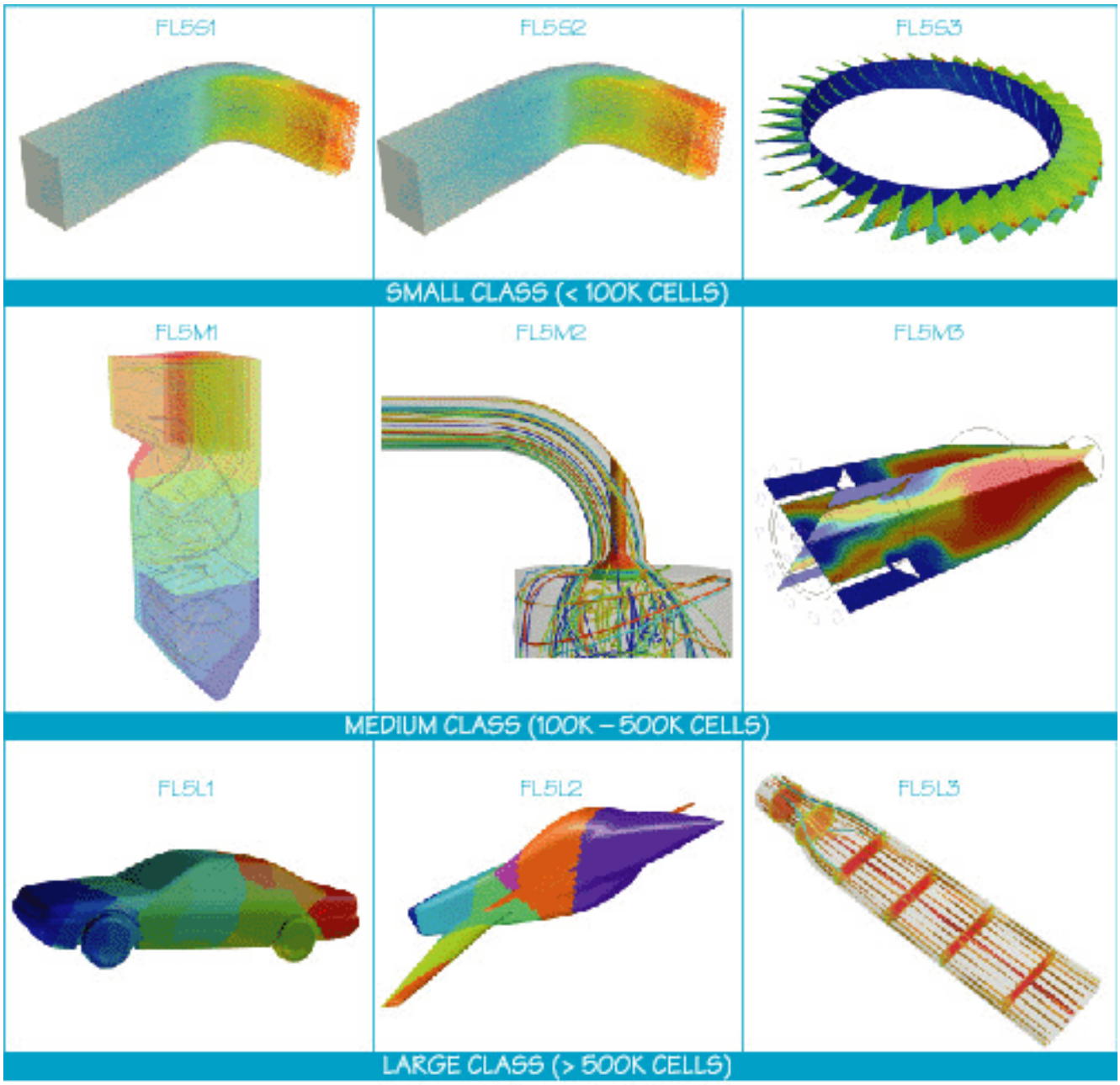
Swiss federal institute of technology
Ecole polytechnique fédérale de Lausanne

N° 12 – Nov. 2000

EPFL SUPERCOMPUTING REVIEW

NUMERICAL SIMULATION FOR SCIENCE AND TECHNOLOGY



Pictorial representation of the nine *Fluent 5* benchmark test cases (see article on page 13)

ECOLE POLYTECHNIQUE
FEDERALE DE LAUSANNE

Contents

Table des matières

Using Smoothed Particle Hydrodynamics for Industrial Flow Simulations

Application de la méthode de Smoothed Particle Hydrodynamics *à la simulation des écoulements industriels*

Mark L. Sawley, Paul W. Cleary & Joseph Ha 3

Visualization Tools and Environments for Very Large Data

Outils et environnements de visualisation pour des très grands volumes de données

Jean M. Favre 9

The Fluent 5 Benchmark Results on the Swiss-T1

Résultats du benchmark de Fluent 5 sur la Swiss-T1

Mark L. Sawley, Trach-Minh Tran & Tom L. Tysinger 13

SFIO,Parallel File Striping for MPI-I/O

SFIO, Système de fichiers distribués pour MPI-I/O

Emin Gabrielyan 17

Stability and α-particle Confinement in the Sphellamak Reactor Concept

Stabilité et confinement des particules α dans le concept des réacteurs du type Sphellamak

W. Anthony Cooper & Olivier Fischer 22

HPC in Computational Structural Mechanics

HPC en mécanique des structures

Pieter Volgers 29

Flow Simulations on High Performance Computers using the NSMB flow solver

Simulation des écoulements sur des ordinateurs à haute performance utilisant le solveur NSMB

Jan B. Vos 32

Editorial

L'inauguration de la Swiss T1 du 23 août dernier à l'EPFL a démontré que l'utilisation d'un ordinateur parallèle basé sur une grappe constituée de composants standard du marché est une solution viable pour des calculs à haute performance. Les résultats obtenus par les différents utilisateurs sur cette machine et présentés durant cette journée ont montré qu'effectivement, le développement ainsi que l'exploitation des applications parallèles sur la Swiss T1 sont possibles dans le domaine de calculs à haute performance (HPC), ceci grâce à l'utilisation généralisée de la bibliothèque standard MPI (Message Passing Interface) *qui permet d'écrire des codes portables et performants.*

*Sur les sept articles de cette édition du Supercomputing Review, cinq présentent les résultats obtenus sur la Swiss T1. En particulier, l'article de J. Vos montre l'efficacité de la Swiss T1 à travers une comparaison détaillée des performances de calcul et du surcoût dû aux communications entre différents ordinateurs. En dehors des calculs purement techniques, il faut relever l'article de E. Gabrielyan proposant une solution efficace pour effectuer des entrées-sorties parallèles, possibilité qui est d'ailleurs prévue dans le nouveau standard de MPI-2. Dans les deux autres articles (*Application de la méthode de Smoothed Particle Hydrodynamics à la simulation des écoulements industriels *de M. Sawley et. al. et* Stabilité et confinement des particules α dans le concept des réacteurs du type Sphellamak *de W.A. Cooper et. al.) les auteurs décrivent des modèles de simulation particulièrement gourmands en ressources de calculs et potentiellement candidats pour une implémentation sur des ordinateurs parallèles.*

Editorial

The Swiss T1 inauguration day on the 23rd August at the EPFL has demonstrated that the utilization of a computer cluster made of computer commodities is an efficient solution for High Performance Computing (HPC). Indeed, results obtained by several Swiss T1 users show that development as well as the exploitation of parallel applications are possible in HPC, using the code portability and efficiency of the MPI *(Message Passing Interface)* standard.

Five out of the seven articles presented in this edition of the Supercomputing Review are dedicated to results obtained on the Swiss T1. More specifically, J. Vos shows the efficiency of Swiss T1 through a detailed comparison of performance and communication overhead between different computers. Beside the purely technical computations, E. Gabrielyan proposes a performant solution for the parallel IO, a feature already included in the new MPI-2 . In the two remaining articles (*Using Smoothed Particle Hydrodynamics for Industrial Flow Simulations* from M. Sawley et. al. and *Stability and α-particle Confinement in the Sphellamak Reactor Concept* from W.A. Cooper et. al.) , the authors describe simulations requiring especially large computer resources and thus are good candidates for an implementation on parallel computers.

USING SMOOTHED PARTICLE HYDRODYNAMICS FOR INDUSTRIAL FLOW SIMULATIONS

MARK L. SAWLEY, PAUL W. CLEARY & JOSEPH HA,
CSIRO MATHEMATICAL & INFORMATION SCIENCES, CLAYTON, AUSTRALIA

La simulation numérique des écoulements industriels par la méthode de **smoothed particle hydrodynamics** *est présentée. Cette technique lagrangienne, qui s'applique sans maillage, se montre bien adaptée aux applications impliquant des géométries et surfaces libres complexes. Ces avantages sont illustrés par l'application aux deux procédés industriels suivants : le moulage mécanique sous haute pression et le moulage des pièces composites par transfert de résine.*

The application of the smoothed particle hydrodynamics method to numerical flow simulations of industrial interest is presented. Such a meshless, Lagrangian technique is shown to be particularly well adapted to applications involving complex geometries and free surfaces. The advantages of this method are illustrated by its application to two industrial processes: high pressure die casting and resin transfer moulding.

INTRODUCTION

Smoothed particle hydrodynamics (SPH) is a meshless Lagrangian method for modelling mass flow and heat transfer. Material properties are approximated by their values at a discrete set of disordered points, or *SPH particles*. SPH is directly based on the resolution of the macroscopic governing equations, such as the Navier-Stokes equations. These equations are written as a set of ordinary differential equations for the mass and heat flux of the SPH particles. The SPH method has been developed over the past two decades, primarily for astrophysical applications [1]. More recently, the method has been extended to incompressible enclosed flows [2],[3].

This article presents the results of a study into the application of the SPH method to the simulation of industrial flow problems. SPH has a number of properties that make it well suited to complex flow simulations:

- momentum-dominated flows are particularly adapted to such a Lagrangian technique,
- complex physics, such as multi-phase flow, realistic equations of state, heat transfer and solidification/curing, can be included in a rigorous manner,
- complex geometries - in particular, complex free surfaces - can be handled in a simple manner,
- extension from two- to three-dimensional flows is straightforward.

Two specific industrial applications are considered in this article: high pressure die casting (HPDC) and resin transfer moulding (RTM). These applications illustrate the use of different aspects of the SPH method. HPDC involves momentum-dominated flows with highly complex free surface behaviour, which are extremely difficult to simulate with conventional numerical methods. RTM involves relatively slow resin flow through a geometrically complex porous medium; SPH is used to provide a mesoscopic-level approach to the numerical simulation of this process.

Being a Lagrangian method, SPH shares with other particle-based methods the characteristic of being compute intensive. Various means to alleviate computational requirements, involving algorithmic improvements and parallel simulation, are briefly presented.

SMOOTHED PARTICLE HYDRODYNAMICS

In the SPH method, the interpolated value of any field A at position $\mathbf{r}$ is approximated by

$$A(\mathbf{r}) = \sum_b m_b \frac{A_b}{\rho_b} W(\mathbf{r}-\mathbf{r}_b, h) \qquad (1)$$

where the value of A associated with particle b at $\mathbf{r}_b$ is denoted by A_b. The mass and density of particle b are denoted by m_b and ρ_b, respectively. $W(\mathbf{r},h)$ is a spline-based interpolation kernel of radius twice the interpolation length h. The kernel is a C^2 function that approximates the shape of a truncated Gaussian function with support radius $2h$; the sum in Eq. (1) is thus restricted to all particles b within a radius $2h$ of $\mathbf{r}_b$.

The use of an interpolation kernel allows smoothed approximations to the physical properties of the material to be calculated from the particle information. The smoothing formalism also provides a means to determine gradients of material properties. The gradient of the function A is given by

$$\nabla A(\mathbf{r}) = \sum_b m_b \frac{A_b}{\rho_b} \nabla W(\mathbf{r}-\mathbf{r}_b, h) \qquad (2)$$

In this way, the SPH representation of the hydrodynamic governing equations can be built from the Navier-Stokes equations.

The most appropriate choice of SPH continuity equation is

$$\frac{d\rho_a}{dt} = \sum_b m_b (\mathbf{v}_a - \mathbf{v}_b) \nabla W_{ab} \tag{3}$$

since it is Galilean invariant, has good numerical conservation properties and can be applied to free surfaces. The following form of the SPH momentum equation is used

$$\frac{d\mathbf{v}_a}{dt} = \mathbf{f} - \sum_b m_b \left[\left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} \right) - \frac{\xi}{\rho_a \rho_b} \frac{4\mu_a \mu_b}{(\mu_a + \mu_b)} \frac{\mathbf{v}_{ab} \cdot \mathbf{r}_{ab}}{\mathbf{r}_{ab}^2 + \eta^2} \right] \nabla W_{ab}, \tag{4}$$

where $\mathbf{r}_{ab} = \mathbf{r}_a - \mathbf{r}_b$, $\mathbf{f}$ is a body force (per unit mass), μ the dynamic viscosity, and ξ and η are constants. The above formulation automatically ensures continuity of stress across material interfaces, and allows multiple materials with highly varying viscosities to be simulated accurately.

SPH is currently implemented as a quasi-compressible method, with the pressure given by the equation of state

$$P = P_0 \left[\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right] \tag{5}$$

where P_0 is the magnitude of the pressure and ρ_0 is the reference density. For the present study, the ratio of specific heats $\gamma = 7$ is used. The speed of sound c_s in the material is given by

$$c_s^2 = \frac{\gamma P_0}{\rho_0} = \alpha V^2 \tag{6}$$

where V is the characteristic or maximum fluid velocity. Typically, α is chosen to be around 100, which ensures that the density variation is less than 1% and the flow can be regarded as incompressible.

