FEMPAR: Scaling Multi-Level Domain Decomposition up to the full JUQUEEN supercomputer

Santiago Badia, Alberto F. Martín, and Javier Principe

Centre Internacional de Mètodes Numèrics a l'Enginyeria (CIMNE),
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

Description of the Code

FEMPAR [1], developed by the members of the LSSC team at CIMNE, is a parallel hybrid OpenMP/MPI, object-oriented framework for the massively parallel Finite Element (FE) simulation of multiphysics problems governed by PDEs. It provides tools for the numerical simulation of a wide range of different physical phenomena, including compressible/incompressible flows, magnetics, solid mechanics, fluid-structure interaction, or thermal coupling. FEMPAR has been designed to tackle multiphysics, nonlinear, and multiscale problems. For such problems, it makes use of scalable implicit massively parallel solvers that are based on Balancing Domain Decomposition by Constraints (BDDC) preconditioning ideas [2, 3], combined with fully-coupled block LU preconditioning [4].

In particular, within the domain decomposition kernel, FEMPAR provides a novel, fully-distributed, communicator-aware, recursive, and inter-level overlapped implementation of the MultiLevel BDDC (MLBDDC) preconditioner [5]. Figure 1 depicts the global structure of computation and communication underlying this kernel. This code weakly scales up to 458,752 JUQUEEN cores for coercive three-dimensional problems (the Laplacian and Linear Elasticity problems). The largest problem solved with FEMPAR up to now involved 30 billion unknowns. FEMPAR is released under the GNU GPL v3 license, and is more than 200K lines of Fortran95/2003/2008 code long.

As an application example, Figure 2 illustrates the vorticity isosurfaces for the incompressible Taylor-Green vortex problem at $Re = 1600$ at four different time steps, starting with the initial condition at the top-left corner and evolving in time from top to bottom, and left to right. These simulation results were obtained with FEMPAR by means of a segregated velocity/pressure algorithm, that involves a pressure Poisson MLBDDC solver per time step.

Results

Before the workshop, we could already scale the MLBDDC solver up to the full JUQUEEN supercomputer. In particular, a 3-level BDDC preconditioner, supplied either with corner and edges, or corner, edge and face constraints, was successfully applied to the Laplacian and Linear elasticity discrete problems with excellent weak scalability results. These experiments were performed with 16 MPI tasks/node, and 1 OpenMP thread/MPI task, so that we were not actually taking any profit from the hardware threads of the IBM PowerPC A2 cores. Given such limitation, the main goal during the workshop was to explore approaches that enable the exploitation of hardware multi-threading.

