Industrial Computing on MIMD machines

Petter E. Bjørstad
Para//ab, Høyteknologisenteret
N-5020 Bergen, Norway

Abstract

The laboratory for parallel computing - Parallab actively pursues the porting of significant industrial codes to parallel computing platforms. This work started already in 1985 with the acquisition of Europe's first 64 processor Intel hypercube. As of today, the laboratory has three MIMD machines, an Intel Paragon, a Parsytec GC/Power-Plus and a DEC α-cluster. As part of the Europort effort, four industrial codes are currently being ported by Parallab scientists. The paper will describe this effort and the issues, trends and experiences learned from working with industrial codes using MIMD type parallel computers.

1 Industrial Parallel Computing

The use of parallel computing technology applied to industrial computing needs started in Bergen in 1985. Christan Michelsen Institute acquired Europe's first parallel 'supercomputer', an Intel IPSc hypercube with 64 processors (i286) each accessing a local memory of 0.5 megabytes! (later upgraded to 4.5 Mb). With increasing demand for computer power stimulated by Norway's offshore oil industry, one of the first industrial projects was the development of a parallel version of the well known reservoir simulator Eclipse[1]. Parallab - Laboratory for Parallel Computing, was established to provide a good link between rapidly evolving research within the field of parallel algorithms, scientific computing, and industrial use of these techniques in large scale simulations.

Today, ten years later one of the projects being carried out under the European Europort initiative appears very similar, indeed the Eclipse simulator is still the focus of a parallelization project[2]. One may be tempted to conclude that the early effort was badly timed and took place much too early. In fact, it has been a 10 year learning period in which experience and knowledge about parallel algorithms and the programming of parallel machines have been gradually refined. Considering the fact that our 'tools' the (parallel) computers, have increased both in speed and (memory) size by almost a factor 1000 during this period, it is quite impressive that the software has not only kept up, but also come a long way towards portability across parallel computing platforms.

In this article we will describe some of the lessons learned over ten years of industrial parallel computing projects with emphasis on the Europort effort currently taking place at Parallab.

2 Emerging standards

Almost any successful industrial software is the result of more than ten years of focused development. The software developers that maintain and market such (mostly technical) software are usually small or

[1] Developed and marketed by Intera Ltd. 11 Foxcombe Court, UK-Abingdon, OX14 1DZ.
[2] This parallelization effort is carried out by EPCC at the University of Edinburgh

medium sized companies. There are two effects; small companies can more easily respond to new demands and may quickly be able to adapt to dramatic changes, on the other hand their business is often in highly specialized market segments and their economy may be very vulnerable in an unstable market. The companies often critically depend on a small number of important customers. Industrial software packages must be able to adopt to new computer hardware and performance expectations without extensive redesign of the code. Portable code that can be sold and maintained on many computing platforms without excessive cost is therefore of utmost importance.

This is precisely the area where parallel computing has been weak in the previous ten year period. A quick look at the history of message passing codes reveals this immediately. In the early years parallel codes were almost without exception tailored to a very specific target machine. The short life cycle of machines resulted in a constant need for rewriting code. In Europe, much work at universities was directed to transputer based systems. These systems often lacked both an industrial programming environment as well as performance and they were never considered for industrial computing in Norway.

Intel, a pioneer with their line of hypercube machines has supported the message passing software NX over a long period of time. People without parallel machines quickly started to write so called 'hypercube simulators' and the origin of message passing software for workstations can be found here [12]. PVM (Parallel Virtual Machine) [13] was the first software that was widely adopted and an early de facto standard. Today, the parallel computing community is in agreement on supporting the MPI (Message Passing Interface) [14], [18] as a standard software layer for this purpose. Why did it take ten years? Simply because it took about ten years of experience to learn what functionality was needed in such a software layer. In the same time the user base and the computer platforms both grew to a critical size where the demand for standardization became a driving force. Today, message passing programs of MIMD or SPMD type can and should be written using MPI. This standard is a long step in the direction of portable parallel programs. An MPI based program should be developed and debugged on a single workstation then moved to the parallel machine.

Still, considerable challenges remain. Today, almost all parallel programs based on explicit message passing in a distributed memory computer, contain the message passing code in the application code itself. This level of programming should really be reserved for specialists and not be visible at the application level. Industry must learn to work more closely in an interdisciplinary fashion with library developers. Big savings in software development costs can be achieved when application codes are built in a more layered fashion. A more modular design approach has been advocated by many computer scientists for years. Despite this, there are very few production codes where, for example, the equation solver can be easily replaced. Obviously, these problems are hard in real life situations. Parallel programs based on message passing provide additional pressure to accelerate a more modular programming design. All message passing code rightly belongs in a layer of library software on top of which the application code should be written. The concepts of *parallelization expert*, *code owner* and *end user* in the Europort project reflects this division of work that we expect is here to stay.