Two types of boundary conditions have been used in the present study. Periodic boundaries are applied by simply replacing SPH particles that move outward across the boundary by inward moving particles at the corresponding opposite boundary. External solid walls are modelled by placing *boundary particles* along the boundary. These particles exert Leonard-Jones forces on the SPH particles in the normal direction; appropriate interpolation of these forces produces arbitrary, smoothly-defined boundaries. In the tangential direction, the particles are included in the summation for the shear force to give non-slip boundary conditions at the walls. Note that the solid walls need not necessarily be stationary; a moving wall is used, for example, to model flow ahead of a piston injecting molten metal into a die. The above set of ordinary differential equations is integrated in time using an explicit predictor-corrector scheme, which has a time step limited by the Courant condition.

Further details of the numerical implementation of the SPH method can be found in [3],[4].

It is important to note that the SPH method does not require a computational mesh. The SPH particles contain all the computational information and are free to move throughout the computational domain.

HIGH PRESSURE DIE CASTING

High pressure die casting (HPDC) is a widely used manufacturing process for mass production of components of aluminium and magnesium alloys, such as automotive transmission housings and gearbox parts. Molten metal is injected at high speed (50 to 100 m s^{-1}) and under very high pressures into a die through a complex gate and runner system. The geometrical complexity of the die leads to strongly three-dimensional fluid flow. Within the die cavity, jetting and splashing results in liquid droplet and possibly atomised spray formation. Crucial to the production of homogeneous cast components with minimal entrapped voids is the order in which the various parts of the die fill and the positioning of the gas exits. This is determined by the design of the gate configuration and the geometry of the die.

The geometry of the die, the gate, the runner and the cylindrical shot sleeve considered in the present study is shown in Fig. 1. The fluid initially fills the cylindrical column and is pushed downward by a plunger at the top of the fluid moving at 15 m s^{-1}. The fluid has a density ρ = 1000 kg m^{-3} and a dynamic viscosity μ = 0.08 Pa s. The Reynolds number at the gate is about 2700, with reference to the gate height of 5 mm. (Simulations have also been performed for Reynolds numbers of 500 and 2.7x10^4.) In these simulations, a resolution of 1 particle/mm was used giving a total of 292,931 particles.

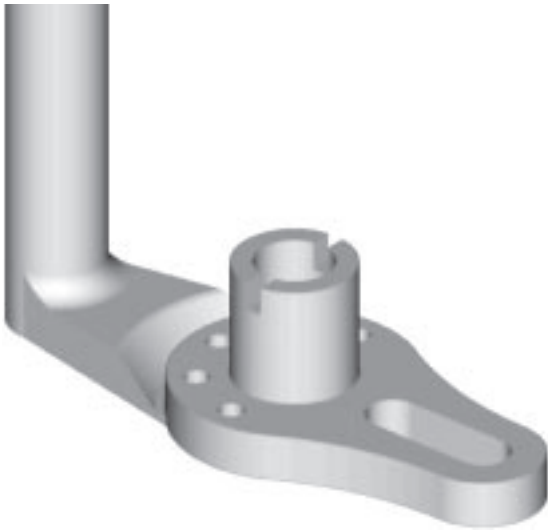


Fig. 1 – Die for a slotted guide, including attached cylindrical shot sleeve and fanned runner and gate

Two perspective views of the filling pattern at different times are shown in Fig. 2. The first frame at 4 ms shows the system after the runner has been filled and the fluid is just entering the die. The second frame at 6 ms shows the fluid entering the vertical cylindrical section. These frames indicate that the leading material consists of fast moving fragments and droplets generated by splashing as the fluid flows around the distinct features of the die cavity. The final two frames, at times 8 and 10 ms, show the fluid converging into the slotted section of the die. Despite the geometrical

symmetry of the die, the flow is observed to be asymmetric. In addition, the jetting of fluid with a high velocity through the gate gives rise to a small unfilled cavity on each side of the die; this cavity remains one of the last regions to be filled.

The extreme complexity of the filling of this relatively simple die geometry illustrates the severe demands imposed on a numerical method by HPDC simulation. The present meshless SPH method excels under such conditions, where the flow is strongly momentum dominated. While this study has been concerned solely with the die filling, it should be noted that other physical phenomena associated with HPDC (eg thermal effects, solidification [5]) can also be incorporated into the SPH methodology in a relatively straightforward manner.

Further details concerning the application of SPH to HPDC, including the results of validation studies, can be found in [5]-[7].

RESIN TRANSFER MOULDING

The resin transfer moulding (RTM) process is used to manufacture polymer composite components [8]. Liquid resin is forced under pressure into a mould containing a preform. The preform is chosen according to the desired properties of the resulting composite component, and generally consists of woven fibres. A number of other factors are also important, including the dynamics of the resin flow. Small clearances may exist between the fibre preform and the mould edges due to the unravelling of fibre bundles during the cutting of the preform, imperfect fitting or deformation of the preform. Such defects influence the resin flow, with high permeability regions leading to *race-tracking*; these edge effects can result in the production of unsatisfactory composite components.

RTM involves the flow in a finite unsaturated porous medium. Before examining such a process, a preliminary investigation was undertaken for flow in an infinite saturated porous medium. The creeping flow of an incompressible, isothermal, single-phase fluid through a porous medium is governed by the well-known Darcy law [9]. A body force $\mathbf{f}$ applied to a Newtonian fluid completely filling a porous medium with permeability $\mathbf{K}$ gives rise to drift velocity $\mathbf{u}_D$ given by

$$\frac{\mu}{\mathbf{K}}\mathbf{u}_D = \mathbf{f} \qquad (7)$$

Here $\mathbf{K}$ is a second-order symmetric tensor, which may have non-zero off-diagonal elements if the porous medium is anisotropic. Equation (7) is a formulation of the Darcy law, and is valid provided that viscous forces dominate over the inertia forces, i.e.

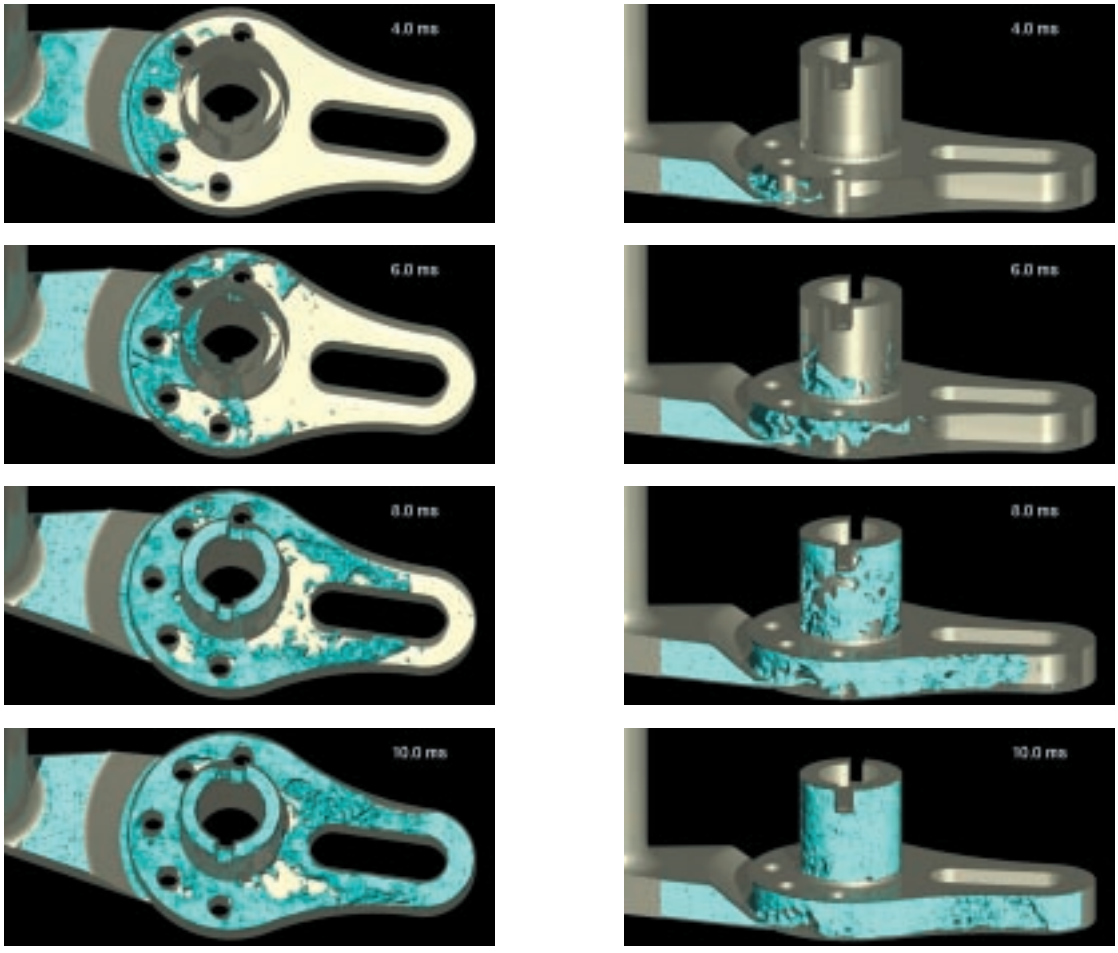


Fig. 2 – Plan and side views at selected times during the filling of the slotted guide die

$$Re_p = \frac{\rho u_p d_p}{\mu} < 1 \qquad (8)$$

where u_p is the average pore velocity and d_p is a characteristic length scale of the pores.

Within the framework of the SPH methodology, a porous medium can be modelled by the inclusion of a number of *fixed particles* within the flow domain. These particles provide a similar contribution to both the continuity and momentum equations as the mobile SPH particles. However, fixed particles are not influenced by the resulting forces, and remain stationary (it being implicitly assumed that the porous medium structure provides the necessary counteracting force).

Since there is considerable freedom in the choice of the number, location and clustering of the fixed particles, such a model allows a large degree of flexibility to tailor the properties of the porous medium. In addition, the interpolation kernel W used in the SPH formalism can be modified to alter the specific nature of the inter-particle interactions. Such flexibility is of particular importance for the modelling of a wide range of porous media flow conditions.

In a saturated porous medium, the entire (non-isolated) pore volume is filled with fluid. A convenient means to construct initial conditions for the SPH modelling of such a medium is to establish an array of particles, of which an appropriate number are randomly assigned as fixed particles. The porosity ε of the medium can be defined by

$$\varepsilon = 1 - \frac{number\ of\ fixed\ particles}{total\ number\ of\ particles} \qquad (9)$$

It should be noted that the present model is not intended to give an accurate representation of the microscopic pore-scale behaviour of the flow. Except for certain academic examples, such a microscopic representation would involve a very complex fluid-solid interface to represent the detailed pore structure. Rather, the present SPH model is intended as an intermediate level, mesoscopic-scale representation. The model aims to provide the correct global behaviour of the flow, while also enabling the possibility to include more fine-scale details that are not available from a macroscopic model.

An example of such a model of a two-dimensional isotropic saturated porous medium is shown in Fig. 3, using a medium constructed from an 80x80 array of particles. A constant body force $\mathbf{f} = f_0 \mathbf{x}$ (from left to right) is applied to the medium. Periodic boundary conditions are imposed at each of the four edges of the array.

Fig. 3 shows that the body force produces a movement of the mobile SPH particles through the void regions surrounding the fixed particles. The preferred path of the mobile particles across the medium has a tortuosity determined by the placement of the fixed particles. While the local velocity of the individual particles contains time-varying **x** and **y** components, the **y** component of the time-averaged drift velocity is found to be negligible.

A series of simulations has been undertaken to provide a more quantitative analysis of the induced drift velocity. Parametric studies have been undertaken varying the following quantities:

- spatial resolution (ie number of SPH particles);
- porosity (ie percentage of fixed particles);
- porous medium properties (ie layout of fixed particles);
- fluid properties (eg viscosity);
- magnitude of applied body force.

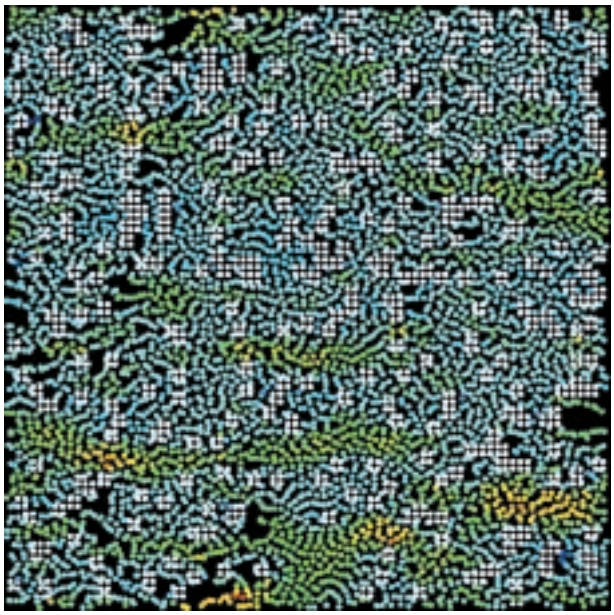


Fig. 3 – Example of flow in a 2D porous medium with $\varepsilon = 0.8$ and $f_0 = 1$. Fixed particles are coloured white, while mobile particles are coloured by their velocity magnitude (Flow is from left to right)

Such studies have enabled a detailed analysis of the properties of our mathematical model of porous media.

As an example of the results obtained, Fig. 4 presents the calculated time-averaged drift velocity as a function of the amplitude of the applied force, for $\varepsilon = 0.8$, $\rho = 1000$ kg m^{-3} and $\mu = 0.1$ Pa s. These results show that for low applied force (for which the inequality (8) is satisfied), the induced drift velocity is proportional to the force. Since the porosity (and hence the permeability) of the medium and the fluid viscosity were constant for the simulations, these results are thus in agreement with the Darcy law, Eq. (7). For higher applied force, corresponding to a Re_p greater than unity, a nonlinear relationship is observed between the force and the resulting drift velocity.

