SaSHiMi : Sparse direct Solvers using Hierarchical Matrices

Résolution de systèmes linéaires creux exploitant des matrices hiérarchiques

Défi 7 > Axe 5 > Simulation numérique intensive pour comprendre, pour optimiser, pour décider

Mathieu Faverge
HiePACS project-team
Inria Bordeaux - Sud-Ouest

Appel à projets générique 2018

Jeunes chercheuses et jeunes chercheurs

Contents

Main Changes with regards to the pre-proposition With respect to the pre-proposition, the only changes we made is the additional funding of two intern students to initiate the work of the PhD and to investigate the support of accelerators. The budget has been adjusted based on updated costs.

Summary table of persons involved in the project:

Partner	Name	Current Position	Contribution *research time*	Involvement throughout the project's total duration
Inria	Mathieu FAVERGE	McF - Bordeaux-INP	Project coordinator	24PM (50%)
Inria	Pierre RAMET	McF - Univ. Bordeaux	He will help in supervising the PhD student in Tasks 1 and 3	6PM
Inria	Aurélien ESNARD	McF - Univ. Bordeaux	He will help in supervising the PhD student in Task 2	6PM
Inria	XXX	PhD student funded by ANR	He will work on Tasks 1, 2, and 3	36PM
Inria	XXX	Engineer funded by ANR	He will work on Task 3	12PM
Inria	XXX	Research intern funded by ANR	He will work on Subtask 1.1	6PM
Inria	XXX	Research intern funded by ANR	He will work on Subtask 1.2	6PM

1. Proposal's context, positioning and objective(s)

The race to exascale supercomputer systems has led to a large increase in the number of computational resources, and especially in large vectored units present in accelerators like GPUs or Intel KNL. However, this increase in the number of computational units is not followed by a rise of the amount of memory available on those systems. For example, the actual leader of the Top500 list, the system Sunway TaihuLight, is built with 1,3PB of memory for more than 10 million cores, which represents less than 128MB per core, versus an average of 2,5GB per core on systems with less accelerators like Occigen2, the current supercomputer at CINES.

This trend in the **reduction of the amount of memory available** per core forces applications not only to be extremely well parallelized, but also to reduce their memory consumption. In many scientific and engineering applications arises the need to **solve large sparse systems** of linear equations $Ax = b$, which is a crucial and time-consuming step. Many algorithmic solutions exist to solve those systems providing various ranges of numerical precision, robustness, memory consumption, and parallelism. Among them, the use of sparse **direct solvers remains the most robust solution** but their downside is their large memory consumption and time to solution.

In order to tackle very large challenging problems, research needs to be undertaken to improve sparse direct solvers that are strategic to this goal. In the last five years, those solvers have been drastically changed by introducing **compression techniques** to exploit the **low-rank** properties of the **matrices**. Many compression formats such as Block Low-Rank (BLR), $\mathcal{H}$, $\mathcal{H}^2$, HSS, HODLR... are available. They all allow a reduction of the memory requirement and/or the time to solution to solve sparse or dense linear systems with a reduced precision or to be used as a fast preconditioner. Depending on the compression strategy, solvers may require the knowledge of the underlying problem or may do it in a purely algebraic fashion. Some have been shown to be efficient while other are difficult to implement with good scalability. **The SaSHiMi project targets a purely algebraic method** to target the largest spectrum of applications possible, with a **hierarchical low-rank format** that will provide block structures to enable the **use of BLAS 3 kernels** and a **good scalability** of the solution proposed. After detailing the related work on compression techniques for dense and sparse matrices, we present the proposed method and the risks we may encounter. Finally, we discuss ordering techniques that will be investigated to reduce the complexity of using hierarchical formats in sparse direct solvers.

1.1. Originality and relevance in relation to the state of the art

Matrix compression techniques rely on the fact that most of the matrices arising from applications are low-rank matrices by construction, and that the Schur complement exploited by the solvers is low-rank too. The main idea of the low-rank compression is based on the most basic format that consists in representing a matrix A of size m-by-n and rank r by the product UV^T, where U, and V are respectively m-by-r, and n-by-r. Thus, if $r(m+n) \leq mn$, the low-rank form of A will be cheaper to store than A. The rank r is chosen accordingly to the accuracy prescribed to approximate A, such that $||A - UV^T|| < \epsilon||A||$ with ϵ the precision of this low-rank approximation.

Based on this idea, multiple matrix formats have emerged to exploit this low-rank property in solvers. They can be characterized by three properties: the block admissibility condition, the block clustering, and the low-rank basis. When considering a given clustering of the matrix, the block admissibility condition defines which off-diagonal blocks are admissible for low-rank compression. A strong admissibility criterion will allow only far distance interaction to be admissible, while a weak admissibility will consider all off-diagonal blocks to be admissible. In both cases, the diagonal blocks associated with a block clustering are always considered non admissible. The block clustering defines the matrix partitioning and can be defined as *flat*, the matrix is partitioned in blocks of similar sizes on one level, this is the BLR format [2, 4], or as *hierarchical*, the matrix is recursively partitioned on multiple levels. Usually bi-partitioning is applied to compute the clustering recursively. In this category, two strategies are possible at each level of the clustering refinement. The first strategy refines only the diagonal blocks and compresses off-diagonal blocks at each level without further splitting. This is the Hierarchically Off-Diagonal Low-Rank (HODLR) format described in [5]. The second strategy refines all blocks based on the admissibility criterion at each level. This is the most general format, the $\mathcal{H}$-matrix format [10, 12, 18]. In order to reduce again, the memory and computational cost of factorizing dense matrices, nested bases have been introduced to represent the low-rank form of the off-diagonal blocks. This is done in the general hierarchical nested format $\mathcal{H}^2$ [12], and in the particular case of weak admissibility criterion, the format is referred to as the Hierarchically Semi-Separable format (HSS) [15, 30].

All those formats have been widely studied in the case of dense matrices to provide efficient and cheap preconditioners or direct solvers when low numerical accuracy is needed by the application. Thanks to those formats, the computational and memory costs of dense matrix factorization have been drastically reduced to go from $\mathcal{O}(n^3)$ time complexity to $\mathcal{O}(r^2 n)$ where r is the rank of the matrix, and respectively from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ for the memory. Additionally to the works cited above, BLR compression has been investigated for dense matrices in an industrial context in [7], and [27] (CEA) to speed-up dense matrix factorization. Another industrial study has been pursued on $\mathcal{H}$-matrices at Airbus [21]. This shows that those solvers are of great importance to both academic and industrial communities as they will allow for larger problems to be solved while keeping the memory consumption low.

In the family of sparse direct solvers, the works on integrating low-rank matrices are more recent, and the difficulty is increased, as it is not always possible to get the expected gains. The initial lower complexity of the solver ($\mathcal{O}(n^2)$ vs $\mathcal{O}(n^3)$) in sparse makes it more difficult to hide the computational overhead that we may encounter during the compression of the blocks. Furthermore, the block structure of the sparse matrices enforces constraints that harden the reduction of the number of flops and may reduce the number of efficient block operations based on BLAS 3 kernels to BLAS 2, reducing the overall efficiency of the solver. The objective of the project is to develop a solver with a low-rank structure that keeps the efficiency of the block structure, with a method that provides good parallelism for scalability. In the field of sparse direct solvers, two methods to perform the matrix factorization coexist with both their pros and cons: the multifrontal and the supernodal method. Both methods rely on the construction, by an ordering library, of a so-called elimination tree that defines the order in which computations will be performed during the factorization with a bottom-up traversal of the tree. The main difference between the two methods resides in the way contributions from children to ancestors are performed. In the supernodal method, for each node in the elimination tree, we will consider all its contributions to all its ancestors independently. As in dense linear algebra solvers,

Table 1: Complexities of multifrontal sparse direct solvers with different compression formats in terms of memory and operations for 3D problems of size n where the rank is assumed to be $\mathcal{O}(n^{1/3})$ [23, 29]

	Full-rank	BLR	HODLR / HSS	$\mathcal{H}$ / $\mathcal{H}^2$
Memory	$\mathcal{O}(n^{4/3})$	$\mathcal{O}(n^{7/6})$	$\mathcal{O}(n^{7/6})$	$\mathcal{O}(n \ln n)$
Flops	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{5/3})$	$\mathcal{O}(n^{4/3})$	$\mathcal{O}(n^{4/3})$

it exists two scheduling strategies for these contributions. The right-looking strategy applies all the contributions from a node to possibly all its ancestors, and the left-looking strategy gathers for each node all the contributions from possibly all the descendants. The first one favors the parallelism of the independent updates, the second one favors the locality of the written data on which contributions are reduced. In the multifrontal method, each node will access only its direct children to gather both the local contributions and those for its ancestors in a so-called frontal matrix. The Schur part of this matrix is transferred from child to parent all along the elimination tree. Thus, the multifrontal method favors the locality of the access, but need extra-storage for the *fronts*. In both methods, the first level of parallelism comes from the elimination tree, the second level comes from the parallelism of the updates in the supernodal method, and from the large dense frontal matrices that can be factorized in parallel in the multifrontal method. Furthermore, the multifrontal intrinsically uses more memory but provides a simpler structure to perform more advanced numerical strategy such as pivoting, while the supernodal method has no extra-memory requirement and exposes more parallelism, thus providing a better solution for scalability, but is more difficult to apprehend to integrate pivoting strategies.