Similarly, the development of (data) parallel Fortran started out mainly fueled by the early success of Thinking Machines Corporation. The effort of revising Fortran took much longer than it should have taken, and many parallel structures were introduced by various parallel computer companies in the interim period. Shortly after the approval of Fortran 90 the effort for a standardization where also data layout on the processor topology could be specified, gained momentum. Largely based on research compiler projects in Vienna and Texas, scientists went ahead with the computer industry and established High Performance Fortran (HPF) as a de facto standard for parallel Fortran. HPF is an extension to Fortran 90 where data distribution can be specified by user directives. Additionally, the FORALL statement which was left out of the F90 standard in the last moment, immediately made its way back.

It is exciting to see that we already today have HPF programs that can be compiled and run on our 16384 processor MasPar MP-2, on our Intel Paragon, on a cluster of Digital workstations and on a single SUN workstation. This can be done without a single change to the source code. The data distribution and communication requirements are of course quite different, as is the performance. There are considerable challenges for parallel compiler software optimization which is currently subject to active research and development. It is much to early to draw definite conclusions about performance and its relation to the underlying computer architecture. Thus, it remains an open question whether the two first letters in HPF will stand up to their promise. The development over the last couple of years is extremely encouraging

to industries having a large inventory of Fortran based software. HPF may provide a much higher level approach to portable, parallel programs than explicit message passing.

3 The Europort Initiative

The primary objective of the European ESPRIT project EUROPORT is to increase awareness of and confidence in parallel high performance computing (HPC) for commercial and industrial applications. This objective will be achieved by porting large serial commercial and in house codes to parallel architectures to enable significant industrial impact.

In EUROPORT-2, 24 codes are being ported to parallel platforms. Parallab is responsible for the porting of four industrial codes under this program. The codes are all large scale simulation codes with the bulk of their market in the offshore sector. All codes have important business potential in several European countries as well as being sold to customers in Asia and America. Parallab will port and benchmark these codes on two quite different parallel computing platforms, a Parsytec GC/Power Plus with 64 processors and a Digital Equipment cluster of 8 tightly coupled workstations. The use of standards like MPI (and PVM) will protect the software investment and make the code portable to other parallel machines. The four projects being carried out by Parallab are briefly described in the next few paragraphs.

3.1 EP-FRONTSIM

The incompressible pressure solver in the reservoir application package Frontsim/Frontline[3] will be ported. Frontsim/Frontline is a specialized reservoir simulator used by the petroleum industry. The use of parallel processing will enable larger simulations to be conducted (improving accuracy) together with reductions in execution time (reducing cost) compared with those systems currently used. Figure 1 shows an example of graphical output of a typical 3-D simulation. The code uses a modern domain decomposition algorithm that have good parallel properties [6]. Frontsim is written in C++, thus requiring the parallel computers to support this language.

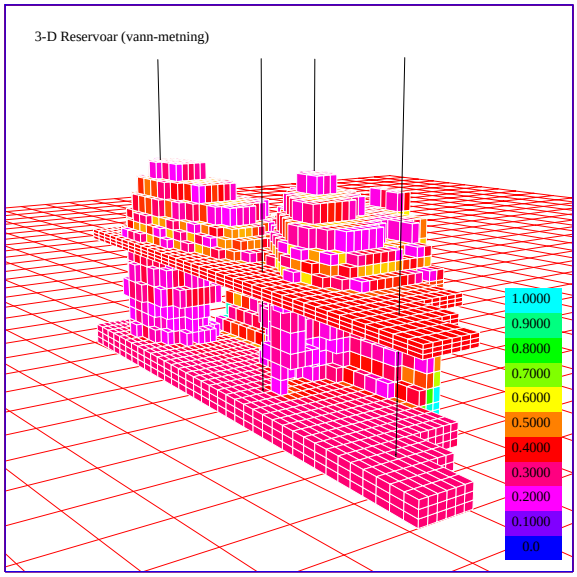

Figure 1: 3-D oil reservoir simulation from Frontsim. The plot shows water saturation in the reservoir at a specified time.

