Numerical simulation of electrical machines by means of a hybrid parallelisation using MPI and OpenMP for FEM

Stefan Boehmer, Tim Cramer, Martin Hafner, Enno Lange, and Kay Hameyer

Institute of Electrical Machines, RWTH Aachen University, D-52056 Aachen

Abstract

In this paper, an hybrid parallelisation approach for the simulation of nonlinear electromagnetic problems by means of the Message Passing Interface (MPI) and the OpenMP Application Program Interface for the Finite Element Method (FEM) is investigated. After an introduction, the metrics applied to evaluate the speedup and the efficiency are outlined. By parallelising the institute's in–house FEM–package "*i*MOOSE" either by MPI or by OpenMP, an evaluation basis for the hybrid approach is being founded. The hybrid parallelisation approach is being evaluated on the high performance computing cluster of university's centre for computing and communication.

1 Introduction

Throughout the past decade, the increase of computational power was partially a result of increasing processing frequency but mainly a consequence of massive parallelisation within the central processing unit (CPU). This multi–core architecture evolved rapidly within the past years and along with their research programs, the road maps of the large CPU manufacturers indicate a break through of more than a hundred cores for a single CPU within the next decade [7]. This massive parallelisation has already proven its potential for general purpose graphic processing units (GPGPU). The architecture classification of CPUs and GPGPUs according to [3] distinguishes two categories: *MIMD* (Multiple Instructions Multiple Data) for the CPU and *SIMD* (Single Instructions Multiple Data) for the GPGPU respectively.
To account for the nonlinear material properties of electrical machines, iterative methods, e.g. Newton–Raphson, are required to solve the FE–system. The iterative schemes modify the corresponding FE–system matrix between two iterative steps, which significantly limits the potential of the GPGPUs due to an increased data transfer between the GPGPU memory and the main memory.
The motivation of this work is to evaluate the performance of the industrial standardised parallelisation paradigms MPI, OpenMP and their hybrid combination on high performance computing clusters based on *MIMD*–architectures to lay the foundation for a decision basis for future FEM–software design for the numerical simulation of electrical machines. Additionally, the evaluation can be extended to *SIMD*–architectures as one of the underlying equation solvers will be supporting GPGPUs in the near future [2].

2 Metrics

In order to compare different parallelisation approaches the following metrics are applied [4]. All measurements are based on the execution time $T(p)$ (*Wall Clock Time*) with p being the number of parallel processes. The sequential execution time $T(1)$ accounts for the exclusively consumed CPU–time as well as for peripheral access time caused by a sequential process. The parallel overhead is described by:

$$T_O(p) = pT(p) - T(1). \tag{1}$$

The speedup for a given number of p parallel processes is defined as:

$$S(p) = \frac{T(1)}{T(p)}. \tag{2}$$

The general definition of the efficiency with respect to the computational resources is given by:

$$E(p) = \frac{S(p)}{p} = \frac{T(1)}{pT(p)} = \frac{1}{1 + \frac{T_O(p)}{T(1)}}. \tag{3}$$

According to Amdahl's law [1] the theoretical maximum speedup for a given program in case of $p \to \infty$ is limited by its sequential fraction α:

$$S(p) \leq S(\infty) \leq \frac{1}{\alpha}. \tag{4}$$

The theoretical maximum speedup of a program with a sequential fraction of e.g. $\alpha = 0,1$ is $S(\infty) \leq 10$.
The upcoming evaluations are based on the numerical problems stated in table 1. Note that the speedup is not related to any time dependency of the problem.

testcase	# DoFs	description
PMSM2D	89.587	synchronous machine 2D
TEAM20	255.804	TEAM20 test problem 2D
PMSM3D	805.206	synchronous machine 3D
LARGE3D	9.050.911	conductor in free space