When introducing low-rank compression techniques into sparse direct solvers, the multifrontal method has been largely preferred and investigated in recent works to explore a large family of compression methods. Table 1 summarizes the asymptotic complexities of sparse direct solvers using low-rank compression that have been studied. For example, the BLR compression format has been successfully introduced and widely studied in the MUMPS solver [2, 3] to reduce the memory peak of the solver and the time to solution of both the factorization and the solve steps. In this work, the current strategy consists in compressing the frontal matrices at each level of the tree which reduces the memory cost of the fronts, and the computational cost of the solver. The final size of the factorized matrix is well reduced by the compression strategy, however the memory overhead inherent to the fronts allocation in the multifrontal method will always exist. The actual strategy of the MUMPS solver, and of other multifrontal solvers, does not reduce this cost as the fronts are allocated in full-rank before being compressed. To reduce this overhead, two main strategies exist: allocating the fronts panel per panel to reduce the peak, however it also reduces the available parallelism in the front, or performing the assembly directly in low-rank format. This last solution may increase the computational complexity of the assembly step as contributions from children do not necessarily match and overlap. In [23], the complexity of the MUMPS BLR solver is given in terms of flops as $\mathcal{O}(n^{4/3}r)$, and final memory as $\mathcal{O}(n \max(\sqrt{r}, \log(n)))$ of the factorized matrix for a problem of size n. In 3D, the rank r is considered to be $\mathcal{O}(r^{1/3})$ giving the complexities in Table 1. This is the main concurrent project to what is proposed in the SaSHiMi project. MUMPS is widely used and distributed and, thanks to the multifrontal strategy, is able to apply pivoting strategies, even when using the BLR format, to provide good numerical accuracy on difficult problems.

Attempts to introduce more complex hierarchical compression schemes have been done in other multifrontal solvers. Aminfar et al [5, 6] have proposed a prototype using the Hierarchically Off-Diagonal Low-Rank (HODLR) in a multifrontal sparse direct solver as a proof of concept to accelerate the elimination of large fronts. In [6], fronts are assembled directly in a compressed format thanks to the preselection of rows and columns based on the adjacency graph structure of the sparse matrix. Thus, the assembly of the children contribution is much cheaper in memory than the full-rank solver. This solution diminishes the complexity of the HODLR matrices assembly, but the final precision is low when the application does not possess the weak admissibility criterion.

Multiple ongoing works are trying to exploit the Hierarchically Semi-Separable (HSS) format in

sparse multifrontal solvers. In [31], Xia et al presented a sequential solver for large structured linear systems of equations. In this work, they were able to take advantage from the structure of the problem to perform the front assembly in the HSS format presenting a near linear solver for 2D problems. One of the drawbacks was their limitation to 2D problems and the requirement for knowledge of the problem rank property. In [29], Xia extended this work to describe an efficient structured multifrontal factorization for general matrices. However, to reduce the complexity of the front assembly, Schur complements are kept in full-rank. Thus, the memory consumption might temporarily increased above the memory complexity of the factorized matrix. A parallel version of this work has been implemented in [28], and further refined in the StrumPACK solver [16, 17]. In its last version, this solver uses HSS matrices for all fronts, and performs the so-called extend-add operation (addition of two low-rank matrices together) involved in the assembly directly in the HSS format thanks to randomization algorithms, and rows/columns preselection. This solver is a parallel implementation and uses MPI for distributed memory, and OpenMP threads to distribute the work within each node, as opposed to MUMPS which mainly profits from multi-threaded BLAS kernels. In chapter 8 of [23], the comparison between these two solvers has shown that for now the StrumPACK solver struggles to compete with the MUMPS BLR solver when the difficulty of the problem arises. However, on very large and easy problems, the HSS format wins over the BLR format giving the advantage to StrumPACK. The difficulty is thus to reduce the memory and computational complexities of the solver while keeping it numerically stable to provide a good preconditioner.

As opposed to this various research on the multifrontal method, only a few works have been produced for supernodal methods. In [14], Chadwick and Bindel presented a left-looking supernodal solver based on the Cholmod solver and using low-rank compression. The main objective of this work is to accelerate the solver for a local usage on a laptop only, and not to study scalability at large scale. Thus, this is only a sequential algorithm, with potentially multi-threaded BLAS. In this work, off-diagonal blocks are compacted together and compressed in a single low-rank matrix representation, and large diagonal blocks are compressed in HODLR format. The Schur complements associated with each supernode are allocated in full-rank one by one before being compressed with randomization techniques. The main contribution regarding supernodal methods has been developed in the PaStiX solver [26], and uses the flat, as opposed to hierarchical, BLR format as in the MUMPS solver. This solution has been implemented in the new release of the solver and benefits from the support of the runtime systems PaRSEC [13] and StarPU [8] for the parallelism. In PaStiX, as opposed to the multifrontal method who often favors multi-threaded BLAS to factorize the fronts, the parallelism is described as elementary sequential BLAS tasks which are either statically or dynamically scheduled on the architecture and benefit from the large degree of parallelism of the right-looking supernodal method. As of today, only the shared memory version of the solver is available and the distributed memory implementation is under development. Two BLR strategies have been proposed in this solver. In both strategies, the large diagonal blocks of each supernode of the elimination tree are compressed in a BLR format as for a dense matrix, and each off-diagonal block of the interactions between those supernodes are compressed independently in a low-rank form.

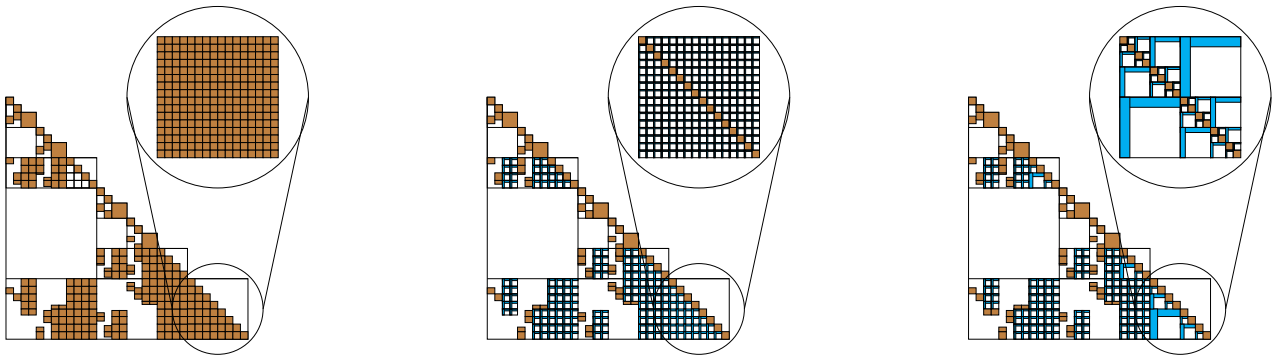
• The first strategy, called *Just-In-Time*, exploits the low-rank property to accelerate both the factorization and the solve steps, but does not reduce the memory peak of the solver. In this strategy, blocks are compressed only when they have been fully updated by their descendants and before they are factorized. Thus, this strategy is really close to the MUMPS *FCSU* (Factor, Compress, Solve, Update) strategy, and takes advantage of the lower cost of the updates to reduce the factorization time.

• The second strategy that has been proposed, called *Minimal-Memory*, is more ambitious as it considers the matrix compressed as input and works only on low-rank structures. Thus, the memory peak of the solver is drastically reduced as there is no assembly of the contributions in full-rank as it is done in most multifrontal solutions for simplicity. Thanks to this method, we managed to solve a 1 billion unknowns Laplacian at an accuracy of $1e^{-6}$ with $2.5PB$ of memory in 6 hours on a 96 cores with $3PB$ of memory machine, saving more than $10PB$ of memory that would have been required in full-rank. The difficulty of this method is the so-called extend-add operation that adds two low-

rank matrices together and which might add extra-cost to the solver. When considering a compressed dense matrix, the block clustering gives a regular partition of the matrix that simplifies the extend-add operation. This regularity is lost when considering sparse matrices, and more especially when using the right-looking supernodal method to solve the system. This irregularity may induce extra-flops when computing the updates from interactions between supernodes. No theoretical proof of a bound to those updates have been found yet. However, we have shown experimentally on Laplacian 3D grids that asymptotically this does not impact the flops reduction gained with the BLR format.