[3] Developed and marketed by Technical Software Consultants, Gaustadalléen 2, N-0371 Oslo, Norway

3.2 EP-MUSIC

The code, MUSIC[4] simulates gas explosions. The package will enhance and replace today's commercial software packages FLACS and GALILEA. These programs are used by many companies in the petroleum industry. By simulation on explosions in refineries or on offshore oil production p he design of the installation in order to minimize risk, d

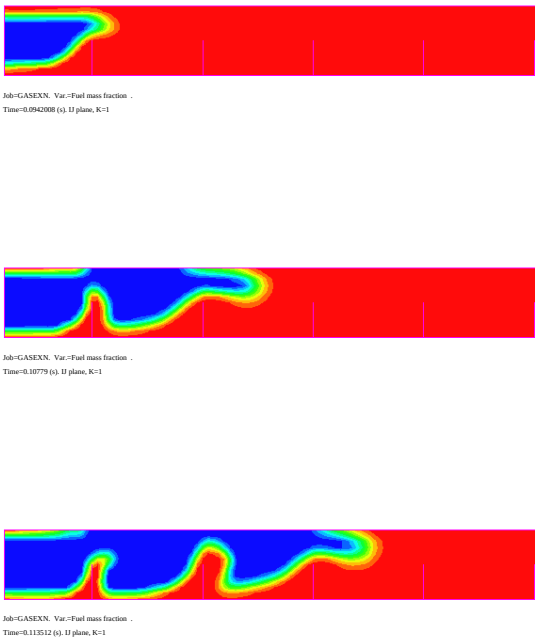

Figure 2: Simulation of a gas explosion flame front using MUSIC.

The simulation of 3-D flame propagation (from a gas explosion) in a complex geometry involves the solution of complex, time dependent partial differential equations. The computational requirements can be very large. The combination of advanced algorithms and the use of parallel computers is needed to advance this field. Figure 2 shows three snapshots of a flame front computed with MUSIC. Realistic output from this code is often by way of computer generated video which can visualize the propagation of many relevant physical parameters in the complex geometry.

The code employs a 3-D domain partitioning combined with an additive preconditioning. Single block iterative solvers based on Krylov subspace algorithms and a selection of preconditioning techniques are also available.

3.3 EP-SESAM

A part of the SESAM[5] application package [20] called SESTRA is being ported to parallel computers. SESAM has set new standards for the solution of large structures problems. Typical end users of SESAM are shipyards, construction companies and oil companies. The code is a result of long development and intimate knowledge of the offshore industries needs and requirements. Figure 3 shows an example of a finite element model of an offshore oil production platform that has been analyzed using the SESAM system.

The code is based on a multi level substructuring technique. The sparse, direct Cholesky factorization is central to the SESTRA module. Parallel execution can take place at a coarse level between different substructures, but also at a finer level within the Schur complement reduction of each substructure. In

[4] Developed and marketed by Christian Michelsen Research, Fantoftveien 38, N-5036 Fantoft, Norway
[5] Developed and marketed by DNV Sesam AS, Veritasveien 1, N-1322 Høvik, Norway

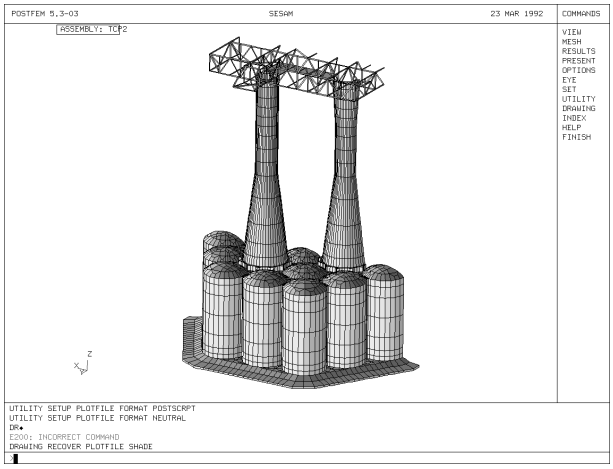

Figure 3: An offshore oil production platform analysis using SESAM.

Europort this will be done by using ScaLAPACK [7] for the parallel execution of a set of matrix algorithms. Several previous projects carried out at Parallab have addressed these issues individually [3], [4], [5], [15]. The current project will result in an integrated, two-level parallel code. Such a code will be portable across a range of different parallel machines. However, being an out of core code, there will always be considerable time spent on I/O. In fact, the code is I/O bound on almost all current computers. The challenge of building inexpensive, scalable secondary storage devices is still open. Most vendors of parallel computers can deliver systems that scale with respect to storage needs. Scalability with the increased processing power provided is still largely a theoretical concept that remains to be demonstrated with realistic large scale application codes. An important task is therefore the efficient transfer of matrix data from a fixed file format to a range of block distributed formats as may be required by ScaLAPACK to insure good performance across a wide range of parallel computers.

