Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Scalability Improvement of the Projected Conjugate Gradient Method used in FETI Domain Decomposition Algorithms

Tomáš Kozubek[a], David Horák[a], Václav Hapla[a], Lubomír Říha[a]

[a]IT4Innovations, VŠB-Technical University of Ostrava (VSB-TUO)

Abstract

This report summarizes the results of the scalability improvements of the algorithms used in Total FETI (TFETI). A performance evaluation of two new techniques is presented in this report: (1) a novel pipelined implementation of CG method in PETSc and (2) a MAGMA LU solver running on following many-cores accelerators: GPU Nvidia Tesla K20m and Intel MIC Xeon Phi 5110P.

Introduction

FETI (Finite Element Tearing and Interconnecting) type domain decomposition methods are powerful tools for constructing numerically and parallel scalable solvers for real engineering problems. The original FETI method, also called FETI-1 method, was originally introduced by Farhat and Roux [10]. The standard scheme for the solution of a linear elliptic boundary value problem using FETI consists of following steps: (1) finite element discretization; (2) applying domain decomposition to decompose problem into sub-problems (using METIS); (3) introducing the Lagrange multipliers to glue subdomains together; (4) construction of the dual formulation in terms of the Lagrange multipliers; and (5) finding a solution using an appropriate iterative method. In many cases the resulting algebraic formulation leads to a quadratic programming problem with convex constraints. Solvers for such problems including our own FETI method called Total FETI are developed at IT4Innovations, VSB-Technical University of Ostrava.

The domain decomposition methods are used to solve large linear and nonlinear engineering problems of mechanics in parallel (see [2], [3]). We have implemented many domain decomposition (TFETI) based solvers into an in-house FLLOP (FETI Light Layer On top of PETSc) library [6]. The library is implemented as an extension of the PETSc framework [5] and its entire development is done at IT4Innovations. It is primarily used for solving constrained quadratic programming (QP) problems on parallel computers in High Performance Computing (HPC) environment. It is designed to be user-friendly, efficient and scalable. Additionally, any combination of the following constraints can be specified: (1) equality; (2) inequality; and (3) box constraints. To allow other packages such as Code Aster and ELMER to use FLLOP solvers a general C array based FLLOP_AIF interface has been also implemented.

In this paper we summarize the scalability improvements of two underlying algorithms used by TFETI solvers: (1) a novel pipelined implementation of CG method in PETSc and (2) a MAGMA LU solver running on following many-cores accelerators: GPU Nvidia Tesla K20m and Intel MIC Xeon Phi 5110P.

[*]Corresponding author. *E-mail address*: david.horak@vsb.cz

FETI benchmark

FETI methods use the Lagrange multipliers λ to enforce equality constraints (gluing conditions) in the original primal problem: $\min \frac{1}{2} u^T K u - u^T f \;\; s.t. \;\; Bu = o$. The primal problem is then transformed into significantly smaller and better conditioned dual equality constrained problem: $\min \frac{1}{2} \lambda^T F \lambda - \lambda^T d \;\; s.t. \;\; G\lambda = o$, with $F = BK^+B^T$. This problem can be solved by means of projectors $P_G = I - Q_G$, $Q_G = G^T(GG^T)^{-1}G$ and CG method applied to the projected system $P_G F \lambda = P_G d$. For this dual problem the classical estimate of the spectral condition number by Farhat, Mandel, and Roux is valid, i.e. $\varkappa(P_G F | Im\, P_G) \leq C \frac{H}{h}$, with H denoting the decomposition and h the discretization parameter.

To be able to fully utilize the massively parallel computers it is essential to maximize the number of subdomains (decrease H) which leads to the reduction of the sub-domain size. This improves the performance due to following reasons: (1) a subdomain stiffness matrix size is reduced which speeds up both factorization and subsequent pseudo-inverse application; and (2) the conditioning is improved which reduces the number of iterations. The negative effect of the large number of subdomains is the increase of dual and null space dimension which slows down the coarse problem (CP) solution (solution of the system $GG^T x = y$). Therefore in this case the bottleneck of the TFETI method is the application of the projector which dominates the solution time. Several parallel direct solvers such as SuperLU_DIST [8] and MUMPS [9] have been tested for coarse problem processing (all running over small number of compute nodes).

