# AN APPROACH FOR LOAD BALANCING FOR SIMULATION IN HETEROGENEOUS DISTRIBUTED SYSTEMS USING SIMULATION DATA MINING

Irina Bernst, Patrick Bouillon, Jörg Frochte*, Christof Kaufmann

*Dept. of Electrical Engineering and Computer Science*
*Bochum University of Applied Science*
*42579 Bochum, Germany*
**E-Mail: joerg.frochte@hs-bochum.de*

**ABSTRACT**

This paper describes an approach to reduce the computation time of finite element simulations on heterogeneous distributed systems. This should be achieved by enhanced load balancing with help of machine learning techniques. Based on the hardware topology and the finite element problem the machine learning algorithm would be trained to predict the computation time in dependence on the geometric partitioning. The learned model will then be optimized to find the best partitioning regarding the computation time. The challenge of load balancing on non-homogenous clusters is to be solved to make distributed computing an accepted method for industrial users in the field of simulations.

**KEYWORDS**

Machine learning; Simulation Data Mining; Load balancing; Distributed Systems; Finite Element Method.

## 1. INTRODUCTION

Machine learning and data mining techniques have proven to be of great practical value in many application areas. However, there are fields which have not got much attention with respect to machine learning. The field of simulation in engineering software is one of these fields and many methods could be improved by these kinds of techniques. Earlier work in this field has demonstrated the ability to support civil engineers at design decisions of a bridge model (Burrows et al. [2011]).

As described in (Toselli and Widlund [2004]) there exist a lot of methods to solve such problems parallel. One of them is called domain decomposition, which involves solving a large system of equations by decomposing them into smaller sub-problems such that they can be solved in parallel. We want to utilize the benefits of distributed computing using heterogeneous computer systems for simulation based on the finite element method (FEM), see (Brenner and Scott [2008]) for details about FEM. Heterogeneous computing systems may suffer from synchronization issues, which lead to non-optimal performance. Load balancing is a technique to optimize the distribution of the sub-problems, for a general insight see e.g. (Kameda et al. [2011]). Different learning methods have been applied to the load balancing task in recent years. These methods, such as simulated annealing (Aydin and Fogarty [2004]) or genetic algorithms (Asadzadeh and Zamanifar [2010]; Wu [2004]) have high costs, hence they are not suited to solve complex problems. Our approach is to provide an assistance system, which offers the user a very efficient setting for the task distribution on his heterogeneous hardware system. As in (Burrows et al. [2013]) we are going to use machine learning and data mining to achieve this goal and provide assistant systems for engineers in the context of simulation parameter settings. One of the contributions of our paper will be a formal description of

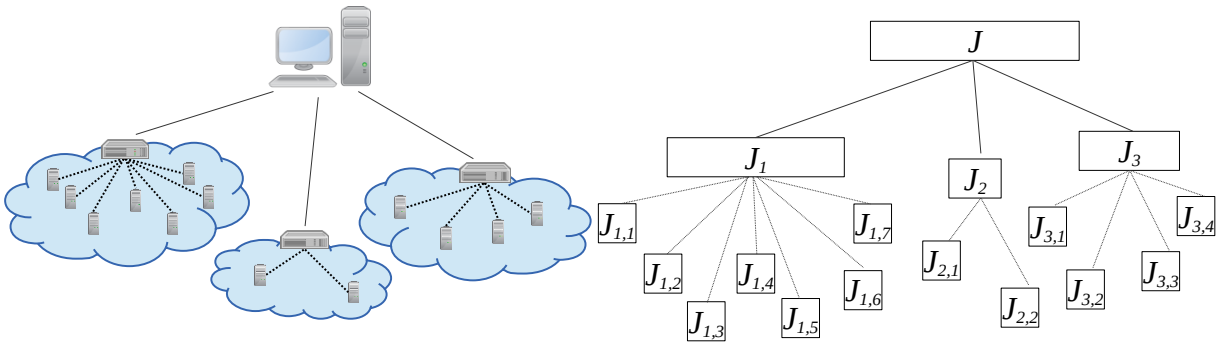
the problem and the extraction of a proper feature set for the purpose of Simulation Data Mining in the sense of (Brady and Yellig [2005]) or (Burrows et al. [2011]).

