Parallel and distributed crashworthiness simulation

G. Lonsdale	*C&C Research Laboratories, NEC Europe Ltd., St. Augustin, Germany*
J. Clinckemaillie	*ESI SA, Rungis, France*
S. Meliciani	*ESI SA, Rungis, France*
S. Vlachoutsis	*ESI SA, Rungis, France*
B. Elsner	*ESI GmbH, Eschborn, Germany*

Abstract

This paper focusses on recent developments and on-going research related to parallel and distributed approaches for crashworthiness simulation, using the PAM-CRASH code from ESI as an example. The developments described here relate directly to the take-up in the automotive industry of the latest High Performance Computing (HPC) technology, since crash and safety analysis is well established in the automotive design process.

Keywords: Parallel, crash simulation, PAM-CRASH, message-passing, HPF

1 Introduction

Over the last decade, crashworthiness simulation, and more recently simulation of occupant safety and the associated systems, has established itself within automotive design. Advances in both the modelling of physical phenomena and performance (arising from computer architecture advances and more efficient numerical algorithms) have led to an increase in the detail and complexity of models which can be handled, to the point where the position of numerical simulation has “moved-forward” in the design cycle. The numerical simulations are being used to influence the early designs of new automobiles, for example, rather than being used to make improvements to the design at a much later stage, when possibilities for modification are more restricted. With numerical simulation offering a more detailed analysis of the whole structure within a crash event, the experimental testing is often used only as a validation of the simulation. An overview of typical simulations, modelling aspects and computational requirements is given in [5]. An example of a simulation at the **current** high-end of computational requirements is shown in Figure 1.

Despite the advances made, both car manufacturers and their suppliers have an increasing demand for computational performance. One reason for this is the expansion in the detail of investigation being carried out: as an example, the occupant safety simulation is constantly increasing the detail of the dummy models being employed, with the expectation that certain biomechanic (human-like) parts will be included in the near future. Another reason comes from the introduction of more restrictive crashworthiness regulations, which results in the need for a higher *number* of different simulations to be performed. These demands have led to the development of message-passing versions of crash codes for parallel and distributed HPC architectures.

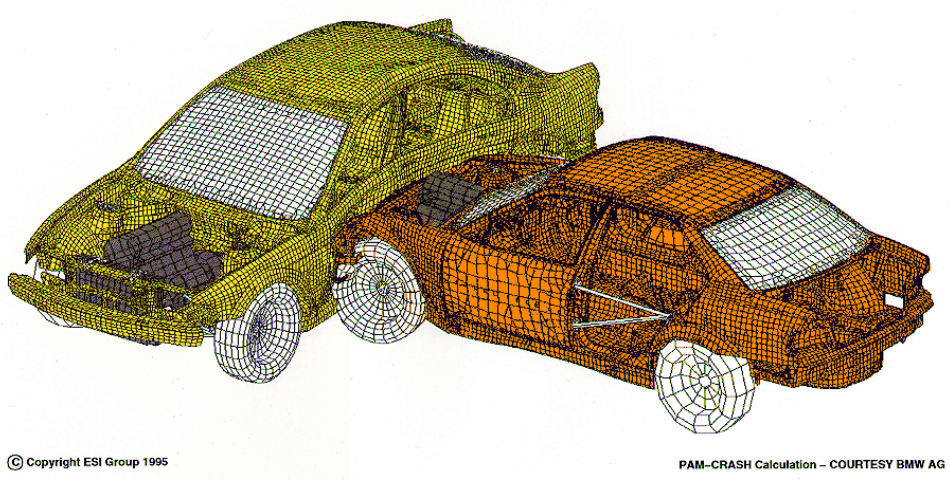

Figure 1: Car-to-car crash simulation *Courtesy of BMW*

Although shared-memory parallel implementations of the PAM-CRASH code have been in existence since 1987, the exploitation of distributed-memory parallel architectures has only recently been possible. In the ESPRIT project CAMAS and its extension to link to the EUROPORT action (see [12,13]), a prototype, portable message-passing version of the PAM-CRASH code was developed and benchmarked using industrially relevant models. The performance of this code version - even on a shared-memory architecture - has led to the acceptance of the, meanwhile matured, version as a viable 'production-use' design tool by automotive manufacturers who already have access to parallel HPC systems. Beyond this, current investigations are focussed on the efficient use of the message-passing version on networked workstations. Not only do networked workstations provide the “economic entry-point” to parallel computing for the smaller automotive suppliers, but they also allow the auto-manufacturers the possibility to both reduce the load on the dedicated parallel platform and to exploit a large computational resource which would otherwise remain idle outside office-hours - for example, the large number of CAD-stations otherwise used only for interactive, graphical work.

