WebFEM: A WEB APPLICATION FOR DISTRIBUTED NUMERICAL SIMULATIONS

Lorenza Ferrario[1] ferrario@itc.it

Cristiana Armaroli[1] armaroli@itc.it

Elena Betta[1] betta@itc.it

Paolo Conci ing.paolo.conci@virgilio.it

[1]ITC-irst, Divisione Microsistemi, Via Sommarive 16, 38050 Povo (TN), Italy
Ph. +39 0461 314463 Fax +39 0461 302040

Abstract

In this work we present the research project aimed to designing and implementing *WebFEM*, a Web application for the efficient simulation of engineering problems whose solution is based on the Finite Element Method (FEM). *WebFEM* is accessible through the Web, and the final user interfaces are the most common Web browser pages. The architecture of the simulator has been accurately designed in order to allow an easy distribution of the computation over cluster of computers linked on the Web. The programming tools used for the implementation have been chosen in order to get portable and flexible Web software, also suited for educational purposes. The distributed architecture allows using *WebFEM* in advanced application contexts, like distance learning and simulation activities of distributed working teams.

Keywords

Finite Element Methods, Web-based simulation, Object-Oriented implementation, Computer cluster

1. Introduction

The FEM (Finite Element Method, [1]) is one of the most common numerical methods, thanks to its usability in many engineering, but not only, contexts. As other numerical methods its main aim is to reduce the solution of complicated PDEs (partial differential equations) set for continuum problems to the solution of a finite set of simpler equations. This can be achieved thanks to the discretization of the original model of the problem with a finite number of elements and nodes, where the equations have to be locally solved according to local boundary conditions. FEM software simulators evolved as local, stand-alone programs. Attention was made to the processing speed, more then to the easiness of use or to the sharing of the program on Internet. Today most spread programs (ANSYS[2], ABAQUS [3]) reflect this evolution. Moving from one release to the following, they pay more attention to add new element types or models then to the software design methodology and to the technology. The traditional FEM simulators are proprietary software, with or without graphical interactive interfaces, accessible locally or via a remote login from remote computers: in fact, the user opens a terminal on the simulator host. The limitations are many: the net load can seriously affect the efficiency of the session; graphical limitations are possible; the limited resources on the simulator host can strongly limit the number of concurrent users. Some optimisation is available in case of batch sessions, where the user connects to the host, sends his/her request and disconnects allowing the host to optimise the computation resource requests arriving from many users. The users re-connect after a certain time to download the results on their own computer. A step forward in the Web direction is proposed by the development of Web interfaces to traditional applications [4].

Today, the simulators field starts to show a strong interest in an evolution of the technology [5][6]. As a proof, we mention two recent research projects of the Swiss ETH Polytechnic (Zurich) and of the German IMTEK laboratory (University of Freiburg). The projects have developed new simulation tools specifically designed for microelectronics problems, introducing for the first time an object-oriented design of the code. In both cases the system runs on a single platform and is a stand-alone system with customized interfaces. A similar project is "FEMEngine", developed at ETH Polytechnic (Zurich) and at IMTEK (Freiburg, Germany) [7][8][9].

Today, there is also a strong research interest in studies and experiments exploring the validity of distributed computing solutions[10][11][12]. *WebFEM* places in this research context. The *WebFEM* project is aimed to define a new software architecture for Web oriented FEM simulation modules. The simulator is designed with the most advanced software engineering techniques and implemented with Internet tools so to become a real Web application. With the term "Web application" we mean an application accessible on the Web through Web browsers, which does not require additional local software (except a Web browser), portable to different platforms and operative systems, adapt to the integration with other Web tools. Orienting the applications to the Web technology has many advantages:

- no local installation of modules is needed

- no local powerful computers are required
- it is possible to share the tools among many users on the net
- the user accesses always an updated release of the simulator
- the access is through familiar interfaces, the Web browsers, with no need of specific training
- it is possible to imagine the connection to other Web applications
- the computing load can be distributed over cluster of computers.