1.2. Objectives and scientific hypotheses

The objective of the SaSHiMi project is to **investigate a genuine hierarchical data sparse format in the context of supernodal sparse direct solvers** to extend the work done within the ongoing PhD thesis of G. Pichon [26]. The envisioned evolution of the solver is presented by Figure 1, with from left to right, the original full-rank storage format, the Block low-rank format from the preliminary work, and the hierarchical format that this project aims at study. The goal of the project with this format, as in the preliminary work, is to provide a **black-box algebraic solver** which does not require knowledge on the problem's origin to perform the compression. The second objective is to choose a not too complex format to be **able to benefit from high performance BLAS 3** kernels. The algebraic aspect is important to target the largest spectrum of applications as possible. Even if applications solving PDE will be the main beneficiary of this work, the algebraic implementation will allow for experimentation on any class of problems. The hierarchical formats represent promising and more advanced solutions as opposed to the flat BLR format in terms of computational complexity and memory consumption as the Table 1 summarizes it, however we can observe that the nested bases do not improve the asymptotic complexity of the solver in the sparse case. Thus, a much simpler format like HODLR or general $\mathcal{H}$ will be preferred. A good hierarchical format correctly exploited should provide a breakthrough to solve problems with millions of unknown on a single node, and potentially billions of unknowns on parallel distributed platforms. Indeed, hierarchical formats offer the possibility to expose more data sparsity in the matrices than the flat formats, and thus reduce the time to solution and offers a higher compression ratio than a flat compression format as BLR. Some of them have been investigated in the context of multifrontal solvers (HODLR, HSS) but they have not yet been investigated in a supernodal framework. We believe that doing it in a supernodal framework is the direction to study as it is the most memory-saving strategy for two reasons. First, it avoids the memory extra cost of allocating the fronts inherent to the multifrontal method. Second, the low-rank assembly prevents to form dense blocks before compressing them. This last overhead might be controlled in a flat format, but grows with hierarchical formats due to the recursive clustering. Furthermore, the right-looking supernodal method provides by nature a more important and flexible parallelism, which is essential to a good scalability for the large problems targeted, and for the emerging extreme scale platforms.



(a) Full-rank (b) Block low-rank (c) HODLR format

Figure 1: Objective of evolution of sparse matrix storage format for the SaSHiMi project. On the left, the original full-rank solver, on the middle, the preliminary work on BLR format, and on the right, one of possible the hierarchical format targeted by the project.

Among all those formats, the most advanced formats, such as HSS and $\mathcal{H}^2$, offer the best asymptotic complexities and seem really promising in dense. However, the first one is really efficient only for very large and simple problems as shown in the comparison with the MUMPS solver [23]. And for both, the complexity intrinsic to those format make them really hard to implement efficiently in the dense case. Thus, it seems too complex and too early to exploit them in a supernodal framework, and particularly since their asymptotic complexities are identical in sparse solvers (Table 1). In this project, we will consider the $\mathcal{H}$ format, and more especially the HODLR subclass. This simpler format with non-nested bases makes it easier to develop efficient kernels based on BLAS 3 operations, while formats with nested bases force to have more memory bound kernels due to their row/column selection.

Reducing the complexity and the memory footprint of the solver will not be enough. An efficient parallel implementation of the solver is mandatory to scale on large distributed systems. The low-rank formats, by nature, do not have the regular pattern of computation of the dense matrix kernels. The use of hierarchical matrices, with irregular sizes like the HODLR format, will break even more this regularity of the computing tasks than with the BLR format. This will also induce more complex task dependencies to take into account the levels of the hierarchical format. That is why all the results of the SaSHiMi project will be integrated in the PaStiX solver. This solver has been recently extended to provide alternative solutions to exploit **heterogeneous architectures** thanks to a **task based implementation on top of generic runtime systems** [20] which brings efficient support for accelerators. Prospective work has been made to support distributed systems with those runtimes in [19] and is currently under integration. This project expects to benefit from this flexible implementation to cope with the irregularity and complexity of exploiting hierarchical matrices, and provide an efficient solution on distributed, and potentially heterogeneous, architectures. Thus, the results of the research effort on hierarchical compression formats will not only study the shared memory impact, but also their impact on the distributed aspect of the solver in terms of scheduling and data exchange. In today's architectures, a key aspect in distributed algorithms is the volume of communication. The low-rank compression scheme will naturally reduce the volume of communication, thus improving again the scalability of the solver.

All the results will be developed in the PaStiX solver, and will be publicly available on the repository of the library (https://gitlab.inria.fr/solverstack/pastix). The PaStiX library is available through different interfaces such as Fortran90, Python, PETSc [9], and Eigen [11]. We expect that moving toward a **hierarchical format** would allow sparse direct solvers and the applications to **break down the memory limit** they encounter. In most cases, this improvement will come for free for applications using the PaStiX library through these interfaces. However, we will also collaborate with external projects which are already using PaStiX to integrate this work in hybrid direct-iterative solvers such as MaPHyS [1] and HORSE [22], and in more complex applications like Jorek and Tokam3X from CEA Cadarache, Alya from BSC, or applications from Cerfacs and Total.

1.3. Methodology and risk management

The methodology that will be used to develop the final solution targeted by the project, a supernodal sparse direct solver exploiting a hierarchical matrix format, will distinguish two cases for the blocks of the factorized sparse matrix. The large diagonal blocks corresponding to supernodes in the elimination tree will be compressed in the hierarchical format, as shown in Figure 1. The off-diagonal blocks of interactions between those supernodes will be simply compressed as block low-ranks following the clustering of the facing diagonal blocks. The **main risk** of the SaSHiMi project with this compression scheme resides in the capacity of developing an **efficient** solution to perform **low-rank to low-rank updates**, or extend-add, in data sparse format of the interaction blocks onto the hierarchically compressed diagonal blocks. To minimize the risks, the project will study different strategies to reduce this constraints and improve the efficiency which can all be studied independently from each other.

As a starting point, similarly to the BLR format, two strategies will be developed. The *Just-In-Time* strategy that compresses the factorized blocks once they have been fully computed and just

before they are used to perform updates. This solution is similar to what is exploited in multi-frontal method. Only dense matrices are compressed. This is a well-known problem as described in Subsection 1.1, and no complex low-rank to low-rank updates are performed. This solution will efficiently reduce the time to solution using simple BLAS-3 operations of lower complexity, but it will not reduce the memory peak of the solver.

Quite the opposite, implementing the *Minimal-Memory* strategy with a hierarchical data sparse format is the main risk of the project as performing efficient extend-add operations reveal to be extremely challenging. In [26], we have shown with the BLR format that the cost of the compressed updates form interactions between supernodes might grow due to the irregularity of the contribution sizes. Indeed, in the compressed format, the complexity depends on the updated block size, and not on the contribution size as in full-rank. Thus, this overhead is limited in the BLR solver by the chosen block size. However, in a hierarchical format, especially when off-diagonal blocks are not refined as in the HODLR format, the size of the blocks receiving the contributions is not limited anymore, and thus the complexity might increased compare to the BLR version. This problem might not reveal as problematic as it seems. The gain on the number of flops obtained from the compression of the large diagonal matrices in hierarchical format might asymptotically compensate this potential extra cost. However, if we want this solution to be efficient on smaller matrices, we need to investigate alternative solutions.

The first solution considered is to investigate reordering heuristics as the one presented in [25], to reduce the number of contributions that need to be applied on the large off-diagonal blocks of the supernode matrices. The objective of the ordering strategy will be to separate the contributions from the children of different of branches and compact contributions from a single child. In addition to the new ordering heuristics, we will develop unknowns preselection strategies to limit the number of low-rank updates. This ordering solution is also completely independent from the initial problem and will benefit to the BLR format too, thus it will be developed independently to be integrated in the Scotch library [24] developed in Inria TADaaM project in Bordeaux.

One extra minor risks might be considered. The complexity of the general $\mathcal{H}$-matrix format which might make it too difficult to be implemented efficiently in a sparse direct solver context. Thus, we will develop the solver with general $\mathcal{H}$-matrix structures, but we might consider only the special case of the HODLR format in this family.