While flow in an infinite saturated porous medium may be a valid approximation for some problems, in a number of applications - such as RTM - the time-dependent filling of the porous medium is of critical importance. For such applications it is necessary to model the finite extent of the porous medium and the propagation of the fluid free surface. Our studies to date have considered a very simplified model of mould filling in the RTM process. Important effects such as fibre wetting, the removal of entrapped air, and resin curing are not considered. However, it should be noted that many of these more complex phenomena can be incorporated in a straightforward manner into the SPH methodology.

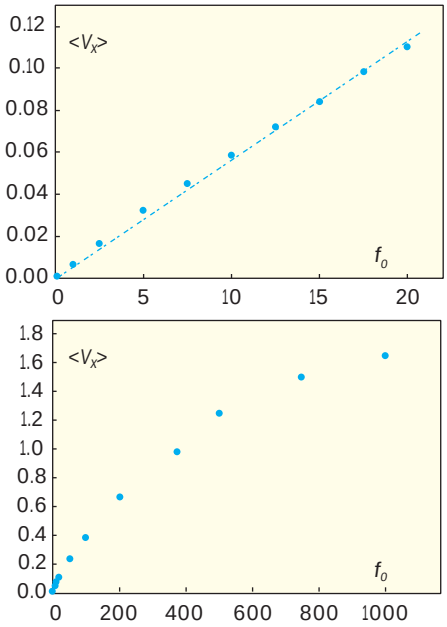


Fig. 4 – Dependence of the drift velocity $\langle V_x \rangle$ on the applied force f_0 for (top) low values and (bottom) wide range of values of applied force

A preliminary study of mould filling, including edge effects, has been undertaken. The geometry considered is presented schematically in Fig. 5, and consists of a two-dimensional rectangular channel of length 220 mm and width 80 mm. The preform, comprised of an isotropic porous material, occupies the entire channel except for a 5 mm gap at one side. Numerical simulations have been undertaken for a number of preform porosities, ranging from $\varepsilon = 0.3$ to 0.9. The fluid (liquid resin) is injected into one end of the channel, ahead of a piston moving at 0.1 m s^{-1}. The resin has a density $\rho = 1200$ kg m^{-3} and a dynamic viscosity $\mu = 0.2$ Pa s.

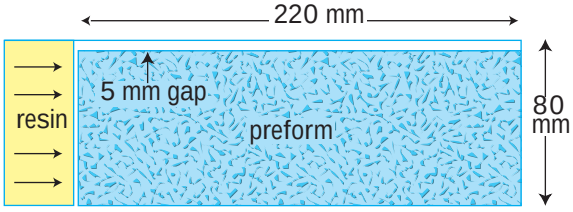


Fig. 5 – Schematic diagram of geometry used for the resin transfer moulding study (The left side section of the resin has been truncated for clarity)

For the SPH simulation of the mould filling process, the porous preform was constructed by randomly removing particles from an array of $N_f = 16{,}200$ particles, the resulting number of fixed particles being given by $(1-\varepsilon)N_f$. In the initial state, the preform was considered to be empty, containing only fixed particles (with no enclosed mobile SPH particles). The resin was modelled by 11,770 SPH particles, corresponding to a resolution of 1 particle/mm. A moving wall condition was applied at the left boundary, which forced the mobile SPH particles into the inter-particle spaces of the porous preform. Fixed wall conditions are applied at the horizontal upper and lower boundaries.

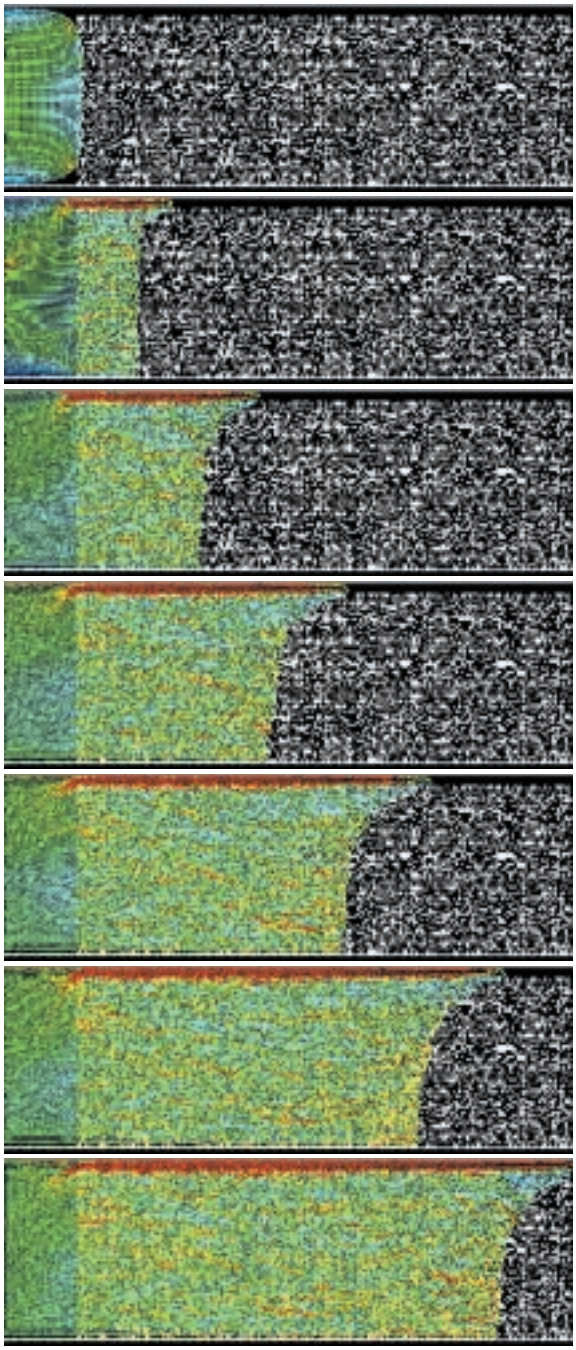


Fig. 6 – SPH particle locations, coloured by velocity, at selected times (top to bottom: 0.32, 0.68, 1.03, 1.39, 1.74, 2.10, 2.45 s) during the mould filling for a preform porosity of $\varepsilon = 0.5$ (Flow is from left to right)

Figure 6 shows, for a preform porosity of $\varepsilon = 0.5$, the location of the fluid particles at selected times during the mould filling. The filling of the mould is observed to give rise to two-dimensional flow. The free surface of the resin is initially convex, due to the Poiseuille-like flow that is established in the channel before the resin reaches the preform. As the resin enters the region partially filled by the preform, the free surface rapidly becomes concave in shape, due to the permeability in the preform being lower than in the gap. As the resin is forced further into the channel, its free surface appears to reach a steady-state shape. This

predicted behaviour of the resin flow in the unsaturated porous preform material appears to be in qualitatively good agreement with available experimental data [10].

COMPUTATIONAL REQUIREMENTS

Like most particle-based methods, SPH simulations generally require significant computational resources, particularly for large-scale 3D applications with complex physical modelling. As an example, the modest-size 3D die casting simulation described in this article was performed on a single 667 MHz Alpha ev67 processor of a Compaq ES40 system, and required about 90 MB and 3.5 days to complete.

Two complementary approaches are being pursued to alleviate computational resource requirements: improved numerical algorithms (eg fully incompressible method, implicit time integration) and code parallelisation. Two different approaches to parallel simulation are being considered:

- *parallel computation* involving extensive code parallelisation to distribute each individual simulation over multiple processors. This can be achieved in a generally straightforward manner for a particle-based method such as SPH, and is presently under consideration using message passing (MPI library). Such a technique will be essential for the simulation of full-scale industrial applications.
- *computation in parallel* involving the distribution across multiple processors of several instances of the same (sequential) computation with different input data. Such an approach, which requires only minor code modification, has been used for the above-mentioned parametric studies of flow in a saturated porous medium. The Swiss-T1 parallel machine (during its installation and testing phase when only the Fast Ethernet interconnect was available) was employed for these studies, generally using 16 processors. Due to the high level of efficiency of these essentially *embarrassingly parallel* computations, a very high throughput was achieved.

For large-scale industrial simulations, availability of computer resources may impose limits on the simulation size; such limitations, however, tend to diminish rapidly with the ever-increasing computational power available. It is generally considered that limitations associated with realistic physical modelling are more severe. While complex physical modelling has not been considered in detail in this article, additional investigations (see eg [3],[5]-[7]) indicate that the SPH method provides a convenient framework for implementation of the necessary advanced physical models.

CONCLUSIONS

The two examples considered in the present study illustrate that the SPH method can be effectively employed for the numerical simulation of complex industrial applications. Such a meshless Lagrangian method excels in the simulation of the complex free surfaces that arise in HPDC due to the high velocity injection of molten metal into a die cavity. The treatment of an extremely fine-detail geometry, such as necessary in a mesoscopic-level simulation of the flow in a porous medium and its application to RTM, has also been demonstrated using SPH.

The computational resources required for this particle-based method may be substantially greater than for conventional mesh-based approaches. However, this is more than counter-balanced by the greater flexibility of the SPH method to handle in a simple manner not only complex geometries but realistic physical modelling.

ACKNOWLEDGEMENTS

The authors wish to thank François Debroux for his assistance in the visualisation of the 3D HPDC simulation results. The Service Informatique Central of the Ecole Polytechnique Fédérale de Lausanne is acknowledged for providing access to the Swiss-T1 machine. The HPDC study was funded by the Cooperative Research Centre for Alloy and Solidification Technology (CAST).

Further information and application examples regarding this work can be found at *www.cmis.csiro.au/cfd* .

REFERENCES

[1] J.J. Monaghan, *Smoothed particle hydrodynamics,* Annual Review of Astronomy and Astrophysics, **30** (1992) 543-574.
[2] J.J. Monaghan, *Simulating free surface flows with SPH*, Journal of Computational Physics, **110** (1994) 399-406.
[3] P.W. Cleary, *Modelling confined multi-material heat and mass flows using SPH*, Applied Mathematical Modelling, **22** (1998) 981-993.
[4] P.W. Cleary and J.J. Monaghan, *Boundary interactions and transition to turbulence for standard CFD problems using SPH*, Proc. 6th International Computational Techniques and Applications Conference, Canberra (1993) 157-165.
[5] P.W. Cleary and J. Ha, *Effect of heat transfer and solidification on high pressure die casting*, Proc. 13th Australasian Fluid Mechanics Conference, Melbourne (1998) 679-682.
[6] P.W. Cleary, J. Ha and V. Ahuja, *High pressure die casting simulation using smoothed particle hydrodynamics*, International Journal of Cast Metals Research, **12** (2000) 335-355.
[7] W. Thorpe, V. Ahuja, M. Jahedi, P. Cleary, J. Ha and N. Stokes, *Simulation of fluid flow within the die cavity in high pressure die casting using smoothed particle hydrodynamics*, Proc. 20th International Die Casting Congress and Exposition, Cleveland (1999) Paper T99-014.
[8] K. Potter, *Resin transfer moulding*, Chapman & Hall (1997).
[9] M. Kaviany, *Principles of heat transfer in porous media*, Springer-Verlag, 2nd ed. (1995).
[10] W.-B. Young and C.-L. Lai, *Analysis of the edge effect in resin transfer moulding*, Composites Part A, **28A** (1997) 817-822. ■

Visualization Tools and Environments for Very Large Data

Jean M. Favre, Sathya Krishnamurthy, CSCS, Swiss Center for Scientific Computing

La facilité avec laquelle les simulations et expériences génèrent des données a souvent surpassé la capacité des outils de représentation graphique disponibles dans les centres de calcul à haute performance. Cela est en train de changer, alors que la visualisation scientifique devient un élément à part entière du calcul à haute performance. Dans cet article, nous étudions quelques systèmes et techniques disponibles pour la compréhension et la représentation de bases de données très larges et décrivons les meilleures méthodes de la chaîne de production de la visualisation scientifique. La gestion de données de la part des serveurs de fichiers, le calcul des représentations graphiques, et le rendu 3D avancé forment les éléments essentiels de cette activité.

The facility with which computational scientists and experiments can generate data has often outpaced the capacity of the post-processing tools available in High Performance Computing environments. This is changing however, as scientific visualization is becoming itself an HPC activity. In this article, we wish to review systems and techniques which are becoming feasible for the ingestion and graphical display of very large data sets. Based on our experience and a review of current practices, an attempt is made to describe the best visualization methods. From data management issues on the files servers, to information extraction, to data processing and advanced 3D graphics, we propose a journey through the data production and visualization events, reviewing some of the latest technologies available.

Introduction

Data visualization is reaching an all-time high. How many times have we heard this statement in the last few years? Year 2000 is already marked by the first publicized 11.5 billion cell visualization [1], and the trend shows no signs of slowing down. Large-scale environments that attempt to optimize the use of resources at all levels of the data visualization chain are supplanting the traditional desktop graphics workstation. File access and data retrieval are the first essential links of this infrastructure. When a dataset does not fit in local memory, or in the local disk space available, a set of non-trivial techniques must be put in place to accelerate its retrieval. Second, comes the information extraction, the essence of data visualization. Parallel or distributed implementations are now possible, paving the way towards computational-steering architectures.

Parallelism has also moved to the graphics hardware systems, with the multi-pipe rendering servers and the large displays. These computational servers can offer a tight coupling between simulation and visualization, or they can be used to generate remote graphics, leaving no much work on the client side but the display of static images. To offer interactivity using remote rendering requires a minimum of 10 frames per second, using potentially over 30 Mbytes per second of bandwidth for full-screen uncompressed images. Generally though, visualization must provide the interactive means to explore data and their representations. To accelerate this process, we will see how improvements in 3D graphics technology are contributing to handling objects made of millions of graphics primitives.

