This is the published version (version of record) of:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30004863

Copyright : ©2002 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Object-Oriented Finite Element Analysis: A Distributed Approach to Mesh Generation

1Yang Xiang, 2Yang Ping, and 1Wanlei Zhou
1School of Computing and Mathematics, Deakin University,
Burwood 3125, Victoria, Australia
{yang, wanlei}@deakin.edu.au
2Department of Civil Engineering,
Hong Kong University of Science and Technology,
Hong Kong, P. R. China
ceyping@ust.hk

Abstract

Recently the object-oriented finite element method (OOFEM) attracts the attention of lots of researchers. Compared with traditional Finite Element Method, OOFEM software has its advantages in maintenance and reuse of software. Moreover, it is easier to expand the architecture to a distributed one. In this paper, we introduce a distributed architecture of object-oriented finite element pre-processor. A comparison between the distributed system and the centralised system shows that the distributed one presented in this paper greatly improves the performance of mesh generation. Other finite element analysis modules could be expanded according to this architecture.

1. Introduction

1.1 Overview of Object-Oriented Finite Element Method

Computing technology nowadays makes it possible to solve complicated problems that are difficult to work out before. Many problems that could hardly be figured out before now can be solved by the Finite Element Method (FEM) [6]. FEM was developed in the 1960’s when the computing facilities were very limited and the programming languages were still juvenile. Almost every Finite Element (FE) program developed at the time was procedure-oriented and written in FORTRAN.

However, procedure-oriented FE has lots of shortcomings. A procedure-oriented finite element program may cost years to develop; yet the reusability of the program is not satisfiable. When a new algorithm or a new kind of element is designed it is difficult to apply it in the software because the alteration of one subroutine may affect the whole program. The risk of errors being generated may be increased greatly. So it is hard to maintain and reuse the traditional FE software because the software engineer has to be in control of every particular detail of the whole program. Since the reusability is a very important norm to judge the performance of the software, the traditional method of designing FE engineering software should be seriously reconsidered.

Moreover, the pre-processing is always the bottleneck of applications of the FE technology. The FE mesh generation is a painful task troubling FE software engineers and FE analysers, since the time-consuming work of pre-processing is very burdensome. Only a selected group of individuals has the technical expertise required to apply this method in professional practice.
When the FE engineering software system becomes more and more complex, the FE software crisis occurs.

Fortunately, coupled with the development of hardware has been the development of software and programming languages. With lots of merits, such as data abstraction and encapsulation, inheritance and polymorphism, the object-oriented technology reflects the concerns of software developers. Then the object-oriented approach becomes attractive for the FE engineering software development. In the last few years many articles on Object-Oriented Finite Element (OOFE) research have been published. Forde, et al., published one of the first publications of the OOFE in 1990[4].

1.2 Distributed Computing in OOFEM

To achieve high performance of FE analysis, researchers naturally think of distributed computing or parallel computing. Parallel computing on multiprocessor computers has been researched for many years for traditional FEM. Many excellent algorithms for parallel finite element analysis have been proposed [1][2][11][9].

Divide and conquer is a traditional technique in order to solve large-scale problem. In finite element analysis, domain decomposition techniques are employed because: (1) Geometry repeats itself; (2) Memory is limited to solve much larger problems; (3) Superior convergence rate of domain decomposition methods over other iterative methods; and (4) Potential for efficient parallelization because of data locality [7]. However, current parallel solutions are poor in flexibility and extensibility. In order to perform FE analysis, we must have some expensive hardware. And these hardware develop slowly, for example, today even the personal computer have the computing capability as the parallel machines yesterday. Moreover, it is difficult to port the source code from one operating system to anther because the compilers and libraries differ from system to system. Therefore, there are lots of disadvantages in traditional parallel FE systems.

An object-oriented programming philosophy is a suitable solution for distributed processing to take advantage of Object-Oriented Programming (OOP)’s natural abstraction and encapsulation of a programme’s components because a fully encapsulated component easily lends itself to distribution to remote machines [3]. On the other hand, it is easy to use a LAN to implement the distributed finite element analysis. No expensive parallel machine is required—a LAN is often available in offices. So we can obtain good quality/price rate by using this method.

