Advances in Engineering Software 66 (2013) 3–9

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Dynamic analysis of structures on multicore computers – Achieving efficiency through object oriented design

R.I. Mackie *

Civil Engineering, School of Engineering, Physics and Mathematics, University of Dundee, Dundee DD1 4HN, UK

ARTICLE INFO

Article history:
Available online 18 April 2013

Keywords:
Component-oriented
Object-oriented
Eigenproblems
Transient analysis
Seismic analysis
Parallel computing
Distributed computing

ABSTRACT

The paper examines software design aspects of implementing parallel and distributed computing for transient structural problems. Overall design is achieved using object and component oriented methods. The ideas are implemented using .NET and the Task Parallel Library (TPL). Parallelisation and distribution is applied both to single problems, and to solving multiple problems. The use of object-oriented design means that the solvers and data are packaged together, and this helps facilitate distributed and parallel solution. Factory objects are used to provide the solvers, and interfaces are used to represent both the factory objects and solvers.

1. Introduction

There have been many changes in computing hardware and software. One of the most recent is that virtually all computers are now multi-core, typically dual or quad core. This has implications for the design of software, which has yet to be fully realised. Techniques for parallel computing have largely been developed for high-performance computing (HPC) on super computers or clusters of workstations. This has some relevance to the new world on everyday computers, but there are other possibilities as well. One of these is the user-interaction and software. HPC is largely geared at solving highly complex problems that require massive computing resources and take a long time. This has some relevance to normal computing as desktop computers are now capable of solving much more complex problems than used to be the case. Furthermore computers are linked together on intranets and the internet, so the creation of clusters of computers is relatively easy. However, the usability of engineering software, and the way the software can be used in design is equally important, and the power and architecture of current computers changes what is now possible. This paper will look at some of the possibilities in the area of dynamic analysis of structures, with particular focus on seismic engineering. However, the ideas presented are much more widely applicable. The paper will also emphasise key software design decisions that facilitate the exploitation of the modern computing environment.

Current software developments are addressing the new environment, in particular version 4.0 of Microsoft's .NET framework. The .NET framework was developed with distributed and multi-threading computing in mind, and has had facilities for simplifying software development for this world. Version 4.0 has introduced the Task Parallel Library (TPL) [1] to facilitate software for multi-core computing. There is a tendency to think that HPC should be performed using MPI and on Linux machines. MPI and Linux definitely have their place, but the reasons for using .NET in the current work are:

- Windows is the most commonly used operating system. Now on supercomputers Unix/Linux based operating systems are by far the most common. However, the overarching motivation behind the current work is that parallel and distributed computing are now part of the mainstream computing world. Therefore it is appropriate to consider the application of technologies designed for mainstream computing.
- The .NET infrastructure is available on all Windows computers, so there is no need to install any further software.
- .NET provides parallelisation and distribution in an object and component oriented fashion, so it is consistent with the overall design philosophy.

Dynamic problems in structures are among the more expensive in computational terms, they also generate an incredible amount of data. So there are two problems that need to be addressed in software development: efficient numerical problems, and data handling. Work on parallelism typically focuses on parallelising

* Tel.: +44 1382 384702; fax: +44 1382 384816.
E-mail address: r.i.mackie@dundee.ac.uk

0965-9978/$ - see front matter © 2013 Civil-Comp Ltd and Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2013.03.006

the solution of a single problem. However, it can also sometimes be useful to solve several problems at once, and in terms of parallelism this is actually simpler as one can expect to achieve greater speed-up. The work described in this paper will apply parallelisation to both aspects. While earthquake engineering provided the motivation for this work, the software engineering aspects of this work are much more widely applicable.

The motivation for the current work is the analysis of space structures under seismic loading. On the one hand, the object-oriented implementation of modal and transient analysis algorithms will be examined. Earthquakes are by their very nature uncertain, so it is useful to look at the behaviour of a structure under several different earthquakes. The new environment makes this much more feasible, and the design of software to facilitate this will be described.

The advantages of parallel and distributed processing can be used in various ways:

1. Application to individual algorithms.
2. Application to solving multiple problems simultaneously.
3. Overall program design.

This paper will look at all three of these aspects, making use of object and component oriented programming design. It should be noted that the primary emphasis of the current paper is on program design rather than numerical efficiency. Therefore the focus is on explaining design approaches to make implementation of flexible parallel and distributed programs easier on mainstream computers.