The central discussion of the paper is contained in Section 2, which gives an overview of the algorithms used within the PAM-CRASH code and the significant features of its message-passing, mesh-partitioned parallelization; in particular, describing the areas of difficulty and requirements for future developments. The degree of scalability achieved with the message-passing approach will be illustrated and a comparison with a shared-memory implementation made.

Although the message-passing approach does provide a high-performance solution, the complexity of the code implementation poses maintenance difficulties. A high-level approach would therefore be favoured, and the possibilities for using High Performance Fortran (HPF) are being investigated. Since the available constructs of HPF-1 (as defined in [7]) were adjudged inadequate for the requirements of irregular, unstructured industrial applications (see [1,2]), the ESPRIT project HPF+ ([9]) is investigating the extension of the language to enable efficient implementations of codes like PAM-CRASH. A brief overview of the project and the status of the PAM-CRASH related developments will be given in Section 3.

2 Message-passing PAM-CRASH

PAM-CRASH is an explicit time-marching Finite Element program used for the numerical simulation of the highly nonlinear, dynamic phenomena arising in short-duration contact-impact problems. It uses a central difference explicit time-marching scheme with unstructured meshes comprised of mechanical elements, of various types (e.g. thin-shells, beams, bars and special elements to model features such as rivets) and associated with a range of material properties, which model the behaviour of the structure under consideration. The Lagrangian formulation used has the following basic computational components: the time-integration at nodal points, "force calculations" on the elements defined by those nodal points. The time-integration calculates the accelerations, velocities and finally new-coordinates of the mesh based on the existing forces at the nodal points, created/generated by the movement of the points at the previous time-step. The major computational costs of the algorithm comes in the calculation of the forces at the nodal points. These force calculations can be broken down into (essentially[1]) stress-strain calculations and contact-impact calculations - the two having very different levels of data locality, when considering parallel implementations.

The stress-strain calculations are performed over the *elements*. The calculation on each element requires as input the latest co-ordinates and velocities from only those nodal points defining the element. Once calculated, the force on the element is distributed as individual forces at the nodal points. Since these calculations produce the largest contribution to the overall computational cost (between 60% and 80%, depending on the particular model) and exhibit a high degree of data locality, the message-passing parallelization employed an element-wise mesh-partitioning.

In contrast to the stress-strain calculations, the contact-impact algorithms used within the code have, in terms of data access, a pseudo-global nature. These contact algorithms serve to detect and correct penetration of structural components. This is achieved by first performing a proximity and penetration search, followed by a penetration correction procedure. Within the PAM-CRASH code, the latter is currently a penalty method, whereby contact forces are introduced at the impacting node and at the nodal points of the impacted segment. An implementation (or practical usage) issue which affects parallelization is that the contact calculations are performed only within user-defined (and not necessarily disjunct) areas.

In the following, we will give a brief overview of the message-passing parallelization. Further information can be found in the references:
In the papers [10,11], the parallelization of the PAM-CRASH code using static domain partitioning was detailed and, most importantly, the problems arising due to the inclusion of the contact-impact algorithms was discussed: the pseudo-global nature of the communications patterns, the static and dynamic load imbalance, the necessity to remove even small parts of non-scalable code. An analysis of load-balancing issues in the message-passing version and the performance bottle-necks occurring in the shared-memory or symmetric multiprocessing (SMP) version is including in the more recent publication [3].

As indicated above, a static mesh-partitioniong approach, based on a partitioning of *elements*, was employed. This static partition was calculated prior to the simulation. For the benchmarking results for EUROPORT, the multilevel recursive spectral bisection algorithm of the DDT tool ([4]) of the University of Southampton (produced in the ESPRIT project CAMAS) was used. The partitions employed a static partition balancing based on contact surface definitions (see [3]). With the unique assignment of elements to processes, the non-contact components of the solution

[1] A simplification is taken here, though the bulk of the computation **is** covered. A description of a range of special features, such as the handling of nodal constraint sets, rigid-body interactions or airbag calculations, is beyond the scope of this paper.

scheme (time-integration + stress-strain force calculations) can be efficiently parallelized by the provision of sub-domain interface communication. In order to enable contact search procedures to be performed locally, more complex communication constructs need to be introduced - based on the dynamically varying results of the search routines. Experience with the EUROPORT benchmark models has shown that load imbalance created by the contact-impact calculations is the major factor governing the scalability of the algorithms. Future research will thus be addressing the use of dynamic mesh re-partitioning, based on the changing computational load as the simulation progresses.