3.4 EP-SWAN

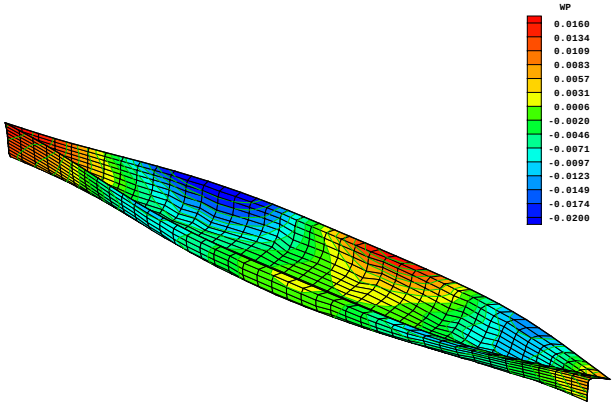

Figure 4: Pressure distribution on a hull calculated with SWAN.

SWAN[6] is used by ship designers, classification societies, and shipyards for wave load simulation. Figure 4 shows the pressure distribution on the under water part of a ship hull due to water resistance and wave-hull interactions. Again, full color graphics and the use of video techniques are used in practice. The quality of the illustrations in this article are obviously not representative. The SWAN code will also be

[6] Developed and marketed by DNV Research AS Veritasveien 11, N-1322 Høvik, Norway

implemented using two levels of parallelism. There is a coarse grain level since SWAN needs to compute over a range of given frequency directions. Additionally, the equations that SWAN is solving must be parallelized. These equations are derived from an integral equation formulation of the problem leading to a dense linear system. The equations are complex and non-symmetric. It is necessary to have both a direct solver and an iterative solver option. In both cases we will use ScaLAPACK, in the iterative case to provide an efficient preconditioner option.

4 The Parsytec GC/Power Plus

A target platform for the industrial codes in the Europort project is the GC/Power Plus (referred to as the GC/PP) manufactured by Parsytec GmbH. The machine has been available since its introduction about one year ago. Parallab installed one of the first machines in September 1994. The machine is relatively new and represents a significant step by establishing a European manufacturer with a product that can be used for large scale parallel computations. We therefore provide a few facts about the machine here. For a more accurate and comprehensive description of this machine as well as of the other computing platforms referred to in this article we refer to relevant technical material provided by the vendors.

The Parsytec GC/PP installed at Parallab is a distributed memory machine with 64 processors and a total of 2 Gigabytes of memory. The machine is designed around a node with 64 Mb of local memory, shared by two processors. The processor is an 80 MHz version of the PowerPC-601 manufactured as a high volume product by Motorola. The 601 processor used by Parsytec is rated at 80 Mflops peak and at 93 SPECfp92. The communication topology is a 2-D grid with each node having 4 links in each of the NEWS directions.

The Parsytec GC/PP runs a light weight kernel called PARIX at each node. We regard this as an advantage since almost all memory is then available for the application code. The support of virtual processors and multi-threaded programming are important strengths in PARIX. We have observed considerable speedup by running 128 PVM processes on our 64 processor machine, demonstrating the kernels ability to hide latencies and effectively use available compute cycles.

Full UNIX capabilities is provided via a multiple front end concept. The Parallab machine uses 4 Sparc-10 workstations from SUN Microsystems as a front end system. Parsytec similarly supports a parallel file system (PFS) implemented via software across the disk drives of the front end machines.

5 The DEC Alpha Cluster

Another target platform in Europort is a cluster of workstations. Parallab provides such a cluster not only for its own computing needs, but as a benchmarking computing platform for all projects in Europort. It should be no surprise that this platform is of particular interest to industrial end users, since this is a computer solution that one realistically can operate locally. Our DEC cluster consists of 8 machines, each running a 233 MHz clock and having a minimum of 128 Mbytes of memory. The CPU delivers a peak of 233 Mflops and is rated at 215 SPECfp92. One machine acts as a fat node with 512 Mbytes of memory and each machine has fast local disks, a total of 34 Gigabytes. The machines are interconnected by a DEC Gigaswitch. This technology provides a dedicated FDDI link to each node, a maximal (bidirectional) bandwidth of 200 Mbyte/s. The latency will, however, still be considerable when compared to more specialized MPP systems. We also have an alternative interconnect technology on order, the DEC memory channel. This approach promises a significant reduction in latency, but is unfortunately not available at the time of this writing.