2. Literature review

Modal analysis and time stepping are both well established, and there are many algorithms for solving these problems. A good description of general techniques relevant for finite element analysis can be found in Bathe [2]. There has been a significant amount of work on parallelisation of both eigenproblems solvers and time stepping algorithms. The main methods used for eigensolution methods are sub-space iteration, the Lanczos method, and component mode synthesis.

PARPACK [3] is a parallel package for large eigenvalue problems. Wu and Simon [4] implemented a parallel Lanczos method for the symmetric generalised eigenvalue problem, and used MPI. Guarracino et al. [5] used a block Lanczos algorithm to solve eigenproblems on multiple computers. Honglin et al. [6] have used a parallel implementation of the sub-space iteration method, and applied it to non-linear problems.

Cross [7] and Aoyama and Yagawa [8] both used parallel implementations of the component mode synthesis method. They reported near ideal speed-up on massively parallel computers. However, the work was based on one dimensional splitting of the structure into sub-domains.

Li et al. [9] described the uses of supercomputers and Nastran and Patran, and use IRAM (implicit restarted Arnoldi method) for symmetric eigenproblems, and achieved up to 75% speed-up efficiency.

Most of the work has used MPI or other parallel methods. There is very little on the object or component oriented implementation, the work of Heng and Mackie [10] being an exception to this. A more general consideration of the use of MPI, Java and C# in parallel and distributed computing can be found in Mackie [11]. It should be noted that another area that is receiving considerable attention is the use of Graphics Processing Units (GPUs) for general purpose computing, seeking to take advantage of the fact that GPUs have many cores and are high-performance [12].

Manolis et al. [13] uses sensitivity and stochastic modelling to help develop retrofit strategies for structures under seismic loading. Such work requires many analyses. Dere and Sotelino [14] also commented on the need for multiple analyses for establishing response spectra in non-linear problems. They implemented a parallel sub-domain solution approach using a group-implicit algorithm. Fu [15] also used a sub-domain approach, but with an overlapping domain algorithm,

3. Object oriented program design

The work in this paper will be described within the context of seismic engineering and space structures, but the work described herein is not limited to this problem area. Rather, it is used as a vehicle for demonstrating various program design principles and methods.

As noted in the introduction, there are various ways in which programs can take advantage of parallelism and distributed processing. The most obvious, and probably the most common, is the application of parallelism to individual problems. If multiple problems need to be solved, then these too can be done in parallel. Within the context of seismic engineering, earthquakes are by their very nature stochastic, and the precise earthquake a structure may have to endure is not known. So it can be useful to subject a structure to a variety of earthquakes. In addition, there is the general design of the program. The presence of multiple processors means that the program can do several tasks at once, so often it is not necessary for the program to stop completely while doing some tasks. This can help with the overall usability and flow of the program. The software described in thus paper was written using C# and the .NET environment, version 4. C# is object and component oriented, and version 4 of .NET has introduced the Task Parallel Library (TPL).

3.1. Parallelising of individual algorithms

The algorithms involved in seismic analysis are: (i) determination of the modal frequencies and mode shapes; (ii) modal superposition; (iii) transient analysis.

Determination of the vibration modes by the subspace iteration method was examined in [16]. The algorithm itself can be parallelised. Further parallelisation can be achieved by using domain decomposition. The paper described the use of a design pattern which has also been used in the design of software for iterative solvers [17]. The work described in [18] has been modified to use the TPL, but the overall design remains the same. [10] described the implementation of component mode synthesis. Work by others has parallelised the Lanczos algorithm.

The results of the modal analysis can be used in the mode superposition method, though naturally this applies only to linear problems.

Transient analysis can be applied to linear and non-linear problems. Since the focus of the current work is on software design aspects, the work in the current paper is limited to the linear case. The Hilber–Hughes–Taylor algorithm is used, and the design pattern used for eigensolution and iterative solvers [16,17] is adopted.

The key feature of the design pattern is the separation of the algorithm from the data. The algorithm for a particular problem remains the same, but it may be implemented for many different data structures. For instance the standard solution would be the use of a single domain, but the algorithm can also be implemented for the domain decomposition case. Furthermore, the data may be stored locally or remotely, or a mixture of the two. Even then, for each of these cases there are a multitude of sparse data storage schemes that can be used. However, despite all these variations,