8th. World Congress on Computational Mechanics (WCCM8)
5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008)
June 30 –July 5, 2008
Venice, Italy

Design of Parallel, Dynamic Load Balancing Framework in OOFEM

*** B. Patzák and Z. Bittnar**

[1] Czech Technical University, Faculty of Civil Engineering
Thákurova 7, 166 29 Prague, Czech Republic
borek.patzak@fsv.cvut.cz, bittnar@fsv.cvut.cz

Key Words: *dynamic load balancing, object-oriented FEM, message passing.*

ABSTRACT

The recent developments in many scientific and engineering disciplines brings in a new challenges for computational science. The software developers have to address demands for large scale computations. The parallel processing represents natural and efficient answer to such needs. Most of the parallel applications are based on message passing programming model, profiting from its portability across many parallel architectures.

The design of parallel algorithms requires the partitioning of the problem into a set of sub-domains, the number of which is equal or greater to the number of available processors. The partitioning of the problem can be fixed in run time (static load balancing) or can change during solution (dynamic load balancing). The later option is often necessary in order to achieve good load balancing of work between processors and thus optimal scalability. There are in general two basic factors causing load imbalance between individual sub-domains: (i) one coming from application nature, such as switching from linear to nonlinear response in certain regions or local adaptive refinement, and (ii) external factors, caused by resource reallocation, typical for non-dedicated cluster environments, where individual processors are shared by different applications and users, leading to time variation in allocated processing power.

The load balance recovery is achieved by repartitioning of the problem domain and transferring the work (represented by finite elements) from one sub-domain to another. The repartitioning is optimization problem with multiple constrains, optimal algorithm has to balance the work, while minimizing the work transfer and keeping the sub-domain interface as small as possible. Other constrains can represent different processing power of individual processors or may be induced by topology of network. In a recent years, many powerful mash (re)partitioning algorithms have been developed [2,3], that can facilitate this task.

The application has to continuously monitor the solution process and detect work imbalance. When imbalance is detected, the decision has to be made whether to recover load balance or continue with existing work distribution, depending on the magnitude of load imbalance and the cost of load recovery. Work transfer requires serialization of problem data (representing parts of solution vectors, elements, nodes, etc) into a byte stream that is sent over the network and unpacked, followed by topology update to reflect new partitioning.

The present contribution deals with design of an object-oriented framework for dynamic load-balancing implemented in the frame of object-oriented fem solver OOFEM [1]. The framework defines several classes representing fundamental components of load balancing algorithm: (i) load monitor, keeping track of solution process and detecting load imbalance (ii) load balancer, responsible for transparent work transfer (iii) load balancer plug-ins to implement analysis-specific data migration, (iv) external domain partitioner interface, and (v) high level communication services.

The role of these classes is to declare an abstract interface, which is implemented by derived classes, representing customized algorithms. Such design using abstract interfaces is necessary in order to achieve highly modular and configurable environment. The data migration task, represented by LoadBalancer class is facilitated by using flexible communication layer, build on the top of message passing library, and using already available serialization services of individual components, that were generalized to work with arbitrary data stream. The implementation ensures the necessary serialization into a byte stream and its transport into destination, unpacking received data, and updating of internal data structure to reflect changed topology of individual sub-problems.

Communication layer provides dynamic communication buffers, facilitating the complex data transfer. Message passing libraries typically provide communication using messages of pre-determined size, which requires to compute the message size in advance to allocate communication buffer and thus requires to implement size counting and packing procedures. This can be error-prone due to the need to maintain consistency. The dynamic messages allow to avoid the determination of message size and provide high-level abstraction. The implementation, using sequence of fixed sized messages, allowing to profit from asynchronous, non-blocking communication.

To facilitate mutual communication with a remote partition a specialized class (called ProcessCommunicator) is introduced. It maintains its communication rank, communication maps, and related communication buffers. Communication maps, which can be thought of as lists of entities (nodes, elements) that participate in the communication, are established according to the mesh partitioning prior to the actual analysis, separately for send and receive operations. In general, the communication maps can vary dynamically during the analysis to reflect the potential repartitioning, for example, due to the recovery of the load balance. In order to fully exploit the features of the non-blocking communication scheme, the communication buffers are provided separately for send and receive operations. It should be emphasized that there are separate instances of the ProcessCommunicator class on each partition, each dedicated to communication with a particular remote partition. This essentially enables an overlapping data exchange between partitions (making the message passing very efficient).

The capabilities and performance of developed framework will be demonstrated on several large-scale engineering problems, showing the scalability of implemented algorithm and advantages of dynamic load balancing when used in non-dedicated cluster environments.

ACKNOWLEDGMENTS
This work was supported by Ministry of Education of the Czech Republic - Project MSM 6840770003.

REFERENCES

[1] B. Patzák, OOFEM project home page, http://www.oofem.org, 2007

[2] K. Schloegel, G. Karypis and V. Kumar, A Unified Algorithm for Load-balancing Adaptive Scientific Simulations, Supercomputing, 2000.

[3] K.D. Devine, E.G.Boman,R.T. Heaphy,R.H. Bisseling and U.V. Catalyurek, Parallel Hypergraph Partitioning for Scientific Computing, In proceedings of IPDPS'06, 2006.