PARACOMP

IT Innovation (formerly PAC)

FEA Ltd

Messier-Dowty

Public Final Report of ESPIRT HPCN PST Activity PARACOMP, Parallel Analysis of Composites.

PAC/PARACOMP /D0.1 Version 2

Tim Cooper, Paul Gordon, IT Innovation Centre
Dave Irving, FEA Ltd.
Richard Newley, Messier-Dowty

8 November 1999

Introduction

This is the revised version of the public final report of the PARACOMP project. It has been revised following the reviewers recommendations from the final PARACOMP review held on 23 June 1999. The following changes have been made:

i) Addition of this introduction;

ii) change of contact details in the synopsis and at the end;

iii) modification to the Introduction section in the Conclusions;

iv) additional paragraph in the 'Resource management on NT' section of the Conclusions;

v) addition of 'business model and business benefits' conclusions.

(Note that this report was originally written when the IT Innovation Centre was called the Parallel Applications Centre, so it contains many references to PAC).

Abstract

The PARACOMP (Parallel Analysis of Composite Materials) project intended to show how a parallel finite element code could be deployed on low cost hardware in such a way as to be transparent to the user and thereby innovate the deployment of numerical simulation at an industrial end user.

Synopsis

This report is the final report of the PARACOMP Project (ESPRIT). The project involves the following partners: PAC (UK), FEA Ltd(UK), and Messier-Dowty(UK). The project was co-ordinated by PAC, and further information may be obtained from

Paul Gordon,
Tel: +44 (0)23 8076 0834,
Fax: +44 (0)23 8076 0833
mailto: plg@it-innovation.soton.ac.uk

Virtual prototyping, through the use of numerical simulation, offers a way for companies to reduce the time to market for new components. However numerical simulation software is not always easy to use and requires staff with a wide range of skills. For many organisations this is costly, and when you add to that the cost of hardware, and the fact that many simulations take hours to run it is not always clear that there is justification in investing in numerical simulation. The PARACOMP project has demonstrated how a parallel simulation code can be deployed to reduce this cost.

Messier Dowty is a world leader in the field of landing gear manufacture. They are investigating the use of composites to reduce the weight of their product. Physical testing is obviously important in such a programme of development, but simulation still has a part to play in identifying where components are likely to fail. This enables the physical testing to focus on important areas, making the process more efficient. Messier-Dowty has a cluster of NT workstations used primarily for CAD. These machines are idle overnight, and Messier-Dowty could gain extra computing power for free if they could find a managed way of running jobs on these machines.

The types of model in which Messier-Dowty are interested range from small components, which take up to a day to run through to full landing gear models which can take up to a week. To reduce the analysis time for a component down to a few hours when run on a cluster of workstations would allow jobs to be run overnight. For large models that take a week on one machine, a reduction in wall-clock time to a few days when run on a number of machines is of no benefit because it prevents engineers from working on the other machines. However if execution can be limited to run only overnight, then there is a benefit as it leaves all machines in the cluster free to run problems.

Messier-Dowty are therefore representative of many of the companies in the aerospace sector in that

1. They work with high tech materials.
2. They invest in staff to run numerical simulations
3. They cannot afford to invest in dedicated hardware to run the problems
4. They have spare computing capacity overnight when their desktop machines are not being used.

Executive Summary

The use of parallel simulation codes has been widely demonstrated in industry. However their take up by smaller organisations has often been limited by the cost of the hardware needed to run the codes. In the PARACOMP project we set out to show how a finite element code could be fully ported to run in parallel on low cost hardware, namely a cluster of NT workstations. The aim of the project was not just to show that the port is technically possible, but also to investigate the business models by which the parallel code could be used. One other aim of the project was to develop a code that could be maintained at minimal extra cost to the code owners.

The approach adopted in the project to meet these aims was to modify existing functionality of the LUSAS software to solve problems in parallel. The problem is broken down into smaller parts called super-elements and each super-element is solved on a separate machine. A global solution phase is then carried out on just one machine. The execution is co-ordinated by Intrepid, a parallel scheduler developed by PAC. Using super-elements does not deliver the optimum performance, in terms of speed up, particularly for a large number of processors. However it does have a number of advantages which outweigh these problems, and on balance the method was felt to meet the needs of FEA.

Virtual prototyping, through the use of numerical simulation, offers a way for companies to reduce the time to market for new components. However numerical simulation software is not always easy to use and requires staff with a wide range of skills. For many organisations this is costly, and when you add to that the cost of hardware, and the fact that many simulations take hours to run it is not always clear that there is justification in investing in numerical simulation. The PARACOMP project has demonstrated how a parallel simulation code can be deployed to reduce this cost.

Messier Dowty is a world leader in the field of landing gear manufacture. They are investigating the use of composites to reduce the weight of their product. Physical testing is obviously important in such a programme of development, but simulation still has a part to play in identifying where components are likely to fail. This enables the physical testing to focus on important areas, making the process more efficient. Messier-Dowty has a cluster of NT workstations used primarily for CAD. These machines are idle overnight, and Messier-Dowty could gain extra computing power for free if they could find a managed way of running jobs on these machines.