With respect to the last item, there are at least two ways the *WebFEM* architecture allows exploiting the computing distribution:

- processing distribution: the possibility to distribute the processing on a cluster of computer (see following sections) allows handling big simulation problems without using expensive multi-processor computer. Further, it allows distributing several concurrent users requests on the cluster, balancing the overall application load (Fig.1)
- simulator modules distribution: basic processing modules (e.g. system solver, mesh generator) could be installed at each node of a *WebFEM* cluster; specific modules (e.g. PDEs, shape functions, post-processing) could differ at each node, according to the research interest of the users accessing each node. Occasionally, the user could need a PDE (or other specific module) not available on his/her cluster node: he/she can access the required module on another computer of the cluster. The possibility to use remote objects (see sections 2 and 3) allows a transparent access to all the modules of a cluster (Fig. 2). Research teams whose members work on different sites (countries, laboratories) are good candidates for an exploitation of simulation tools bases on this new architecture.

In section 2 is reported the description of the first experiment of the project: the implementation of a Web oriented FEM simulator (WebFEM1.0) and the distribution of the processing on cluster of 3 computers. In section 3, the first tests about the efficiency of the distribution of *WebFEM* on the cluster are reported. In section 4 considerations about the future development of the project are reported.

2. Approach and methods

The aforementioned targets (modularity, usability through the net, portability,...) have been pursued with the systematic use of the object-oriented paradigm and of software engineering techniques like the UML (Unified Modelling Language) [13][14]in all the stages of the project: analysis, design, implementation and test. The main programming language used, JAVA, is object-oriented. The HTML, XML and SOAP (the languages for the development of the user interfaces and for the network communication) are completely compatible with the most used Web servers and browsers. Where the tools could result not efficient for the problem (e.g. JAVA based scientific computation is still inefficient) the distribution of the processing on the cluster can introduce a sufficient compensation and avoid the need of dedicated powerful and expansive multi-processor computers.

The first step of the project has been the design of a basic FEM simulator, implemented as a Web application. The WebFEM1.0 [15] is the first release. It allows the simulation of 1D, 2D elastic static problems (Fig. 3). An accurate object-oriented design, based on UML analysis technique, allows adding easily new elements, a new set of shape functions and new PDEs, without changing the previous code, but adding independent JAVA classes. The simulator is completely implemented in JAVA and the interfaces are JServlet-based Web browser pages.

The JAVA language is less efficient with respect to other traditional scientific languages, e.g. C, Fortran, C++. Nevertheless the distribution of the processing on a cluster of computer can give adequate performances. FEM simulations can distribute the following algorithms which could benefit of a parallelization:

- mesh refinement
- stiffness matrix assembly
- solver.

This is possible thanks to the extreme object-oriented design, which created separated objects, classes (and, during the processing, independent instances) for elements, equations, shape functions and the organization of the solver.

The simulation is distributed over the computer cluster via remote calls managed by the Java Web Service Technology [16]. The methods solving the problems are implemented in Java objects which are replicated on the computers of the cluster and executed according to the remote calls activated on the master node (Fig.1 and Fig. 4).

In order to evaluate the efficiency of the distribution of modules processing we have set a cluster, made of three computers connected to a LAN. The full WebFEM1.0 has been installed on one workstation of the cluster, and copies of the classes that execute the parallel simulation algorithms are available also on the other two computers (a SUN workstation and a portable PC). I.e., the parallelizable parts of the code have been distributed on different computers. The user accesses the simulator on the central installation (i.e. where the whole WebFEM is installed) and when he/she runs a simulation, a scheduler sends calls to the cluster. Each call asks to one of the computer specified by the scheduler to run a piece of code and to return the result to the central installation. The communication among the computers is based on the SOAP protocol and the JAVA Web Services technology.

In detail, follows the description of the hardware and software set-up:

- hardware
  - o SUN Sparc Ultra5 OS5.8 - 128 MB of RAM - micro 167 MHz of clock
  - o SUN Sparc Blade100 - 1664 MB of RAM - micro ultrasparc 2e 500MHz of clock.
  - o Portable PC DELL Windows2000 OS - 128 MB RAM – Pentium III Intel 702 MHz of clock
- software:
  - o the JAVA2 Standard Edition in the version j2sdk1.4.0_02 [17];
  - o the JAVA Web Services Developer Pack in the version jwsdp-1_0_01, an integrated toolset that allows developers to build, test and deploy XML applications, Web services, and Web applications through SOAP protocol. It integrates a Tomcat Web server [16][18].