6 Examples and Experiences

This section will present computational examples relevant for the four Europort projects at Parallab. The examples serve as illustrations for ongoing work. There are no performance measurements in this section that will remain valid at the end of the project.

	Processors			
N	4	8	16	32
1024	11	8	6	3
2048	14	11	9	6
4096	14	13	11	10
8192	-	-	12	11

Table 1: Mflops per processor for ScaLAPACK LU on Parsytec GC/PP. Note the discussion in the text describing implementation details.

6.1 ScaLAPACK

Since we intend to base the fine grain level of parallelism in both SESAM and SWAN on ScaLAPACK, one of the important tasks is the implementation of this package on the Parsytec machine. This approach is consistent with a multi layered software library design supporting a higher level application code. The ScaLAPACK software [7] is a distributed memory implementation of some important routines from the well known library LAPACK [1]. The implementation is based on a set of communication subprograms BLACS, [10],[11], while the numerical kernels running on each node are the familiar BLAS [8], [9], [16], [17]. The design of LAPACK addresses data locality, thus the software is well suited to modern RISC processor architectures with a memory hierarchy. In order to achieve performance one should then focus on the efficient implementation of the BLAS and the BLACS on the target machine. In this paper we will report on results from the initial unoptimized implementation of ScaLAPACK and compare with a similar implementation on our Intel Paragon. The reader should immediately be warned that comparisons like this must be interpreted with considerable care. Our purpose is not to identify the fastest machine, but rather to use this as an example to discuss a few trends and relate these to differences in implementation and computer capabilities.

Table 1 shows the Mflops per processor achieved by a ScaLAPACK algorithm on our Parsytec GC/PP. That is, the overall computational rate based on elapsed time can be found by multiplying the numbers in a column of the table with the number of processors for that column. The problem is a double precision LU factorization of a matrix of size N. We show this effective processor rate for variable problem size and for a variable number of processors. The matrix is distributed block cyclic, with each block having dimension NB. We always report the best performance among possible values of NB. For the two smallest values of N the best performance is obtained with $NB = 2$ or $NB = 4$. For large values of N and larger numbers of processors the best value increases to $NB = 8$ and $NB = 16$. The communication library was PVM 3.2 for Parsytec, while the standard Netlib BLAS (written in F77) without any special optimization was used in the computation. We used revision 1.5 of the Motorola F77 compiler. We see that the fastest computational rate of 14 Mflops is obtained for the smallest number of processors. The rate per processor naturally decreases as the communication overhead becomes more important when more processors participate in the computation. We also observe that the computational rate per processor increases substantially as we increase the problem size N with a fixed number of processors. A natural definition of scalability is to keep the problem size in terms of memory usage constant per processor and consider the computational rate as we increase the number of processors. Since we use N^2 storage this corresponds to comparing a given entry with one that is one row below and two columns to the right. We observe that the machine falls a little short of this goal.

Table 2 shows the Mflops per processor achieved by the same ScaLAPACK algorithm on our Intel Paragon. We have the same distribution of the matrix as above, but on this machine we found the block size $NB = 4$ to perform consistently well. The BLACS was implemented directly on Intel's native NX communication library. Again we used the standard Fortran BLAS from Netlib.

We observe a fastest node performance of 12 Mflops. The qualitative behavior is the same as for the Parsytec machine, but we observe that the Paragon is indeed capable of maintaining the computational rate as we move one step down and two steps to the right in the table. The machine shows near perfect scalable results in this very limited experiment.

	Processors			
N	4	8	16	32
1024	10	9	7	5
2048	12	11	10	8
4096	-	12	12	11
8192	-	-	-	12

Table 2: Mflops per processor for ScaLAPACK LU on the Intel Paragon. Note the discussion in the text describing implementation details.

Lastly, a few words on comparing the data obtained from the two machines. We can deduce from the data that the Parsytec GC/PP has a higher floating point performance per processor than the Paragon. This should come as no surprise given that the Motorola 601 processor is a much newer design than the i860-XP from Intel. It is also clear that the communication performance of the Paragon is better than that of the Parsytec. This is easy to see by comparing the same rows in the two tables. Finally, let us note that the communication was implemented using three layers BLACS-PVM-PARIX on the Parsytec, but only two layers BLACS-NX on the Paragon. This will tend to favor the Paragon. We could have used highly tuned BLAS routines on the Paragon, but decided to leave this exercise to the compiler in order to get a more similar comparison. Optimized BLAS routines are not yet available for the Parsytec, but should be a high priority item. It is also likely that the Intel compiler may be somewhat more optimized for the BLAS since it has been around considerably longer. We note that there are a few extra entries in the Parsytec table. Due to the larger node memory and the smaller PARIX kernel we are able to solve larger problems with a fixed number of processors.