The types of model Messier-Dowty are interested in range from small components, which take up to a day to run through to full landing gear models which can take up to a week. To reduce the analysis time for a component down to a few hours when run on a cluster of workstations would allow jobs to be run overnight. For large models that take a week on one machine, a reduction in wall-clock time to a few days when run on a number of machines is of no benefit because it prevents engineers from working on the other machines. However if execution can be limited to run only overnight, then there is a benefit as it leaves all machines in the cluster free to run problems.

Messier-Dowty are therefore representative of many of the companies in the aerospace sector in that

1. They work with high tech materials.
2. They invest in staff to run numerical simulations
3. They cannot afford to invest in dedicated hardware to run the problems
4. They have spare computing capacity overnight when their desktop machines are not being used.

Contents

1 State of the Art

1.1 The numerical analysis of composites

The techniques for performing the numerical analysis of composites are still being developed. Composite materials are made up of layers of fibre held together by resin. Within any one layer the fibres are parallel, but different layers have fibres of different orientation. For a model such as Messier-Dowty's, the code must deal with

1. Non-linear geometrical behaviour of the whole piece
2. Non-linear behaviour within a layer
3. Delamination (where layers of fibre separate)

At the start of the project, the code from FEA was able to model each of these individually, but not all in one problem. This is obviously a problem to the engineers at Messier-Dowty, who need to be able to consider the effects of all three types of behaviour at the same time.

Messier-Dowty use FE analysis to model the behaviour of a whole range of problems, from small components to full bodies. This modelling work is carried out to add value to the physical testing that they do. The numerical models can provide information about where failure of a piece will start and how it is likely to propagate through the component. However composite FE analysis takes a long time. The development of valid models is a lengthy process, and even when finalised, the iterative nature of the modelling means that the execution of a problem can take anything from days to weeks.

At present Messier-Dowty carry out their modelling work on dedicated NT workstations.

1.2 Parallel FE on NT clusters

At the start of the project, many FE solvers were available on NT machines. However parallel codes were almost exclusively available only on Unix machines. The use of NT machines for running parallel jobs was seen as not being feasible for the following reasons

- Slow interconnect speeds on distributed clusters. This is a problem that applies equally to clusters of Unix workstations.
- Lack of supported libraries to enable parallel computing.
- Market inertia still favoured Unix as the default operating system for scientific work

Users of numerical analysis fall into three basic categories;

- Large companies who have teams of experts specialising in different techniques
- Small to medium size companies who have small teams of one or two people expert in techniques core to the company business.
- Consultancy houses which have expertise in a number of different techniques

The computing needs of these organisations vary considerably. Large investors typically use large parallel machines for the computationally intensive analysis, with desktop resources used for pre- and post- processing. The parallel resource is controlled by some kind of resource management solution. Small to medium size companies usually have a cluster of CAD workstations. The simulation experts run their jobs either on a similar machine (with perhaps an extra processor and extra disk) or on a low cost SMP platform. The cluster is sometimes used over-night and in that case the local owners of the machines manage how jobs are run. Consultancy houses are more like the small to medium size companies. However they are likely to have a smaller pool of CAD stations available for use overnight.

Messier-Dowty falls into the middle category. They wish to be able to use spare resource, but at the same time keep the machines in the cluster firmly under the control of the engineers who use them day-to-day. For Messier-Dowty, then, the issue is one of being able to guarantee that jobs submitted to the cluster are only executed overnight and that the user is able to continue using the machine during the day.

1.3 Computational Resource Management tools for NT

Computational resource management tools (which can manage the execution of heterogeneous loads on heterogeneous machines, and at the same time be able to manage data transfers) for NT were not available at the start of the project. Indeed for a company such as Messier-Dowty, the cost of investing in a resource management tool just to run parallel FE analysis was too high.

The PAC has a resource management tool, Intrepid, which has been developed for Unix machines (IRIX, AIX, Solaris and HP-UX). The tool was developed in three ESPRIT projects (HPC-VAO, TOOLSHED and PROMENVIR) to manage heterogeneous work on shared resources. At the start of the project, it had been demonstrated in the Promenvir project to manage a WAN experiment of 1000 tasks on 100 CPUs at 6 sites.

One of the issues in going from a single machine (where the engineer who runs he analysis owns the machine) to a clustered solution is that large jobs are being run on machines which the engineer does not necessarily own. In order for Messier-Dowty to be able to manage this, it was felt appropriate to incorporate a resource management tool into the parallel code. Therefore it was decided to port Intrepid to NT and integrate it with the LUSAS solver.

1.4 The maintenance of parallel simulation codes

One of the hindrances to the take up of parallel simulation is that parallel codes can be hard to maintain. A parallel code typically requires at least one developer full time to maintain, and for a small software house such as FEA this can be hard to justify. One of the aims of the PARACOMP project was to demonstrate a parallelisation technique that would be cheaper to maintain than standard parallel codes, by building it on existing functionality.