We conclude this section with an illustration of the performance of the message-passing PAM-CRASH protoype on a distributed-memory machine (the IBM SP2) and an SMP machine (the Silicon Graphics Power Challenge). Details of the EUROPORT benchmark models and further results can be found in [3,11]. The models themselves are illustrated, by their final deformed states, in Figure 2.

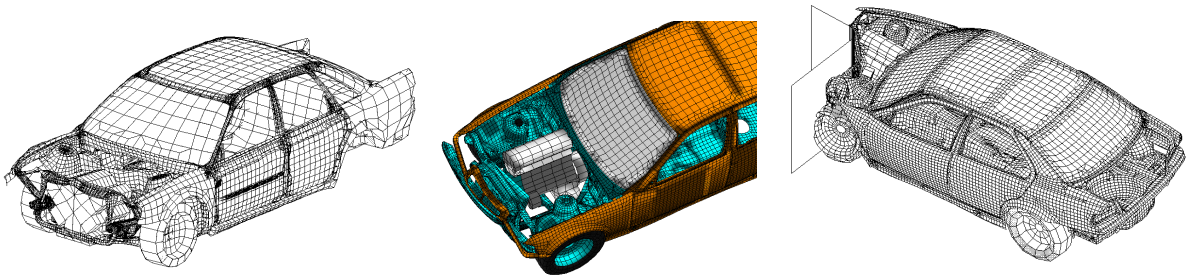

Figure 2: CAMAS-Link EUROPORT Benchmark models: AUDI-Crash30, BMW-Crash15 and BMW-Crash50, respectively. *Courtesy of Audi, BMW*

The BMW-Crash50 model causes the greatest loss in scalability of the three, since it includes the high percentage of elements within contact calculations (45,497 contact segments from the 61,039 elements). Nevertheless, the results on the IBM SP2 shown in Figure 3 demonstrate an impressive speed-up, with sequential supercomputer performance achieved at relatively low processor numbers.

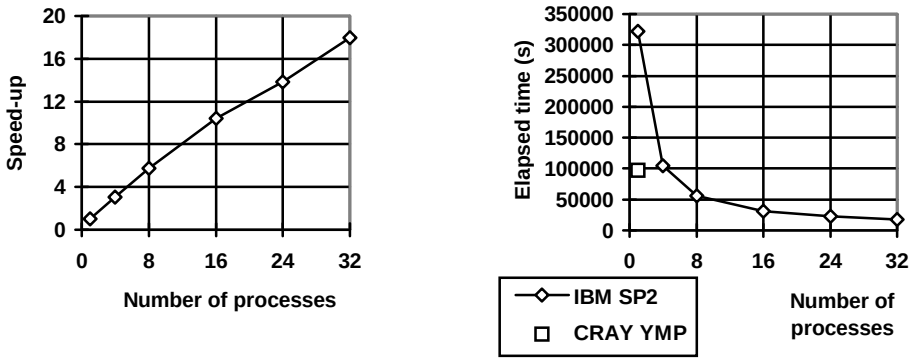

Figure 3: BMW-Crash50 Performance

Both the code portability and the advantage of the message-passing parallelization approach were clearly demonstrated by the benchmarking performed on the SGI Power Challenge. Tests with a variety of models, using numbers of contact segments ranging from 10% to over 70% of the total number of elements in the model, have shown that the message-passing version of PAM-CRASH (in this case using PVM) not only outperforms the shared-memory version at low processor

numbers but also scales significantly better. This is exemplified by the speed-ups obtained on an 18-processor machine shown in Figure 4. The cases tested are two of the above models supplemented by a third BMW model, which is very similar to the BMW-Crash50 case but of slighly smaller size and involving a frontal rather than an offset crash. This additional model is denoted BMW-xtra in the figure.