Currently, the Message Passing Interface (MPI) is commonly used for process-to-process communications. [8] However, to build a distributed FE system by OOP is still a new area, no distributed object-oriented finite element system has been built so far. So the purpose of the current research is to design and implement a portable distributed object-oriented program architecture that is flexible for further expansion, with low computing cost and high computing performance.

In this paper, we first introduce a fundamental object-oriented finite element system architecture, and then pay particular attention to distributed pre-processor of an object-oriented finite element system. A distributed object-oriented mesh generation scheme is described in detail. An example demonstrates the benefits of distributed scheme. The distributed object-oriented FE architecture presented in this paper improves the performance of mesh generation greatly with low cost and easy implementation.

2. The OOFEM System Architecture

2.1 Software Structure

The distributed approach presented in this paper is based on an object-oriented finite element system: MEG+. We first introduce this fundamental OOFEM system architecture. The software architecture of MEG+ has the object-oriented style based on data abstraction and object-oriented organization. Data representations and their associated primitive operations are encapsulated in an abstract data type. The organization structure is shown in Figure 1.
As illustrated in Figure 1, the system consists of six modules. The graphical user interface links the user and the system. The user establishes a graphical objective model for finite element analysis and checks the result of the computation. The database management module exchanges data among different parts. The core of the system is made up of the pre-processor, the solver core and the post-processor. After the mesh generation of the pre-processor, the database manager sends the information of the mesh to the user interface. Then the user validates the information and sends it to the solver. The post-processor gets the information of results and shows it to the user.

The objects in OOFE can be classified into four categories as follows, Mathematical classes, FE-Object classes, CAD classes and Control classes. The following sections outline the components of some of these classes.

2.2 Mathematical classes

The mathematical classes include Matrix, Vector, EquationSolver, and so on. These classes are used to build up and solve the finite element equations.

2.3 FE-Object classes

The FE-Object classes include Domain, Mesh, BackMesh, Element, Node, GaussPoint, DOF (Degree Of Freedom), Constraint, Material, Load, and so on. The FE-Object classes are the most important and comprehensive classes. There are various definitions of classes, however, in this paper, the objective of classification is to build clear, methodical classes and to seek the efficient solving of FE equations.

- **Domain.** A geometric concept, the Class Domain, represents the field of FE analysis. It is built on multiple inherited classes. SubDomain divides the system into sub-parts such as SubDomain. SubDomain is divided into FE mesh, which includes the elements and the nodes.

- **Mesh and BackMesh.** The Class Mesh is the core class in the pre-processor. It generates the mesh data automatically, stores the data to the database, and draws the mesh in the CAD system. After the FE mesh is generated and checked in the graphical interface, the data of mesh, the element and node data will be transferred to the database.

- **Element.** As an abstract basic class, the Element consists of different element types. In OOFE, the work of coding is greatly decreased in element design because of the inheritance of object-oriented technology. When the basic element class is set up, other subclass elements can be easily added. Because lots of virtual functions can be used to express the methods, which are identical for all the elements derived from the same basic element class. Therefore we can view it as a good example of the advantages of OOFE over the conventional FE. This means the effort to make large-scale FE software becomes easier in contrast to the procedure FE software engineering. The attributes of class Element include the element number, type, node information, etc. The functions are calculation of stiff matrix, calculation of stress or strain matrix, calculation of Jacobi matrix, etc. It also can store the data to the database.

- **Node.** The Node class has some functions of input and output, and other calculations. Moreover, the number of nodes, DOF information of the node, coordinate system of nodes are the attributes of this class.

- **Material.** The attributes of the Material class include the number of materials, the material parameters, etc. The functions include input and output the data, modify the data, etc. The Material class also has the link between the graphical interface and the database.

- **Load.** The FE analysis includes several load steps and different load types. The Load class is similar to the
Element class because it has the basic class and the particular load classes. In the upper level, the Load class can be classified into NodeLoad and ElementLoad. NodeLoad is the load applied on the node, for example, the convergence force on a node, or temperature on a node. ElementLoad includes surface load, body load, etc. The control classes control the load because in the whole analysis the load may be divided into lots of steps and applied at a different time.