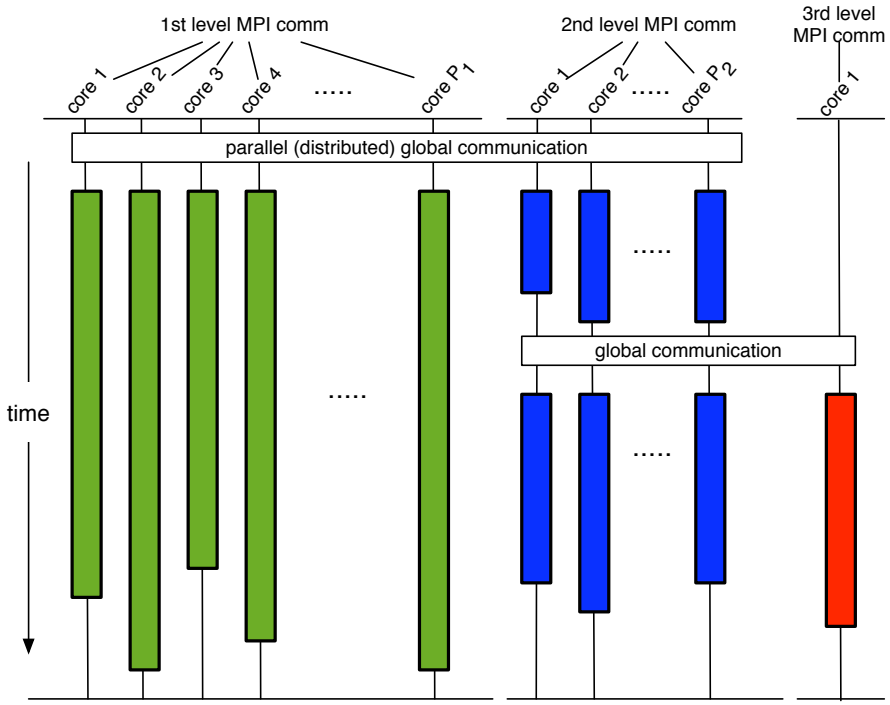

Figure 1: Computation and communication structure of the fully-distributed, communicator-aware, recursive, and inter-level overlapped implementation of the MLBDDC preconditioner.

Although FEMPAR supports hybrid MPI/OpenMP execution, it (currently) only exploits OpenMP for some phases during the computation. In particular, in the solution of *local* Dirichlet/Neumann problems (at each intermediate level), and in the solution of the coarsest-grid problem. For the solution of such problems on JUQUEEN, FEMPAR relies on HSL_MA87 [6], a numerical library for the multi-threaded sparse direct solution of SPD linear systems. Although for "large" load per core these kernels consume the bulk of the computation, there is intrinsically a serial bottleneck for increasing number of threads due to the parts which are not parallelised. On the other hand, arithmetic complexity of sparse direct methods is well known to grow as $O(n^2)$ for 3D problems, with n being the size of the coefficient matrix. These two factors render slower (for same global problem size) those hybrid configurations which use less MPI tasks than physical cores (and more OpenMP threads).

Given such scenario, we have two possible (efficient) approaches for the exploitation of hardware threads. On the one hand, a hybrid MPI/OpenMP approach with 16 MPI tasks/node, and 2/4 threads/MPI task (core). On the other hand, a pure MPI approach with 32 or 64 MPI tasks/node. The first approach, although convenient, could not be explored during the workshop, due to a memory related issue that is still under investigation with the help of JSC staff. In particular, heap and mmap *system memory* consumed by HSL_MA87 with 2/4 OpenMP threads significantly increases with respect to the 1 thread case. Besides, this increase is not reproducible, and may become "large" depending on how the tasks performed by the threads are scheduled/synchronised by the underlying software/hardware stack. This prevents the code from solving problems with a load close to the 1 GB/core limit, precisely those loads for which we expect the largest performance benefit from the exploitation of hardware threads via