The main objective of the work described in this report is to achieve better scalability of FLLOP's FETI solvers. We have used (1) the novel pipelined CG algorithm (PIPECG) and (2) the novel hardware architectures (GPU and Intel MIC accelerators) to evaluate their suitability for FETI solvers. The following tests have been performed:

1. the performance comparison of the general CG and the pipelined CG algorithms when applied to the following problems:
    a) primal block-diagonal, assembled ($A = K, b = f$),
    b) primal decomposed and penalized, unassembled – PFETI ($A = K + \rho B^T B, b = f$),
    c) dual decomposed and projected, unassembled – TFETI ($A = P_G F, b = P_G d$).
2. the performance evaluation of the following functions from the MAGMA library designed for many-core accelerators (GPU and Intel MIC):
    a) LU factorization,
    b) solve function,

    running on (1) multicore CPU only ; (2) combination of CPU and GPU; and (3) combination of CPU and MIC. LU factorization and triangular solves are the essential functions used for the solution of the CP ($A = GG^T, b = y$).

The numerical experiments have been performed using following supercomputers:

- Anselm Bull cluster at IT4Innovations – used for GPU and MIC tests with MAGMA library,
- Sisu Cray XC30 at CSC Helsinki – used for large scalability tests of PIPECG algorithm.

For the numerical testing a loaded elastic cube is used, see Figure 1. To be able to test our FETI solvers on large scale problems, that are expected to run on exascale machines, we have used a parallel PermonCube benchmark developed by our team at IT4Innovations. It enables to generate data of large-scale problems decomposed into thousands of subdomains in parallel. In our case, the loading is f_z = 77 N/mm^3, Young's modulus E = 2e5 MPa, and Poisson's ratio μ = 0.33.

Comparison of PIPECG for primal and dual problems solution on Sisu

This section presents the performance results for benchmark described in Figure 1 of PIPECG implementation available in PETSc 3.4 for tests 1a – 1c as described in the previous section. The main idea of PIPECG is “talk less and work more” and it is illustrated in Figure 2, see also [11], [12]. In this case it is the hiding the communication needed in two dot products computation (red colour) behind the matrix by vector multiplication (blue colour), some additional work with several additional auxiliary vectors compared to CG is necessary. The success of PIPECG profits from the balancing of these two operations, if the multiplication dominates and dimension of the problem is huge, i.e. also the dimension of additional vectors is huge, then PIPECG can have even worse performance than CG.