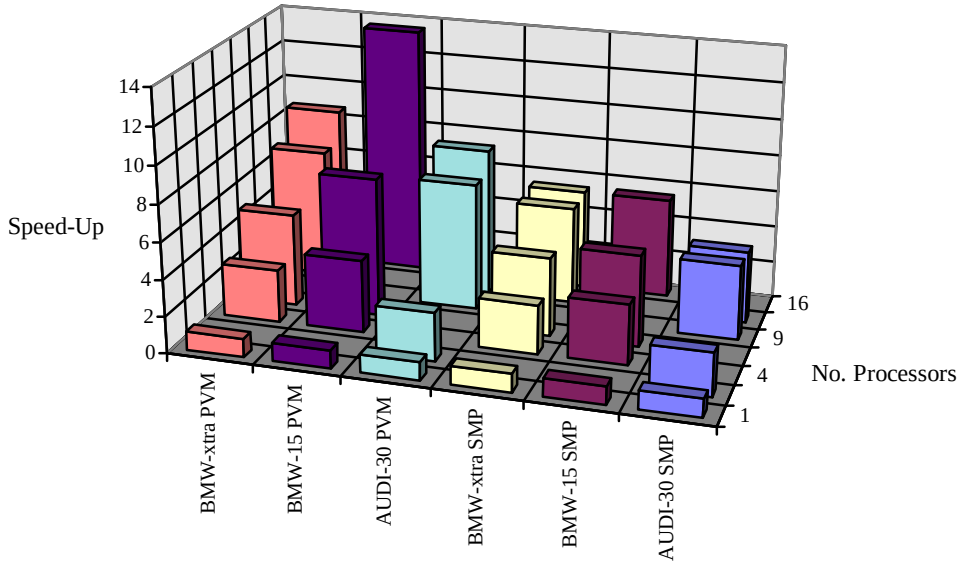

Figure 4: Speed-up for Shared versus Distributed Code Versions on SGI

3 HPF+ Project and PAM-CRASH investigation

The central aim of the ESPRIT Long Term Research project HPF+ is to support and accelerate the development of the HPF language in order that a much wider range of applications may be efficiently parallelised using the high-level language approach. To do this, the project has focussed on complex, scientific applications which involve irregular constructs: unstructured meshes, irregular data structures, computational tasks with dynamically changing costs and data accesses. A sequence of benchmark kernels representing various features and levels of completeness from the codes FIRE (from AVL), IFS (from ECMWF) and PAM-CRASH (from ESI) will be developed and used to define language extensions necessary for these types of irregular codes. A close cooperation with the HPF-2 effort ([8]) will guarantee that HPF+ will not divert from the standard except where this is necessary for providing functionality not being considered under HPF-2. The HPF+ Consortium consists of the application developers listed above[2] together with both academic and commercial language, compiler and tool developers from the Universities of Vienna and Pavia and NA Software.

At the time of writing, the project has completed the definition of first benchmark kernels and an initial definition of the HPF+ language extensions. For the PAM-CRASH -related investigation, concentration has been on the “stress-strain” calculations as described in Section 2, the definition of kernels representing the more complex contact-impact tasks will be handled in the forthcoming project periods. Due to space restrictions, detailed examples of code fragments will not be included here. However, the following discussion gives an indication of the areas of difficulty and

[2] C&C Research Laboratories, NEC Europe are contributing to the project via a collaboration with ESI

(*draft*) language extensions to solve them. Since only language *extensions* are discussed, a knowledge of HPF-1 is assumed.

The dual requirements of the unstructured finite-element mesh - basic variables stored and calculated at nodal points interacting with element-based force computations as described in Section 2 - means that the regular data distributions of HPF-1 are unlikely ever to result in efficient parallel code. Building on experience with the mesh-partitioning approach, an appropriate solution would be the clustering together of elements together with corresponding nodes, since this should minimise necessary communication for the initial kernel. A natural implementation of such would exploit the **DYNAMIC** and **REDISTRIBUTE** features, since the partitioning will be computed or read at run-time, coupled with either a general block distribution (**GEN_BLOCK**) or a completely irregular (array)element-to-processor mapping, **INDIRECT** (see, for example, [1,2]). Turning then to the force calculations, the simplest 'generic', sequential kernel features a loop over elements within which necessary nodal data is gathered, a subroutine is called to calculate the force on the element, corresponding forces are scattered back to the constituent nodal points. Indirect addressing is used for the gather-scatter operations. The HPF+ parallelization of this kernel (currently) focusses on the use of the **INDEPENDENT** directive, linked to the do loop, supplemented by constructs to: specify the distribution of the computation over the processors, **ON HOME**; indicate that a reduction operation takes place, **REDUCTION**; specify that the subroutine execution neither influences other do loop iterations nor introduces communication; indicate to the compiler, that the communication pattern which it may generate is invariant (i.e. it will not change for *all* executions of the particular loop), **REUSE_SCHEDULE**. Whereas, the former options may be found (perhaps in a modified form) in the HPF-2 developments, the latter belongs to the *work-in-progess* category.