We have run a first test on the distribution of the assembly of the stiffness matrix of 2D geometries in static elastic simulations. The assembly (the WebFEM1.0 KMatrixAssembly class) is basically a loop of calls to the algorithm which calculates the stiffness matrix for each element of the mesh (the WebFEM1.0 “TermKElement” class). A copy of this algorithm has been installed on all the computers of the cluster. WebFEM1.0 implements a scheduler which is called by the stiffness matrix assembly algorithm loop. The scheduler instanciates calls (JAVA RPC calls) to local or remote objects. Each call can run one or more time the remote object methods. In order to allow the asynchronous return of the results from the remote object (i.e.: the scheduler can run a further call before the previous has returned its result) the JAVA thread class and its methods have been used. Fig. 4 shows the programming implementation of the distribution.

3. Results

In order to have a first evaluation of the efficiency of the proposed processing distribution architecture we have compared the same number of stiffness matrix assembly operations both on the stand-alone *WebFEM* and on the version distributed on the described computer cluster. We have run assembly operations referred to rectangular planes meshed with 100 and 484 elements. These meshes correspond to 6400 and 30976 nodal operations.

The stand-alone WebFEM is installed on the Dell PC. In the stand-alone simulations the number of nodal operations to be executed is the only varying parameter. In Tab.1 we report the average processing time for the different simulations (i.e. 100 and 484 elements meshes) in the central column.

The simulations run on the cluster of computers can exploit processing distribution varying two parameters: the global number of operations to be executed on each computer and the number of threads. On the master computer (i.e. where the central simulator is installed, the portable Dell PC) it is possible to activate one or more thread directed to each computer of the cluster. Each thread, through a remote call, runs a packet of remote operations. It is possible to better exploit the cluster sending more then one packet to each computer, i.e. running two or more parallel packet on each computer. For this reason we have tried to vary the number of threads among which to distribute the global number of operations in order to see if there is a better combination. Changing this parameter from 3 (the situation where we run one packet of operation on each computer of the cluster) to 90 (30 parallel packets running on each computer) we have found that in our cluster seem to be an optimum number of threads, between 30 and 50. This optimum point is common in both cases: 6400 and 30976 operations. Net traffic factors as well as the computer resources are responsible for this fact and in future development of the project we will investigate on balancing algorithms that could help in automatically finding the optimum number of thread for each computer of the cluster. In Table 1, in the rightmost column, we report the best results obtained in the distributed simulations. In Fig. 5 we report two series of global processing times plotted versus the number of parallel threads activated: the first series refers to 6400 operations (100 elements mesh), the second to 30976 operations (484 elements mesh), both cases processed by the version of *WebFEM1.0* distributed on our cluster of computer.

We have started to compare the matrix assembly times of our system with those of a commercial software, ANSYS [2]. In the first benchmarks we found an average ratio of 5:1 in favour of ANSYS. But further tests, with a broader cluster, are needed in order to have affordable results.

Operation number	**Average global processing time (msec) – stand alone WebFEM1.0**	**Average global processing time (msec) – distributed WebFEM1.0**
6400	3462	3402
30976	14883	16800

Table 1 – Average processing times of the simulations run

4. Conclusions and future work

The first results show a good velocity of the distributed architecture with respect to the traditional stand-alone programs. Considering that in this first test the distribution has been applied only to one of the time-consuming FEM algorithms, i.e. the stiffness matrix algorithm, these results are favourable to the JAVA RPC technology introduced and encourages to keep on developing the original idea of a distributed Web FEM tool.

Future steps of the research project will be extending the distribution to other time-consuming algorithms, e.g. the solver and the meshing and mesh refinement, studying the system behaviour in larger clusters and improving the benchmarks with commercial software.

The generality of the technical solutions adopted suggests testing the proposed architecture on other numerical methods or time-consuming computations. Emerging fields where the aforementioned ideas could be exploited are the distance learning and the e-business.