2.4 CAD classes

The CAD classes include Point, Line, Polygon, Text, etc. In a 3D situation, these classes may include other solid objects. In MEG+, all the input of an FE user can be done in the CAD system. Although this system is simple, it has most functions an FE user needs.

2.5 Control classes

The Control classes include GlobalControl, Time, and so on. The Control classes are used to control the non-linear and time stepping.

- GlobalControl. The GlobalControl class plays the key role of the FE analysis. It controls the analysis type, for example, the linear analysis or non-linear analysis and controls the global time stepping. So generally speaking, it controls the FE analysis.

- Time. The Time class is applied in time stepping. The attributes of Time class are the length of the time stepping, the number of stepping, etc. It has links with the database.

The MEG+ system includes three parts, the FE-Object classes, the CAD classes and the Control classes. The Control classes are the centres to control the process and exchange information between different parts. This part controls the mesh auto-generation and the solver. The FE-Object classes can manage the data by themselves.

3. Distributed System Design

3.1 Mesh Generation

Many mesh generation algorithms such as advancing front method, tri/tet method, octree/quadtree method, have been developed and implemented by researchers [5]. In this paper, we use advancing front technique to generate 2D unstructured quadrate finite element mesh.

Although this method tends to be somewhat heuristic, the resulting elements are generally of very high quality. In the 2D version of advancing front method, the input to the method is a set of meshed “loops”. A loop consists of a set of oriented geometric curves. Nodes have been placed along the curves at a predefined spacing. Curve meshing is a subject of research, but suffice to say that an algorithm has been employed to place nodes on curves. Most advancing front methods will form a data structure called a “front”, which consists of a segment bounded by two adjacent nodes on a curve. The initial fronts are formed from the initial curve mesh nodes. The main loop of the algorithm involves processing each front, one at a time, building the mesh from the boundary, advancing towards the interior of the domain until no fronts remain in the mesh. Unfortunately this method is among the less efficient (slower) methods. In section 4, we improve the performance of this method by using a distributed approach to obtain a better quality and higher speed.

3.2 Distributed Scheme

A scheme to implement the distributed mesh generation is presented in this section. The main idea is to partition the original domain into subdomain and then distribute them to different processors (computers). Each subdomain could be meshed independently when the boundary condition is defined. The scheme is shown as figure 2. At the first step, the domain is meshed coarsely. Then according to the computational ability of each node of the LAN, the domain is partitioned into several subdomains. After that, the information of subdomains is distributed to different processors. When each processor finishes mesh generation, it transfers the mesh data to the central
controller.

Generate initial background coarse mesh in domain

Partition the background mesh into a number of subdomains

Define the boundary condition of each subdomains

Distribute the data of each subdomain to different processor

Each processor generate the mesh data independently

Transfer each part of mesh data to central controller

Figure 2. Distributed Scheme

3.3 Extension to a Distributed Architecture

The distributed object-oriented mesh generation is based on the software structure of figure 1. The pre-processor module is extended to a distributed version, while other modules are still run on a centralised machine.

The distributed pre-processor module is based on the client/server model. The sub-mesh generators (application programs) are executed in each workstation. The server stores the database, manages the domain partition, collects the mesh data and does other work required by FE analysis. The server and clients (workstations) communicate with each other during the pre-processing stage.

Some classes are added or modified to the MEG+ system as follows. First, two classes are added, CentralControl and SubControl. These two modules are the most important for server and clients to communicate with each other. Second, the mesh generation classes are shifted from a centralised version to the distributed version. All the components run on a workstation are managed by the SubControl class.

3.4 Control classes

In the centralised computing module, only the GlobalControl class is used to control the whole system. In distributed architecture, however, the management and communication classes such as CentralControl and SubControl classes must be employed.

The CentralControl class partitions the whole domain into several parts according to the number of client machines linked to the server. Then it passes the domain information to every client application. A PULL model is used in this class. That means we can find out when the mesh data is ready on each workstation and pull the mesh data from it.

The SubControl class plays an important role on each workstation. It collects the definition data for mesh generation. Then it uses the data to generate the mesh independently and automatically. When the central control module pulls the mesh data, it transfers the data to the server.