2. Project organization and means implemented

2.1. Scientific coordinator and his team

Mathieu Faverge holds an Assistant Professor position at Bordeaux INP since September 2012, and he is a member of the Inria HiePACS team. He defended his PhD thesis in December 2009 on hybrid static-dynamic scheduling for sparse direct solvers on large NUMA architectures. Then, he worked as a Research Scientist at the University of Tennessee in the group of Jack Dongarra. His research interests focus on numerical linear algebra algorithms for sparse and dense problems on massively parallel architectures, and especially on task based (or DAG) algorithms relying on dynamic schedulers. He is the main contributor in reference libraries for distributed dense linear algebra: Chameleon and DPLASMA, as well as for sparse linear algebra with the PaStiX library. He has experience with hierarchical shared memory, heterogeneous and distributed systems, and his contributions to the scientific community include multiple algorithmic solution to enhance those libraries on modern architectures. Recently, in the context of the FastLA Inria Associate Team, he co-advises the PhD thesis of Grégoire Pichon with Pierre Ramet, from the University of Bordeaux, and in collaboration with Eric Darve, from the University of Stanford, on exploiting low-rank compression techniques in a sparse direct solver to reduce both the memory footprint and the time to solution of the solver. Mathieu Faverge has a strong collaboration with the Inria Storm team which is developing the StarPU runtime system, and with the University of Tennessee which is developing the PaRSEC runtime system. These collaborations will help to integrate the last development of the runtime systems to support distributed and heterogeneous systems for the irregular tasks and data that represents the

low-rank matrices. As the project coordinator, he will advise the PhD student and the engineer that will be hired for the project, and maintain the work done in the project.

The project will be developed within the HiePACS team (Inria Bordeaux - Sud-Ouest). HiePACS research involves massively parallel computing and the design of highly scalable linear algebra algorithms and codes with advanced numerical schemes to be executed on petaflop (and beyond) platforms. HiePACS is the authors of multiple solvers library to investigate numerous numerical schemes and parallel strategies like MaPHyS, HIPS, PaStiX, and Chameleon with a strong development around task based solution on top of runtime systems to provide good scalability and portability of the performances. The HiePACS team is collaborating with the Inria TADaaM team on ordering libraries in the Scotch project, and all the members have a strong collaboration with the Inria STORM team to exploit efficiently the StarPU runtime system. HiePACS has ongoing collaborations with French industrial partners (such as Airbus, CEA, Total). These collaborations will be used to keep track on what is being done in industrial production, and what is needed by their applications. Airbus and the CEA collaborations are of particular interests as they developed their own low-rank solver for large dense matrices with the help of runtime systems. Furthermore, HiePACS has also international collaborations with the Berkeley National Laboratory and the University of Stanford that will give opportunities for collaborations on the development of ordering techniques and kernels adapted to the hierarchical format in sparse direct solvers.

Two members of the Inria HiePACS team will have a stronger implication in the project. Pierre Ramet, who is an associate professor at the University of Bordeaux, is the co-developer of the PaStiX library with Mathieu Faverge. He will participate in advising the PhD student on the solver aspect and in the library validation and integration. Aurélien Esnard, who is an associate professor at University of Bordeaux. He is actual research interest focused on graph partitioning techniques for code coupling. He will help in the study of ordering heuristics adapted to low-rank format in the Task 2 of the project.

2.2. Means of achieving the objectives

The SaSHiMi project is organized in four major tasks as follows:

Task 1 is the core task of the project and is the integration of hierarchical data format in the PaStiX sparse direct solver. Outcomes of this task will be new releases of the PaStiX solver integrating the hierarchical matrix format and scientific publications on the complexity and performance analysis of the solver.

Task 2 is the study of ordering heuristics to extend the classic nested dissection to consider low-rank properties of the matrices. This work consists in extending the heuristics from the Scotch library to limit the number of extend-add operations and preselect important rows and columns.

Task 3 is the stabilization and integration of the work done in the project in applications from academic and industrial partners and in hybrid solvers.

2.2.1. Task 1: Integrating hierarchical data structure in a sparse direct solver

leader: Mathieu Faverge				
Involved partners	Mathieu Faverge	Pierre Ramet	Research Interns 1 & 2	PhD
Involvement	8	3	2×6	18
External collaborators: IRIT, KAUST, LBNL, Univ. of Stanford				
Tools and environment: PaStiX, StarPU, PaRSEC, PlaFRIM, GENCI platforms				

The objective of this task is to provide a sparse direct solver using a supernodal method with a hierarchical compression scheme of the blocks arising in the sparse matrix factorization. This task is decomposed in three subtasks, decomposed as follow: the main development of the solver using hierarchical matrices, the scalability study of the solver and strategies to improve it, and finally the comparison with the other two main solvers using compression techniques to speed up sparse direct solvers.

Subtask 1.1: Integrate hierarchical data sparse format in PaStiX

Goals: Provide a supernodal sparse direct solver using a non nested hierarchical matrix format.

Detailed work program: This task is the main task of the project as it is the modification of the solver to integrate hierarchical matrix format in the solver. First, the solver will be modified to describe the factorized matrix L (or the symbol matrix) to a hierarchical matrix format that conserves the supernodes information from the original elimination tree. This new data structure will require to adapt the classic *flat* algorithm of the sparse direct solver to a hierarchical traversal of the structure adapted to the runtime systems supported by the PaStiX library. This work, with the study of its impact on the full-rank and the BLR solver, will be the objective of the intern student hired at the beginning of the project.

The next step will be to describe each diagonal block associated with a supernode with a hierarchical matrix structure and to develop new kernels adapted to this format. The outcome will be the supernodal sparse direct solver using hierarchical matrix compression format that will serve as basis for the latter development of the project.

First, the *Just-In-Time* strategy will be studied in terms of performance, and the *Minimal-Memory* strategy will be refined all along the project with the work from Tasks 1.2 and 2.

This work will be done in the continuity of our collaboration with E. Darve (Univ. Stanford), and with G. Pichon who is expected to have a post-doctoral research position at Lawrence Berkeley National Laboratory (LBNL) at the time of the project.

Subtask 1.2: Flexible left-right looking algorithms and scalability of the solver

Goals: Provide a flexible left-right looking scheduling to reduce the number of extend-add operations in the *Minimal-Memory* strategy.

Detailed work program: Multiple strategies will be investigated to reduce the cost of the extend-add operations in the *Minimal-Memory* strategy, and to study the scalability of the solver. On the first aspect, we will investigate strategies mixing both left and right looking scheduling. The right looking strategy provides parallelism and is what is looked for to reach a good scalability. However, it increases the number of extend-add operations that may slow-down the solver. A solution to this problem is to gather subsets of contributions together to apply them in a left-looking fashion to reduce the number of updates and thus make the solver more efficient. This solution will be implemented by inserting specific extend-add tasks in the runtime systems, while the original task will just gather the contributions. This way, we should be able to provide a flexible solution to study the interest of such hybrid scheduling strategy with a hierarchical format.

This first study, as well as the move toward a hierarchical format, will impact the scalability of the solver. It introduces a lot of irregularities in the kernel computations which makes it more difficult to predict the execution time, especially because the ranks of the blocks are not a priori known. We will analyze the behavior of the dynamic scheduling provided by the runtime systems used in the PaStiX solver, to detect if this task irregularity perturbs the scalability of the solver or not. As we consider static mapping, this might create load unbalance in the system, thus we will study strategies based on expected maximum ranks to better estimate the cost of the kernels and better balance the load.

Finally, to improve the scalability of the solver, an intern student will be hired to study the integration of GPU kernels for the low-rank operations in the runtime system version of the library. This work will be done through our collaboration with H. Ltaief from KAUST and the K-BLAS library they develop.

The output of this task will be reports, trace analysis of the solver in distributed and heterogeneous environments, and a new mapping strategy based on maximum rank estimation.

Subtask 1.3: Comparative study with the MUMPS and StrumPACK solvers.

Goals: Provide a detailed report on the existing strategies to solve a sparse linear system with low-rank compression.

Detailed work program: The goal of this task is to perform a comparative study with the MUMPS and StrumPACK solvers. This study will be performed in collaboration with both teams developing those solvers and will compare the solvers in terms of computational flops, efficiency, accuracy and memory.

The goal of this will be to produce scientific publications on the different strategies actually existing to solve sparse matrices using direct methods with low-rank compression formats.

To efficiently conduct this task, a three-month visit of the PhD student to the LBNL is planned to collaborate with StrumPACK developers on the comparison. Visits to and from IRIT are also planned to exchange on and compare the MUMPS and PaStiX solvers.

Deliverables:

T0+07 [D1.1]: Study of the impact of the hierarchical data structures in the solver (report),
T0+12 [D1.2]: Study of the initial performance of $\mathcal{H}$-matrix version (report),
T0+36 [D1.3]: New mapping strategy for the distributed memory,
T0+42 [D1.4]: Comparative study with the MUMPS and StrumPACK solvers (report).