4 Concluding Remarks

Investigations into the use of HPF for irregular codes such as PAM-CRASH are in their infancy, but it is already clear that a further development of the language will be necessary if efficient code is to be generated. Although, the simplified code maintenance offered by a high-level programming approach is most desirable, HPF does lag behind message-passing in terms of exploitability and performance for complex, irregular industrial codes. The message-passing version of the PAM-CRASH code has, thanks (to a large extent) to the EUROPORT demonstration of performance, achieved industrial acceptance as a design tool. Although the obtained scalability is not perfect and subject to improvement, a nearly linear speed-up is obtained for processor numbers far beyond what is required to match traditional vector supercomputer performance and this at considerably lower cost. Besides allowing a decreased turn-around, the possibility has now been created to solve larger problems with acceptable delays. In comparison with the shared-memory programming model, the message-passing code version increases the effective peak performance of the machine, allowing greater flexibility in its exploitation.

Acknowledgements

The authors would like to acknowledge the collaboration of their colleagues in the ESPRIT projects CAMAS, CAMAS-Link and HPF+, which led to the results reported on in this paper - the financial support of the European Commission for those projects is also gratefully acknowledged.

References

1. Chapman, B. et.al., ‘High Performance Fortran Languages: Advanced applications and their implementation’ *Future Generation Computer Systems* **11** (1995)
2. Chapman, B. et.al., ‘Extending HPF for advanced data parallel applications’, World-Wide Web document:

   http://www.vcpc.univie.ac.at/activities/r_and_d/Publications.html
3. Clinckemaillie, J. et. al., ‘Performance issues of the parallel PAM-CRASH code’, *Int. J. Supercomputer Applications*, (accepted for publication)
4. Floros, N. et.al., ‘Comparitive efficiencies of domain decompositions’, *Parallel Computing* **21**, , pp. 1823-1835 (1995)
5. Haug, E. et. al., ‘Transport vehicle crash, safety and manufacturing simulation in the perspective of high performance computing and networking’, *Future Generation Computer Systems* **10**, pp. 173-181 (1994)
6. Hertzberger, B. & Serazzi, G. (Eds), Proceedings of the HPCN '95 Conference, *Lecture Notes in Computer Science **919***, Springer-Verlag (1995).
7. High Performance Fortran Forum. *High Performance Fortran Language Specification Version 1.1*, World-Wide Web document:
   http://www.crpc.rice.edu/HPFF/hpf1
8. High Performance Fortran Forum. *HPF-2*, World-Wide Web document:
   http://www.crpc.rice.edu/HPFF/hpf2
9. HPF+ ESPRIT LTR Project No. 21033. *Optimizing HPF for Advanced Applications.* World-Wide Web document:
   http://www.par.univie.ac.at/hpf+
10. Lonsdale, G. et al., ‘Communication requirements in parallel crashworthiness simulation’, Proceedings of the HPCN '94 Conference, *Lecture Notes in Computer Science* **796**, Springer-Verlag, pp. 55-61. (1994)
11. Lonsdale, G. et al., ‘Experiences with industrial crashworthiness simulations using the portable, message-passing PAM-CRASH code’, pp. 856-862 in [6]
12. Mierendorff, H. *et.al.*, ‘Europort-1: Porting industrial codes to parallel architectures’, pp. 806-812 in [6]
13. Stüben, K. *et.al.*, ‘Parallel industrial Fluid Dynamics and Structural Mechanics codes”, Proceedings of the HPCN '96 Conference, *Lecture Notes in Computer Science **1067***, Springer-Verlag, pp.90-98 (1996)