3.5 FE Objects on Workstation

The FE Objects on workstation are SubDomain, SubMesh, BackMesh, BackElement, BackNode, and so on. The SubDomain receives the information passed from the server; the data it contains is used to generate the mesh. The Class SubMesh is the core class in a workstation. It generates the mesh data automatically, stores the data to the database, and waits for the SubControl class to transfer data. After the FE mesh is gathered and checked in the graphical interface of the central machine, the data of mesh, the element and node data will be transferred to the database for further analysis.

Figure 3 shows the relationship between the classes.
4. An Example

4.1 Problem

The proposed distributed object-oriented finite element pre-processor was implemented in an array of computers (LAN) (one 800MHz/Pentium-3 server, and other five 400MHz/Pentium-2 workstations). The switch of the LAN is 100 Mbps. The workstations used are from 1 up to 5.

An annulus is meshed by using this architecture. The inner diameter is 25cm and the outer diameter is 70cm. Left and right edges are defined as the mesh-dense areas. The coarse mesh generated by the CentralControl is shown in figure 4. A 5-workstation example is shown in the following figures. The domain is partitioned automatically by the CentralControl and then passed to different client machine. The ID number identifies to which workstation the information of SubDomain should be passed.

The final mesh generated by the FE system is shown in figure 5a. The submesh generated in workstation (ID=1) is shown figure 5b. Number of total elements generated is 9764.

4.2 Results and Discussion

The time costs for mesh generation on a single machine (centralised version) and the distributed version are shown in table 1. We also use speed-up to compare the performance between difference systems. Speed-up is
defined as the ratio of centralised system time cost divided by the distributed system time cost. The speed-up histogram of different workstations used is shown in figure 6. The speed-up of 1 server plus 1 workstation is less than the centralised one. It is because the distributed version needs more time for server and workstation to communicate with each other.

Table 1. Time Cost and Speed-up

<table>
<thead>
<tr>
<th>Number of Workstation(s)</th>
<th>Time cost (seconds)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (centralised version)</td>
<td>35.763</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>42.374</td>
<td>0.84</td>
</tr>
<tr>
<td>2</td>
<td>21.646</td>
<td>1.65</td>
</tr>
<tr>
<td>3</td>
<td>17.360</td>
<td>2.06</td>
</tr>
<tr>
<td>4</td>
<td>15.071</td>
<td>2.37</td>
</tr>
<tr>
<td>5</td>
<td>14.505</td>
<td>2.47</td>
</tr>
</tbody>
</table>

Figure 6. Speed-up of different workstation(s) used

As more workstations are involved, the performance of mesh generation improved. That is, the speed-up increased. This is due to the computation work distributed to different workstation is decreased and each workstation should generate less elements. However, as more workstations are involved, the communication time between the server and workstations increases. Therefore, the deviation of the speed-up curve slows down. The communication cost negatives the benefit the distributed system brings. We use Efficiency to describe this effect. Let efficiency=(time cost of 1 workstation)/(time cost of \(N\) workstations \(\times\) number of workstations).

That is, efficiency=100% when one server plus one workstation system is employed. The efficiency decreases when more workstations are involved. The efficiency is shown in figure 7.

From the results we can find out that by using the distributed object-oriented system, the performance of mesh generation is improved, and the time cost decreases. However, the relationship between computing efficiency and the number of distributed workstations of the system still needs more research.

Figure 7. Efficiency of different workstation(s) used

5. Conclusion

In this paper we presented a distributed object-oriented architecture to solve the mesh generation problem. The classes of this distributed system is introduced in detail. An example implementation on a LAN is also presented.

From the results of mesh generation we can find that the performance of pre-processing of object-oriented finite element analysis is improved. However, because of the communication cost between the server and workstations, the efficiency of the computation decreases.

The architecture presented in this paper has potential benefits in maintenance, reusability, code readability, expendability, fast prototyping, computing efficiency, error tolerance, etc. In this paper we have shown that, the distributed architecture can solve large-scale mesh generation problems in a LAN with low cost. For further research we plan to extend other components of the
object-oriented finite element system to a distributed version according to this architecture. Other areas for further research include the scaling and the cost of communication.

References