. In this paper, we focus on one aspect of this description of the computational services: the use of meta-data called *abstract parameters*. Our approach allows the automatic discovery and the exploitation of new services throught the concept of *scenario*.

## 1 Introduction

The main goal of the GRID-TLSE Project is to design an expert site that provides an easy access to a number of direct solvers for solving sparse linear systems, allowing their comparative analysis on user-submitted problems, as well as on matrices from collections also available on the site. The site provides user assistance in choosing the right solver for its problems and appropriate values for the control parameters of the selected solver. It is also intended to be a testbed for experts in sparse linear algebra. A computational Grid is used to deal with all the runs arising from user requests. For more details see **http://www.irit.enseeiht.fr/tlse**.

The expert site asks the user through a WEB interface (called WebSolve) to describe his problem as well as, optionally, the characteristics of the computers and the software that he plans to use. The expertise kernel (called Weaver) takes into account the user requirements, the internal expertise scenarios and the Grid state to build experience plans which are run using the DIET middleware [3] ( http://graal.ens-lyon.fr/~diet/). The results and metrics are used to produce

* funded by the French Ministery of Research throught ACI «*Globalisation des Ressources Informatiques et des Données*»

synthetic graphics which help the user in choosing the best tools – and the corresponding value of control parameters – for his problem (according to some metric e.g. minimizing execution time).

In sparse linear algebra, similarly to other areas of scientific computing, there exists a lot of different algorithmic approaches for solving the same problem with different features and performance (e.g. several algorithmic variants for factorizing a sparse matrix).

As a consequence, the description of the computational services provided by each component is much more complex than usually advocated in software engineering (typically restricted to service name, type of input / output parameters). The computing services have functional parameters and results – as usual – but also make use of parameters and results for algorithmic control and execution metrics that depend on the numerical algorithms used. Controls (usually parameters) allow to adapt the algorithm to the user performance requirements. Metrics (usually results) provide the users insights on the results quality and on the way the computer was used.

We describe in the next sections the approach used within the GRID-TLSE Project. It has been initially designed for allowing experts in sparse linear algebra, that are not always grid computing specialists, to deploy easily software over the grid and to use it within the expertise process they describe using scenarios. This approach is generic and may be used in other areas.

## 2 Sparse direct solvers for linear systems

### 2.1 Sparse direct solvers

The main service used in the GRID-TLSE Project aims at solving $Ax = b$ where $A$ is sparse using direct solvers.

The direct approach for solving $Ax = b$ consists in factorizing the matrix $A$ into a product of simpler matrices (so called factors) and then computing the solution $x$. There exists different factorizations of $A$: $A = LU$, $A = QR$, $A = LL^\top$, $A = LDL^\top$, ....

Several algorithms can be used for solving the same linear system. They all use the same functional input parameters $A$ and $b$ and produce the same functional result $x$. However, they do not always have the same set of input / output parameters for algorithm control. They also provide execution metrics (execution time, amount of memory used, number of flops, ...) that may not be similar.

The performance of the sparse solvers depends on the exploitation of the structural and numerical properties of the matrix $A$ and on the target computing platform characteristics. For the sake of simplicity, we focus on the $LU$ factorization in the following sections.

### 2.2 Algorithm controls and execution metrics

A computational service may possess a lot of input / output parameters for algorithm controls and execution metrics that may vary with its implementation.

In the general case, $A$ is factorized into $PQ_RD_RAD_CQ_CP^\top$ where :

- $D_R$ and $D_C$ are diagonal scaling matrices for respectively rows and columns of $A$ ;
- $Q_R$ and $Q_C$ are unsymmetric permutations for respectively rows and columns. Solvers often use only one.
- $P$ is a symmetric permutation whose purpose is to reduce the size of the factors during the factorization of $A$.

The problem to be solved is then $\hat{A}\hat{x} = \hat{b}$ where $\hat{A} = PQ_RD_RAD_CQ_CP^\top$, $\hat{x} = PQ_C^\top D_C^{-1}x$ and $\hat{b} = PQ_RD_Rb$. These transformations are usually computed in the first phase of the algorithm referred to as symbolic analysis. The permutations and scalings are also performed during this step. Algorithmic control parameters are tuned according to the properties of the matrix for improving execution.