Overall it is striking how similar the performance is. Both processors have about the same peak rating and achieve about 15 percent in this test. The use of highly tuned BLAS would have made the performance numbers more sensitive to the quality of the communication software and hardware.

Finally, a warning about scalability. We looked at this in order to illustrate the concept by way of some elementary data. The idea of designing scalable algorithms (of which the LU ScaLAPACK is an example) is generally worthwhile if one intends to use the same implementation on a wide range of computers with possibly hundreds of processors. There are problems for which scalable algorithms are not known and the concept may be misleading if what one is after is the best possible performance for a given problem on say, a moderate number of processors. Thus, based on our small, illustrative example having 4 to 32 processors, we used the term solely in order to discuss the concept. One should not interpret these results beyond their intention.

	Single 601	DEC Alpha	
	Elapsed	CPU	Elapsed
Merge	140	15	27
Reduction	522	127	184
Retracking	292	67	179
Total time	1381	234	465

Table 3: A small tubular joint test example computed by SESAM.

Parallel computers like the ones discussed here sometimes receive negative comments due to the 10-20 percent (compared to peak) performance. We would like to emphasize that this criticism always should include some measure of price performance. If a product is five times less expensive then maybe it is still competitive even if its performance falls below 20 percent of peak. For industrial computations it is often the availability of a large total memory that may be the overriding important factor.

6.2 Preliminary status with the Europort codes

This section should be read as a progress report, many or most results indicated here are likely to change fundamentally before the project has been completed. The 'snapshots' provided should serve as illustrations for the kind of issues that one typically are faced with when porting industrial codes. It would be wrong and misleading to draw any performance conclusions from the data in this section. We provide this only in order to illustrate the process of porting industrial code.

233 MHz	1 DEC Alpha	3 DEC Alpha)
Reduction	682 min	383 min
Retracking	171 min	72 min
Total time	853 min	455 min

Table 4: A realistic ship computation by SESAM.

SESAM is by far the largest code. Over the past fifteen years it has tested (and broken!) operating systems, compilers, loaders, file systems, and debuggers of most current computers. In doing so, the code has contributed to improving the quality of many computer systems. We are pleased that SESAM at the time of this writing runs correctly on a single Parsytec node. The single node run is often what requires most work on a new computer platform. In Table 3 we compare a single node run with a run on a 233 MHz DEC Alpha. Three important partial times are listed together with the total time. The Parsytec elapsed times are from two to five times slower than the DEC Alpha, but this is only on a single node. We expect a parallel execution to improve this ratio, but we notice that the 'Merge' part may pose a problem as it will only be subject to a coarse grain parallelism, leaving large tasks of this kind to single node execution also in the future. A significant effort with the system software is most likely required for the Parsytec machine to become competitive with a cluster concept for this type of application.

In Table 4 we show parallel performance of SESTRA on our DEC Alpha cluster. We observe an overall factor of two gain when using three machines. This may correspond to a typical industrial computation using locally available resources. This analysis has 212.000 degrees of freedom and requires about 6 Gbytes of secondary storage. The example has also been run on a 2 CPU, 100MHz Pentium based PC taking 65 hours of computing time. Thus, there is still a considerable gap between a high end PC and a state of art workstation cluster.

The FRONTSIM code is written in C++, and our compiler is at the time of this writing still unsupported. This code has already been tested in parallel on a workstation cluster. The final port to the Parsytec GC/PP has therefore been moved out in time as we wait for better compiler support.

Additionally, the Frontsim code has a substantial sequential part, called the front tracking phase, which determines the saturation. Unfortunately, the memory requirements of this code segment currently scales with the memory used in the parallel pressure solver. As a consequence of this the code needs at least one fat node with a very large memory. On the GC/PP this can only be handled by executing this part on a front end SUN having sufficient memory. This in turn requires a PVM with support for heterogeneous computer systems which at the time of writing, is not yet available.

The front tracking part of the code can and should be parallelized, but this task is outside the scope of the Europort project. A lesson learned; the benefit from the parallel code may be limited unless the entire application can be scaled up on a distributed memory machine. It may well be advisable to parallelize a large code in stages, but definite plans for the entire application code should be in place.

Processors	1	2	4	8
pipe11	22	16	13	-
pipe21	898	511	322	231

Table 5: Two small tests showing elapsed time in seconds with the code MUSIC on the Parsytec GC/PP.