The images and movies produced during visualization remain static snapshots of the dataset under study. An additional added value of a visualization environment is its ability to reproduce results, and to facilitate side-by-side comparisons. To this effect, we will see the role played by scripting languages in visualization. They provide the indispensable programs necessary such that the deployment of a complete visualization chain becomes less tedious.

File, data and process management

The need for some form of Data Management for efficient data access becomes obvious when dealing with datasets beyond a few hundreds of megabytes. Data access should be optimized on a variable basis, on a computational block basis, and at specific time-steps for transient solutions analysis. Such access should be provided without the need for searching. It should provide support for self-descriptive and context-free data manipulation. For example, MemCom [2] is a data management system that has been specifically designed for engineering applications, like computational solid mechanics, computational fluid dynamics, and coupled multi-disciplinary applications. MemCom consists of a wide range of functions for data definition and data manipulation, as well as auxiliary tools. The data manipulation functions are not tied to specific applications and APIs for C, C++, Fortran, as well as a CORBA interface exist. Access to large collections of data has been optimized in MemCom, and it fits very well the concept of *load-on-demand*.

A user-defined reader created to provide a native interface to MemCom databases for the EnSight software (*www.ensight.com*) was created at the Swiss Center for Scientific Computing (CSCS) [3], and took advantage of the very explicit and clear hierarchies of template sub-routines specified by the EnSight environment which matches extremely well the part, block, variable, and time-step access sub-routines offered by MemCom.

OPTIMIZED FILE ACCESS

Waiting to load data into memory when it is needed is just one facet of optimizing access. To handle very large data sequences, a user should also take into account the non-uniform access times of different storage technologies. If data have to be retrieved across a network from magnetic storage, it is advisable to stage the data, i.e. to perform a pre-fetch operation that will get the files ready when they are required. We have run such a visualization script at CSCS for the simulation of a large laminar-to-turbulent transition in a supersonic boundary layer [4, 5]. A file server moves data files from a slow storage area (Timberline, Redwood or Eagle tape drives at a maximum speed of 12 Mbytes/sec) to a RAID5 disk array (disks can now be found which operate between 120 and 350 Mbytes/sec). To optimize data reading without filling up the RAM, files can also be memory-mapped. We have made use of special features of the AVS/Express software (*www.avs.com*) to accomplish this. Often, though, many UNIX shell commands are also necessary to facilitate file management, and it is important to have the ability to execute and synchronize shells scripts from within the visualization softwares.

DATA-STREAMING AND DATA PARALLELISM

With data larger than RAM, there is the option of streaming the data, i.e. partitioning it into smaller and independent subsets that can be processed one at a time. The pv3 system pioneered this technique in an application allowing a network of heterogeneous computers to process data in chunks, sending them to a collector process responsible for gathering the final image. The Vtk software, an object-oriented library of data and visualization classes improves on this model by providing application-independent components that support multi-threaded Data Streaming [6]. However, it is currently limited to handle only structured data (images or volumes) which are cleanly separable, and where the result is independent of the number and size of the data chunks.

Data parallelism is essential for contemporaneously processing independent subsets of data. In Vtk, the implementation of data parallelism does not require any additional changes to the toolkit. To write a program that expresses data parallelism:

- copies of the same modules are run in each process,
- these data parallel modules process independent subsets of data,
- the results of the last data parallel module are usually merged to create a single process result.

TASK PARALLELISM

Client-server applications are the first examples of distributed computing. They are generally found under the following three scenarios.

TRANSFERRING DATA TO THE CLIENT

The network serves as a data communication medium where the data is tagged to use the MIME-typing concept to divert it to a particular application. Here, the software must be installed and run on every user's desk. The idea has been explored in the Vis5D system [7]. Meteorological data can be tagged with MIME-type 'application/vis5d', and a browser configured to pass data of that MIME-type to Vis5D. The user then selects the options in Vis5D to create whatever visualization is required.

TRANSFERRING SOFTWARE

This is a Java based approach whose origins are traced to early examples such as the NPAC Visible Human Viewer. These examples were designed for cases where the data was closely associated with the software, and indeed located on the same server. However for many applications this is inappropriate - the data usually is associated with the user rather than the software provider.

TRANSFERRING GRAPHICS

This idea originated with the server-based systems. It is a very general approach, and several similar systems have been proposed. A logical extension of this approach in the case of Modular environments is to split their operation into two parts: a windowed front-end that can execute on the client, and a pipeline from the server, which connects with the client. EnSight, IRIS Explorer, IBM Data Explorer etc. use this principle. Most softwares use the X-server and the Java networking methods for this communication. Others (e.g. EnSight) connect through a proprietary network connection mechanism. Another interesting example is the Visapult [8] system; it has a decoupled computational back-end that performs parallel software volume rendering using MPI and a front-viewer communicating over a custom IPC layer built using TCP sockets. The viewer is itself a parallel application programmed with OpenGL and pthreads.

In another development, the Vtk visualization library is also extended to support multiple processes [9]. A system process object encodes whether the system is distributed (via MPI), or shared memory (via pthreads or sprocs). In many of the recent visualization softwares, execution is based on the data-flow approach. To make parsimonious use of the computing resources when confronted with large data, it is then recommended to drive the execution in an event-driven fashion. Multiple visualization parameters and queries can then be grouped before requesting the data required. This is in contrast to environments which are demand-driven, and whose performance under heavy loads suffers from too many update requests.

DOWN THE GRAPHICS PIPELINE

Visualization is traditionally achieved by the creation of geometric representations, or of pixel-based imagery. Many opportunities exist to optimize this graphical data display. OpenGL offers now many features useful for the display of engineering data. Vertex arrays for example, are a recent feature (OGL 1.1) which allows to send entire list of primitives (triangles, or quadrilaterals for example) by a single call to the rendering API, specifying at once, all the pointers where coordinates, colors, textures can be found in block-optimized regions of memory on the client side. Others ways to minimize data exchange between the OpenGL client and its server process are display lists, or textures objects.

To improve interactivity, one may also use different *levels of detail* (LOD) — bounding boxes, clouds of points, surface decimations with a smaller number of primitive cells or higher order representations such as parametric surface hulls — for interactive display, switching then to full resolution graphics for static image production.

VOLUME RENDERING

The visualization technique where the best contributions from hardware rendering and the newest features of OpenGL are evident is Volume Rendering. It is a technique used to display 3-D data without the intermediate step of deriving a geometric representation like a solid surface. The graphical primitives being characteristic for this technique are called voxels, derived from volume element and analog to the pixel. However, voxels describe more than just color, and in fact can represent opacity or shading parameters as well. The data structures employed to manipulate volumetric data come in two flavors:

- the data may be stored as a 3-D grid, or
- it may be handled as a stack of 2-D images.

The computational demands of volume rendering require the use of a high degree of hardware parallelism. In addition, volume rendering is a memory intensive operation; the design of the memory system is critical in volume rendering architectures. Texture mapping hardware, which is a common feature of modern 3D graphics accelerators, can be exploited for volume rendering by applying a method called planar texture resampling. The volume is stored in 3D texture memory and resampled during rendering by extracting textured planes parallel to the image plane. Lookup tables map density to RGBA color and opacity. The resulting texture images are combined in back_to_front visibility order using compositing. There are certain limitations that are encountered while going from 2D-texture to the 3-D texture approach:

- the memory required by the triple image stack is a factor of three larger than the original data set, which can be critical for large data sets as they are common in medical imaging or microscopy,
- the geometry sampling of the volume must be aligned with the 2-D textures concerning the depth, i.e. arbitrary surfaces constructed from a triangle mesh cannot easily be colored depending on the properties of a surrounding volume.

For this reason, advanced rendering architectures support hardware implementations of 3-D textures (e.g. Mitsubishi's VolumePro Hardware board [10]). The correspondence between the volume to be rendered and the 3-D texture is obvious. Any 3-D surface can serve as a sampling device to monitor the coloring of a volumetric property. I.e., the final coloring of the geometry reflects the result of the intersection with the texture. Following this principle, 3-D texture mapping is a fast, accurate and flexible technique for looking at volumetric data. It is now part of the OpenGL 1.2 specification [11].

MULTI-PIPE RENDERING ENVIRONMENTS

Going one step further is to include parallelism into the graphics rendering hardware, as found in the Silicon Graphics Onyx2. Parallelism is now found in the geometry engines and raster managers responsible for lighting calculations, geometric transformations, image-processing functions such as convolution and histogram equalization, and represent a more effective approach than multiple CPUs. Pixel operations, including z-buffer testing, color and transparency blending, texture mapping, and multisample anti-aliasing are then possible at interactive rates. While the cost of parallel graphics hardware remains high, there are several utilities developed to deliver these advanced visualization capabilities and performance to the desktop. The OpenGL Vizserver enables the SGI Onyx2 visualization workstation as a graphics server, transmitting to remote desktops, over standard high-speed networks, the compressed images from the Onyx2 frame buffer [12]. A similar system developed at Sandia National Laboratories provides a four-way parallel remote console system, by splitting, compressing and delivering the high-resolution displays of a visualization server over an ATM network [13].

Transition for code development between workstation and multi-pipe renderings architectures can be done with the Multi-pipe Utility *www.devprg.sgi.de.devtools/tools/MPU*), a programming interface for OpenGL. It allows the development for large-scale environments such as CAVEs, Power-Walls, ImmersaDesks and the like. As an example, it allows a multiprocessor, multi-pipe application to be developed and tested on a single-graphics board desktop workstation, without recompilation. Another library developed at Lawrence Livermore National Laboratories is the Virtual Display Library, VDL [14]. VDL provides a simple API for threaded, multi-pipe rendering through basic display management services, such as window and thread creation, and double-buffer window synchronization.

Programming these graphics supercomputers is thus becoming much easier, and more affordable, since their power can be harnessed and shared by remote users.

PROGRAMMING SUPPORT FOR SCRIPTING

Controlling the execution of visualization softwares remains an important issue. Many softwares were traditionally created for interactive usage. Journaling is often available,

recording every mouse click and menu selection to a file for later replay, but this often creates a very large amount of transient UI events that can be difficult to edit.

Scripting is important for rapid prototyping before compiling an application, for batch mode submission when software rendering is possible, or to be able to repeat the same image creation with a new dataset. The programmer can be confronted with proprietary scripting languages (e.g. AVS5, AVS/Express, EnSight) that are more difficult to comprehend. Other approaches offer interface via a language that is more widespread like Tcl (e.g. Vtk, ICEM Visual3). Important in all scripting support, is the ability to write loops and control flow structures, and to favor code reuse for callable macros containing common sets of instructions. A difficulty of editing journal files is also that they can be very state-dependent. Execution must be carried in a specific order, often dependent also on the number and names of the variables created. Finally, not all commands available through the User Interface are always possible to script. For example, in EnSight, the selection of parts can be done via the common UNIX syntax of wildcard naming. Yet, this is translated into a script command explicitly selecting parts by names, and it cannot be programmed directly via a helper application. When one of the difficulties of handling large data is also due to the large number of computational blocks and their derived graphics representations, using a helper application to automatically write the script is also critical. A program of this kind was created to support visualization of MemCom databases in aerodynamics [3]. The NSMB code development for example generates between 100-1000 blocks for detailed simulations of aircrafts [15]. Surface extraction and other routine data extraction must be completely automatized to save time and reduce scripting errors (fig. 1).

Fig 1 – Flow simulation around the F18 of the Swiss Army, by Alain Gehri and Jan Vos, CFS Engineering, Lausanne. An EnSight script automatically queried a MemCom database storing 194 computational zones for 4.4 million nodes. The graphics is delivered via a 100Mbytes/sec line from a multi-CPU file server to a graphics workstation. Interactive queries are carried over TCP sockets.

CONCLUSION

Scientific visualization is no longer constrained to small data handling. From PC-based consumer boards which can process volumetric data in real-time, to large HPC environments including data access, visualization extraction, and graphical representation, all processed in parallel, the environments available today can easily process distributed data and deliver 3D results to remote desktops. A mix of expensive hardware and many advanced software developments can make the visualization of tera-bytes of data a common feat. Yet, other emerging activities will clearly complement the advanced environments of today. Feature detection for automatic searches through very large Fluid Dynamics databases is becoming available in commercial products for the identification of vortex cores, shock waves, separation lines and surface flow topology [16]. These will replace the very tedious and error-prone interactive queries that can render visualization systems completely ineffective. Data Mining and Knowledge Discovery are also contributing to gaining insight from large protein databases and others huge data depositories. In this context, and for very large data handling, visualization will perhaps turn back to a batch-oriented activity trained to deliver only the quintessential features of large data banks.