Depending on the software, the permutations are either symmetric ($P$), unsymmetric (either $Q_R$ or $Q_C$), left ($PQ_R$) or right ($Q_CP^\top$). Many algorithms - called orderings - are available for computing permutations, for example AMD (Approximate Minimum Degree [1]), Metis (graph partitioning [11]), MMD (Multiple Minimum Degree [12], Matrix bandwidth reduction [4]). Some packages provide several orderings and a control parameter is used to select one.

The $LU$ factorization of $\hat{A}$ is performed next. During this factorization phase, the static symmetric ordering $P$ can be completed by a dynamic ordering $P_N$ (referred to as the numerical permutation) monitored using a pivoting threshold. The linear system is then $P_N\hat{A}\hat{x} = P_N\hat{b}$. The pivoting threshold is not always available as an algorithm control.

The last step ("solve") computes $\hat{x}$ using the factors $L$ and $U$.

Most of the direct algorithms for solving a sparse linear problem are using these three steps (symbolic analysis, factorization and solve) in sequence. It is therefore possible to share the symbolic analysis between several factorizations (with different values for the pivoting threshold) and to share a factorization between several solves (with different values of $b$). One of the main benefit is to be able to use the ordering available within one sparse solver as an input for the factorization of another solver. This implies that a functional description of the package must be available to be able to call separately ordering, factorization and solve and to recover the corresponding outputs.

## 3 The GRID-TLSE reflexive approach

We use a component approach with a dynamic discovery of component characteristics. This approach relies on meta-data – called *abstract parameters* - describing all the possible features for all available service implementations. This approach is usually referred to as reflexive as it relies on services managing services. Note that one package may be deployed in several places and several versions, i.e. there may be several services implementing the same sofware.

There are two kinds of services within the GRID-TLSE Project:

- Computational services that correspond to sparse softwares or tools for processing sparse matrices (visualization, ...)
- Scenarios that are a high level level description of the expertise process. The interpretation of scenarios by the Weaver software layer generates the workflows executed over the Grid. Scenarios are specified by sparse linear algebra experts.

## 4 Use of abstract parameters for describing services

From the Web interface to define the objective and parameters of the user request up to the construction of scenarios, we use the same description of services based on common meta-data.

To describe a computational service, we specify:

- its functionalities: assembled/elemental entries, type of factorisations ($LU$, $LDL^T$,$QR$), multiprocessor, multiple Right-Hand-Side Members, ...;
- and its algorithmic properties: unsymmetric/symmetric solver, multifrontal, left/right looking, pivoting strategy, ....

To describe a scenario, in addition to service input / output parameters (as usual), we specify:

- its execution metrics sent back by the solver executions: memory, numerical precision, execution time, ...
- its control: type of graphic visualization for post-processing, level of user (expert, non-expert, intermediate user).

More expert is the user, more control he may have on the parameters of the expertise process.

### 4.1 Expressing dependencies between abstract parameters

The abstract parameters are used to express constraints and/or relations that forms the basis of the expertise scenario. We can thus express qualitative and quantitative dependencies between values of metrics and control parameters within scenarios. This feature allows to limit the combinatorial explosion inherents to the expertise process. Here are some examples:

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

The first item illustrates how it is possible to limit the number of experiments performed over the grid: when the user is non-expert and when the targert matrix is symmetric, only symmetric solvers are experimented (while an expert user may want to run an unsymmetric solver on a symmetric matrix).

### 4.2 Example: description of the MUMPS software

We illustrate this by considering the MUMPS software ([2]). The abstract parameters describing this software (this is not an exhaustive list) looks like:

- Functional decomposition: Symbolic analysis, Factorization, Solve (the three steps are available and can be called independently)
- Control parameters: Symmetric Permutation, Unsymmetric Permutation, Pivoting Threshold
- Possible values : Symmetric Permutations available are { AMD, Metis, ... }. Unsymmetric Permutations are ...
- Metrics: estimated flops from the symbolic analysis, effective time for the whole solution, numerical precision after the solve, ...
- Dependency: numerical precision depends on the pivoting threshold values

### 4.3 Structuring Abstract Parameters: illustration with symmetric permutations

An ordering is a heuristic to permute the graph of the initial matrix with the aim to limit the cost of the numerical factorization; the ordering has a strong impact on both the number of operations and the memory used by a solver. Orderings involves symmetric or unsymmetric permutations. We focus on the abstract parameter associated to symmetric permutations.