	# PROCESSORS	WALL CLOCK	SPEED-UP
Sequential code	1	6245 min	
Parallel code	6	1054 min	5.9

Table 6: Parallel performance for the SWAN code: 140 combinations of speeds and headings for a tanker hull using DEC Alpha (233 MHz).

Parallel SWAN solver (complex arithmetic)				
Processors	3	6	12	24
Factorization	63.0	33.5	19.9	12.7
Observed speedup	(1.0)	1.9	3.2	5.0
Linear speedup	1.0	2.0	4.0	8.0

Table 7: Timing of the linear solver in SWAN on the Parsytec GC/PP.

The MUSIC code is written in Fortran 90. A Fortran 90 node compiler has been delivered from NAG[7] and is operational, working together with PVM on our system. We report on two very small test problems in Table 5. The current run uses BicgSTAB [19] with a polynomial preconditioner. Turbulence has not yet been included in the test.

We show results from the coarse grain parallel phase of SWAN in Table 6. This is a realistic computation performed on 6 DEC Alpha machines. We observe an almost perfect speedup on this large problem. In Table 7 we similarly show computational results from the fine grained parallel implementation of the equation solver. This is based on ScaLAPACK, and one would expect similar numbers to the ones reported in Table 1. The SWAN code still needs improved I/O performance on the GC/PP similar to SESTRA and we do not report on runs where both levels of parallel code execute simultaneously.

As this report from ongoing work shows, industrial codes can benefit greatly from high performance computing. Industrial codes have a broad requirement to system software. Just in these four codes we needed four different compilers, message passing supporting both homogeneous and heterogeneous nodes, fat nodes with substantial memory, and very high performance distributed I/O file systems. Thus, the main challenge is not to build the parallel hardware, but in a timely fashion to create and support a sufficiently advanced and stable programming environment for the machine.

7 Trends

There are several trends within the field of large scale computing that should be noticed. The Parsytec GC/PP using the Motorola 601 processor signals the use of PC components even in supercomputers. It is clear that PC components and products are attacking the traditional workstation market. One can today run applications on a Pentium based PC that just recently required a workstation to run. The previous generation of parallel supercomputers claimed that they took advantage of high volume off the shelf components, but this was only partially true and the price certainly testified to this fact. The tradeoff between state of art (expensive) processors, possibly running at a very high clock rate, and the use of slightly more (inexpensive) conservative technology, but compensating by using a higher number of processors remains to be settled. This issue is directly coupled to high aggregate memory bandwidth vis-a-vis the cost of high speed processor interconnect providing sufficient bisectional bandwidth.

Workstation clusters are still popular as compute servers, but until recently they have had a severe handicap with high latency in their interconnecting technology. This issue is being addressed and low latency interconnecting technology like the SCI standard offered by Dolphin[8] or the reflective memory of

[7] The Numerical Algorithms Group Ltd. Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK

[8] Dolphin Interconnect Solutions AS, Olaf Helsets Vei 6, 0621 Oslo, Norway

Digital will become available in the near future. The memory hierarchy and the performance variations depending on data reuse and the data locality of algorithms are here to stay. The balance of computational speed, hierarchy of caches and ultimate memory bandwidth is now better understood by software library developers. There is a clear trend towards increased use of these systems for large scale industrial computations. The technical challenge of scaling these systems to very large size remains.

Similarly, parallel programming standards like MPI and HPF are gaining acceptance and have already improved portability of parallel programs. The task of mapping data to a distributed memory and the possible redistribution of data during program execution will keep optimizing compiler writers employed for many years to come. It remains to be seen if the application programmers view of large MPP systems again may return to the single (shared) memory with a global addressing scheme. The KSR machine was an example of a nonuniform memory access machine that failed, however new products that support this model will emerge. It is a safe assumption that programs based on explicit message passing will perform well on such machines, the challenge are the programs that have been left behind in the move to distributed memory machines.

The trend towards SPMD as the preferred programming model and the advantage of light weight system software kernels on most nodes in a parallel computer to be used for scientific computing is also evident. Fat nodes running a full UNIX will get more integrated. The trend away from front end computers has been around since the FPS-164 in the early eighties, but a front end is still an easy way out when new technology is rushed to market.

Available memory is getting very large, but the need for high speed transfers from memory to disk storage will not disappear. The design of truly scalable, high speed parallel file systems from high volume components has not been very successful to date. Such systems often claim impressive performance figures, but when measured from industrial application codes the actual performance is often worse than the gap between observed and peak computational performance.