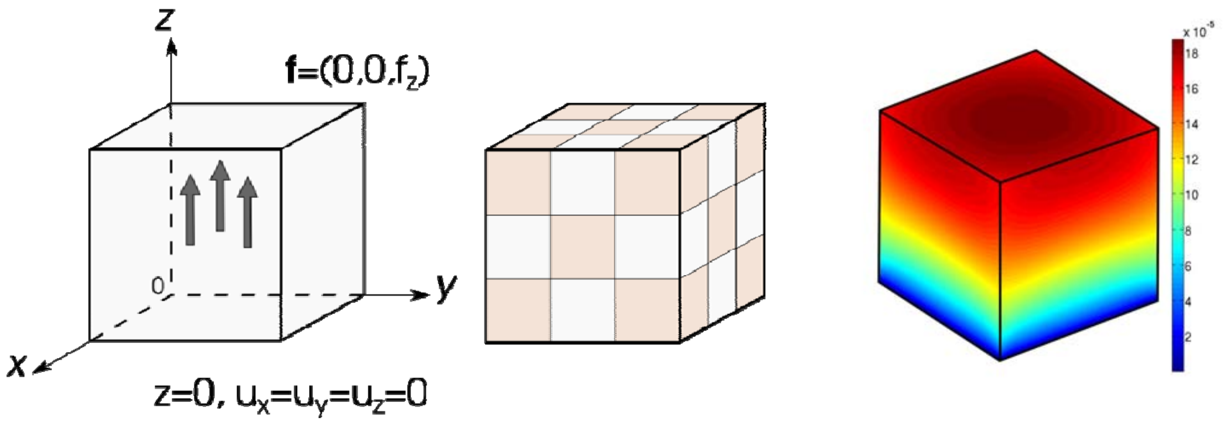
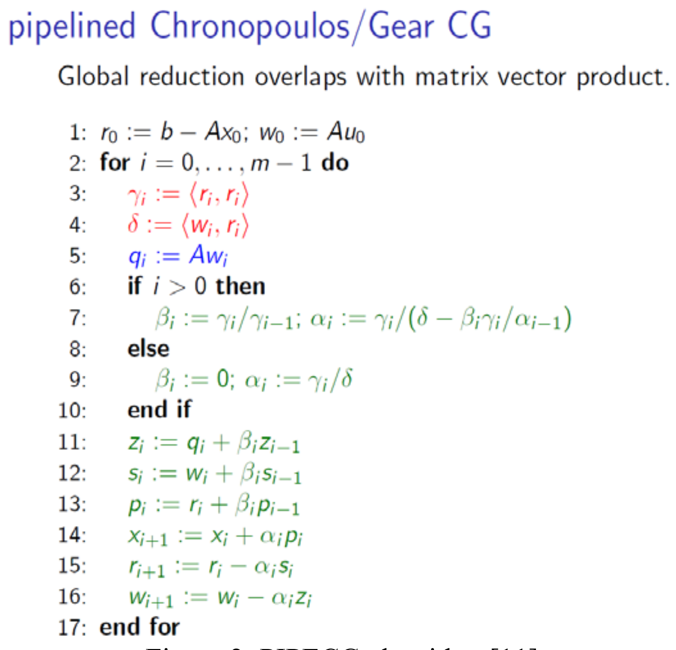


Figure 1: Cube benchmark (left), the regular decomposition (centre) and total displacements (right)



pipelined Chronopoulos/Gear CG

Global reduction overlaps with matrix vector product.

```
$r_0 := b - Ax_0$; $w_0 := Au_0$
for $i = 0, \dots, m-1$ do
    $\gamma_i := \langle r_i, r_i \rangle$
    $\delta := \langle w_i, r_i \rangle$
    $q_i := Aw_i$
    if $i > 0$ then
        $\beta_i := \gamma_i / \gamma_{i-1}$; $\alpha_i := \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$
    else
        $\beta_i := 0$; $\alpha_i := \gamma_i / \delta$
    end if
    $z_i := q_i + \beta_i z_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $p_i := r_i + \beta_i p_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $w_{i+1} := w_i - \alpha_i z_i$
end for
```

Figure 2: PIPECG algorithm [11]

The results are summarized in Figures 3-7. Figure 3 depicts that in case 1a the PIPECG performs better when subdomain size is small, but it is less efficient for larger subdomains. Figure 4 shows that for subdomains smaller than 11^3 the PIPECG algorithm is superior regardless of number of sub-domains/running MPI processes.