# 2. PROBLEM SETTING

## 2.1 Hardware topology

The problem consists of distributing a computational task over a distributed system in such a way the simulation processing time reduces, while taking into account the energy cost as a second optimization variable. The distributed system we will consider throughout this paper is shown as schematic in Figure 1.

Figure 1: Hierarchical distribution of the job $J$ on a heterogeneous system with three clusters.




An important part displayed in Figure 1 is the host computer, which triggers the task and manages the computation. Because of the heterogeneity of the computer system, which means non-equal computation power per computation node across different clusters, non-equal connection speed between the host computer and the sub-systems, an efficient distribution of the tasks is a non-trivial challenge. We have parts of the system, up to three in this example, that are homogenous among themselves. A more detailed description of a typical application can be found in (Frochte et al. [2014]), in which each of the homogenous parts of the system might be hardware rent by a different cloud provider. To identify the features for the machine learning task in the Simulation Data Mining workflow we need a more formal description of the problem.

Let $J$ be a computational task (or job) that can be scheduled in parallel. We want to distribute it on a heterogeneous system with a number of major sub-systems, like e.g. cloud hardware. For example we consider a system with three subsystems (Figure 1). Each of them has a number of computational nodes: $n_1, n_2, n_3$. We combine them in a vector $\boldsymbol{n}$. Let $c_1, c_2, c_3$ be the numbers of compute nodes that we use, such that

$$c \in \{(c_1, c_2, c_3) \in \mathbb{N}^3 | 1 \leq \sum c_i \wedge 0 \leq c_i \leq n_i\}$$

The job $J$ can be decomposed into $\boldsymbol{N} \leq \boldsymbol{3}$ main-jobs $J_1, J_2, J_3$ (see Figure 1) in the first level with different sizes. These main-jobs can again be decomposed into sub-jobs in the second level up to the number of available compute nodes in the cloud, i.e. $c_i \leq n_i (i = 1,..,3)$. So we have a job hierarchy. Therefore the total number of the jobs is

$$C = \sum c_i$$

Based on the hardware topology and the execution time the machine learning algorithm would be trained how to partition the job optimally.

## 2.2 Mesh partitioning

The geometric mesh provides the basis for the finite element method and consists of tetrahedral elements. For the simulation we use a domain decomposition approach (Toselli and Widlund [2004]) from the class of Schwarz methods. To apply this, it is necessary to split the mesh into several partitions. This also allows for parallelization of the simulation and the partitions can be associated with sub-jobs. If one just addresses a homogenous scenario on a single cluster a major goal is quite clear; all of these partitions must have the same size so that a perfect synchronization of the computation time between the nodes is possible. For more details concerning partitioning see e.g. (Nikishkov [2007]). In our heterogeneous application case we have to take much more aspects into account to receive good results. We will discuss this in section 3 more deeply.

For partitioning the mesh we use the program METIS[1] (Karypis and Kumar [1998]). An important input parameter for mesh partitioning in METIS is an array which length corresponds to the requested number of partitions. Each entry of the array specifies the desired weight for the corresponding partition, which determines the size of the partition.

To apply the Schwarz method on the described hardware scenario we use a hierarchical structure. Firstly, we choose the relative weights of the three clusters (see Figure 1). We denote these weights by $w_1, w_2$ and $w_3$ and collect them in a vector

$$w \in \{(w_1, w_2, w_3) \in \mathbb{R}^3 | \sum w_i = 1 \wedge w_i \geq 0\}.$$

These represent the sizes of the main-jobs $J_i$ in the first hierarchy level. However, computation is done on the compute nodes in the second level. Hence, the main-jobs are further partitioned into sub-jobs $J_{i,k}$ with $k < c_i$. The corresponding sub-partitions below a main-partition are equally weighted with $w_i/c_i$, since all nodes within one cluster are equal. If $w_i$ is zero we turn off the specific subsystem by choosing $c_i = 0$.

# 3. FEATURE SELECTION

## 3.1 Feature selection of the hardware topology

To understand the feature set it is important to see the parameters our software architecture provides for the generation of the jobs. Because we consider a heterogeneous system we have isolated the following features per homogenous sub-structure: *number of nodes in a cluster*, *normalized computational node speed*, *normalized data transfer rate in a cluster*, *normalized data transfer rate to the host*. The normalization means in this context a range from 0 to 1. Additional there is the global feature *number of clusters*.