Risks and backup solutions: As explained in Subsection 1.3, the main risk of this task is the efficiency of the *Minimal-Memory* strategy due to the difficulty of the extend-add operation. This version however will be further refined with Task 2, and the *Just-In-Time* strategy already represents a high breakthrough as it will reduce the asymptotic time of the factorization and solve steps of the solver.

Secondly, the large and ambitious amount of work required by Subtask 1.2 may represent a risk, as it may be difficult to fulfill integrally this task for the duration of the project. Based on the results of Subtask 1.1 for the *Minimal-Memory* strategy, priority will be given either to the left-right looking approach to reduce the cost of the extend-add operation, or to the distributed mapping and scheduling if original results are satisfying enough. GPU kernels will be studied as an opening research to improve the solver efficiency.

2.2.2. Task 2: Ordering heuristics

leader: Mathieu Faverge			
Involved partners	Mathieu Faverge	Aurélien Esnard	PhD
Involvement	8	6	12
External collaborators: Inria TADaaM, LBNL			
Tools and environment: Scotch, PlaFRIM, GENCI platforms			

One of the important techniques in low-rank matrix compression is the choice of the clustering to get a good representation of the original matrix without loosing more than the prescribed accuracy. When considering dense matrices, two techniques stand out: k-partitioning for the flat BLR format, and recursive bi-partitioning for hierarchical formats. In our study, those techniques do not consider the supernode interactions which are the generators of contributions. Thus, to get an efficient *Minimal-Memory* strategy in our solver, it is mandatory to reconsider the clustering techniques used to take into consideration those external contributions. We will do that through two different aspects: the clustering and ordering technique used on each supernode, and the modification of the nested dissection algorithm.

Subtask 2.1: Supernode ordering heuristics for low-rank matrices

Goals: Study the ordering heuristics used for the unknowns within the subset of each supernode to adapt to sparse direct solvers with low-rank compression.

Detailed work program: In preliminary work [25], we have shown that in sparse direct solvers, unknowns within each supernode can be reordered without changing the cost of the solver to

improve the task granularity and better compact contributions from interactions with other supernodes. In this task, we will pursue this work to reorder unknowns of each supernode with two objectives: 1) minimize the number of contributions that requires a complex extend-add updates, and 2) preselect unknowns to have a better admissibility criterion and avoid the cost of trying to compress data that will end to be stored full-rank when all contributions will be accumulated. Both of those objectives will help to use more efficient kernels and increase the overall efficiency of the low-rank solver in both *Just-In-Time* and *Minimal-Memory* strategies.

The main idea behind this work is that the nodes of a supernode directly connected to its children in the elimination tree may be of greater importance than the others. Thus, they need to be preselected for non admissibility to avoid useless compression, and the remaining nodes must be clustered in such a way that contributions from children overlap as less as possible on multiple clusters to reduce the number of extend-add operations.

This study will be done using existing graph libraries such as the Scotch library developed by the Inria TADaaM team, and used by many sparse direct solvers.

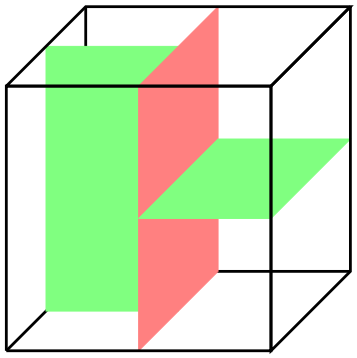
This will continue the collaboration started with M. Jacquelin from LBNL on reordering strategies for sparse direct solvers, and exchanges with LBNL will be planned in that context, by organizing mini-symposiums in SIAM PP and/or CSE conferences.

The output will be the implementation of a supernode ordering heuristic within the Scotch library, and reports on their impact on both the BLR and the hierarchical matrix version of the PaStiX solver in terms of flops count, timing and memory.

Subtask 2.2: Supernode alignment strategy

Goals: Study variants of the nested dissection algorithm to align supernodes.

Detailed work program: In previous Subtask 2.1, we propose to provide ordering heuristics to preselect unknowns from a supernode directly connected to children for non admissibility criterion, and get better kernel efficiency. However, when we consider a separator in the nested dissection algorithm as the red plan in the figure on the right, the next two separators in the recursive nested dissection are chosen independently. As illustrated here with the two green separators, the connection of those separator, *i.e.*, their projections, onto the parent separator (in red) do not match. Thus, the number of nodes preselected by the previous task may be higher than expected and it may badly impact the compression of the solver.



We propose to study variants of the nested dissection algorithm used in ordering libraries to align, if possible, the projections of separators from similar level in the recursive algorithm. This will reduce the number of preselected unknowns and, level after level, create the expected hierarchy to compress efficiently the matrices associated with supernodes.

To perform this task, we will work with the Scotch library and with the MetaPart package, developed by A. Esnard (UNiv. Bordeaux, HiePACS), that interfaces multiple graph libraries in a single one, and provide to the user a common API to access them. Since aligning the supernodes may introduce a too strong constraint that will degrade the ordering quality and increase the fill-in of the matrix, the ordering at each level will choose the best quality solution among the alignment version and the original algorithm. In order to that, it will be mandatory to have a good decision criteria to select the correct solution at each level without degrading the overall quality of the ordering.

The output will be the implementation of the alignment strategy within the Scotch library and reports on their impact on both the BLR, and the hierarchical matrix version of the PaStiX solver.

Deliverables:

T0+24 [D2.1]: New ordering heuristics adapted to hierarchical compression format in the Scotch library (report + soft),
T0+24 [D2.3]: Study on the impact of the ordering strategies on the PaStiX solver with hierarchical matrix compression.

Risks and backup solutions: The main risk that we may encounter in this task is the cost of computing such heuristics, and this applies to both subtasks. However, both are part of the preprocessing stage which may be applied only once for multiple factorizations and/or solves, and they operate on integers only which make them cheaper even if we reach the same asymptotic complexity as the factorization. The other risk is that the gain from this work on the solver complexity might be little compared to its complexity, but preliminary study with and without the reordering strategy of our preliminary work showed a large benefit on both the factorization time and memory compression. Thus, the impact expected on hierarchical format should be higher.

2.2.3. Task 3: Validation and integration

leader: Mathieu Faverge				
Involved partners	Mathieu Faverge	Pierre Ramet	Engineer	PhD
Involvement	8	3	12	6
External collaborators: CEA Cadarache, Inria NACHOS				
Tools and environment: MaPHyS, HORSE, Jorek, Tokam3X, PlaFRIM, GENCI platforms				

This task will gather the main contribution of the engineer hired for the project. It is related to the stabilization and diffusion of the research development of Tasks 1 and 2, and its integration in two classes of codes: hybrids solvers and applications from our collaborators.

Subtask 3.1: Validation of the PaStiX library

Goals: Validate and maintain the PaStiX solver at every step of the project, and propose standard APIs to interact with both the solve and the low-rank kernels.

Detailed work program: At every step of the SaSHiMi project, the developments done in the library will be made available to the community through the public Git repository of the library. All the work developed in the context of the project will be integrated in the unitary tests and the continuous integration system that has been put in place on the Inria gitlab repository to improve the quality of the software produced and ease its integration in applications.

The goal of this task will also be to provide the right API to ease the integration of the solver in user's applications, with a particular interest into the applications targeted by Subtasks 3.2 and 3.3. Additionally, a public API will be developed to make available the direct interaction with the low-rank kernels of the PaStiX solver for external usage. This development will be critical to have a good interaction with the hybrid solvers such as MaPHyS and HORSE that may use the Schur complement directly in its low-rank form. Thus, a data structure and functionalities to operate on it will be defined with these solvers to ease the use of PaStiX with low-rank matrices.

For a better validation of the solution proposed in this project, a second objective of this task is to maintain and extend the numerous interfaces available to access the PaStiX library. This will give us the opportunity to extend the number of users of the library and the amount of feedback.

Subtask 3.2: Integration in hybrid solvers

Goals: Integrate the low-rank PaStiX solver in MaPHyS and HORSE solvers.

Detailed work program: Once the data structure will be defined in Subtask 3.1, the PaStiX library will be integrated in hybrid solvers such as MaPHyS and HORSE. Those two solvers are using domain decomposition with a direct solver to factorize and solve what is called the

interior of the domains, and then an iterative solver computes the solution on the interfaces of the domains. In current work, the Schur complement of the sparse direct solver factorization is returned in full-rank to the hybrid solver. Having a low-rank sparse direct solver will open new research on this field as the precision of a full-rank solver may not be required and low-rank compression could be used. To efficiently couples the direct method in those solver, the Schur complement must be returned in low-rank format, and solutions to manipulate it must be provided. In current solution, the Schur complement is returned as a full-rank matrix and the memory gain observed in PaStiX is lost when returning it to the Hybrid solver. The API designed in Subtask 3.1 will thus be developed to consider this case and integrated into those solvers to have a finer interaction and a real memory gain. This work will benefit to both the flat and hierarchical format of compression in the solver.