REFERENCES

[1] *www.ensight.com/11.5billioncells.htm*
[2] *www.smr.ch/Products/memcom.html*
[3] Jean M. Favre, *An EnSight native reader interface for MemCom databases*, Swiss-Tx project, in preparation.
[4] Jean M. Favre, *Towards Efficient Visualization Support for Single-block and Multi-block Datasets*, IEEE Visualization 1997 Proceedings.
[5] Ch. Mielke, et al., *Laminar-turbulent transition in a supersonic boundary layer*, Phys. Fluids, Vol. 11, Nr. 9, 1999, pp. S10
[6] C. Law et al., *A Multi-Threaded Streaming Pipeline Architecture for Large Structured Meshes*, IEEE Visualization 1999 Proceedings.
[7] Ken Brodlie, et al., *Harnessing the Web for Scientific Visualization*, VisFiles By Bill Hibbard, February, 2000
[8] Wes Bethel, *Visualization Dot Com*, Visualization Viewpoints, IEEE CG&A, vol. 20, no. 3, August 2000.
[9] J. Ahrens et al., *A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets*, Los Alamos National Laboratory, Technical Report #LAUR-00-1620.
[10] Hanspeter Pfister, *Architectures for Real-time Volume Rendering*, Future Generation Systems, Vol. 15, 1999
[11] *www.opengl.org/Documentation/OpenGL12.html*
[12] *www.sgi.com/software/vizserver/*
[13] J. Friesen and T. Tarman, *Remote High-Performance Visualization and Collaboration*, IEEE CG&A, vol. 20, no. 4, August 2000.
[14] D. Schikore et al., *High-resolution Multiprojector Display Walls*, IEEE CG&A, vol. 20, no. 4, August 2000.
[15] *www.cerfacs.fr/cfd/research/aerodyn/aerodyn_home_page.html#NSMB*
[16] R. Haimes and D. Kenwright, *On the Velocity Gradient Tensor and Fluid Feature Extraction*, AIAA Conference, Norfolk, VA, July 1999, 99-3288 ■

The Fluent 5 Benchmark Results on the Swiss-T1

Mark L. Sawley, Visiting Scientist, Fluid Mechanics Laboratory, DGM - EPFL & Trach-Minh Tran, Service informatique central EPFL & Tom L. Tysinger, Fluent Inc. USA

Les résultats de performance obtenus pour la suite de benchmarks **Fluent 5** *sur le Swiss-T1 sont présentés et analysés. Ces résultats démontrent la capacité d'un tel système parallèle de calculer un éventail de problèmes d'écoulements industriels. On observe que l'utilisation du réseau d'interconnexion performant TNet fournit une amélioration de performance significative comparée à un réseau standard Fast Ethernet.*

The performance results obtained for the *Fluent 5* benchmark suite on the Swiss-T1 are presented and analysed. These results demonstrate the capacity of such a commodity cluster to compute a wide range of industrial flow problems. Use of the high-bandwidth, low-latency TNet interconnect is observed to provide significant performance improvement compared to a standard Fast Ethernet switched network.

Introduction

Performance benchmarking is commonly used to assess the ability of a computer system to undertake particular tasks, such as computation, input/output and visualisation [1]. Frequently, benchmark results obtained for relatively simple computational tasks (eg, solving a dense system of linear equations, as in the LINPACK benchmark [2]) are extrapolated to provide performance estimation for significantly more complex problems (eg complex multi-physics simulations). Clearly, the more representative is the benchmark test case of the real problems to be solved, the more useful the benchmark results will be.

Fluent 5 is a state-of-the-art commercial software package for modelling fluid flow and heat transfer in complex geometries [3]. The governing equations are resolved on an unstructured mesh. A number of different mesh types are supported (2D triangular / quadrilateral, 3D tetrahedral / hexahedral / pyramid / wedge, as well as hybrid meshes and mesh refinement). A wide variety of numerical (segregated implicit / coupled explicit / coupled implicit) and physical models (including incompressible / compressible, inviscid / laminar / turbulent, convective / conductive / radiative heat transfer, chemical reactions / combustion, two-phase and free-surface flows) are available to adapt to the extensive range of simulation problems that can be addressed. Due to this large degree of flexibility and applicability, *Fluent 5* has been chosen as a standard software package at the EPF-Lausanne; it is installed on a number of computer systems, and used by researchers in a number of different laboratories.

Fluent 5 is written in the C language, and makes use of such features as dynamic memory allocation, efficient data structures and flexible solver control. A client / server architecture permits the separation of interactive control and visualisation (generally performed on a desktop workstation) from the solver computation (which can be performed on a remote compute server). In addition, computations can be performed in parallel; a distributed-memory model based on domain decomposition is used, with message passing via the MPI library for inter-processor communication. Such a model can be employed on a wide range of computer platforms, including both shared-memory and distributed-memory systems.

To obtain a valid assessment of the performance of such an extensive software package on different computer platforms is not a simple task. Indeed, any performance evaluation must address the specific needs of the particular user. To assist in such an assessment, a series of benchmark test cases has been defined by Fluent Inc. [4]. These test cases consist of industrial problems that have been chosen, not because they perform well using *Fluent 5*, but rather because they represent a broad range of simulations typical of those investigated by the user community.

The purpose of the study reported here is to assess *Fluent 5* on the Swiss-T1 parallel commodity cluster installed at the EPF-Lausanne. The compute nodes of the Swiss-T1 consist of 32 Compaq DS20e bi-processor (500 MHz Alpha EV6, 2*L1 cache of 64 kB) boxes, each with an L2 cache of 4 MB and a main memory of 1 GB. The boxes are connected via two networks: a conventional Fast Ethernet switched network, and a high-bandwidth, low-latency TNet network developed by Supercomputing Systems, Zurich. Compaq Tru64 UNIX 5.0 is the operating system used on the compute nodes, while the Codine load management software of Gridware Inc. is used for the submission of batch jobs. (See [5] for more details regarding the Swiss-T1.)

Fluent 5 has been ported to the Swiss-T1 by re-compiling using the appropriate MPI include file and linking with the necessary libraries. *Fluent 5* can run on the Swiss-T1 using either the freely-available MPICH library [6] via the Fast Ethernet network, or a proprietary MPI library implemented on top of the FCI communication library which uses TNet. This enables a direct comparison between the performance obtained on a commodity cluster having communication characteristics similar to a Beowulf cluster [7] and that of a system with an enhanced communication interconnect.

Class	Test case	Cells	Mesh	Models	Solver	Description
small	FL5S1	32,000	hexahedral	ke	segregated implicit	turbulent flow in a bend
	FL5S2	32,000	hexahedral	ke	coupled implicit	turbulent flow in a bend
	FL5S3	89,856	hexahedral	ke	coupled implicit	flow in a compressor, rotor 37
medium	FL5M1	155,188	tetrahedral	ke 6spe react DPM P1	segregated implicit	coal combustion in a boiler, with particle tracking
	FL5M2	242,782	hybrid, hanging-node	ke	segregated implicit	turbulent flow in an engine valve port
	FL5M3	352,800	hexahedral	ke 6spe react	segregated implicit	combustion in a high velocity burner
large	FL5L1	847,746	hexahedral	ke	coupled explicit	transonic flow around a fighter
	FL5L2	3,618,080	hybrid	RNG ke	segregated implicit	external aerodynamics around a car body
	FL5L3	9,792,512	hexahedral	RSM	segregated implicit	turbulent flow in a transition duct

Table 1 – Detailed properties of the nine benchmark test cases

THE FLUENT 5 BENCHMARKS

The test cases defined in the *Fluent 5* benchmark suite cover a wide spectrum of application problems, as shown pictorially in the figure on the cover of this issue. These test cases can be characterised by:

- problem size (number of mesh cells);
- mesh type (hexahedral, tetrahedral, hybrid);
- numerical method (segregated/coupled solver);
- physical modelling (turbulence, combustion).

The test cases are divided into 3 classes: small, medium and large, depending on the number of mesh cells employed; each class contains 3 different cases. A detailed presentation of the nine benchmark test cases is given in Table 1.

Due to the large computational requirements of these application problems, a fixed number of iterations was generally used for the benchmark results. Care was taken to select an iteration interval representative of the overall performance of the flow simulation. To facilitate the comparison of different computer systems, Fluent Inc. has defined the following global performance measures:

Rating – Rating is the primary metric used, and is defined as the number of times the test case can be run on a given machine (in sequence) in a 24-hour period. It is computed by dividing the number of seconds in a day by the number of seconds required to run the benchmark. A higher rating means faster performance.

Redzone – The redzone is the region of greater than 66% efficiency. It is arbitrarily selected as the region that is favourable to perform a production run (to limit the amount of time a processor is idle while waiting, for example, for the communication phase of the calculation to complete).

Redzone Rating – The redzone rating is defined as the maximum rating achieved with a parallel run of a given benchmark and platform that occurs within the redzone.

Peak Rating – The peak rating is defined as the maximum rating achieved for a parallel run of a given benchmark and platform with any number of processors.

For convenience, the benchmark suite is run via a standard shell script. Various output files are generated, such as a log file and a results file containing the above-mentioned performance measures. While measurements are also made of input/output performance, in this study we have concentrated solely on computational performance. The performance quoted in this article is thus for the core solver only, and neglects overhead in the simulation due to job spawning on multiple processors, or reading to / writing from disk.

SWISS-T1 PERFORMANCE RESULTS

The benchmark results on the Swiss-T1 reported here were obtained during the first week of August 2000, using version 5.4.7 of *Fluent 5*. While the machine was fully functional at this time, some of the system software (such as the Codine batch system) was still in an experimental state. In addition, while significant performance optimization of the MPI/FCI library had already been undertaken, some planned additional optimizations (particularly for global communications) were still to be completed.

The performance rating measured for each of the nine test cases using both TNet and Fast Ethernet are presented in Fig. 2. The Fast Ethernet interconnect was used for up to 32 processors, while up to 64 processors were used with

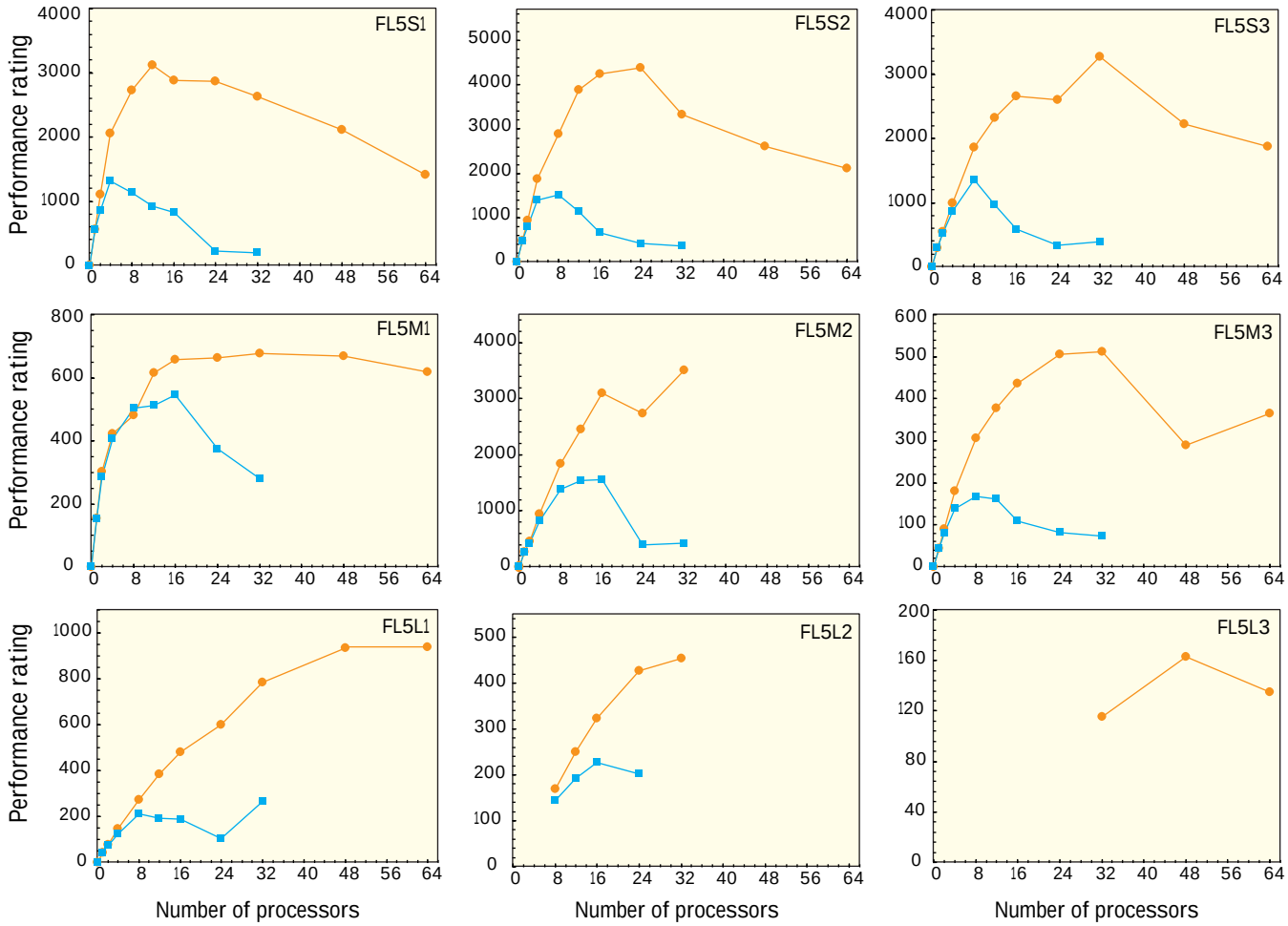


Fig. 2 – Performance rating measured for the nine test cases for different number of processors, using TNet (orange circles) and Fast Ethernet (blue squares)

TNet. For the largest problem sizes (FL5L2 and FL5L3), the memory requirement forced swapping to disk when a small number of processors was used; since this resulted in significant performance degradation, the results for these cases are therefore not taken into consideration. The values of the above-defined global performance measures determined for the Swiss-T1 using both TNet and Fast Ethernet are presented in Table 2.

A number of observations can be made from the performance results of Fig. 2 and Table 2.

The performance of *Fluent 5* is strongly dependent on the characteristics of the test case, in particular, the number of mesh cells and the complexity of the physical modelling.

The number of processors required to obtain peak performance increases - and consequently the scalability improves - as the problem size increases, due to the increased ratio of computation to communication. An associated increase in the redzone is also generally observed with increasing problem size.