8 Acknowledgements

This is a revised and updated version of [2]. Several scientists at Parallab contributed to this work. Jeremy Cook provided the data on SESAM, Mohammad Talal Rahman provided the data on MUSIC, Ove Sævereide described the SWAN example and Yngve Pettersen compared ScaLAPACK on the two machines. Credit is also due to our industrial partners whose computational problems and software provide a never ending challenge. The author expresses his warmest thanks.

References

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. D. CROZ, A. GREENBAUM, S. HAMMERLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK Users' Guide*, Siam, 1992.

[2] P. BJØRSTAD, *Experience with industrial applications on MIMD machines*, in Supercomputer 1995 Anwendungen, Architekturen, Trends, H.-W. Meuer, ed., K. G. Saur, 1995, pp. 98–117. FOKUS Praxis Information und Kommunikation, Band 13.

[3] P. BJØRSTAD, J. BRÆKHUS, AND A. HVIDSTEN, *Parallel substructuring algorithms in structural analysis, direct and iterative methods*, in Fourth International Symposium on Domain Decomposition Metho ds for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. Meurant, J. Périaux, and O. B. Widlund, eds., Philadelphia, 1991, SIAM, pp. 321–340.

[4] P. BJØRSTAD AND J. COOK, *Large scale structural analysis on massively parallel computers*, in Linear Algebra for Large Scale and Real-Time Applications, M. Moonen, G. Golub, and B. D. Moor, eds., Kluwer Academic Publishers, 1993, pp. 3–11. NATO ASI Series.

[5] P. E. BJØRSTAD, *A large scale, sparse, secondary storage, direct linear equation solver for structural analysis and its implementation on vector and parallel architectures.*, Parallel Computing, (1987).

[6] P. E. BJØRSTAD, R. MOE, R. OLUFSEN, AND E. VAINIKKO, *Domain decomposition techniques in parallelization of the 3-dimensional frontsim code.*, 1995. To be presented at ZEUS'95, May 17-18, Lindkoping, Sweden.

[7] J. CHOI, J. DONGARRA, D. W. WALKER, AND R. C. WHALEY, *Scalapack reference manual*, Tech. Rep. ORNL/TM-12470, Oak Ridge National Laboratory, April 1994.

[8] J. DONGARRA, J. DU CROZ, I. DUFF, AND S. HAMMARLING, *A set of level 3 basic linear algebra subprograms: Model implementation and test programs*, ACM Trans. Math. Soft., 16 (1990).

[9] J. DONGARRA, J. DU CROZ, S. HAMMARLING, AND R. HANSON, *An extended set of Fortran Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 14 (1988).

[10] J. DONGARRA, R. A. VAN DE GEIJN, AND R. C. WHALEY, *A users' guide to the blacs*, tech. rep., Oak Ridge National Laboratory, December 1993.

[11] ——, *Lapack working note draft:a users' guide to the blacs v1.0*, tech. rep., Oak Ridge National Laboratory, January 1995.

[12] T. H. DUNIGAN, *Hypercube simulation on a local area network*, Tech. Rep. ORNL/TM-10685, Oak Ridge National Laboratory, November 1988.

[13] A. GEIST, A. BEGUELIN, J. J. DONGARRA, W. JIANG, R. MANCHEK, AND V. S. SUNDERAM, *Pvm 3 user's guide and reference manual*, Tech. Rep. ORNL/TM-12187, Oak Ridge National Laboratory, May 1993.

[14] W. GROPP, E. LUSK, AND A. SKJELLUM, *Using MPI*, MIT Press, 1994.

[15] A. HVIDSTEN, *A parallel implementation of the finite element program SESTRA*, PhD thesis, Department of Informatics, University of Bergen, Norway, 1990.

[16] S. L. JOHNSSON AND L. F. ORTIZ, *Local basic linear algebra subroutines (lblas) for distributed memory architectures and languages with array syntax*, Tech. Rep. TR-09-92, Harvard University, 1992.

[17] C. L. LAWSON, R. J. HANSON, D. R. KINCAID, AND F. T. KROGH, *Basic Linear Algebra Subprograms for Fortran usage*, ACM Trans. Math. Soft., 5 (1979).

[18] *MPI: A message passing interface standard*, The International Journal of Supercomputer Applications and High Performance Computing, 8 (1994). Special Issue.

[19] H. VAN DER VORST, *Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[20] VERITAS SESAM SYSTEMS, P.O. BOX 300, N-1322 HØVIK, NORWAY, *SESAM technical description*, april 1989.