The abstract parameter **SymPerm** that corresponds to symmetric permutations is implemented as an enumeration of large size. One of the symmetric ordering often used is the Approximate Minimum Degree (AMD [1]) available in MUMPS and other sparse solvers. Each software may have its own implementation of the AMD ordering. One representative of the set of AMD implementations over all the sparse solvers available might be enough in most cases but they may perform differently. How to define/select a representative implementation of AMD since it may change from time to time is a quite complex issue.

Furthermore when studying the impact of using various symmetric orderings, one may not want to test all possible values of the symmetric permutation. On some matrices a subclass of orderings may be known to be superior. A non-expert user only wants to capture major differences between orderings, thus using a "good" representative of a subclass may be enough. This is a crucial issues for limiting the combinatorial complexity of this expertise process (i.e. avoiding to explore / execute all possibilities).

The "permutation" abstract parameter can be represented as a tree where:

- we define a default representative at each level of the tree,

– and a default realization for each leaf of the tree.

When managing expertise scenarios, it helps in designing more dynamic server pages by adapting the web pages to the level of the user (normal, expert, debugger), and in limiting cost of scenarios.

Figure 1 illustrates the structuration of the abstract parameter corresponding to permutations (only the symmetric permutation subtree is detailled).

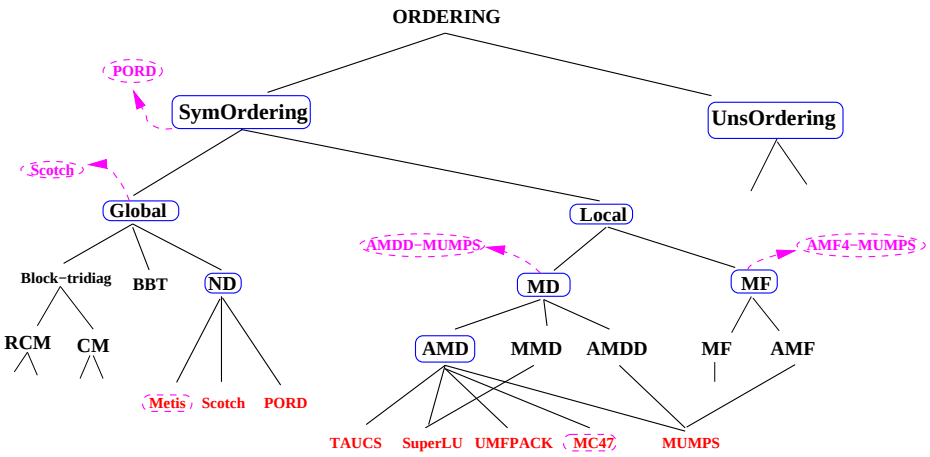


**Fig. 1.** Structuring the Permutation Abstract Parameter

## 5 Using abstract parameters within the GRID-TLSE Project

The TLSE Weaver expertise kernel relies on two levels of services : the expertise scenarios exploited by users and the solvers used by scenarios and experts.

Extensibility is a key point in TLSE : new scenarios and new solvers will be integrated in the expertise site regularly. New scenarios should be able to use old solvers and new solvers should be used by old scenarios without modification. Modifications should only be required if scenarios want to use new specific features from solvers.

All services do not provide the whole set of controls and metrics. Input / output parameters should then be optional with either default values, values computed by other services or values explicitly provided. Tools may use or produce values in a slightly different manner for the same control or metric. It is therefore necessary to add a wrapper around each tool in order to adapt its real interface to the common one. New solvers may provide additional controls and metrics. Their interfaces should therefore be extensible.

The solution chosen in the TLSE Project relies on the definition of an easy to extend set of features for each service which will be wrapped around each tool. Scenarios are then using these features.

The meta-data framework used within TLSE can be summarized as follows:

- Solvers are described using meta-data and wrappers translate meta-data values to/from solver's parameters and results.
- Scenarios require solvers to provide specific meta-data and process experiments which are sets of meta-data.
- The middleware exchanges sets of meta-data with the wrappers of solvers.
- The Web interface is dynamically built from scenarios and their corresponding meta-data and solver meta-data and their values.