For the small class test cases, peak performance is achieved for a relatively low number of processors. Increasing the number of processors beyond this value results in a substantial decrease in performance, associated in part with the increased overhead of global communications.

For the large class test cases, performance scalability is obtained using TNet for a substantial range of the available processors. (The decrease in performance for the FL5L3 test case using 64 processors is attributed to a non-optimal mesh partitioning for this particular case.)

Peak performance for test case FL5M1 is obtained for a relatively small number of processors. This case couples a continuous gas phase calculation with a discrete phase (particle) calculation. Contrary to the continuous phase (as described above), the discrete phase is parallelised in *Fluent 5* using a shared-memory model. This provides for only very limited parallel speedup on the Swiss-T1 (which is based on bi-processor boxes), and therefore severely limits the maximum parallel performance that can be achieved by *Fluent 5* for this test case. (A distributed-memory parallelisation of the discrete phase is planned for a future code version.)

Comparing test cases FL5S1 and FL5S2 shows that the coupled numerical method performs better than the segregated method for this relatively simple problem.

For all test cases, the use of the higher performance TNet interconnect results in a significant improvement of the benchmark performance; this improvement is generally observed even for a small number of processors (ie nproc ≥ 4). For example, for the small class test cases, peak performance

Test case	1 cpu	TNet (MPI/FCI)				Fast Ethernet (MPICH)			
		Redzone		Peak		Redzone		Peak	
FL5S1	564.7	2054.6	4 91%	3125.8	12 46%	848.8	2 75%	1316.5	4 58%
FL5S2	484.8	2889.6	8 84%	4379.6	24 33%	1399.3	4 72%	1515.9	8 44%
FL5S3	293.8	1863.1	8 79%	3275.8	32 35%	869.9	4 74%	1359.6	8 58%
FL5M1	152.8	421.6	4 69%	675.8	32 14%	284.8	2 93%	546.5	16 22%
FL5M2	261.3	3094.0	16 74%	3501.5	32 42%	825.8	4 79%	1550.5	16 37%
FL5M3	44.5	377.9	12 71%	511.8	32 36%	138.1	4 78%	167.7	8 47%
FL5L1	41.3	481.5	16 73%	937.3	64 35%	123.7	4 75%	266.6	32 20%
FL5L2	-	-	- -	453.4	32 -	-	- -	226.6	16 -
FL5L3	-	-	- -	162.8	48 -	-	- -	-	- -

Table 2 – Global performance measures determined for each of the nine test cases using both TNet and Fast Ethernet. In the small boxes are noted the number of processors required to obtain the redzone or peak ratings and the corresponding parallel efficiency.

occurs using TNet for nproc ≤ 32, but for a significantly lower number, nproc ≤ 8, using Fast Ethernet. The higher performance using TNet is also reflected in a considerably larger redzone for all test cases.

Finally, it should be noted that excellent serial performance is obtained on the Swiss-T1 (see results presented in [4]) through the use of high-performance RISC processors, for which *Fluent 5* is particularly well suited.

CONCLUSIONS

Running the *Fluent 5* benchmark suite on the Swiss-T1 has enabled an evaluation of such a commodity cluster to compute a wide range of flow applications. Good performance scalability has been demonstrated for all test cases, provided that the number of processors employed does not exceed a problem-dependent optimal value. It has been demonstrated that small problems can be computed an order-of-magnitude faster in parallel on the Swiss-T1 than by serial computation. Large problems that can not be run on a single-processor workstation (due to memory limitations) can be computed efficiently using a large number (up to 64) of processors.

The use of high-performance processors on the Swiss-T1 to achieve excellent single-processor performance implies that a high-performance network is required for balanced parallel computation. The results obtained in the present study indicate that a Fast Ethernet interconnect (such as commonly used in Beowulf clusters [7]) does not provide adequate communication performance for a significantly large number of processors. The superior capability of the TNet interconnect is reflected in a substantial improvement of performance for the entire *Fluent 5* benchmark suite.

REFERENCES

[1] R.W. Hockney, *The science of computer benchmarking*, SIAM (1996).

[2] J.J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart, *LINPACK User's Guide*, SIAM (1979).

[3] *Fluent 5 User's Guide*, Fluent Inc. (1998); see also: *www.fluent.com/software/fluent*

[4] For full details regarding the Fluent 5 benchmark definitions and results, see: *www.fluent.com/software/fluent/fl5bench*

[5] P. Kuonen and R. Gruber, *Parallel computer architectures for commodity computing and the Swiss-T1 machine*, EPFL Supercomputing Review, **11** (1999) 3-11; see: *sic.epfl.ch/publications/SCR99*; see also: *sewww.epfl.ch/SIC/SE/servcentraux/generalites.html*

[6] W. Gropp, E.L. Lusk, N. Doss and A. Skjellum, *A high-performance, portable implementation of the MPI message passing interface standard*, Parallel Computing, **22** (1996) 789-828; see: *www.mcs.anl.gov/mpi/mpich*

[7] For detailed information regarding Beowulf clusters, see: *www.beowulf.org* ■

SFIO, Parallel File Striping for MPI-I/O

Emin Gabrielyan, EPFL, Computer Science Dept. Peripheral Systems Lab., Emin.Gabrielyan@epfl.ch

Cet article présente l'architecture d'un système de fichiers distribué (SFIO) pour la gestion des entrées/sorties parallèles dans un environment MPI. Différentes techniques d'optimization des communications et d'accès aux disques sont présentées. A l'aide de types dérivés MPI, on peut transmettre sur le réseau des données fragmentées à écrire sur disque à l'aide d'une seule commande MPI. Nous présentons les performances d'entrée/sorties du système de fichiers distribué sur le superordinateur Swiss-Tx formé de noeuds de calcul et E/S de type Compaq Alpha.

This paper presents the design and evaluation of a Striped File I/O (SFIO) library for parallel I/O in an MPI environment. We present techniques for optimizing communications and disk accesses for small striping factors. Using MPI derived datatype capabilities, we transmit fragmented data over the network by single MPI transfers. We present first results regarding the I/O performance of the SFIO library on Compaq Alpha clusters, both for the Fast Ethernet and for the TNet Communication networks.

Motivation/Introduction

For I/O bound parallel applications, parallel file striping is an alternative to Storage Area Networks (SAN). In particular, parallel file striping offers high throughput I/O capabilities at a much cheaper price, since it does not require a special network for accessing the mass storage sub-system [6].

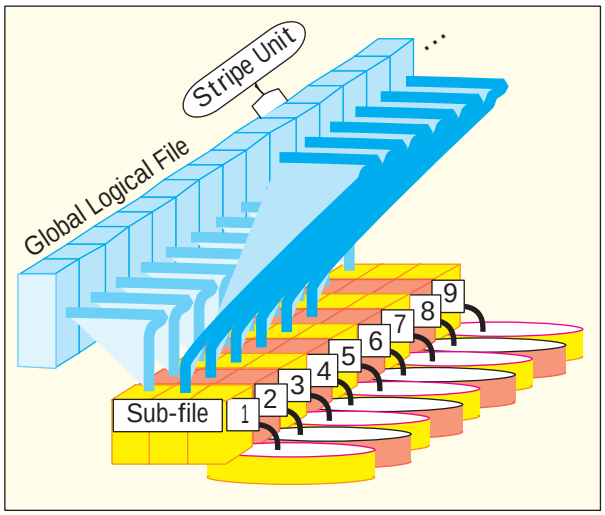


Fig. 1 – File striping

Important aspects of parallel I/O systems are highly concurrent access capabilities to the common datafiles by all parallel application processes and linear increase in performance when increasing the number of I/O nodes and processors. Parallelism for input/output operations can be achieved by striping the data accross multiple disks so that read and write operations occur in parallel (see Fig. 1). A number of parallel file systems where designed ([1], [2], [3], [5]), which make use of the parallel file striping paradigm.

MPI is currently the most used standard framework for creating parallel applications running on various types of parallel computers. A well known implementation of MPI [9], called MPICH, has been developed by Argone National Laboratory. MPICH is used on different platforms and incorporates MPI-1.2 operations [10] as well as the MPI-I/O subset of MPI-2 ([11], [12], [13]). MPICH is most popular for cluster architecture supercomputers, based on Fast or Gigabit Ethernet networks. MPICH's MPI-I/O underlying I/O implementation is completely sequential and is based on NFS ([4], [14]).

Due to the locking mechanisms needed to avoid simultaneous multiple accesses to the shared NFS file, MPICH MPI-I/O write operations can be carried out only at a very slow throughput[1].

Other factor reducing peak performance is the read-modify-write operations useful for writing fragmented data to the target file. Read-modify-write requires sending the full data covering the written data fragment over the network, modifying it and transmitting it back. In the case of high data fragmentation, i.e. small chunks of data spread over a large dataspace in the file, network access overhead may become dominant.

To be able to provide the highest level of parallelization of access requests as well as a good load balance, small striping units are required. However low stripe unit size increases the communication and disk access cost. Our SFIO parallel file striping implementation integrates the relevant optimizations by merging sets of network messages and disk accesses into single messages and single disk access requests. The merging operation makes use of MPI derived datatypes.

The SFIO library interface does not provide nonblocking operations, but internally, accesses to the network and disks are made asynchronously.

Section 2 presents the overall architecture of the SFIO implementation as well as the software layers in order to provide an MPI-I/O interface on top of SFIO. The SFIO

1 When 7 T1 compute nodes access one shared NFS file in an interleaved maner, write throughtput performance on MPICH MPI-IO is 35 KB/s per node

interface description, small examples as well as the details of the system design, caching techniques and other optimizations are presented in Section 3. First performance results are given for various configurations of the Swiss-Tx supercomputer [7]. Section 5 presents the conclusions and future work.

Global Architecture on T1

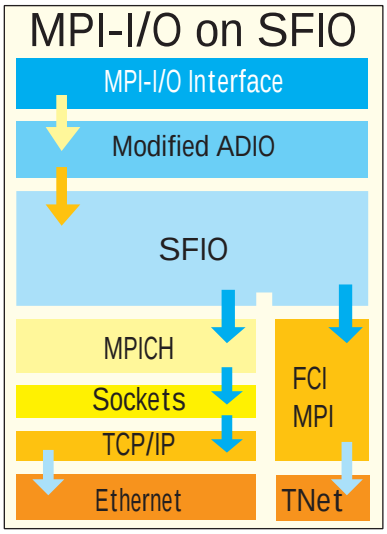


Fig. 2 – Integration of SFIO

The SFIO library is implemented using MPI-1.2 message passing calls. It is therefore as portable as MPI-1.2. The local disk access calls, which depend on the underlying operating system are non-portable. However, they are separately integrated into the source for Unix and Windows-NT versions.

The SFIO parallel file striping library offers a simple Unix like interface. We also intend to provide an MPI-I/O interface on top of SFIO. The intermediate level of MPICH's MPI-I/O implementation is ADIO [14]. We successfully modified the ADIO layer of MPICH to route calls to the SFIO interface.

On the Swiss-T1 machine, SFIO can run on top of MPICH as well as on top of FCI-MPI using the low latency and high throughput network TNet [8].

Unix like interface for parallel striped file I/O

Interface

Two functions, *mopen* and *mclose* are provided to open and close a striped file. Note that a file should be opened by all compute nodes irrespectively of whether that node uses the file or not. This restriction is placed in order to ensure correct behavior of future collective parallel I/O functions. Additionally, the operation of opening as well as of closing a file, implies a global synchronization point in the program. The generic functions to read and write to a file are respectively *mreadc* and *mwritec*.

The multiple I/O request specification interface allows an application program to specify multiple I/O requests within one call. This permits optimizations which otherwise would not be possible. The multiple I/O request operations are *mreadb* and *mwriteb*.

The following source gives a simple SFIO example. The striped file with a stripe unit size of 5 bytes consists of two sub-files. A single compute node accesses the striped file. It is assumed that the program is launched with one compute node MPI process.

```
#include <mpi.h>
#include "mio.h"
int _main(int argc, char *argv[])
{
 MFILE *f;
 f=mopen
 (
 "t0-p1,/tmp/a1.dat;"
 "t0-p2,/tmp/a2.dat;"
 ,5
 );
 mwritec(f,0,"Hello World",11);
 mclose(f);
}
```

Below is an example of multiple compute nodes accessing a striped file. Again the striped file with a stripe unit size of 5 bytes consists of two subfiles. It is accessed by three compute nodes. Each of them writes at different positions simultaneously.

```
#include <mpi.h>
#include "../mpi/sfio/mio.h"
int _main(int argc, char *argv[])
{
 MFILE *f;
 f=mopen
 (
 "t0-p1,/tmp/a1.dat;"
 "t0-p2,/tmp/a2.dat;"
 ,5
 );
 if(rank()==0)
 {
 mwritec(f,0,"Hello*World,*",13);
 }
 else if(rank()==1)
 {
 mwritec(f,13,"I*am*a*program*",15);
 }
 else if(rank()==2)
 {
 mwritec(f,28,"written*with*SFIO.",18);
 }
 mclose(f);
}
```

We assume that the program is launched with three compute and two I/O MPI processes. At the end the global file contains the text combined from the fragments written by the first, second and third compute nodes, i. e. "**Hello*World,*I*am*a*program*written*with*SFIO.**" The text is distributed accross the two sub-files. The first sub-file contains "**Hellod,*I*progritten*SFIO**" and the second "***Worlam*a*am*wr*with.**" (Fig. 3).

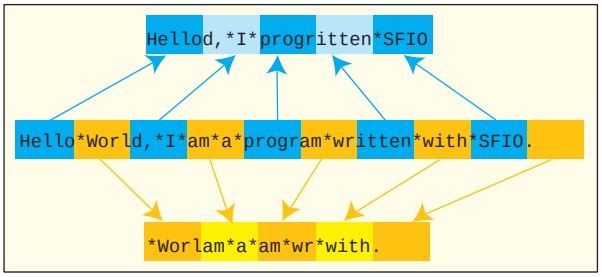


Fig.3 – Distribution of striped file accross sub-files

Function Calls

In this sub-section we present the SFIO library application programmer interface.