2. Approach taken

The approach taken in the project was to build the parallelism using the existing super-element functionality of the LUSAS code. This functionality allows a part of a model (the super-element) to be analysed independently from the rest of the model. The results at each node within the super-element are expressed in terms of the values on the interface. Super-elements have a number of different uses (for instance, if a large part of the model does not change, they allow the static part of the problem to be represented by a super-element). For this reason they were attractive to use as the they are already maintained, thereby reducing the cost of maintaining the parallel code. Once the solution for the super-element is completed, the whole solution is found in a global operation.

The process of performing a super-element analysis is performed as follows

1. The FE matrix is calculated
2. The matrix is reduced to a near-diagonal form where each value is stored in terms of the values on the interface. During this process, the reduced parts of the matrix are written to disk.
3. The global operation is performed, whereby the reduced matrices are assembled and then reduced to calculate the values on the interfaces.
4. Finally, the solution over the whole model is calculated.

The LUSAS code supports super-elements for a number of problem types, but not, at the start of the project, for composite problems. Extending it to cover non-linear problems was not a trivial task. For non-linear problems, analysis needs to iterate on two loops. The outer loop is the loading loop. Because of the non-linear geometry of the problem, the loading has to be applied in small incremental steps. Not only is the geometrical response non-linear, but the physical formulation of the problem is also non-linear. For this reason the inner loop is the iterative, proceeding until the residual energy is reduced to zero.

Super-elements intuitively lend themselves to parallel execution. The approach taken in the project was therefore

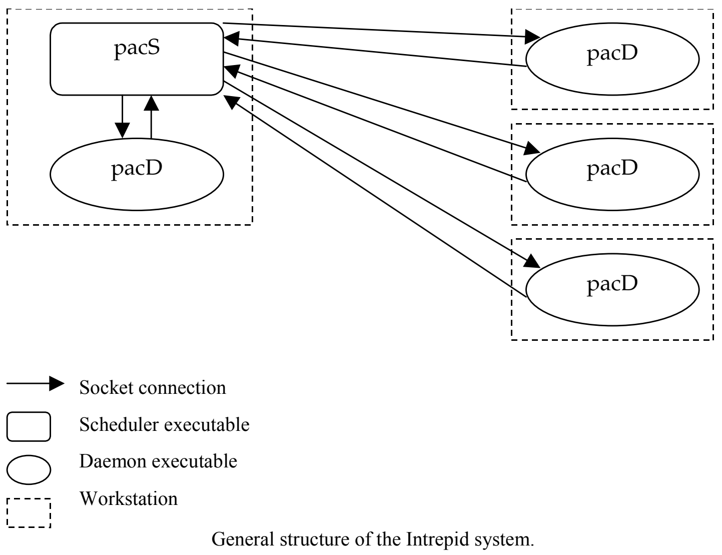
1. To develop non-linear super-elements for the LUSAS code.
2. To develop a tool for co-ordinating the execution of super-elements (including the transfer of data)
3. To integrate these two tools
4. To develop a tool for creating super-elements which will be evenly matched to the resources available.

2.1 The development of non-linear super-elements

2.2 The port of Intrepid to NT

The Intrepid parallel scheduler is a tool that allows the scheduling, control and execution of a number of tasks (programs) on a heterogeneous network of workstations. It allows the user to control both the resources used by the scheduler to run programs on, as well as the programs running themselves.

The general structure of the Intrepid parallel scheduler is shown in **Figure 1**.



General structure of the Intrepid system.

Figure 1

As can be seen from the figure, the Intrepid parallel scheduler consists of two main components:

pacS: This is the main scheduler executable and the program that contains the global network and schedule information for the system. This module takes all the scheduling decisions in Intrepid. There is only one instance of pacS running in a functioning Intrepid system.

pacD: This is the daemon that provides support to the Intrepid parallel scheduler. There is one of these daemons in each "node" in the network. An Intrepid daemon's main function is to pass back information to pacS about the state of the "node" which it is controlling, and to launch and monitor applications. Secondary functionality includes all the necessary actions to start an application (e.g. creation of temporary directories if required, transfer of all input files before and output files after the execution, etc.).

Communications between pacS and the various pacD daemons are implemented via TCP/IP sockets. This allows Intrepid to control heterogeneous networks. The user of Intrepid submits a “schedule” to the system. A schedule consists of a number of tasks (processes) to be executed and the dependencies between them. The Intrepid system then assigns tasks to specific processors in the network, so that the overall time of execution for the whole schedule is minimal. The system also performs any data transfers that might be required to allow a task to run on a specific machine. Any number of users can submit any number of schedules to Intrepid simultaneously and the system is able to deal with the added complexity.

The Intrepid system was originally implemented on UNIX systems. The dependencies of the source code on libraries and development tools were kept minimal (even in the UNIX world there are a large number of different flavours of UNIX and Intrepid was designed to be portable). Intrepid relied on the following external dependencies:

- C++ compiler.
- Socket library. Win32 supports TCP-IP sockets.
- Environment variables. Although NT support environment variables, a more elegant way is to use the NT registry.