The service profile is composed of an abstract parameter set. It qualifies the following aspects of the service : the name of the tool; the service semantics; the functional parameters and results; the parameters and results for algorithm control; the parameters and results for execution metrics.

Each abstract parameter is defined using:

- its values (type, possible values, variation (linear, logarithmic, normal, Gaussian, ...));
- its mode : input or output;
- its constraint : mandatory, optional, with default value, with value computable by another service;
- the expertise level of the users (novice, standard, advanced, expert, manager);
- some documentation related to its purpose (several levels may be defined according to the user level);
- dependencies with other features for expressing incompatibilities, dependence upon a parameter and other constraints.

It is quite similar to an interface in the component world but extended in order to enable an easy integration of the tools that provide the same service with quite different algorithms (therefore different controls and metrics).

## 6 Use of Abstract Parameters within Expertise scenarios

The expertise scenarios are used by the expertise kernel to build experience plans according to the user request. These experience plans are worflows executed over the Grid. The results of one experience plans may be used to biuld the next experience plan and thus the workflows executed are dynamic since they may depend of results of a previous executions.

The scenarios are structured hierarchically in a dataflow like approach. Scenario inputs and outputs are connected to the sub-scenario inputs and outputs. It can also contain internal links between sub-scenario inputs and outputs. A scenario may also use internal operators for creation, modification, execution and filtering of experience plans. A given scenario may then build several internal

experience plans, executes these plans, and finally produces new plans depending on the results from the previous ones. Scenarios are therefore fully dynamic and may depend on the availability of services and the results of experiences in order to generate new experiences. In order to ensure that a scenario will stop, there must be no internal cyclic links between sub-scenarios. Experience plan creation and execution operators use service description in order to assign a value to experience abstract parameters. Some instances may not qualify if some of their abstract parameters have values that are different from the ones required in the experiences.

The "Ordering sensitivity" scenario consists into studying the effect of using the available orderings on the solution of the linear systems in terms of the metrics selected by the user (execution time, memory, number of flops, ...). We only generates runs for default solvers (defined by experts), which is some kind of leaf cleaning and limits the combinatorial complexity of the expertise. The first box called "AllOrdering" corresponds the search of all available symmetric orderings. The second box, called "Exec", requires the executions of all the permutations sent by first box. The final results, in term of the metrics asked by the user, are then produced in a graphical way. The scenario is described by expert using a graphical interface called "GEOS" that is interpreted by Weaver to build experience plans. It is reported in Figure 2.

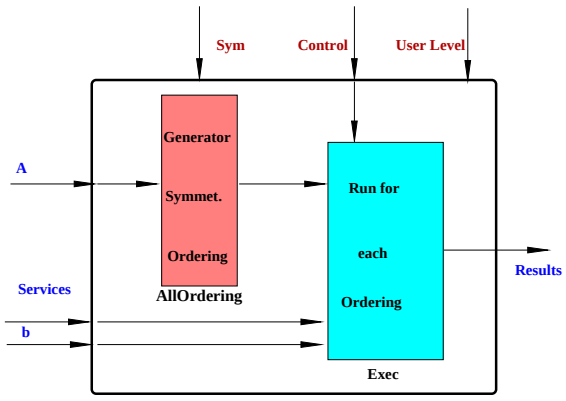


**Fig. 2.** Ordering Sensitivity Scenario.

In the "Minimum time" scenario displayed in Figure 3, we try to identify which combination of ordering and factorization achieves the best execution time. Some branch cleaning is effected by selecting only one possibility at each level of tree of available permutations. This is expressed by the sequence of the two boxes: "AllOrdering" (used in the previous example) and "Select". We then only execute the default solvers (defined by the sparse linear algebra experts) which corresponds to some leaf cleaning.