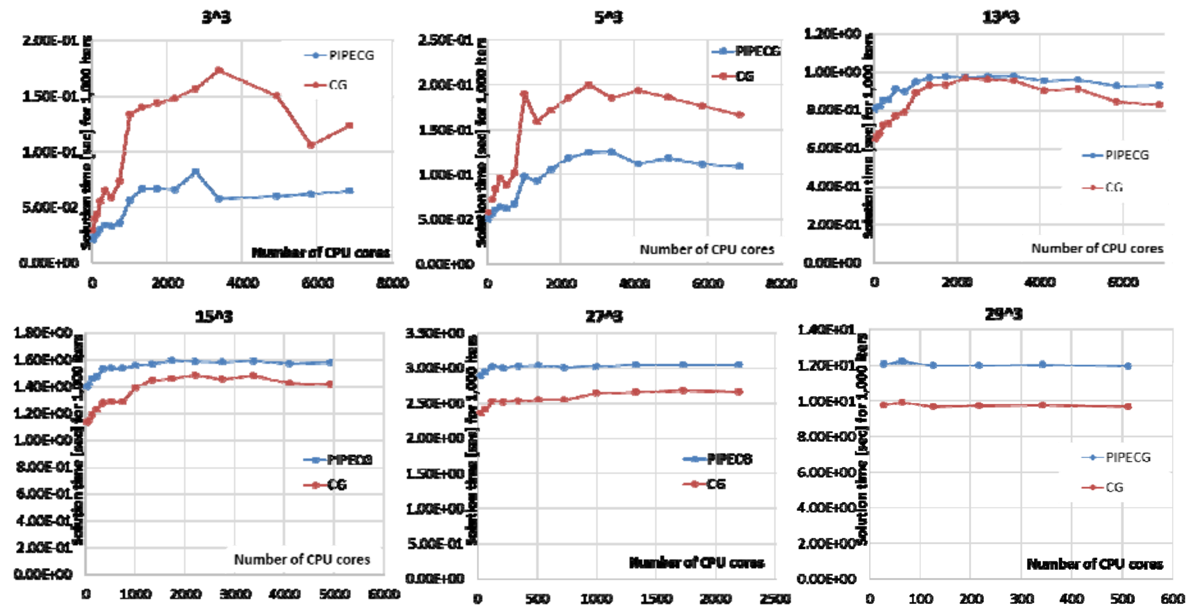


Figure 3: Performance of CG vs. PIPECG for various subdomain sizes - case 1a

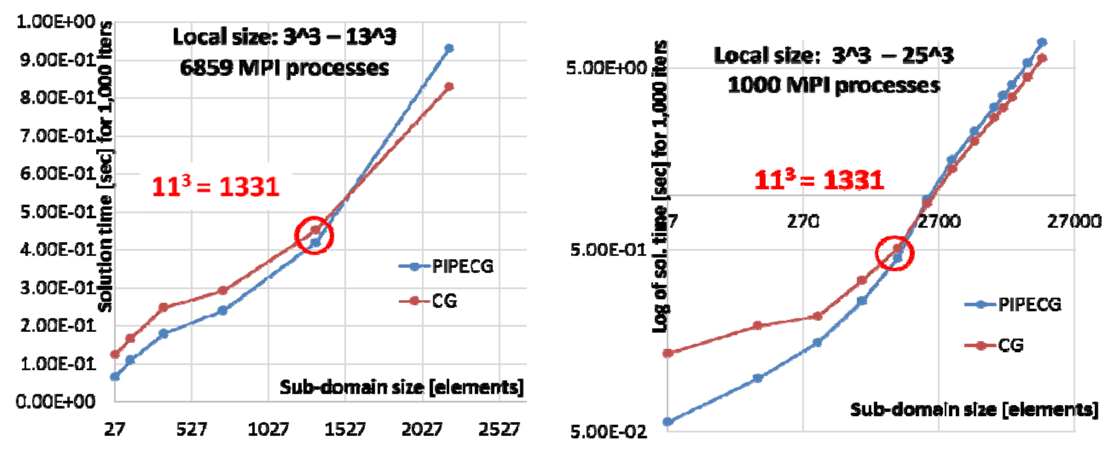


Figure 4: Performance of CG vs. PIPECG for various subdomain sizes – the limit size - case 1a

Similar observations can be made from Figures 5 and 6 which show results for case 1b. In Figure 5, we can see that for case 1b PIPECG performs better than CG when subdomain size is small and the performance gets worse with increasing subdomain sizes. PIPECG behaves similar to case 1a, even worse than CG for large problems as the benefit from the overlap of matrix by vector multiplication and two dot products computation is eliminated. The first operation exceeds significantly the second one and an additional work with longer vectors has more overhead. Figure 6 shows for which subdomain sizes of the case 1b the PIPECG algorithm is better - again for subdomain sizes less than 11^3.