The tight coupling in the hybrid solver will open new studies on the accuracy criterion used in the low-rank sparse direct solver. Variable accuracy criteria respectively for the interior factorization and for the computation of the Schur complement to better reduce the cost of the direct solve part while keeping enough information in the Schur complement. Study on this strategy will be done conjointly with the MaPHyS team, and with Stéphane Lanteri (Inria NACHOS) for the HORSE solver.

Subtask 3.3: Integration and evaluation in applications

Goals: Validate and study the integration of the low-rank PaStiX solver in the CEA Cadarache applications Jorek and Tokam3x.

Detailed work program: PaStiX is already used in its full rank version is two applications from CEA Cadarache: Jorek and Tokam3x. Both applications are hybrid (Fortran90, MPI + OpenMP) parallel code developed for fusion simulations in the ITER project. CEA Cadarache is becoming one of the leader for 3D modeling of tokamak edge plasma turbulence in the field of the magnetic confinement fusion energy research.

In that context, previous work has been done during X. Lacoste PhD to integrate PaStiX in the Jorek simulation code of the ITER project. In this task, we will work on the integration of the new PaStiX interface that has been developed for the BLR solver, and that will slightly evolve to follow the needs of the hierarchical matrix format.

The Tokam3x code has recently started investigating optimizations for many-core architectures and more specifically on Intel Xeon Phi KNL. In order to produce challenging simulations of edge plasma turbulence in X-point geometry, the linear solvers, such as PaStiX, need to handle very large sparse matrices within the limited amount of memory available on such architectures. Compression techniques will help to reduce memory consumption and time to solution to study larger test cases per node.

The objective of this task will be to validate the new releases of the PaStiX solver in those two applications, following the release schedule, and study the impact of the final version of the low-rank factorization will have on those applications.

Deliverables:

T0+12 [D3.1]: First release of the PaStiX library with the hierarchical data format,
T0+24 [D3.2]: Second release of the PaStiX library using the new ordering heuristics,
T0+36 [D3.3]: Third release of the PaStiX library with the new mapping,
T0+48 [D3.4]: Final release of the PaStiX library including all development from the project,
T0+48 [D3.5]: Report on the use of low-rank direct solver in hybrid solvers,
T0+48 [D3.6]: Report on the use of low-rank direct solver in ITER applications.

Risks and backup solutions:

As leading projects for the development of ITER, those applications will continue their development during the time of the project and do not represent a risk. However, the later cancellation of the Intel KNL next generation may reduce the need for a solver with a really low memory

footprint. Still the evolution trend of the architecture shows a reduction of the memory per core which keeps the interest in solutions reducing the memory consumption.

2.2.4. Tasks schedule, deliverable and milestones

We sum up the Task schedule graphically in Table 2.2.4 and a synthetic view of the deliverables is presented in Table 3.

Table 2: Tentative task schedule for the program's objective

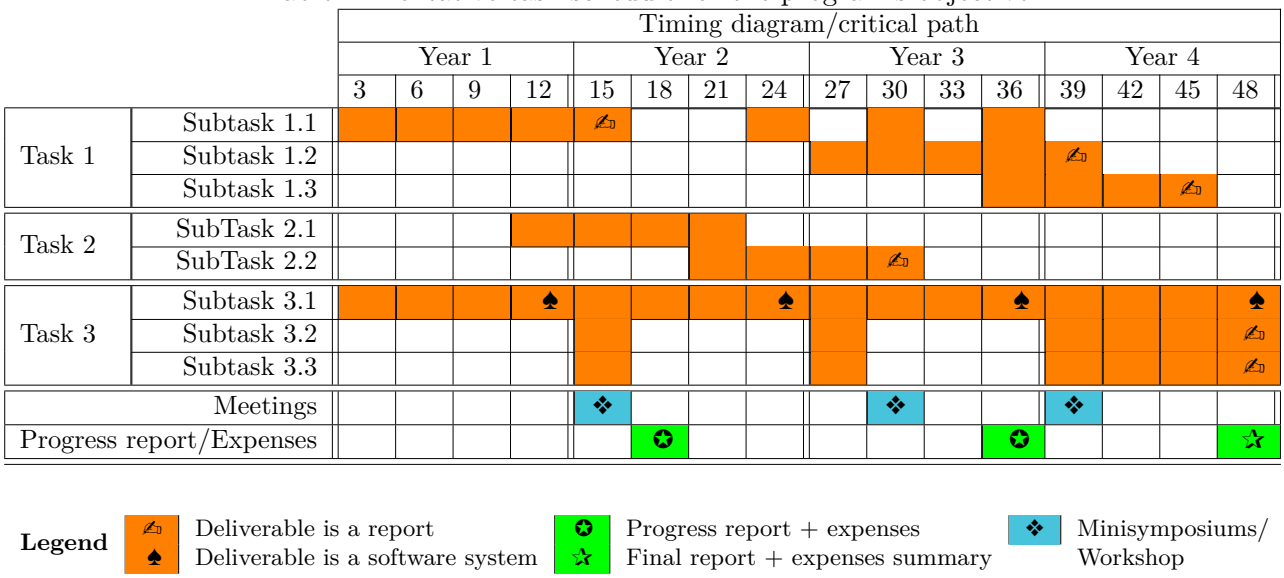


Table 3: Deliverables and Milestones

Task	Title and substance of the deliverables and milestones	Delivery	Participants
T1: Integrating hierarchical data structure in a sparse direct solver			
D1.1	Study of the impact of the hierarchical data structures in the solver	T0+07	Intern
D1.2	Study of the initial performance of $\mathcal{H}$-matrix version	T0+12	PhD
D1.3	New mapping strategy for the distributed memory	T0+36	PhD
D1.4	Comparative study with MUMPS and STRUMPACK	T0+42	PhD
T2: Ordering heuristics			
D2.1	New ordering heuristics adapted to hierarchical format	T0+24	PhD
D2.3	Study on the impact of the ordering strategies	T0+24	PhD
T3: Validation and integration			
D3.1	First release of the PaStiX library with the hierarchical data format	T0+12	-
D3.2	Second release of the PaStiX library using the new ordering heuristics	T0+24	-
D3.3	Third release of the PaStiX library with the new mapping	T0+36	-
D3.4	Final release of the PaStiX library with the new mapping	T0+48	Engineer
D3.5	Report on the use of low-rank direct solver in hybrid solvers	T0+48	Engineer + PhD
D3.6	Report on the use of low-rank direct solver in ITER applications	T0+48	Engineer + PhD

2.3. Scientific and technical justification of the requested means

2.3.1. Costs supported by ANR

Staff. Four people will be hired on the project:

1. Two research intern (master-level, 6 months each, 2×4k€) will be hired on the 1^{st} and the 25^{th} month. The first one will start the work on integrating hierarchical matrix format by doing the preliminary work of Subtask 1.1 which is the modification of the solver to integrate hierarchical structures and validate the modifications in both the full-rank and BLR format. This will define

the initial framework in which the integration of hierarchical matrix compression format will be studied. The second intern will study the integration of GPU kernels in both the BLR and the hierarchical format solver.

2. One PhD student (36 months, 114k€) will be hired on the 6^{th} month to follow up on the intern's work. He will work on the design and evaluation of the integration of the hierarchical matrix format into the PaStiX solver (Task 1) and will also perform the research on ordering heuristics for the low-rank format (Task 2).
3. One engineer (12 months, 50k€) will be hired on the last year of the project. He will mainly work on Task 4 by (i) validating and integrating the final version of the code developed by the PhD student in the library to make the software available to the community; (ii) integrating the new release of the solver in the hybrid solvers; and (iii) assisting with the integration in application and experiments.

Mission, traveling and workshop. The traveling costs are evaluated as follows:

1. Attending two international conferences for two members of the project during 4 years, including the organization of mini-symposiums in conferences SIAM PP, SIAM CSE and/or PMAA and international conferences with proceedings. *Cost:* $2 \times 2 \times 4 \times 2.5$k€ $= 40$k€.
2. One three-months visit for the PhD student to the LBNL partner (Tasks 1 and 2). *Cost:* 1×6k€ $= 6$k€.

Inward billing Two laptops will be bought for the PhD student and the engineer to be recruited by this project: 6k€.