- rshd. NT does not support 'rshd', but does provide NT services which run continuously. Intrepid launches daemons using an 'rsh' command. These daemons were replaced with equivalent NT services.
- lex/yacc.
- Data transfers using 'cp'. NT does not support 'cp' copy commands. All data transfer had to be carried out using drive mappings.

2.3 Interfacing Intrepid with Lusas

Lusas consists of a number of components. The two main ones are the solver itself (Lusas solver) which is the legacy Fortran code for the finite element analysis, and the GUI front end (Lusas modeller) which is a Windows32 application and is used to design the model and present results.

The usual mode of operation for Lusas is that a user will design a model using the modeller, use the solver on the created model and finally display the results on the modeller window. To start the solver on a model all a user has to do is click a button.

It was important to keep the same style of operations in PARACOMP. However, there are a number of steps involved in generating the parallel solution that have to be hidden from the user. When a model is ready to submit to the parallel solver, the first thing that PARACOMP must do is split the model into the appropriate number of sub-domains in order to perform the finite element analysis in parallel. Appropriate data files have to be generated for each of the sub-domains of the model. An input file to Intrepid has to be generated (the task graph file) that contains information about the dependencies between the composite parts of the finite element analysis. For each of the sub-domains a different process running the parallel solver will have to be spawned on the target machine. Before the process is started the necessary data files that represent the domain have to be copied over to a local directory on the target machine. When the finite element analysis run is finished the result files have to be copied back to the machine from which the user submitted the job. All this is handled by the Intrepid system and a small Visual Basic script that performs the domain decomposition and essentially "glues" Lusas and Intrepid together. All the user input required, is the number of sub-domains that the model must be split into.

There is also a need for another GUI program, which configures Intrepid for the specific network. Because of time limitations and because PARACOMP only addressed networks of Windows NT workstations, the software that performs this function does not have the full functionality of the UNIX GUI for setting up Intrepid. It assumes that the network only contains NT machines and the options that the user can set at configuration time are comparatively limited.

Finally, LUSAS is an iterative solver. In order to save time the super-element processes keep all their data in memory between iterations. This means that processes never terminate, and the resource manager has no way of knowing when to start the next process. In order to deal with this, Intrepid was modified to open a socket to LUSAS over which co-ordination messages could be relayed.

2.4 Load Balancing

The time taken to solve a parallel problem using super-elements is given by the equation

$$T_{sol} = \max(T_{sol,i}\ ; i = 1, ND) + T_{comm}$$

where

$T_{sol,i}$ = time to solve domain i,

ND = number of domains,

$T_{comm} = k \times N_{\text{interface}}$,

k = constant.

$N_{\text{interface}}$ = number of nodes on interface,

The first term ($\max(T_{sol,i}; i = 1, ND)$) is minimised by ensuring that each domain solves in the same time, i.e. is computationally load balanced. However this does not guarantee that the sum, T$_{sol}$, is minimised since time to eliminate interface elements, T$_{comm}$, may be excessive. Introducing a load imbalance may thus be beneficial if it causes T$_{comm}$ to reduce significantly.

The solution adopted in this project was to build super-elements using the recursive Mallone method. A visual basic script was written using the LUSAS API that created super-elements according the following algorithm

1 Order elements to ensure that the front width is minimised

2 Add elements to the super-element in order from this list until the required size is reached.

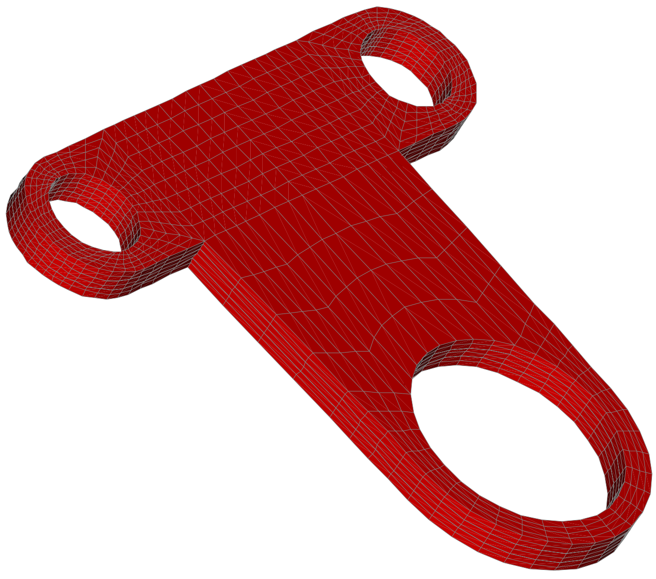
3 Start a new super-element

4 Repeat until all elements are used up

By selecting target sizes for the super-elements that are matched to the resources available it should be possible using this technique to create super-elements that will execute in the same time, even on a heterogeneous cluster. One obvious drawback to this method is that each time the resources available change, the user will have to create a new set of super-elements. The ease with which super-elements can be created will determine how much of a problem this is.