Therefore we end up with a high dimensional feature vector ***h*** for the hardware topology, which includes the information from the vector ***n*** as well. We consider scenarios with two or three homogenous subsystems and in this case we end up with 9 respectively 13 features for the hardware topology.

## 3.2 Feature selection for the mesh and the FEM model

From the mesh side a big set of aspects have an influence on the performance of the load balancing because one has to keep in mind that the computation of different FEM models like e.g. an model for linear elasticity or for a laminar flow with the same method e.g. a Schwarz approach behaves differently on the same mesh. So to reduce the problem we right now concentrate on linear elasticity with a homogenous material parameter on the same mesh. This makes it possible for the selection of features to concentrate on a few meta-features which just depend on the geometry and their discretization within a mesh. We are right now working with the

[1] http://glaros.dtc.umn.edu/gkhome/views/metis

following global features: *total number of degrees of freedom, biggest and smallest diameter of a tetrahedron in the mesh, average density of vertexes per volume.* The number of partitions is not involved because it is redundant with $(c_1, c_2, c_3)$. Additional to these four global features we have got the same corresponding features per major partition, in our case up to three. So we have already 12 features. These features do not include aspects of the boundaries between the domains which are very important for the convergent speed of the domain decomposition methods. Here we have the effect that concerning the network traffic small boundaries are desirable while the convergent speed is increased when more data is exchanged per iteration step. Thus we have again a non-linear effect that is mainly associated with the number of vertices that are exchanged between two neighbor domains. Here it seems reasonable to catch the amount of data exchange by using the quotient of *overall exchanged degrees of freedom* divided by *total number of degrees of freedom* as global feature and on each main-partition. Because of the symmetry we have got only three new features. Let *A, B* and *C* be the homogenous sub-systems then we are interested in the quotient for the exchange between *A*&*B*, *A*&*C* and *B*&*C*. To sum this up we have a vector $\boldsymbol{m}$ of 16 features for the mesh.

## 4. LEARNING APPROACH FOR LOAD BALANCING

Our approach proposes to predict with a learned function $\boldsymbol{f}$ both the computational cost $e$ which is approximately proportional to the energy consumption, and the time $t$ a user has to wait for a simulation result. This function depends on the job distributions $\boldsymbol{c}$, the partition's relative sizes $\boldsymbol{w}$ and the mesh features $\boldsymbol{m}$. The hardware features are denoted by $\boldsymbol{h}$, which we assume to be constant here. However this enables flexibility in terms of hardware configurations for future work.

$$\binom{t}{e} = \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{c}, \boldsymbol{w}, \boldsymbol{m})$$

Then we define a weighted norm

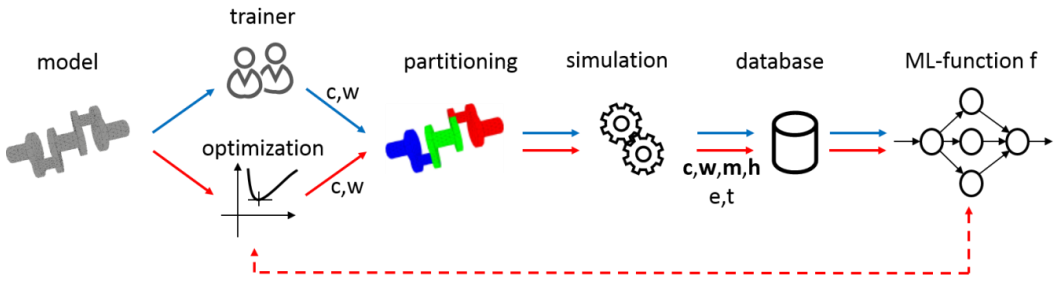
$$g(t, e) = \sqrt{\alpha t + \beta e}$$

with appropriate weights $\alpha$ and $\beta$. Once $\boldsymbol{f}$ is learned we have for a given situation in which $\boldsymbol{h}$ is constant and $\boldsymbol{m}$ is constant or depends on $\boldsymbol{c}$ and $\boldsymbol{w}$. Thus the resulting $\boldsymbol{F_{h,m}}$ is only depending on $\boldsymbol{w}$ and $\boldsymbol{c}$. We define a scalar function $g \circ \boldsymbol{F_{h,m}}$ with – summing up $\boldsymbol{c}$ and $\boldsymbol{w}$ – six degrees of freedom:

$$\left(g \circ \boldsymbol{F_{h,m}}\right)(\boldsymbol{c}, \boldsymbol{w}) \text{ with } \boldsymbol{F_{h,m}}(\boldsymbol{c}, \boldsymbol{w}) = \boldsymbol{f}(\boldsymbol{h}, \boldsymbol{c}, \boldsymbol{w}, \boldsymbol{m}(\boldsymbol{c}, \boldsymbol{w}))$$