File management operations are *mopen*, *mclose*, *mchsize*, *mdelete* and *mcreate*.

```
MFILE* mopen(char *name, int chunk);
void mclose(MFILE *f);
void mchsize(MFILE *f, long size);
void mdelete(char *name);
void mcreate(char *name);
```

All the presented file management operations are collective. Operation *mopen* returns to the compute node a pointer to the logical striped file descriptor. The striped file name, required for the *mopen*, *mdelete*, *mcreate* commands is a string containing the full specification of the number, sequence, locations and paths of sub-files representing the global striped file. The format of the name is a sequence of sub-files, spearated by ";": "`<host>,<path>;<host>,<path>;<host>,<path>…`". For example "`t0-p1,/tmp/a1.dat;t0-p2,/tmp/a2.dat;`"

There are single block and multi-block data access requests.

```
void mread(MFILE *f, long offset, char *buffer,
unsigned size);
void mwrite(MFILE *f, long offset, char *buffer,
unsigned size);
void mreadc(MFILE *f, long offset, char *buffer,
unsigned size);
void mwritec(MFILE *f, long offset, char *buffer,
unsigned size);
void mreadb(MFILE *f, unsigned blknum, long
   offsets[], char *buffers[], unsigned sizes[]);
void mwriteb(MFILE *f, unsigned blknum, long
   offsets[], char *buffers[], unsigned sizes[]);
```

The data access requests are blocking and non-collective. *mreadc* and *mwritec* functions are the optimized versions of the *mread* and *mwrite* functions.

Error management functions are given by *merror* and its collective counterpart *merrora*.

```
void merrora(unsigned long *ioerr);
void merror(unsigned long *ioerr);
void prioerrora();
```

merror and *merrora* return an array of error statistic accumulated on all the I/O nodes. At the same time, they reset the error counters on all the I/O nodes. Statistics are accumulated for operating system I/O calls and listed according to *open*, *close*, *creat*, *unlink*, *ftruncate*, *lseek*, *write* and *read* functions. *prioerrora* is a collective operation which prints the error statistic to the standart output of the application.

Implementation Details

In our programming model, we assume a set of compute nodes and an I/O subsystem. The I/O subsystem is represented as set of I/O nodes running I/O listener processes. Both compute nodes and I/O listeners are MPI processes within a single MPI program. This allows the I/O subsystem to optimize the data transfers between compute nodes and I/O nodes using MPI derived datatypes. The user is allowed to directly use MPI operations only across the compute nodes for computation purposes. The I/O nodes are available to the user only through the SFIO interface.

When a compute node invokes an I/O operation, the SFIO library takes control of that compute node. The library routes the requests to the corresponding I/O listener proxy on the compute node, caches the routed requests and does an optimization of requests queued for each I/O node in order to minimize the cost of disk accesses and network communications. After actual transmission of the messages, the I/O listener(s) prepares a reply which is sent back to the compute node.

Optimization

In order to optimize the disk accesses on the remote I/O node, the algorithm implemented on the compute node tries to combine all overlapping or consecutive I/O requests collected in the cache (Fig. 4). Requests queued for each I/O node are sorted according of their offsets on the remote disk subfile.

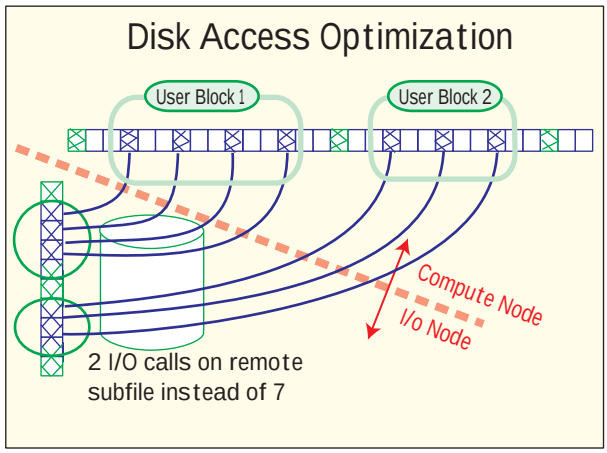


Fig. 4 – Disk Optimization

Queued I/O node access requests cached on the compute node are launched either at the end of the function call or when the buffer size reserved on the remote I/O listener for data reception may become full. Memory is not a problem on the compute node, since data always stays in user

memory and is not buffered. When launching I/O requests, the SFIO library performs a single data transmission to each of the I/O nodes. It creates dynamically a derived datatype which points to the set of pieces in user space memory related to the given I/O node and transmits the data in a single stream without additional copy. The I/O listener at the same time receives the data as a contiguous block.

PERFORMANCE RESULTS

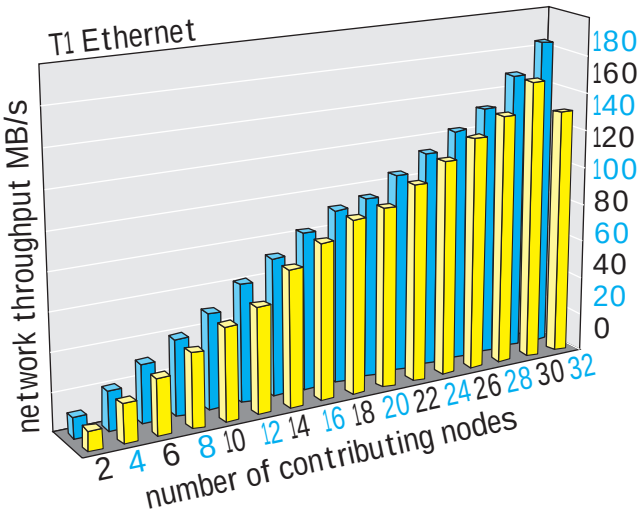


Fig. 5 – Ethernet scalability
(blue = peak; yellow = average)

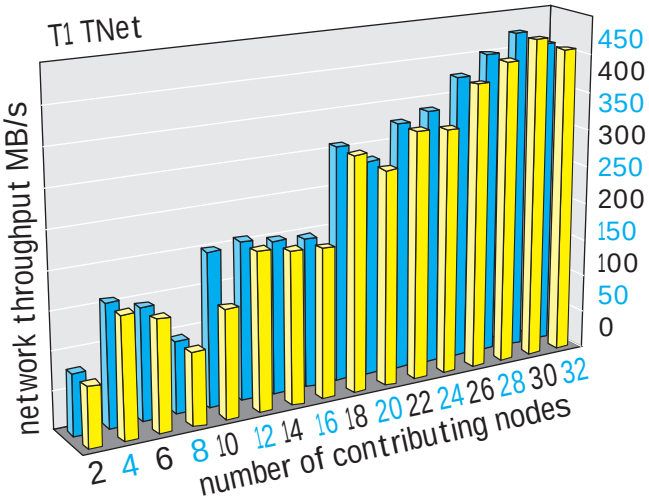


Fig. 6 – TNet scalability

Let us explore the scalability of our parallel I/O implementation (SFIO) as a function of the number of contributing I/O nodes. Performance results have been measured on the Swiss-T1 machine [7]. Swiss-T1 consists of 64 Alpha processors grouped in 32 nodes. Two types of networks are used, TNet and Fast Ethernet. To have an idea about the network capabilities, throughput as a function of number of nodes is measured by a simple MPI program for both networks. The nodes are equally divided into transmitting and receiving nodes and maximal all-to-all traffic is created.

Figure 5 demonstrates cluster throughput scalability with a Fast Ethernet Network and Fig. 6 with TNet. With Fast Ethernet, each node is connected to a Fast Ethernet crossbar switch. The underlying topology of TNet consists of eight 12-port full crossbar switches. The blue graphs show the peak performances and the yellow graphs the average performances.

Let us now analyze the performances of the SFIO library on the Swiss-T1 machine for MPICH on Fast Ethernet and FCI-MPI on TNet. Let us assign the first processor of each compute node to a compute process and the second processor to an I/O listener (Fig. 7).

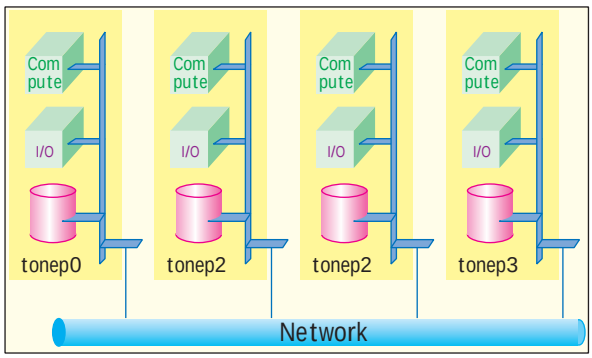


Fig. 7 – SFIO Architecture on Swiss-T1

SFIO performance is measured for concurent write access from all compute nodes to all I/O nodes, the striped file being distributed over all I/O nodes. The number of I/O nodes is equal to the number of compute nodes.

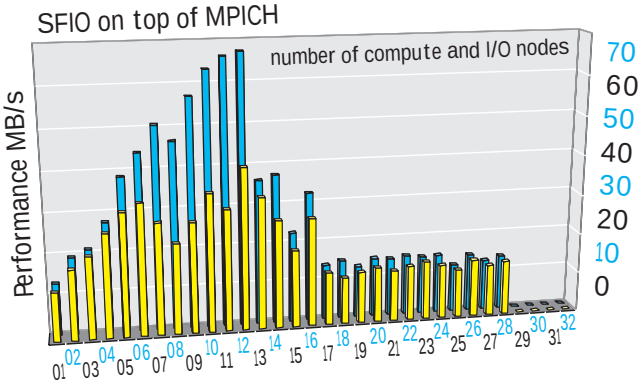


Fig. 8 – SFIO all-to-all I/O performance on Fast Ethernet

The size of the striped file is 2Gbyte and the striped unit size is 200 bytes only. The application's I/O performance as a function of the number of compute and I/O nodes is measured on both Fast Ethernet and TNet and presented in Fig. 8 and Fig. 9. The blue graphs show the peak performances and the yellow graphs the average

performances. We are very surprised with the performance results of SFIO on top of MPICH. This result needs further investigation.

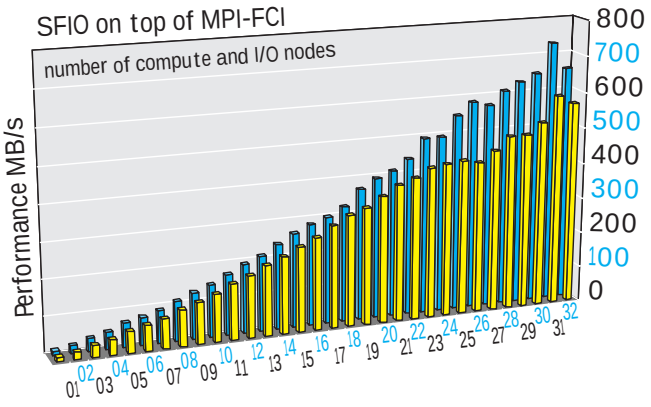


Fig. 9 – SFIO all-to-all I/O performance on TNet

With MPI-FCI the situation is much better. It is highly scalable. When more than 23 nodes participate in the I/O operations, performances may decrease due to TNet's particular communication topology. The effect of topology on the I/O performance will be further studied.

Conclusion and future work

SFIO is a cheap alternative to Storage Area Networks. It is a light-weight portable parallel I/O system available for MPI programmers. Integrated into standard MPI-I/O, SFIO may become a high performance portable MPI-I/O solution for the MPI community.

We plan to realize SFIO benchmarking and check scalability for larger numbers of processors on large supercomputers, e.g. at Sandia National Laboratory.

We intend to implement nonblocking parallel I/O function calls. Disk access optimizations may also be further improved.

Finally we are planning to implement the collective operations as follows: collective operations assume that all compute nodes issue an I/O request at the same logical step in the program. The compute nodes, under control of SFIO library, consult each other to arrive at a common I/O strategy. The I/O nodes are informed about the strategy by the compute nodes and SFIO creates the optimized data flow.

References

[1] Sachin More, Alok Choudhray, Ian Foster, Ming Q. Xu, *MTIO a multi-threaded parallel I/O system*, Proceedings of the 11th International Parallel Processing Symposium (IPPS '97), pages 368-373

[2] Ron Oldfield and David Kotz, *The Armada Parallel File System*, Dartmouth College Dpt. of Compute Science, November 22, 1998, pages 1-14, *www.cs.dartmouth.edu/~dfk/armada/design.html*

[3] Benoit A. Gennart, Emin Gabrielyan, Roger D. Hersch, *Parallel File Striping on the Swiss-Tx Architecture*, EPFL Supercomputing Review, Nov. 99, pp. 15-22, *sic.epfl.ch/publications/SCR99/scr11-page15.html*

[4] Rajeev Thakur, William Gropp, Ewing Lusk, *On Implementing MPI-IO Portably and with High Performance*, *Sixth Workshop on I/O in Parallel and Distributed Systems*, ACM, May 1999, pp. 23-32.

[5] V. Messerli, O. Figueiredo, B. Gennart, R.D. Hersch, *Parallelizing I/O intensive Image Access and Processing Applications*, IEEE Concurrency, Vol. 7, No. 2, April-June 1999, pp. 28-37

[6] Martha Bancroft, Nick Bear, Jim Finlayson, Robert Hill, Richard Isicoff and Hoot Thompson, *Functionality and Performance Evaluation of File Systems for Storage Area Networks (SAN)*, 17-th IEEE Symp. on Mass storage systems, University of Maryland, March 2000, *esdis-it.gsfc.nasa.gov/msst/conf2000/PAPERS/A05PA.PDF*

[7] Pierre Kuonen, Ralf Gruber, *Parallel computer architectures for commodity computing and the Swiss-T1 machine*, EPFL Supercomputing Review, Nov 99, pp. 3-11, *sic.epfl.ch/publications/SCR99/scr11-page3.html*.