2.5 The Messier-Dowty test case

The original Messier-Dowty test case was a lug that forms part of the landing gear. The aim of the test case was to verify that the LUSAS code would predict the onset of delamination correctly. In addition to this, a test case based on a simple beam was also developed.



3 Results, Achievements and Benefits

3.1 Load balancing

In the event, the load balancing part of the code proved to be the hardest to implement. This was not because of any fundamental failing in the algorithm, but because of functionality in the LUSAS modeller used to create the super-elements. The LUSAS modeller builds FE models out of geometrical entities (lines, surfaces, volumes etc). It uses these to create the mesh that will actually be used to solve the problem. Any load-balancing algorithm must work with the connectivity of the mesh, as it is this that which determines the numerical work to be done. Without information on the connectivity, incorrect assumptions about performance will be made. Unfortunately the way LUSAS modeller creates super-elements is to mesh the entire model, then build up the super-element out of geometrical entities, inheriting the existing mesh. This means that the load-balancing algorithm is constrained when building the super-element, in that once an element is selected all the other elements associated with that geometry must be selected.

Changing this would require more effort than was available in the project, and so it was agreed that the test case would have to be built using small geometrical entities. This constraint is potentially serious, as it means that old designs may not run well in parallel, but the work around is practicable.

3.2 Performance of the parallel code.

3.2.1 The controls and goals

Because of the problems with the load balancing, a single model was created decomposed into four super-elements. The idea was to experiment with the distribution of the five parts of the problem (four super-elements and one combination phase) onto a variety of hardware, and the effect these different distributions would have on the overall timing of the calculation. This was felt to be acceptable by Messier-Dowty.

The parameters we were interested in investigating were

- Performance on a range of machines, including
  - The speedup as the number of processors increased. This allows us to assess the number of processors to be used to guarantee a result in a given time.
  - The affect on the speedup of choosing processors from single and multi-processor machines in a cluster.
  - The affect on other code parameters (memory and disk) of running in parallel.
  - The use of heterogeneous machines in a cluster
- The use of machines when execution is suspended.
- The ease of use of the parallel code

3.2.2 Performance of the code

The results for 10 iterations of the parallel solver can be seen in Table 1. The first line shows the result of running the test case through the sequential version of the Lusas solver. The second line shows the same problem running on one computer, but using the PARACOMP executables (ie running the 4 super-elements serially and the recombining the results). The run is more than twice as slow as the sequential run, as one would expect with the added synchronisation phases that are imposed on the calculation.

# of machines	# of CPUs / machine	Memory used / machine	Duration
1	Sequential Lusas used	~45MB	3:12:34
1	1	~213MB	7:01:28
2	1	~132MB	1:12:58
2	2	~132MB	1:58:42
3	1	~132MB	1:02:34
4	1	~86MB	0:52:10

Table 1. Results for 10 iterations of the solver.

From these tables it can be seen that using 4 processors the speed up is marginally less than 4 (3.7). This means that a job, which would have taken a morning, is now complete in less than an hour. If we scale this further (by allowing the solver to run for a full 100 iterations) then we see that a task which previously took a whole day can be run overnight.

An important result concerns the usability of the code in a heterogeneous environment with machines used for other tasks. The tests described were carried out on dual processor machines at the PAC offices. In the 4-machine case (where there is one process per machine) the users of the other machines did not know that their machine was being used for simulation, except by accessing the task manager. Even CPU intensive operations, such as compilation and image manipulation, appeared to be unaffected.

The figures shown in the third column in the tables (namely the "memory used per machine" column) contains the approximate maximum virtual memory usage for any of the machines involved in the run taking into account only the processes involved in the test. It seems that the memory usage of the parallel Lusas solver is approximately 41MB irrespective of the size of the input problem. The reason for this is most likely that the memory requirement of LUSAS follows a stepped profile, and all of the cases tested were within the same band. Memory usage figures also contain the memory used by the Intrepid executables as well, but they impose comparatively minute virtual memory requirements. The figures represent the largest virtual memory usage of the machines involved in the run. For example, in the case where four machines have been used, one of the machines will be running both a super-element parallel Lusas executable and the recombination phase parallel Lusas executable, while the other machines will be running only a super-element calculation each. The 86Mbyte figure given in the table corresponds to the machine that is most loaded in terms of virtual memory.

The Intrepid parallel scheduler also allows administrators to define in advance “unavailability periods” for specific machines. During these periods, Intrepid does not launch any new tasks on the computers that are controlled by it. This would effectively release the CPU for other processes (like a user logging in and checking the e-mail) for the duration of that period. This was one of the goals of the PARACOMP project that the end-user (Messier-Dowty) desired.

The “unavailability periods” were tested and found to work. When a computer enters an unavailability period the processes of an ongoing test through PARACOMP are still in the pool of running processes. They are, however, blocked waiting for receipt of a socket message and thus consume no CPU time until the unavailability period finishes. They do consume memory resources though, which can potentially make the computer appear very slow to whoever is using it at the time. It is up to FEA to decide whether they wish to enhance their code to release memory when it enters an unavailable period of execution.