Bibliography

[1] O. C. Zienkiewicz, R. Taylor, *The finite element method Voll. I-II* (McGraw-Hill: London, 1991)
[2] *Ansys Multiphysics* (http://www.ansys.com)
[3] *ABAQUS* (http://www.hks.com)
[4] *MOPLE* (http://www.cimne.upc.es/projects/mople/)
[5] G. Fox, Introduction to Web Computing, *Computing in Science and Engineering", March-April*, 2001, 52-53
[6] G. Fox, W. Furmanski, Computing on the Web. New approaches to Parallel Procssing. Petaop and Exaop Performance in the Year 2007, *Internal Report Northeast Parallel Architectures Center, Syracuse University*, 1997
[7] *ISE-TCAD* (http://www.ise.ch)
[8] Emmenegger M., *An Object-Oriented Design for Efficient Microsystem Simulation Ph. D. Thesis No. 1334.* (ETH Zurich, Verlag der Fachvereine, Zurich, 1999)
[9] Taschini S., *Modelling of slender structures in microsystems, Ph. D. Thesis No. 13764* (ETH Zurich, Verlag der Fachvereine, Zurich, 2000)
[10] http://www.npac.syr.edu
[11] D. P. Anderson, J. Kubiatowicz, The World Wide Computer, *Scientific American*, March 2002, 28-35
[12] http://www.aspenleaf.com/distributed/distrib-projects.html
[13] R. I. Mackie, *Object-Oriented Methods and Finite Element Analysis* (Saxe-Coburg Pubblications, 2001)
[14] G. Booch, J. Rumbaugh, I. Jacobson, *UML: The unified modeling language - User Guide* (Addison-Wesley 1999)
[15] L. Ferrario, C. Armaroli, WebFEM: a Web based FEM simulator, *Virtual Prototyping Today: Industrial Impacts and Future Trends*, Bergamo, Italy, October 3-4 2002
[16] http://java.sun.com/webservices
[17] http://java.sun.com
[18] http://jakarta.apache.org/tomcat/index.html

Figures

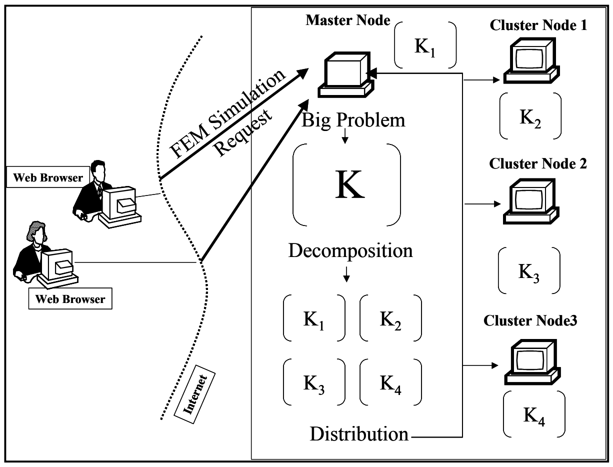


Fig. 1 – Distributed architecture: distributing the processing load

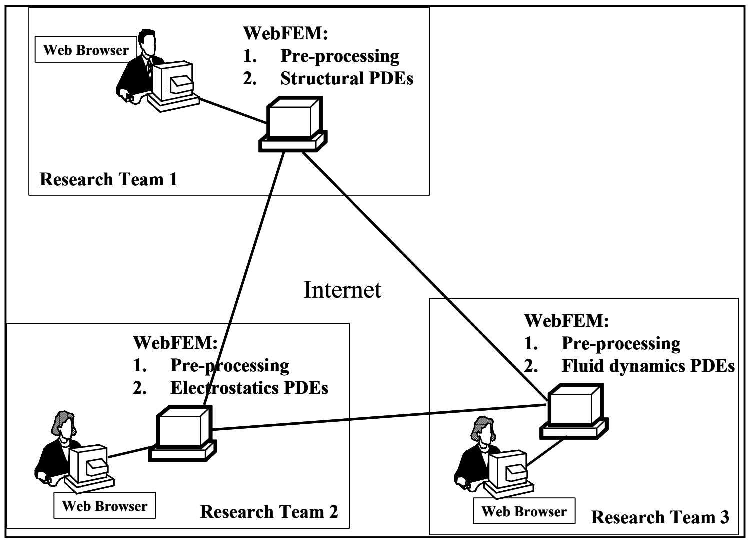
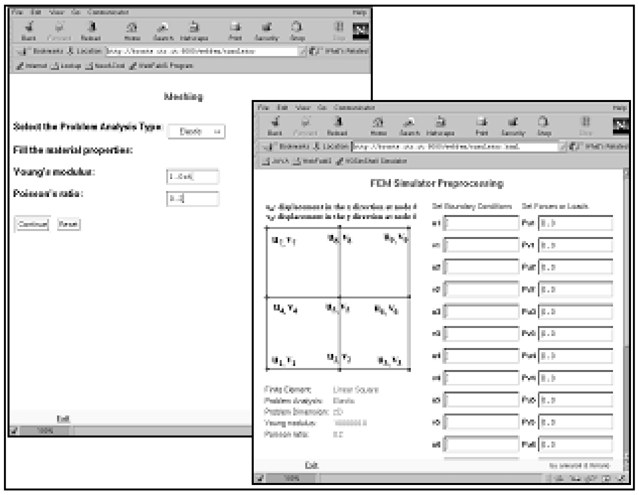


