doi: 10.21495/71-0-61

25th International Conference
ENGINEERING MECHANICS 2019
Svratka, Czech Republic, 13 – 16 May 2019

# ON TUNING OF FINITE ELEMENT LOAD BALANCING FRAMEWORK

**M. Bošanský [*], B. Patzák [**]**

**Abstract:** *Many engineering problems are extremely demanding to solve even on recent hardware. Numerical solutions of these problems on parallel computers can significantly reduce computational time by performing selected tasks concurrently. This paper deals with tuning up the parallel load balancing framework of the finite element software, which is based on domain decomposition paradigm for distributed memory model. The paper describes the technique to determine the actual weights comparing computational performance of individual processing units. These weights are fundamental inputs for mesh (re)partitioning that has to be performed at the beginning of the simulation and whenever the load imbalance is significant. The capabilities and performance of the proposed technique are evaluated on the benchmark problem and discussed.*

**Keywords: Parallelization, Load balancing, Distributed memory, Message passing, Domain decomposition.**

## 1. Introduction

The idea of any parallel algorithm is partitioning of the problem into a set of smaller tasks, that can be solved simultaneously. The partitioning of the problem can be fixed in time and this case is known as so-called static load balancing. On the other hand, when the partitioning of the problem can change to reflect the evolving workload of individual tasks, one speaks about dynamic load balancing, see Schloegel, Karypis, Kumar (2002). The scalability of the solution process is one of the most important characteristics of any parallel algorithm.

The ideal scalability is difficult to obtain due to overhead cost of the parallel algorithm, as in reality, it is almost impossible to partition the problem into a set of independent tasks. In reality, synchronization and communication between tasks is necessary and often some parts of the problem are essentially sequential. To obtain optimal scalability, the load has to be distributed between processing units proportionally to their processing power. Other factors that can be taken into account include, for example, the communication bandwidth between nodes, available memory, etc.

The Finite Element Method (FEM) has become a widely used tool for solving engineering problems described by systems of partial differential equations. In FEM the differential equations are converted to the algebraic system of equations by using variational methods. In many cases the resulting system of equations is nonlinear, caused by various sources of non-linearity. In structural mechanics, the non-linearity typically originates from geometrical or constitutive relations. Nonlinear problems could not be solved directly, an iterative solution algorithm has to be employed, typically based on different variants of Newton method. The problem is solved in a series of load or displacement increment controlled steps in which the equilibrium state is iteratively obtained.

When solving the real problem on the parallel computer, the load balance can change during the solution. The first source of imbalance originates from the character of the problem. For example, in nonlinear problems, the transition from initial elastic material response to nonlinear regime is often associated with

[*] Ing. Michal Bošanský: Czech Technical University in Prague, Faculty of Civil Engineering, Thákurova 7, 166 29 Praha 6; CZ, michal.bosansky@fsv.cvut.cz

[**] Prof. Dr. Ing. Bořek Patzák: Czech Technical University in Prague, Faculty of Civil Engineering, Thákurova 7, 166 29 Praha 6; CZ

increased computational cost and this transition is often associated only to certain regions of overall domain. The second source of load imbalance includes external factors, which can change performance of individual processing nodes or communication network. This typically happens in non-dedicated cluster environments, where processing nodes and communication infrastructure is shared between users. In both cases, the gradual grow of imbalance can have significant effect on performance and on scalability.

The only way how to reflect growing imbalance is to adaptively redistribute work between processing units to restore load balance and thus to ensure optimal use of resources. The load imbalance can be detected by monitoring the time on individual processing units required to perform allocated work. The differences in processing time indicate imbalance. After imbalance is detected, the decision whether to restore the load balance or whether to continue has to be taken, This can be a complex task, as the load redistribution may be in fact a very complex problem with non negligible time requirements. The cost of load re-balancing may be higher than the cost of continuing further with slight load-imbalance. All these aspects have to be considered and are, unfortunately, problem and implementation specific.