3.2.3 Ease of use

Transparency to the users was also an issue. The aim was to develop a code whereby the user defines a problem, the machines on which to run it and everything else is taken care of. In principle this is what happens. There is a once only installation, whereby the administrator needs to define the workstations that are to be controlled by Intrepid. During the installation phase the system administrator can also determine the “unavailability periods” for each machine, in order to limit the usage of the computing resources at certain times during the day. The user then simply has to start the Visual Basic script that initiates the parallel solution of the currently loaded model and then select the number of super-elements that he or she wishes to split the model into. However in practice PARACOMP partially failed to deliver this for two main reasons. Firstly, there is no way of automatically generating the super-element input files from the Visual Basic script (for the test case the super-element input files were supplied manually altered by FEA for use with the parallel scheduler). The format of the input deck has to be manually changed. Secondly, super-elements are built up from geometrical entities, making load balancing a problem.

3.3 User assessment

The ability to solve large complex composite models is a priority for Messier-Dowty because accurate predictions of behaviour can significantly reduce the cost of ‘manufacture and test’ programmes to optimise the design of composite parts. Accurate predictions give confidence in the technology of structural composites and thus aid the acceptance of composite components on large landing gear of civil aircraft and their application in other critical aircraft structures.

The successful demonstration of the parallel code offers the potential both to reduce the execution time of problems and make better use of the hardware. In the immediate future we shall make use of the recently introduced sparse solvers in the LUSAS code (see next section). However we are always looking to run larger problems and the ability to distribute problem size across the network that has been demonstrated in PARACOMP project will be a useful feature when it is implemented in the standard LUSAS code. When the parallel version is released, the extra time savings and the extra processing flexibility will produce further benefits.

4 Developments within the lifetime of the project

The parallel LUSAS code developed in the PARACOMP project delivers a number of benefits to the user. These include a distribution of the memory and disk requirements of the code and transparent execution, but the principle benefit to the user is speed. There are three stages to the execution: preparing the super-element matrices, solving the super-elements and solving the global problem. The PARACOMP project has reduced the execution time in the first and last phases so as to reduce the overall execution time by a factor of 3.7 on 4 machines. The execution time in the 'solve' phase is reduced by only a small amount. The reason for this is that by splitting the model the front width of each submodel is actually is increased. Since the solution time is proportional to the square of the front width time the number of equations the solution time of the submodels remains about the same. In fact if the model is not split at the optimal positions the solution time may increase. Now the current solver does not help in this respect as the frontwidth optimisers do not account for retained freedoms. The new multi fontal solver does however and therefore it makes sense to concenrate effort on implementing this for parallel solutions.

During the lifetime of the project, FEA have implemented core sparse matrix solver routines developed by a third-party. This solver can run up to 10 times faster on a single machine for very large problems by speeding up the solution of equations, an area the PARACOMP project has not addressed. Unfortunately the sparse matrix solver does not currently support super-elements. This is however currently being implemented.

Given the benefits the sparse matrix solver will provide over the parallel code FEA have decided to focus their efforts in delivering the sparse matrix code to their customers first. This development does not mean that FEA will not exploit the functionality developed in the PARACOMP project. Indeed the PARACOMP work is complementary with the sparse matrix solver speed up and FEA remain committed to creating a commercial parallel code. The time scales have however moved since the start of the project.

Figures 3 and 4 below show a comparison of four versions of LUSAS; the current serial version, the current parallel version, the new solvers and the parallel version with the new solvers. The first three are based on measured values, but the last is predicted. There are three phases to the solution; pre-processing, solution and post-processing. Figure 3 shows values for a standard, linear elasticity problem and figure 4 shows figures for a non-linear dcb analysis. From these figures we see that for the non-linear problem the parallel version gives a speed up overall, but this is roughly the same improvement as the improvement the new solvers give on a single processor. It is estimated that by combining the two may give a speed up of between 5 and 7. For linear problems the parallel version gives a poor speed up, whereas the new solvers can give a speed up of up to 10. Combining the two may give a speed up of 15 to 20 on four processors.