2.3.2. Requested means by item of expenditure and by partner

		Inria
Staff expenses		172k€
Instruments and material costs (including the scientific consumables)		6k€
Building and ground costs		0k€
Outsourcing / subcontracting		0k€
General and administrative costs & other operating expenses	Travel costs	46k€
	Administrative Management and structure costs	17.92k€
Requested		241.92€

2.3.3. Costs non supported by ANR

Experimental resources For the experimental platforms we will rely on:

- our collaborators from LBNL and KAUST to have access to their respective supercomputers Cori and Shaheen;
- and finally, on the local PlaFRIM cluster as our development and experimental setting before performing larger experiments on the supercomputers cited above.

Additional travels and workshop We will seek for complement for a second three-month visit of the PhD to our LBNL collaborator to make a visit at the beginning of the second year to work on the ordering heuristics, and one at the beginning of the third year to finalize the comparison with their local solver. This complement will be sought through with different our partners through the Inria Associated Team call which provides funding for exchanges.

Additional visits to and from IRIT for the comparisons with the MUMPS solver will be funded by Inria, and visits to our industrial partners will be funded through ongoing collaborative projects.

2.3.4. Access to very large research infrastructure

Our annual GENCI project provide the HiePACS team an access to the Frecnh national supercomputer for the last 20 years. For the years f the project, we will repeat each year our demand of one million of CPU hours on the recent infrastructures: Cobalt and Occigen2. This will allow us to validate our research on large test cases issued from applications of industrial partners.

3. Impact and benefits of the project

Impact. This project aims at answering the *"Défi 7 > Axe 5 > Simulation numérique intensive pour comprendre, pour optimiser, pour décider.* Indeed, as a core kernel to many simulation applications, we expect to obtain a large spectrum of HPC applications: from electromagnetism to computational fluid dynamics, including geophysics, astrophysics, or fusion plasma. An indirect impact to those applications will also be made through the opening of new opportunities for research in the field of domain decomposition solvers that will benefit from the memory footprint reduction induce by a low-rank Schur complement.

At the solver level, we expect that the project will provide significant contributions to the sparse solvers area with:

1. new ordering heuristics adapted to the low-rank compression scheme that will benefit to all sparse solvers,
2. a large reduction of the memory footprint and the computational cost of sparse direct solvers.

All those results will not be only have an impact on the PaStiX solver, but are of interest to many people in the linear algebra community, and to the industrial community as testified by the attached letters to this document.

Scientific communication. We expect to publish the results obtained with submissions to the major venues of the field (SuperComputing, IPDPS, EuroPar), and journal papers in leading international journals (TPDS, CCPE, SIAM). Furthermore, we expect to organize multiple mini-symposiums at SIAM CSE, SIAM PP and PMAA conferences to follow up on current research with ongoing partners and trigger new collaborations and fruitful discussions with new ones. This project will pursue our ongoing collaboration with E. Darve's team at the University of Stanford on fast-multi-pole methods and data sparse formats, and will reinforce the starting collaboration with LBNL on ordering heuristics.

Software. The results of this project are expected to be integrated in the PaStiX library. PaStiX is freely available under free software license (Cecill-C) on its gitlab repository. The ordering heuristics will be developed and made public in the MetaPart package and the Scotch library (LGPL).

We expect to integrate the final release in the hybrid solvers HORSE and MaPHyS, as wells as in industrial applications from CEA Cadarache on which Subtask 3.3 will focus, but also in the applications from Airbus, BSC, Cerfacs, or Total. Many more applications will *transparently* benefit from this project through the multiple interfaces available to access PaStiX: Fortran90, Python, PETSc, or Eigen that will will be maintained and extended to cover the results of the SaSHiMi project.

Other. The SaSHiMi project will continued developing the research on low-rank matrices started by Mathieu Faverge and more generally by the HiePACS team. The results are expected to open new directions of research for hybrid solvers like the MaPHyS solver developed by the HiePACS team, and for research on communication schemes in the sparse direct solvers to reduce the volume of data exchanged. The low-rank kernels developed in the context of this project will also benefit to the dense linear algebra library Chameleon, developed by Mathieu Faverge, in which new dense linear algebra algorithms using low-rank will be investigated.

The results of this project will be presented in international PRACE PATC schools like the *Parallel Linear Algebra* formation led by the HiePACS team every two years.

4. References related to the project

[1] E. Agullo, L. Giraud, and L. Poirel. *Robust coarse spaces for Abstract Schwarz preconditioners via generalized eigenproblems*. Research Report RR-8978. INRIA Bordeaux, Nov. 2016. URL: https://hal.inria.fr/hal-01399203.

[2] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. "Improving Multifrontal Methods by Means of Block Low-Rank Representations". In: *SIAM Journal on Scientific Computing* 37.3 (2015), A1451–A1474.

[3] P. Amestoy, A. Buttari, J.-Y. L 'excellent, and T. Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*. Research Report. INPT-IRIT, CNRS-IRIT, INRIA-LIP, UPS-IRIT, Apr. 2017. URL: https://hal.archives-ouvertes.fr/hal-01505070.

[4] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. "On the Complexity of the Block Low-Rank Multifrontal Factorization". In: *SIAM Journal on Scientific Computing* 39.4 (2017), A1710–A1740. DOI: 10.1137/16M1077192. eprint: https://doi.org/10.1137/16M1077192.

[5] A. Aminfar, S. Ambikasaran, and E. Darve. "A fast block low-rank dense solver with applications to finite-element matrices". In: *Journal of Computational Physics* 304 (2016), pp. 170–188.

[6] A. Aminfar and E. Darve. "A fast, memory efficient and robust sparse preconditioner based on a multifrontal approach with applications to finite-element matrices". In: *International Journal for Numerical Methods in Engineering* (2016).

[7] J. Anton, C. Ashcraft, and C. Weisbecker. "A Block Low-Rank Multithreaded Factorization for Dense BEM Operators". In: *SIAM Conference on Parallel Processing for Scientific Computing (SIAM PP 2016)*. Paris, France, Apr. 2016.

[8] C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier. "StarPU: a unified platform for task scheduling on heterogeneous multicore architectures". In: *Concurrency and Computation: Practice and Experience* 23.2 (2010), pp. 187–198. DOI: 10.1002/cpe.1631. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1631.

[9] S. Balay et al. *PETSc Web page*. http://www.mcs.anl.gov/petsc. 2017. URL: http://www.mcs.anl.gov/petsc.

[10] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*. 1st. Springer Publishing Company, Incorporated, 2008.

[11] G. G. Benoît Jacob. *Eigen Web page*. http://eigen.tuxfamily.org. 2017. URL: http://eigen.tuxfamily.org.

[12] S. Börm, L. Grasedyck, and W. Hackbusch. "Introduction to hierarchical matrices with applications". In: *Engineering Analysis with Boundary Elements* 27.5 (2003). Large scale problems using BEM, pp. 405–422. ISSN: 0955-7997. DOI: https://doi.org/10.1016/S0955-7997(02)00152-2.

[13] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Hérault, and J. J. Dongarra. "PaRSEC: Exploiting heterogeneity to enhance scalability". In: *Computing in Science & Engineering* 15.6 (2013), pp. 36–45.

[14] J. N. Chadwick and D. S. Bindel. "An Efficient Solver for Sparse Linear Systems Based on Rank-Structured Cholesky Factorization". In: *CoRR* abs/1507.05593 (2015).

[15] S. Chandrasekaran, M. Gu, and T. Pals. "A Fast ULV Decomposition Solver for Hierarchically Semiseparable Representations". In: *SIAM Journal on Matrix Analysis and Applications* 28.3 (2006), pp. 603–622. DOI: 10.1137/S0895479803436652. eprint: https://doi.org/10.1137/S0895479803436652.

[16] P. Ghysels, X. S. Li, C. Gorman, and F.-H. Rouet. "A Robust Parallel Preconditioner for Indefinite Systems Using Hierarchical Matrices and Randomized Sampling". In: *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. May 2017, pp. 897–906. DOI: 10.1109/IPDPS.2017.21.

[17] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov. "An Efficient Multicore Implementation of a Novel HSS-Structured Multifrontal Solver Using Randomized Sampling". In: *SIAM Journal on Scientific Computing* 38.5 (2016), S358–S384.

[18] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Vol. 49. Springer Series in Computational Mathematics, 2015.

[19] X. Lacoste. "Scheduling and memory optimizations for sparse direct solver on multi-core/multi-gpu cluster systems". PhD thesis. Talence, France: Bordeaux University, Feb. 2015.

[20] X. Lacoste, M. Faverge, P. Ramet, S. Thibault, and G. Bosilca. "Taking advantage of hybrid systems for sparse direct solvers via task-based runtimes". In: *HCW'2014 workshop of IPDPS*. Phoenix, United States: IEEE, May 2014.