## 2. Parallelization strategy in FEM

The idea of parallelization strategy in FEM is based on the domain decomposition paradigm. In general, two dual partitioning techniques for the domain decomposition can be recognized, see Krysl and Bittnar (2001). In this paper, the node-cut partitioning approach is used, where the cut dividing of the mesh into the partitions runs through nodes (and thus through element boundaries). The nodes on mutual partition boundaries are so-called shared nodes, the nodes inside partitions are so-called local nodes. The node-cut partitioning scheme can be interpreted as mesh decomposition using cuts passing trough shared nodes of the mesh without crossing any element, see Fig. 1. The cut strategy leads to duplication of the shared nodes between partitions.

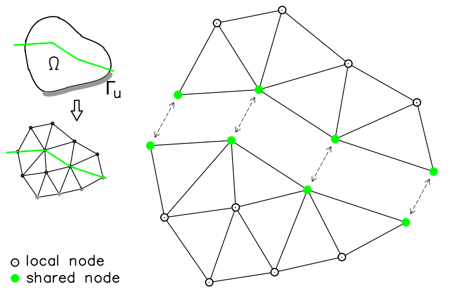

*Fig. 1: Node cut partitioning*

In the subsequent parts of the paper, we will consider that problem is nonlinear and is iteratively solved in a series of loading steps. The load imbalance is monitored by measuring the computational time required to process assigned work (process given partition). In a ideal case the individual times should be equal, indicating that the work load is ideally balanced. When imbalance is detected, the work (expressed in terms of individual elements) has to be redistributed to reflect actual performance of individual processing nodes. Individual elements have assigned weights, reflecting the relative computational costs. These depend on element type, material state (elastic, plastic, etc.). These weights are typically obtained in advance from measurements on estimated. Also the actual processing power of individual processing units has to be determined. The load re-balancing process in this paper is based on re-distributing the computational work proportionally to performance of individual processing units. In principle, additional factors including available communication band-with between individual processing units or available memory can be taken into account. Any load balancing should not only distribute the work according to processing powers, but also try to minimize the communication cost between individual processing units. In FEM, this means that the cuts between partitions (number of shared nodes) should be minimized. Failing to meet the secondary criteria can significantly impact overall performance, as the cost of communication (in terms of time required) is much higher than cost of computation. The load re-balance should also try to minimize reallocation of elements as possible to minimize the communication costs.

## 3. Mesh partitioning

The mesh partitioning is itself a complex problem. In this work, the ParMETIS library is used, see Walshaw (2007). The library provides parallel load balancing and rebalancing algorithms for general unstructured graphs and meshes, based on the parallel multilevel k-way graph-partitioning. The library allows to take into account the weights expressing the computational performance of individual processing nodes, as well as weightse expressing different computational requirements of individual elements. The determination of the weights of individual processing units is a subject of this paper. The appropriate weights are prerequisite for optimal load balancing.

### 3.1. Determination of weights

In this work, the approach to determine the actual processing weights of individual processing units is based on formulation and implementation of series of so-called micro-benchmark tests, that can be run on individual processing units at very low cost. The individual micro-benchmarks are procedures performing selected linear algebra tasks. The performance is estimated by measuring the computational times required to execute individual micro-benchmarks. The weighted sum of individual times is evaluated on each processing unit and from this the relative computational performances are determined. The micro-benchmarks are formulated in a way, that should represent typical operations during the standard solution process. In the case of this work related to FEM, the micro benchmarks include procedures solving small linear system of equations using Gauss elimination, numerical integration of simple polynomial, etc.

At present, this approach is implemented only to determine the processing weights for static load balancing. The results are presented and discussed in next section. The straight forward extension towards dynamic load balancing is the subject of ongoing work.