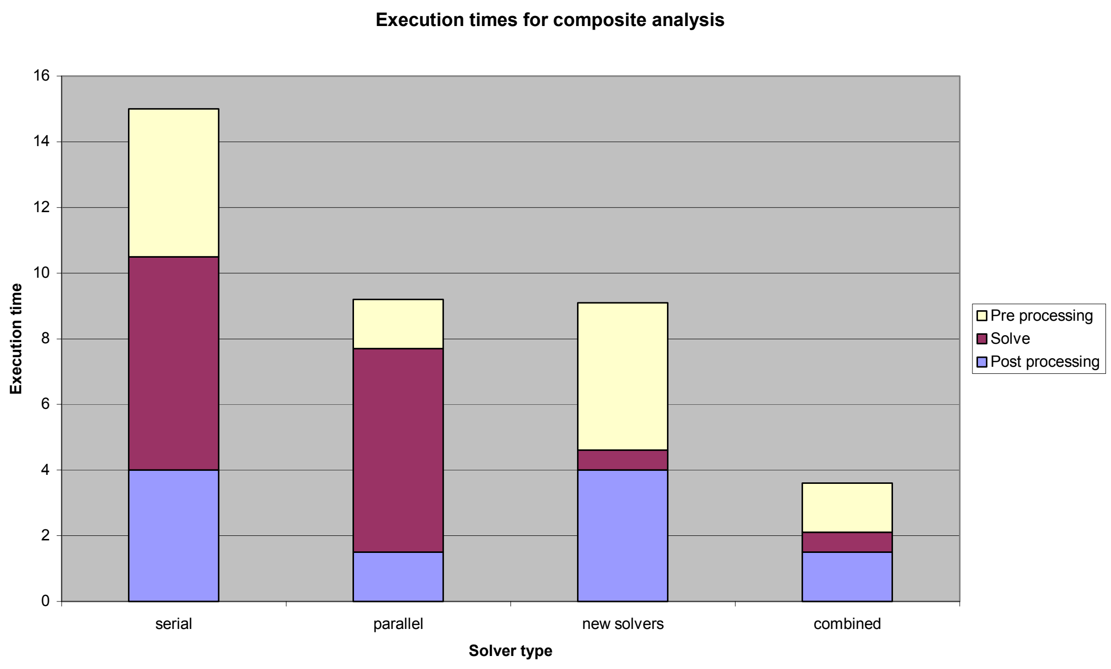


Figure 3

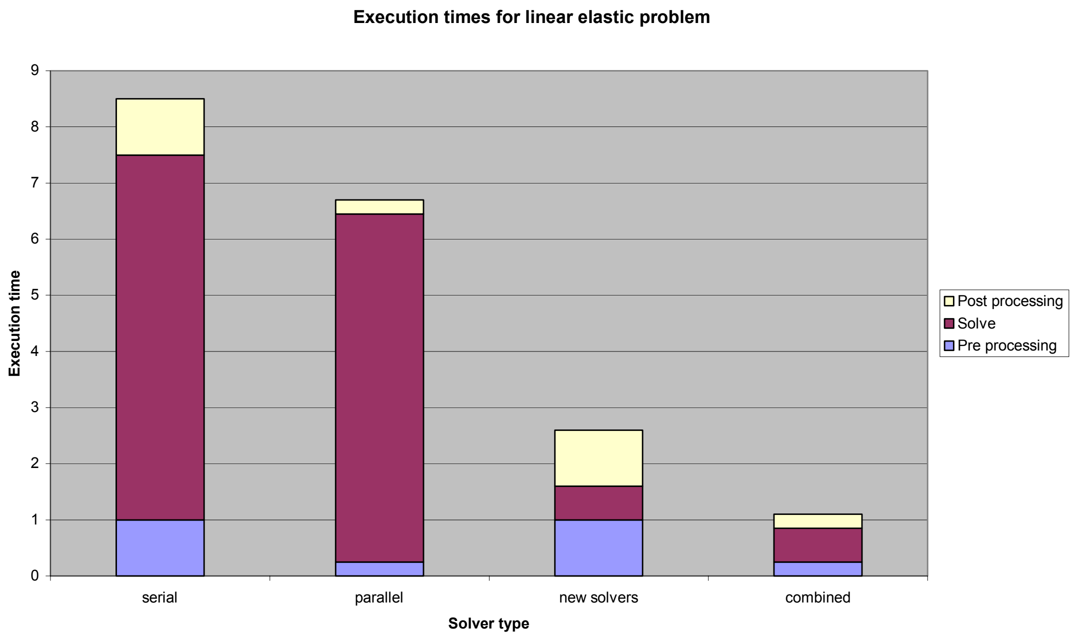


Figure 4

This analysis shows two things. Firstly, the new solvers outperform the parallel version in almost all situations. For this reason, FEA must deliver them to market first. However it is also clear that a parallel version using the new solvers will give still further benefits. The only thing preventing FEA from developing a parallel version with the new solvers is that they do not currently support super-elements. As soon as this functionality becomes available, FEA will invest to bring a parallel version to market.

5 Dissemination

Dissemination of material from the PARACOMP project has focussed on two distinct audiences. In the first audience are software houses. An important result of the project is that a functionality which many FE code vendors maintain can be used to support parallel execution. This reduces the cost of maintaining the code, making it attractive offering to software houses. The second group is industrial end-users of numerical simulation, particularly small to medium sized companies.

5.1 Technical Papers

The following papers have been submitted and accepted at conferences

'Case Studies of four industrial Meta-applications', Cooper T, HPCN Europe 99

'Porting legacy applications to NT', Cooper T, Allsopp N, Ftakas P, Usenix 99

Both of these papers dealt with the technical issues of developing and maintaining parallel codes on NT machines. They were therefore aimed at software houses.

5.2 Dissemination Events

NAFEMS A conference was organised with NAFEMS on the theme 'Changing the way we use simulation'. The conference disseminated the results from a number of PST activities including PARACOMP. The conference was aimed at those within industry whose expertise is numerical simulation. These range from engineers to managers in industry.