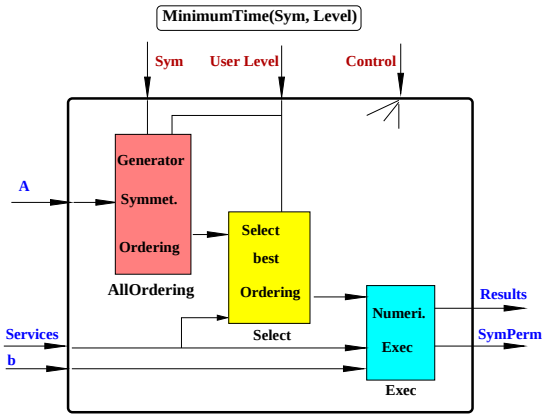


**Fig. 3.** Minimum Time Scenario.

## 7 Conclusion

We have described the main aspects of the component framework used in the GRID-TLSE Project. This high level description of scientific software is used within the scenarios for generating the dynamic workflows that perform expertise. The main benefit is that adding / removing solvers does not require to update scenarios (they will be automatically discovered). New scenarios make use of all the deployed softwares.

This type of reflexive approach is commonly used for the dynamic discovery of services (for example, in the Java language or the Corba middleware). Similar approaches have been described for the use of object-oriented technologies for scientific computations in order to combine several algorithmic solutions: for example centralized and distributed matrix structures, (see F. Guidec [8], E. Noulard and N. Emad [6]), the SANS (Self Adapting Numerical Software) Project (see [5]), or the Salsa Project (see [7]).

Our component framework combines two approaches : a static approach for accessing the functional parameters and the results for a given service; a dynamic approach for accessing the controls and metrics of a service. The set of meta-data used within the TLSE Project can be easily extended which is not always the case in the approaches mentioned above.

The GRID-TLSE Project focus on sparse solvers. The corresponding abstract parameters are defined using a graphical interface called PRUNE. Adding abstract parameters or specifying an entire set of new parameters is easy. As a consequence, the approach described in this paper can be extended to other areas providing that an adequate set of abstract parameters has been derived.

An important requirement in our approach is to be able to give an accurate description of the computation done by a given service according to its functional parameters and results. The service semantics could also be described using algebraic specification technologies. This semantics could then be used for service

trading. This point is currently under investigation (see [10]). The use of accurate semantics allows to combine basic services in order to provide more sophisticated ones. This trading approach can also be combined with a scheduler for finding the best service combination (see [9]).

## References


1. P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905, 1996.
2. P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng.*, 184:501–520, 2000.
3. E. Caron, F. Desprez, E. Fleury, F. Lombard, J.-M. Nicod, M. Quinson, and F. Suter. Une approche hiérarchique des serveurs de calculs. In Françoise Baude, editor, *Calcul réparti à grande échelle*, pages 129–151. Hermès Science Paris, 2002. ISBN 2-7462-0472-X.
4. E. Cuthill. Several strategies for reducing the bandwidth of matrices. In D. J. Rose and R. A. Willoughby, editors, *Sparse Matrices and Their Applications*, New York, 1972. Plenum Press.
5. J. Dongarra and V. Eijkhout. Self-adapting numerical software and automatic tuning of heuristics. In *Proceedings of the International Conference on Computational Science, June 2–4 2003, St. Petersburg (Russia) and Melbourne (Australia)*, 2003.
6. N. Emad E. Noulard. A key for reusable parallel linear algebra software. *Parallel Computing*, 27(10):1299–1319, 2001.
7. Victor Eijkhout and Erika Fuentes. A proposed standard for numerical metadata. Technical Report ICL-UT-03-02, Innovative Computing Laboratory, University of Tennessee, 2003.
8. F. Guidec. Object-Oriented Parallel Software Components for Supercomputing. In Peters D'Hollander, Joubert and Trystram, editors, *Parallel Computing: State of the Art and Perspectives. Proceedings of PARCO'95 (Parallel Computing), Gent, Belgium*, Advances in Parallel Computing. North-Holland, 1995.
9. A. Hurault, M. Pantel, and F. Desprez. Recherche de services en algèbre linéaire sur une grille. 5-8 Avril 2005. Rencontres Francophones en Parallélisme, Architecture, Système et Composant (RenPar'16), Croisic, (France).
10. Aurélie Hurault and Marc Pantel. Mathematical service trading based on equational matching. In *Calculemus 2005, 12th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, Newcastle, United Kingdom*, July 18-19 2005.
11. G. Karypis and V. Kumar. MeTiS – *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices – Version 4.0.* University of Minnesota, September 1998.
12. J. W. H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11(2):141–153, 1985.