Fig. 2 – Distributed architecture: access to remote simulation objects


Meshing

Select the Problem Analysis Type: Elastic

Fill the material properties:

Young's modulus: 1.0e6

Poisson's ratio: 0.2

FEM Simulator Preprocessing

Set Boundary Conditions

Set Forces or Loads

Finite Element: Linear Square

Problem Analysis: Elastic

Problem Dimension: 2D

Young modulus: 1000000.0

Poisson ratio: 0.2

Fig. 3 – WebFEM1.0: the selection of the PDE and of the boundary conditions

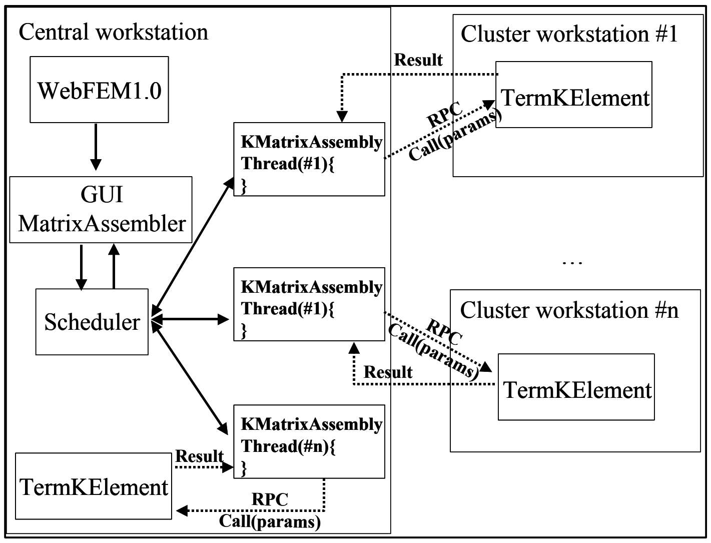


Fig. 4 – Distribution programming solution

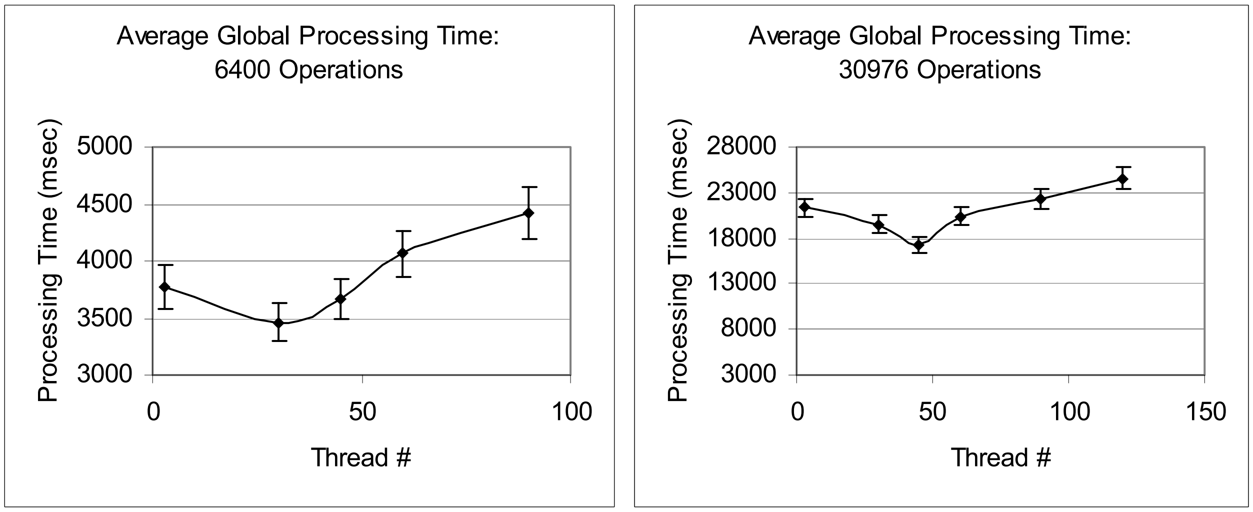


Fig. 5 – Processing times of the simulations run on the cluster of computer