[21] B. Lizé. "Fast direct solver for the boundary element method in electromagnetism and acoustics : $\mathcal{H}$-Matrices. Parallelism and industrial applications". Theses. Université Paris-Nord - Paris XIII, June 2014. URL: https://tel.archives-ouvertes.fr/tel-01244260.

[22] S. L. Ludovic Moya. *HORSE Web page*. http://www-sop.inria.fr/nachos/index.php/Software/HORSE. 2017. URL: http://www-sop.inria.fr/nachos/index.php/Software/HORSE.

[23] T. Mary. "Block Low-Rank multifrontal solvers: complexity, performance, and scalability". Theses. Université Paul Sabatier - Toulouse III, Nov. 2017. URL: https://tel.archives-ouvertes.fr/tel-01708791.

[24] F. Pellegrini. *Scotch and libScotch 5.1 User's Guide*. User's manual, 127 pages. Aug. 2008.

[25] G. Pichon, M. Faverge, P. Ramet, and J. Roman. "Reordering Strategy for Blocking Optimization in Sparse Linear Solvers". In: *SIAM Journal on Matrix Analysis and Applications* 38.1 (2017), pp. 226–248. DOI: 10.1137/16M1062454.

[26] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman. "Sparse Supernodal Solver Using Block Low-Rank Compression". In: *18th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC 2017)*. Orlando, United States, June 2017. URL: https://hal.inria.fr/hal-01502215.

[27] M. Sergent. "Scalability of a task-based runtime system for dense linear algebra applications". Theses. Université de Bordeaux, Dec. 2016. URL: https://tel.archives-ouvertes.fr/tel-01483666.

[28] S. Wang, X. S. Li, F.-H. Rouet, J. Xia, and M. V. De Hoop. "A Parallel Geometric Multifrontal Solver Using Hierarchically Semiseparable Structure". In: *ACM Trans. Math. Softw.* 42.3 (May 2016), 21:1–21:21.

[29] J. Xia. "Efficient Structured Multifrontal Factorization for General Large Sparse Matrices". In: *SIAM Journal on Scientific Computing* 35.2 (2013), A832–A860. DOI: 10.1137/120867032. eprint: https://doi.org/10.1137/120867032.

[30] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. "Fast algorithms for hierarchically semiseparable matrices". In: *Numerical Linear Algebra with Applications* 17.6 (2010), pp. 953–976. ISSN: 1099-1506. DOI: 10.1002/nla.691.

[31] J. Xia, S. Chandrasekaran, M. Gu, and X. Li. "Superfast Multifrontal Method For Large Structured Linear Systems of Equations". In: *Siam Journal on Matrix Analysis and Applications* 31 (2009), 1382–1411.

5. Support letters

Prof. Dr. Ir. G.T.A. Huijsmans
IRFM, CEA Cadarache
Eindhoven University of Technology
E-mail: guido.huijsmans@cea.fr

Cadarache, 26 March 2018

Subject: Recommendation letter Dr. Mathieu Faverge,
ANR project SaSHiMi : Sparse direct Solvers using Hierarchical Matrices

To whom it may concern,

I hereby would like to express my strong support for the project proposal "SaSHiMi : Sparse direct Solvers using Hierarchical Matrices " by Dr. Faverge within the ANR JCJC program.

ITER, the next generation experiment for the generation of fusion energy is now under construction in Cadarache France. The first experiments will start in about 10 years' time. In the meantime, large scale numerical simulations are essential for the prediction of the performance and the preparation for the operation of ITER.

Sparse direct matrix solvers, notably the PaStiX library, are an integral part of a number of the codes used for the simulation of fusion plasmas. The direct solvers often constitute a bottleneck in memory and computing time and are limiting factor in the scalability towards large numbers of cpus.

The use of compression techniques in the direct solvers holds the promise of a significant reduction in the memory requirements and with the expected reduction in communication volume also an improvement in scalability. This major advance in sparse direct solvers would be of immediate benefit to simulation codes such as the non-linear magneto-hydrodynamic code JOREK, allowing a significant and much needed improvement in the level of detail.

Given the proven excellence of the people involved, I am fully convinced of a successful outcome. I therefor wholeheartedly recommend this project.

Yours sincerely,

Prof. Dr. Guido Huijsmans
Full Professor at the Technical University of Eindhoven
ITER Scientist Fellow

March 26, 2018

To whom it may concern,

This letter is to express our support to the SaSHiMi (Sparse direct Solvers using Hierarchical Matrices) project proposal presented by Dr Mathieu Faverge in response to the ANR JCJC 2018 program.

The SaSHiMi project aims at investigating the use of low-rank compression formats within parallel sparse direct solvers; as such, this project is closely related to the work we have done in the MUMPS project and solver since 2010 and that has led to several publications (including two PhD theses) and to the production of a release of the MUMPS software that includes this feature. The SaSHiMi project has the potential to provide significant scientific contributions to this field thanks to the study of the following original ideas.

First, whereas our work, and most of the related efforts, focused on the use of low-rank techniques within solvers of the multifrontal type, the SaSHiMi project targets supernodal solvers. These two types of solvers have an equivalent cost in terms of operations but have a different memory consumption and communications profile and lend themselves differently to the use of low-rank approximations. As a result, a number of interesting algorithmic questions have to be addressed. Among these is the issue related to the complexity of the low-rank assembly operations which is critical in the supernodal approach. The SaSHiMi project proposes to address this difficulty by means of heuristics for the ordering of separators that reduce the overhead imposed by such operations.

The SaSHiMi project, moreover, plans to investigate the use of hierarchical low-rank formats, namely the HODLR one, as opposed to the flat Block Low-Rank (BLR) format which we have relied upon. Hierarchical formats are the method of choice for dense linear algebra operations for which they can theoretically achieve better compression rates. The SaSHiMi project will assess the HODLR format in the context of a sparse supernodal approach in comparison with two multifrontal low-rank solvers: the StrumPACK solver based on the HSS hierarchical format and the MUMPS solver based on the BLR format.

We believe that the research work proposed within the SaSHiMi project is complementary to the work performed in the "multifrontal community" and support the SaSHiMi application to the ANR JCJC 2018 program.

Patrick Amestoy	Alfredo Buttari	Jean-Yves L'Excellent
Professeur INPT-IRIT	CR CNRS-IRIT	CR Inria-LIP-ENS-Lyon



Esmond G. Ng
Head, Applied Mathematics Department
Computational Research Division

March 23, 2018

To Whom It May Concern:

Subject: Letter of Support for Dr. Mathieu Faverge's SaSHiMi Proposal

I am writing to express my enthusiasm and strongest support for the SaSHiMi proposal by Dr. Mathieu Faverge. This proposal is concerned with the exploration of the use of hierarchical compression matrices in supernodal direct solvers for the solution of sparse systems of linear equations. In particular, Dr. Faverge is proposing to work in the context of PaStiX, one of the well-established sparse linear equations solvers. The outcome of the proposed project would greatly reduce the computational and memory complexity of such solvers.

We are working on a similar project in the Applied Mathematics Department at Lawrence Berkeley National Laboratory. The package is STRUMPACK, which uses the hierarchical semi-separable format and incorporates the compression technique in a multifrontal-based sparse linear equations solver. Depending on the level of compression, STRUMPACK can be used as a direct solver or to produce preconditioners for iterative solvers.

The approach taken by Dr. Faverge and the PaStiX team will provide a different parallel scheme, using the hierarchically off-diagonal low-rank format, which is less restrictive but should work in a broader spectrum of cases. It will be interesting to compare if this approach enables lower ranks than the techniques used in STRUMPACK.

In addition, we are working on another sparse supernodal linear solver, symPACK, which is similar to PaStiX in philosophy but is different in terms of algorithms and data structures. Thus, the results from Dr. Faverge's proposed project would be of interest to symPACK as well.

Our team is already collaborating with Dr. Faverge and his colleagues in Bordeaux by sharing and comparing results on ordering techniques for sparse matrix factorization. I am therefore excited about the SaSHiMi proposal and I am looking forward eagerly to its success, which will provide additional collaboration opportunities between Lawrence Berkeley National Laboratory and INRIA Bordeaux, such as working jointly on new ordering heuristics for low-rank matrices, and by sharing and comparing results on STRUMPACK and PaStiX.

Sincerely,

Esmond G. Ng
Senior Scientist, and
Head, Applied Mathematics Department
Computational Research Division

LAWRENCE BERKELEY NATIONAL LABORATORY
ONE CYCLOTRON ROAD | MS 50A-3111 | BERKELEY, CALIFORNIA 94720 | PHONE: 510.495.2851 | FAX: 510.486.6900 | EMAIL: EGNG@LBL.GOV