Finally, we optimize this function over $\boldsymbol{c}$ and $\boldsymbol{w}$. The learning process can be drawn out by an artificial neural network or another appropriate method. The whole scenario in the sense of Simulation Data Mining is illustrated in Figure 2. To handle different physical models (diffusion, linear elasticity, convection dominated problems) there can be thought of one learned function $\boldsymbol{f}$ for each model, since the physics can make a difference for the job distribution.

Figure 2. Schematic of the learning process. The initial learning is indicated by the blue arrows. Thereby, to feed the database, some arbitrary parameters $\boldsymbol{c}$ and $\boldsymbol{w}$ are chosen for simple models. When the artificial neural network (ANN) is ready trained, the red arrows show the usual use case, where the ongoing learning process will still improve the accuracy of the ANN.



## 4. CONCLUSION AND FUTURE WORK

The application of Simulation Data Mining for distributed FE simulations presented in this work is being researched as an effort to develop an assistant system based on machine learning for engineers to (semi-) automatize a complex process.

The main contribution of our work relates to the description of a full Simulation Data Mining workflow applied to the task of the load balancing problem for heterogeneous distributed system for an FEM simulation. In this workflow the feature set we proposed plays an important role. The high dimension of the feature set – 34 features – is an additional challenge for the efficiency of the machine learning we address in our current and future work. However, the machine learning approach promises to be an effective method for the load balancing problem in heterogeneous systems.

## ACKNOWLEDGEMENT

This work is supported by the BMBF (Federal Ministry of Education and Research, Germany) under grant 03FH01812.

## REFERENCES

Asadzadeh, L. and Zamanifar, K., 2010. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, Vol. 52, pp. 1957-1965.

Aydin, M.E. and Fogarty, T.C., 2004. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. *Journal of Intelligent Manufacturing*, Vol. 15, pp. 805-814.

Brady, T. F. and Yellig, E., 2005. Simulation Data Mining: A New Form of Computer Simulation Output, *Proceedings of the Thirty-Seventh Winter Simulation Conference*', Orlando, Florida, pp. 285-289

Brenner, S.C. and Scott, L.R., 2008. *The mathematical theory of finite element method*. Springer-Verlag, New York-Berlin-Heidelberg, 3rd edition.

Burrows S., Frochte J., Völske, M., Martinez Torres, A.B. and Stein, B., 2013. Learning Overlap Optimization for Domain Decomposition Methods, *17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 13)*, pp. 438-449.

Burrows S., Stein, B. Frochte J., Wiesner, D. and Müller, K., 2011. Simulation Data Mining for Supporting Bridge Design, *Proceedings of the Ninth Australasian Data Mining Conference*, Balarat, Australia, ACM pp. 163-170.

Frochte, J., Kaufmann, C. and Bouillon, P., 2014. An approach for secure cloud computing for fem simulation, *Proceedings of the 11th International Conference Applied Computing*, Porto, Portugal (submitted).

Kameda, H., Li, J., Kim, C. and Zhang, Y., 2011. *Optimal load balancing in distributed computer systems*, Springer-Verlag, Berlin, Germany.

Karypis, G. and Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, Vol. 20, No. 1, pp. 359-392.

Nikishkov, G. P., 2007. Basics of the domain decomposition method for finite element analysis, *in Mesh Partitioning Techniques and Domain Decomposition Methods*, Saxe-Coburg Publications, Kippen, Stirling, pp. 119-142.

Toselli, A. and Widlund, O., 2004. *Domain Decomposition Methods – Algorithms and Theory*. First edn, Vol. 34 of Springer Series in Computational Mathematics, Springer, Leipzig, Germany.

Wu, A.S., 2004. An incremental genetic algorithm approach to multiprocessor scheduling. *Parallel and Distributed Systems*, Vol. 15, No.9, pp. 824-834.