Table 1: Evaluated test cases

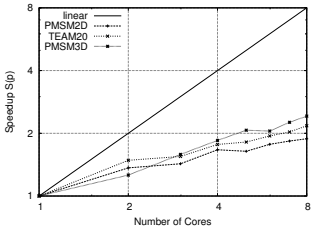

Figure 1: OpenMP speedup

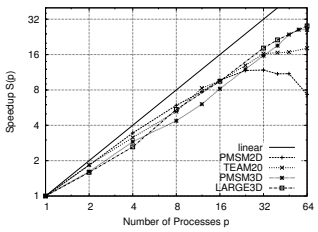

Figure 2: MPI speedup

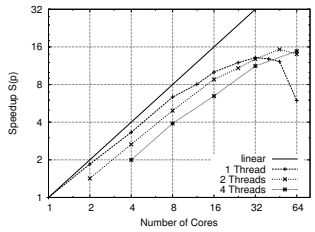

Figure 3: Hybrid PMSM2D

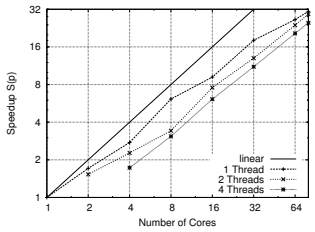

Figure 4: Hybrid LARGE3D

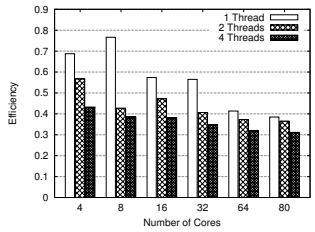

Figure 5: Hybrid efficiency LARGE3D

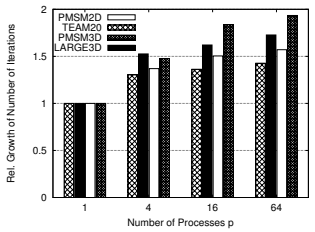

Figure 6: Increased number of CG–iterations with MPI

3 Parallelisation paradigm: OpenMP vs. MPI

OpenMP is designed to work on architectures having a common memory addressing space and can be used to parallelise loops and computationally independent program fragments with a minimum of additional code. Critical memory access, e.g. during the assembly of the system matrix element by element within a parallel loop, must be guarded by locking mechanisms.
MPI relies on an explicit communication between the parallel processes requiring an a–priori mesh decomposition due to data decomposition. The decomposition allows for a locking free assembly of the system matrix but requires an explicit data exchange for solving and incorporating boundary constraints. The amount of additional code is considerably larger compared to OpenMP.

3.1 Parallelisation based on OpenMP

According to Amdahl's law (4) the estimated theoretical maximum is $2,4 \leq S(\infty) \leq 3,6$, which is in accordance to the measured speedups of the studied problems shown in Fig. 1.

3.2 Parallelisation based on MPI

The required mesh decomposition of the MPI parallelisation is performed by PARMETIS [5]. Due to the parallel execution of all processes, a sequential code fraction limiting the speedup as seen for the OpenMP parallelisation cannot be observed (Fig. 2). The limiting factor is the explicit communication during solving, which can be observed for the relatively small 2D problems and $p > 24$. Furthermore, the number of iterations of the Krylov–Subspace method increases with p depending on the problem investigated (Fig. 6) – a well known fact.

3.3 Hybrid Parallelisation

The hybrid parallelisation decomposes the discretisation of the problem to a given number of p sub–meshes. Within each process the matrix assembly is performed by OpenMP parallelisation. The applied equation solver is LIS [6] which is based on a hybrid parallelisation as well. It can be seen, that the hybrid approach allows for increasing the scalability for higher numbers of p whereas the speedup of the MPI only parallelisation is already declining (Fig. 3). This drop cannot be observed for the large problem (Fig. 4) and thus, MPI only parallelisation performs best until the speedup stagnates (Fig. 5).

4 Discussion and Conclusions

This paper gives an evaluation of the numerical simulation of electrical machines by means of a hybrid parallelisation of the FEM–package iMOOSE using the standardised paradigms OpenMP and MPI. Especially when the speedup of the MPI only parallelisation stagnates, the hybrid approach is advantageous compared to either OpenMP or MPI. This will become of interest throughout the forthcoming years with CPUs unifying hundreds of cores being available. A detailed performance evaluation and a thorough discussion of the results will be given in the full paper.

References

[1] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (Spring)*, page 483, Atlantic City, New Jersey, 1967.

[2] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. *PETSc Web page*. 2009. http://www.mcs.anl.gov/petsc.

[3] M. J. Flynn. Some computer organizations and their effectiveness. *Computers, IEEE Transactions on*, C-21(9):948–960, 1972.

[4] A. Grama, G. Karypis, V. Kumar, and A. Gupta. *Introduction to Parallel Computing*. Addison Wesley, 2 edition, Jan. 2003.

[5] G. Karypis and V. Kumar. Parallel multilevel graph partitioning. In *Parallel Processing Symposium, Proceedings of IPPS '96, The 10th International*, pages 314–319, 1996.

[6] A. Nishida. Experience in developing an open source scalable software infrastructure in japan. In *Computational Science and Its Applications – ICCSA 2010*, volume 6017 of *Lecture Notes in Computer Science*, pages 448–462. Springer Berlin / Heidelberg, 2010.

[7] Various. *Intel Technology Journal Volume 13, Issue 4: Addressing the Challenges of Tera-scale Computing*. Intel Press, 2009.