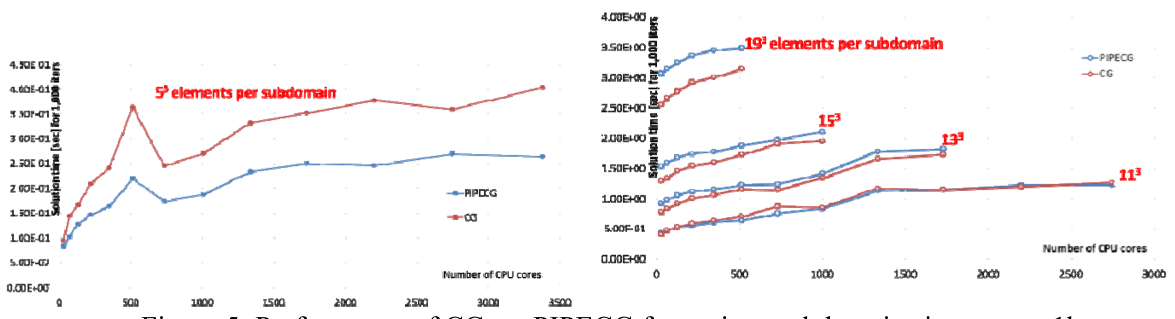


Figure 5: Performance of CG vs. PIPECG for various subdomain sizes - case 1b

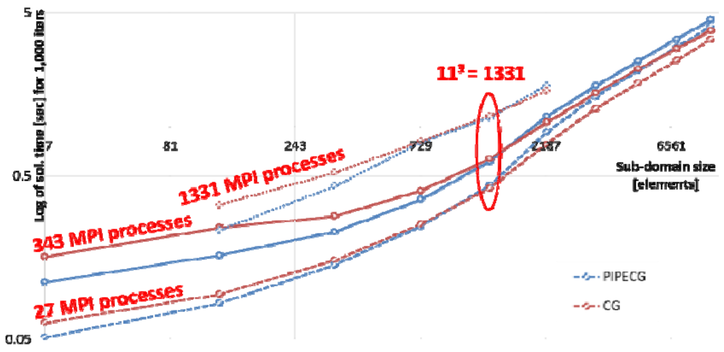


Figure 6: Performance of CG vs. PIPECG for various subdomain sizes – the limit size - case 1b

On the other hand, as seen in Figure 7, when PIPECG is employed in TFETI solver - case 1c - performance is almost identical to standard CG. This behaviour is caused by the significant amount of communication included in the dual operator $P_G F$.

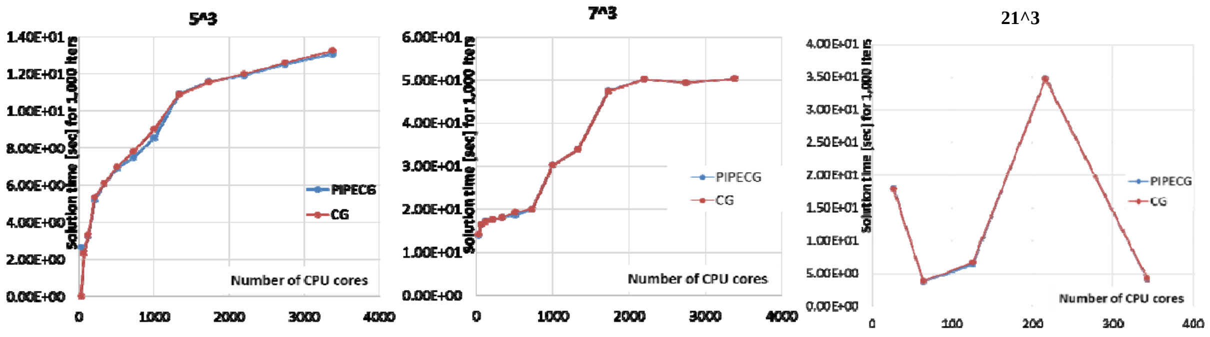


Figure 7: Performance of CG vs. PIPECG for various subdomain sizes - case 1c

MAGMA LU solver for CP solution on accelerated nodes on Anselm

This section presents the performance evaluation of the CP solution using GPU and MIC accelerators. The performance evaluation of the MAGMA LU dense direct solvers [7] applied to CP was evaluated on a single compute node of Anselm supercomputer. Three hardware configuration have been tested: (1) CPU only (16 OpenMP threads – two 8-core CPUs); (2) CPU + GPU (16 OpenMP threads + 1 Tesla K20m GPU), and (3) CPU + MIC (16 OpenMP threads + 1 MIC - 240 OpenMP threads). The results are presented in Table 1 (note: 1x solve = 1 solution of a dense CP).
In the largest test case, the LU factorization using CPU + MIC is 2 times faster and using CPU + GPU the same operations is 4 times faster when compared to MAGMA running on CPU only. On the other hand, a single solve (CP solution) is much faster using MIC than using GPU. For smaller problems the solves done by accelerators can be slower than the ones done by CPU only because the factors can fit into the cache of CPU.