## 4. Example

The proposed approach has been evaluated using a benchmark problem of 3D finite element model of a nuclear containment reactor. The benchmark problem (marked as Jete250k) consists of 87k nodes, 959k tetrahedral elements with linear interpolation. The total number of equations is approximately 250k. The linear elastic response of structure subjected to self-weight has been analyzed. The problem has been solved in parallel on the two workstation with different performances. This setup has been chosen by purpose, to demonstrate the role of using appropriate processing weights. The first one (running Ubuntu 16.04 OS) with Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz processor with fourth cores consists of two logical processors per core. The second workstation has Intel® Core$^{\text{TM}}$ i3-2370M CPU @ 2.40GHz processor with two cores consisting of two logical cores. Workstations have been connected by faculty network on 100 Megabytes. Testing machines have 15GiB and 8GiB of system memory. The problem has been solved using object-oriented FEM code OOFEM, see Patzak (2011). The iterative linear equation solver from PETSc library has been used with block Jacobi preconditioner, see Belay (2001). The obtained results are presented in Fig. 2 and compared to results obtained with static load balancing using the equal processing weights. The obtained results confirm expected fact that when appropriate weights are used, the better performance is to be obtained.

## 5. Conclusions

The presented contribution deals with determination of actual processing weights of individual processing units. The processing weights are obtained by running a series of micro-benchmark procedures on each processing unit, that characterize typical operations of the problem under interest. The obtained weights are important inputs for any load (re)balancing algorithm. Particularly, in nonlinear problems, which are solved iteratively in a series of load increments, the initial load balance can be perturbed during the solution and optimal performance can only be obtained by load re-balancing that should take into account actual processing weights among other factors. The paper illustrates the importance of using actual processing weights on evaluating the performance on benchmark problem of linear elastic analysis of complex 3D structure with static load balancing.

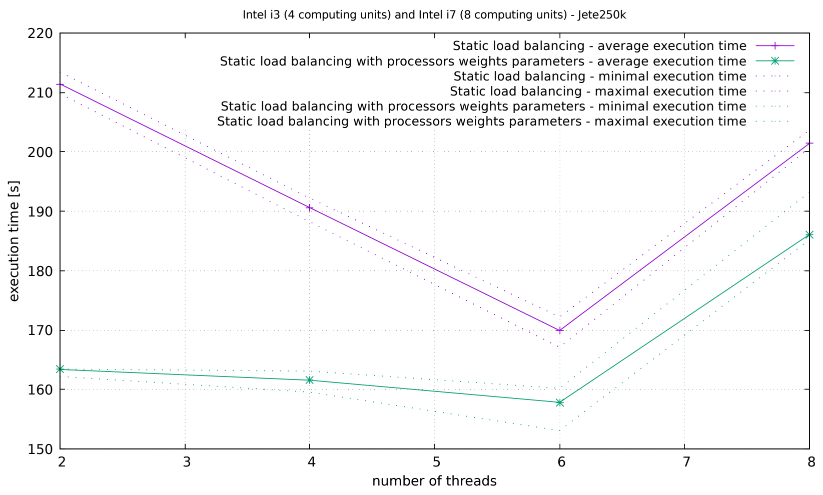

*Fig. 2: Execution times using the actual estimated processing weights compared to uniform processing weights.*

## Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS16/038/OHK1/1T/11 - Advanced algorithms for numerical modelling in mechanics of structures and materials. The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 ”Research Center for Informatics”.

## References

Schloegel K, Karypis G, Kumar V., (2002) Parallel static and dynamic multi-constraint graph partitioning. *Concurr Comput Pract Exper*, 14(3) pp. 27–58

Krysl P, Bittnar Z. (2001) Parallel explicit finite element solid dynamics with domain decomposition and message passing; deal programming scalability *Comput Struct*, 79(3):345-60

Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. (1996) MPI: the complete reference. *Boston: MIT Press*

Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley MG, McInnes LC, et al. (2001) PETSc Web page *http://www.mcs.anl.gov/petsc.*

Patzak B. (2011) OOFEM project home page *http://www.oofem.org.*

Walshaw C. (2007) Parallel multilevel graph-partitioning software – an overview *Magoules F, editor. Mesh partitioning techniques and domain decomposition techniques. Civil-Comp Ltd.*, pp. 27–58

Boman E, Devine K, Fisk LA, Heaphy R, Hendrickson B, Leung V, et al. (1999) Zoltan home page *http://www.cs.sandia.gov/Zoltan*