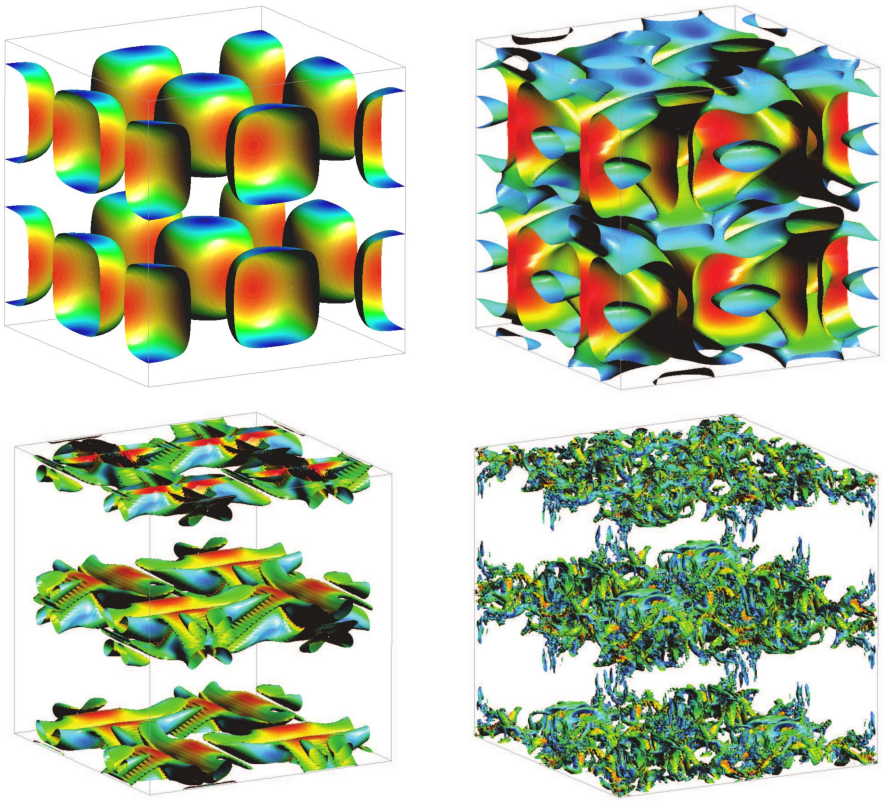

Figure 2: Vorticity isosurfaces for the incompressible Taylor-Green vortex problem at $Re = 1600$.

OpenMP.

In light of these memory issues, we decided to put on hold the hybrid MPI/OpenMP approach, focusing ourselves on the pure MPI approach. Although we also performed experiments with 32 MPI tasks/node, the results with 64 MPI tasks/node confirm a higher profit from the hardware threads in terms of aggregated efficiency. The usage of 64 MPI tasks/node implies a very moderate amount of memory of 256 MB/MPI task, and a 4-fold increase in the coarse-grid problem size to be solved at each level of the hierarchy. In order to cope with a smaller load per core, and larger coarse-grid problems, we decided to add an additional level in the preconditioning hierarchy, and study a 4-level BDDC preconditioner. In particular, Table 1 reports the configuration of the experiment that we performed with 64 MPI tasks/node, with the number of MPI tasks (subdomains) at each level, and the loads per MPI task (subdomain) tested. We applied this algorithm to the Laplacian problem discretised with Q1 FEs, and studied the weak scalability of the code on JUQUEEN with the BDDC space supplied with either corner and edge (ce), or corner, edge, and face (cef) constraints, for 3 different loads per core on the first level. To keep the presentation simpler, we will focus on the results that we obtained with the 4-level MLBDDC(cef) solver, with the largest load of 25^3 FEs/core.

A first bottleneck that we had to face was related to the initialisation stage of the code. In such stage, the MPI tasks in the global communicator are split into two disjoint subcommunicators via a call to `MPI_Comm_split`. One of these two includes the MPI tasks in the first level of the hierarchy, while the other one those which belong to the rest of levels (2nd, 3rd, and 4th in Table 1). With default settings, `MPI_Comm_split` was scaling as $O(P^2)$, with P being the number of MPI tasks in the global communicator. The workaround recommended by JSC staff was to activate the `PAMID_COLLECTIVES_MEMORY_OPTIMIZED` environment variable. This set-up

Table 1: Configuration of the 4-level BDDC preconditioner for the FEMPAR experiments with 64 MPI tasks/node performed during the workshop.

Level	# MPI tasks				FEs/core
1st	592.7K	884.7K	1.26M	1.73M	$10^3/20^3/25^3$
2nd	9.26K	13.8K	19.7K	64K	4^3
3rd	343	512	729	1K	3^3
4th	1	1	1	1	n/a
	615K	918K	1280K	1795K	

switches to a different algorithm within the underlying message-passing stack. As shown in Table 2, it also has a tremendous *positive* impact on performance/scalability, at the price of a moderate increase in memory consumption.

Table 2: Performance/scalability of `MPI_Comm_split` and average memory/1st level MPI task consumed by the 4-level BDDC(cef) solver, with the largest load of 25^3 FEs/core, once the preconditioner is set-up.