5.3 Newsletter

A newsletter on the parallel LUSAS code has been prepared. It will be included in the next edition of LUSAS News, and a flier based on it will be made available to the TTN for distribution.

5.4 Exploitation

In order for FEA to exploit the parallel version, an agreement has been reached between PAC and FEA whereby the sources to the code have been delivered to FEA to support and develop.

6 Conclusions

The PARACOMP project set out to achieve three things:

- to show how a finite element code could be fully ported to run in parallel on low cost hardware, namely a cluster of NT workstations;
- to develop a code that could be maintained at minimal extra cost to the code owners;
- to investigate the business models by which the parallel code could be used

The conclusions can be broken down into these three areas, plus some general comments.

6.1 Introduction

The LUSAS code has been ported from what, at the start of the project, was a serial code available on Unix and NT to a parallel code on NT. It has been integrated with the Intrepid scheduler from PAC and has been demonstrated on a number of different clusters of NT machines. The code has reduced the execution time for a parallel composites analysis down from nearly 4 hours to just under 1 hour on 4 machines. Scripts and user interfaces have been developed which allow the user to create super-elements that, subject to the constraints of the LUSAS modeller, are load balanced with respect to the resources available. A set of scripts has been written which allow the user to install Intrepid onto the NT network and to select the machines on which to solve the problem. The code has been tested by the end user with encouraging feedback.

There were a number of unforeseen problems that have hindered the development in the project. The fundamental way in which LUSAS deals with super-elements turned out to have a large effect on how effectively the problem could be load-balanced. In addition the integration of in-core sparse solvers which showed similar, if not better speed-ups means that the parallel code will not now be productised until next year.

6.2 Parallel FE on NT - low-cost parallel finite element

The code has been demonstrated to give real speed ups on an NT cluster. Because of the way in which data is handled, the transfer of data files across the network has been minimised, an important result for users of the code.

Composite analysis is traditionally I/O intensive. Because each machine in the network has its own disk, the I/O is distributed across the network and this gives speed-ups in its own right. There is a further consequence of this. The limiting factor which determines the size of the problem when running a composite analysis in LUSAS is the scratch space required to write the intermediate files. Because these files are distributed across the network, parallel LUSAS is able to process much larger jobs than was previously the case. For Messier-Dowty this is an important result, as they have a long term desire to be able to model full landing gear.

6.3 Resource Management on NT - code maintenance and usability

As NT becomes increasingly popular as a desk top platform, there is increasing pressure for simulation users to take advantage of this resource for running jobs. This means that robust resource managers must be developed for NT.

The Intrepid resource manager, originally developed on UNIX machines has been ported to NT and integrated with LUSAS. The port made use of the Win32 socket library to perform communications between the various modules and the Microsoft Foundation Class libraries to perform data transfers. Both have been proven to be reliable and efficient ways of controlling the network.

The absence of an 'rsh' daemon on NT has been shown not to be a hindrance to the development of resource management techniques on NT, and these results have been presented to the leading NT conference.

The use of standard components (LUSAS and Intrepid) to implement the parallel FE solution means that code ownership and maintenance is straightforward. Technical support for FE analysis is provide as standard by FEA, and this support applies to the parallel code. The use of simple scripts means that there should be no complex user support issues for Messier-Dowty to deal with in-house.

6.4 Business model and business benefits

Messier-Dowty is a medium-sized engineering company with a cluster of NT work stations used during the daytime for CAD work. They are typical of such engineering companies. This project has shown how the available computing resource may be used for running problems previously outside the scope of those that could be run.

The main perceived business benefits to such small- to medium-sized engineering companies from using parallel finite element analysis running on existing low-cost workstations are:

- Greater responsiveness to customer demands (the faster solution time enables Messier-Dowty to attack more complex problems more quickly);
- Increased customer confidence in a proposed engineering solution (by running simulations more quickly, Messier-Dowty can improve the accuracy of their simulations and better model component behaviour);

Both of these benefits will make Messier-Dowty more competitive in the aerospace component industry.

In addition, with the faster simulation capability, Messier-Dowty expect to be able to build components from composite materials with greater confidence. This has a tremendous knock-on effect for the aircraft manufacturer and airlines - composite components offer excellent weight saving over conventional components and this leads to lower aircraft running costs.

All of these benefits can be realised with very little investment in hardware, so the return on investment is potentially very rapid.

7 Exploitation

Key to the exploitation of this code is the ease of maintenance for FEA. This project has demonstrated how, by building the parallelisation using existing functionality, a parallel code can be developed which is cheaper to maintain than one which is developed using bespoke functionality.

In order for FEA to be able to offer the parallel code in the future, PAC and FEA have reached an agreement on the use of Intrepid.

8 Contact details

(formerly PAC)	Paul Gordon	http://www.it-innovation.soton.ac.uk	Coordinator, developer of Intrepid
	Dave Irving	http://www.lusas.com	LUSAS FE code
Messier Dowty	Richard Newley	Messier-Dowty, Cheltenham Road, Gloucester, UK	End user