	CPU only		**CPU+GPU**		**CPU+MIC**	
CP size	LU factorization	100x solve	LU factorization	100x solve	LU factorization	100x solve
384	0.002	0.01	0.004	0.06	0.002	0.12
3,072	0.125	0.46	0.075	1.87	0.123	0.37
6,000	0.769	1.90	0.267	4.32	0.374	0.86
24,576	46.066	23.29	11.349	22.75	22.495	8.29

Table 1: Comparison of MAGMA LU dense solver (LU factorization / 100x CP solve) for various CP sizes on CPU only, CPU + GPU, and CPU + MIC on Anselm

Conclusions

The new pipelined version of CG (PIPECG) brings significant time savings for primal block-diagonal, assembled and primal decomposed, penalized, unassembled problems, if the number of subdomain elements is less than 11^3. In case of the dual problems (TFETI) there is no performance difference between the general CG and the pipelined CG algorithm due to excessive communication contained in the dual operator: (1) multiplication by a constrained matrix gluing the subdomains together; (2) multiplication by the transpose of this matrix; and (3) application of the projector. With respect to our current analysis it seems that pipelined CG will be ideally suited for the parallel solution of the TFETI CP ($A = GG^T, b = y$) – it is very promising and it is a topic for future research.
Concerning the direct dense MAGMA LU solver used to solve the coarse problem, the following observations have been made: (1) LU factorization: CPU + MIC is 2 times faster than CPU (22.5 sec); CPU + GPU is 4 times faster than CPU (11.3 sec); CPU only (46.1 sec); (2) 100 calls of the triangular solves: CPU + MIC is 3 times faster than CPU (8.29 sec); CPU + GPU is 1.02 times faster than CPU (22.75 sec); CPU only (23.29 sec). To conclude, the combination of CPU + MIC is more efficient if large number of solve calls/iterations is required while CPU + GPU is better if solver can find a solution in smaller number of iterations.

References

[1] Říha, L., Hapla, V., Horák, D., Kozubek, T. et al.: Implementation of Hybrid Total FETI (HTFETI) Solver for Multi-core Architectures with Heterogeneous Accelerators, in preparation

[2] Čermák, M., Kozubek, T., Sysala, S., Valdman, J. A TFETI domain decomposition solver for elastoplastic problems, Applied Mathematics and Computation 231, pp. 634-653, 2014.

[3] Dostál, Z., Kozubek, T., Markopoulos, A., Vondrák, V., Horyl, P. A theoretically supported scalable TFETI algorithm for the solution of multibody 3D contact problems with friction, Computer Methods in Applied Mechanics and Engineering 205-208 (1), pp. 110-120, 2012.

[4] Jarošová, M., Kozubek, T., Menšík, M., Markopoulos, A. Hybrid Total FETI method, ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers pp. 6653-6663, 2012.

[5] http://www.mcs.anl.gov/petsc/

[6] http://industry.it4i.cz/produkty/fllop/

[7] http://icl.cs.utk.edu/magma/software/

[8] http://crd-legacy.lbl.gov/~xiaoye/SuperLU/

[9] http://mumps.enseeiht.fr/

[10] Farhat, C., Roux, F-X. An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems, SIAM J. Sci. Stat. Comput. 13: 379–396, 1992.

[11] Ghysels, P. and Vanroose, W.: Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. Technical Report 12.2012.1, Intel Exascience Lab Flanders, Leuven, 2012.

[12] Selvitopia, R. O., Ozdalb, M., Aykanata, C.: A Novel Method for Scaling Iterative Solvers: Avoiding Latency Overhead of Parallel Sparse-Matrix Vector Multiplies, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6766662&tag=1.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. PRACE-2IP: 283493.
This publication was also supported by the Projects of major infrastructures for research, development and innovation of Ministry of Education, Youth and Sports (LM2011033).