		Default settings		`PAMID_..._OPTIMIZED`	
bg_size	P	Time (sec.)	Mem. (MB)	Time (sec.)	Mem. (MB)
9609	615K	n/a	115.3	2.46	127.8
14344	918K	n/a	122.5	3.92	143.3
20002	1280K	365	131.7	5.74	163.2
28047	1795K	862	n/a	8.09	187.8

Once the bottleneck related to `MPI_Comm_split` was overcome, we proceeded with the actual weak scalability test. Table 3 reports the number of PCG solver iterations and total computation time in seconds for the 4-level BDDC(cef) solver, when applied to the discrete Laplacian problem using 64 MPI tasks/node, and the largest load of 25^3 FEs/core. Total computation time includes both preconditioner set-up and the PCG phase. These results confirm remarkable scalability for the approach that we pursue for the extreme scale implementation of the MLBDDC preconditioner. In particular, with a 4-level BDDC(cef) preconditioner, we were already able to strike a balance such that computation/communication related to coarser-grid levels in the hierarchy (i.e., 2nd, 3rd and 4th in Table 1) are completely absorbed (i.e., masked) by the finest-grid level duties (i.e., 1st level in Table 1) due to the effect of inter-level overlapping (see Figure 1). Besides, on smaller scale test cases, we compared the computation times of the codes using 16 and 64 MPI tasks/node (with the same number of MPI tasks/level in both cases), confirming an approximately 50% saving in aggregated efficiency by the exploitation of hardware multi-threading (i.e., the computation time with 64 MPI tasks/node was approximately twice as much as the one with 16 MPI tasks/node).

Finally, we would like to remark that we expect that the achievements resulting from our participation in the workshop will have a high impact on the scientific computing community in general, and in the development of fast parallel solvers tailored for FE analysis in particular [5].

Table 3: Weak scalability for the FEMPAR 4-level BDDC(cef) solver with 64 MPI tasks/node and the largest load of 25^3 FEs/core.

bg_size	P	#PCG iterations	Total time (sec.)
9609	615K	25	22.1
14344	918K	26	22.6
20002	1280K	27	22.9
28047	1795K	27	23.0

Acknowledgments

This work has been funded by the European Research Council under the FP7 Programme Ideas through the Starting Grant No. 258443 – COMFUS: Computational Methods for Fusion Technology. A. F. Martín was also partially funded by the Generalitat de Catalunya under the program "Ajuts per a la incorporació, amb caràcter temporal, de personal investigador júnior a les universitats públiques del sistema universitari català PDJ 2013." We acknowledge GCS for awarding us access to resource JUQUEEN. We gratefully acknowledge JSC's staff in general, and Dirk Brömmel in particular, for their support in porting/debugging FEMPAR and its dependencies to/on JUQUEEN.

References

[1] FEMPAR web page. `https://web.cimne.upc.edu/groups/comfus/fempar.html`

[2] S. Badia, A. F. Martín and J. Principe. Implementation and scalability analysis of balancing domain decomposition methods. *Archives of Computational Methods in Engineering*. Vol. 20(3), pp. 239–262, 2013.

[3] S. Badia, A. F. Martín and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*. Vol. 36(2), pp. C190–C218, 2014.

[4] S. Badia, A. F. Martín and R. Planas. Block recursive LU preconditioners for the thermally coupled incompressible inductionless MHD problem. *Journal of Computational Physics*, Vol. 274, pp. 562–591, 2014.

[5] S. Badia, A. F. Martín and J. Principe. Multilevel balancing domain decomposition at extreme scales. In preparation, 2015.

[6] J. Hogg, J. Reid and J. Scott. Design of a Multicore Sparse Cholesky Factorization Using DAGs. *SIAM Journal on Scientific Computing*. Vol. 32(6), pp. 3627–3649, 2010.