[8] Stephan Brauss, *Communication Libraries for the Swiss-Tx Machines* EPFL Supercomputing Review, Nov 99, pp. 12-15, *sic.epfl.ch/publications/SCR99/scr11-page12.html*.

[9] Peter S. Pacheco, *Parallel Programming with MPI*, by Morgan Kaufmann Publishers, pages 137-178, 1997

[10] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra, *MPI - The Complete Reference*, Volume 1, The MPI Core, MIT Press, pages 123-189, 1996

[11] William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, Marc Snir, *MPI - The Complete Reference*, Volume 2, The MPI Extensions, MIT Press, pages 185-274, 1998

[12] William Gropp, Ewing Lusk, Rajeev Thakur, *Using MPI-2 Advanced Features of the Message-Passing Interface*, MIT Press, pages 51-118, 1999

[13] Message Passing Interface Forum, *MPI-2 Extentions to the Message-Passing Interface*, University of Tennessee, pages 209-300, 1997

[14] Rajeev Thakur, William Gropp, Ewing Lusk, *A Case for Using MPI's Derived Datatypes to Improve I/O Performance*, *www.supercomp.org/sc98/TechPapers/sc98_FullAbstracts/Thakur447/, pages 1-9, 1998* ■

STABILITY AND α-PARTICLE CONFINEMENT IN THE SPHELLAMAK REACTOR CONCEPT

W. ANTHONY COOPER AND OLIVIER FISCHER, EPFL — CENTRE DE RECHERCHES EN PHYSIQUE DES PLASMAS

Le Sphellamak est un système hybride sans noyau central composé par des éléments de Tokamak, de Stellérateur et de Sphéromak. L'absence de colonne centrale permet la réalisation d'un système toroïdal compact puisque le manteau de protection interne ne devient plus nécessaire. Avec un profil de courant piqué, une séquence d'équilibres Sphellamak de dimension d'un réacteur est calculée numériquement en variant le courant des bobines helicoïdales I_{hc} tout en fixant le courant toroïdal du plasma I_p=-30 MA ainsi que la moyenne volumique β=7.3%. Les modes globaux externes du type **kink** *sont faiblement instables mais suffisent à garantir la stabilité pour I_{hc} > 138 MA.*

Les critères de stabilité magnétohydrodynamique idéale locale sont réalisés pour des courants de 42 MA < I_{hc} < 122 MA. Le courant toroïdal piqué produit localement des valeurs maximales pour le module du champ magnétique dans la région centrale du plasma ce qui implique des conditions favorables pour le confinement des particules énergétiques et thermiques. Cette conclusion est confirmée à travers le calcul d'un taux de perte très faible des orbites du centre de guidage des particules α.

The Sphellamak is a coreless hybrid system with Tokamak, Stellarator and Spheromak features. The absence of a central conductor permits the realisation of a compact toroidal system as internal shielding becomes unnecessary. With a peaked toroidal current profile, a sequence of reactor-sized Sphellamak equilibria is computed numerically in which the current in the helical coils I_{hc} is varied while the toroidal plasma current I_p=-30 MA and the volume average β=7.3% remain fixed. Ideal global external kink modes are weakly unstable but indicate stability for I_{hc} > 138 MA. The local ideal magnetohydrodynamic stability criteria are satisfied in the range 42 MA < I_{hc} < 122 MA. The peaked toroidal current generates local maxima of the modulus of the magnetic field strength in the central region of the plasma which has very favourable implications for energetic and thermal particle confinement. This is confirmed through the computation of a very small α-particle guiding centre orbit loss fraction.

INTRODUCTION

Magnetic fusion offers the potential of a clean and everlasting source of energy to satisfy the requirements of humankind. For a plasma to ignite and burn, typically its density must be $n \sim 10^{20}$ particles/m^3, its temperature must reach $T \sim$10keV and the confinement of the thermal particle energy must approach $\tau \sim 1s$. This is often combined to yield the criterion $nT\tau \sim 10^{21}$kev - s/m^3 for a relevant power plant. First generation reactors are anticipated to rely on Deuterium (D_1^2) and Tritium (T_1^3) as the reaction

$$D_1^2 + T_1^3 \rightarrow n_0^1 + He_2^4 + 17.6\ MeV \qquad (1)$$

constitutes the easiest to realise experimentally. The energy carried by the neutron that can be recovered in a blanket surrounding the discharge chamber amounts to 14.1 Mev per reaction. The heat generated in the blanket as the neutron delivers its energy through collisions with the structure in turn can run a conventional turbine. The 3.5 MeV α particles (the helium nuclei) remain confined in the plasma as they are electrically charged and provide the source of energy required to sustain the plasma temperature through collisional processes with the background plasma particles and the generation of waves that may also interact with the plasma fluid.

$$\beta = \frac{2\mu_0 \int d^3x p}{\int d^3x B^2} \qquad (2)$$

The cost of a magnetic fusion reactor is closely linked with the magnetic field required to confine the plasma. The magnitude of this field, labelled B, depends on the current that must be driven in the coils which in turn dictates the size of the coils, the material from which they must be constructed and the cooling capabilities. Magnetohydrodynamic (MHD) instabilities can be triggered whenever the B-field strength becomes too weak. Specifically, instabilities can arise when the ratio of the kinetic pressure of the confined gas to the magnetic pressure of the confining fields exceeds a critical value. This parameter is defined as β,

$$\sqrt{g}\delta B^s = \sqrt{g}B \bullet \nabla\xi^s \qquad (3)$$

The kinetic pressure p is proportional to the product nT and the requirement for fusion reaction that this product approach 10^{20}keV/m^3 entails that B must be sufficiently large to avoid instabilities that can destroy the plasma column.

It has been demonstrated theoretically and experimentally in a Tokamak plasma confinement system that the maximum stable β values that can be achieved increase with decreasing aspect ratio. The aspect ratio $A=R/a$ is the ratio of the major radius to the minor radius. The major radius R is the distance from the major axis to the centre of the plasma column and the minor radius a is the average radius of the plasma column. As $A \rightarrow 1$, β values in excess of 15% can be realised. However, in a reactor concept, the necessity to shield the coil structure and the central column to neutron bombardment makes any design with low aspect ratio extremely difficult to fulfill.

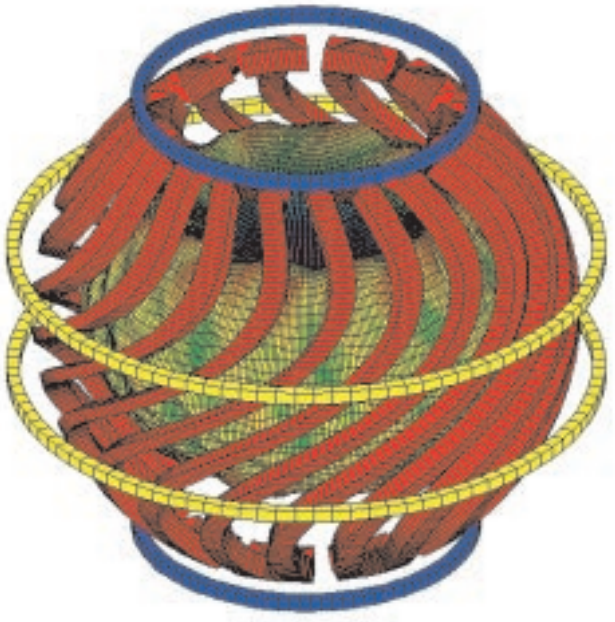


Fig. 1 – The coil system of a Sphellamak reactor concept. The helical coils in red are wound on a spheroidal surface of 7.5m radius. The coil width is roughly 0.75m. The vertical field compensation coils that cancel the current flowing in the arc segments that connect the helical coil legs are shown in blue. They carry half the current in the helical coils but in the opposite direction of that in the arc segments. The vertical field coils in yellow control the plasma position. The B^2 distribution on the outermost flux surface appears in shades of green, yellow and red. The Boozer magnetic coordinate grid is also shown.

The Tokamak systems utilise the coils to produce the longitudinal magnetic field the long way round the torus and a plasma current to generate the poloidal field the short way around the torus. This combination of fields serves to confine the charged particles in the plasma. The resulting magnetic field structure maintains a very high degree of symmetry in the toroidal direction. Stellarator systems can generate the confining fields entirely with external coils but usually at the expense of the symmetry in the toroidal direction. An investigation of α-particle orbits in nonsymmetric Stellarators reveals that a large fraction of them are lost almost instantaneously due to enhanced magnetic gradient and curvature drift effects and thus cannot contribute to the sustainment of the background plasma temperature. New Stellarator designs have been identified which, although three-dimensional (3D) in physical appearance, can still produce a magnetic field structure that resembles that of a two-dimensional Tokamak. These are known as quasiaxisymmetric Stellarators [1]. Furthermore, other 3D systems have been conceived such as the Wendelstein VII-X in construction in Germany that guarantee adequate confinement of α-particles through the poloidal closure of the second adiabatic invariant. These are referred to as quasi-isodynamic systems [2].

To highlight the principal physics issues in magnetic fusion reactor systems, MHD instabilities impose a limit on β which implies that the magnetic field B must have a minimum value. Furthermore, the confinement of α particles imposes an adequate level of symmetry properties in the magnetic field structure in addition to constraints on the magnitude of B. On the other hand, cost constraints favour systems that can satisfy the physics criteria (β, α-particle confinement) at the lowest possible B. One of the challenges of fusion physicists is to identify configurations and scenarios that optimally meet these conflicting conditions.

THE SPHELLAMAK CONCEPT

The Sphellamak concept [3], developed at CRPP/EPFL in collaboration with T. N. Todd of Culham Laboratories, UK, is a hybrid system that combines features of a Tokamak, a Stellarator and a Spheromak. A Spheromak is a coreless device that carries a toroidal current and relies on plasma instabilities and turbulence to generate the necessary toroidal magnetic field for confinement through a process of helicity conservation called the *dynamo* action. The Sphellamak, like the Spheromak, is a coreless concept that employs Stellarator windings on a spheroidal surface. These coils produce seed paramagnetism that is significantly amplified by the toroidal plasma current. As the system is 3D, charge conservation ($\nabla \bullet \mathbf{j}=0$, where $\mathbf{j}$ is the plasma current density) implies that the toroidal plasma current generates not only the poloidal magnetic field, but also the toroidal magnetic field without a need for the dynamo effect and the instabilities associated with it. One of the main attractions of the Sphellamak is the potential to realise a very low aspect ratio device in a reactor as the absence of a central column eliminates the need for internal shielding. A reactor-sized version of a Sphellamak device is displayed in Fig. 1. The helical coils are shown in red. The current in them flows up one leg, across the connecting arc segment near the upper pole, down the adjacent helical leg and then back across the arc segment near the lower pole. There are 10 modular coils in this device. The vertical field coils in blue near the polar regions carry half the current of the helical coils, but in the direction opposite that of the flow in the neighbouring arc segments to cancel the effective vertical fields produced by

the circulation of currents in these arcs. The vertical field coils in yellow control the plasma position and counteract the outward hoop force induced by the toroidal current in the plasma. The distribution of B^2 on the outermost flux surface in shades of yellow, green and red appears within the coil structure.

MAGNETOHYDRODYNAMIC EQUILIBRIA

The free boundary version of the 3D VMEC equilibrium code [4] is employed to numerically compute Sphellamak equilibrium sequences. This code imposes perfectly nested magnetic flux surfaces like the layers of an onion on a doughnut shaped system for the equilibria that are calculated. The input required for this code are the radial, vertical and toroidal components of the vacuum magnetic fields produced by the currents in the external coil windings, the pressure and current profiles. Also required are the magnitudes of the toroidal magnetic flux $2\pi\Phi$ (1), the pressure at the magnetic axis $p(0)$ and the total toroidal current enclosed within the last flux surface $2\pi J$ (1). The contribution of the helical coils to the vacuum fields are computed using the Biot-Savart law on a toroidal domain of rectangular cross section within the helical coils, where each coil is modelled as a short straight segment. Four filaments, separated by a distance of 0.75 m are employed to model the finite dimensions of these coils. The elliptic integral formulation of the Biot-Savart law is applied to determine the contributions of the vertical field coils to the vacuum magnetic field which are composed of four circular filaments separated by a distance of 0.375m. A sequence of configurations have been calculated by varying the current in the helical coils from 42 MA to 122 MA. The outer vertical field coil currents are adjusted to carry a 1/15 fraction of the helical coil current.

The VMEC code also requires the pressure profile which we have prescribed as

$$p(s) = p\,(0)\,(1 - s^2)^2 \qquad (3)$$

where we have varied p (0) to maintain β =7.3% and the toroidal current profile

$$2\pi J'(s) = 2\pi J'(0)\,(3\,(1 - s)^5 + (1 - s^5)^2)/4 \qquad (4)$$

where $0 \le s \le 1$ is the radial variable that labels the flux surfaces and is proportional to the volume enclosed. The value of $2\pi J'(0)$ is chosen such the the total toroidal current within the plasma is –30 MA. An initial guess is provided for the shape of the plasma and the toroidal magnetic flux at the boundary $2\pi\,\Phi$ (1) is varied until a converged equilibrium is obtained.

The plasma volume and the average magnetic energy density in the plasma are displayed in Fig. 2 for the sequence of equilibria investigated as a function of the current in the helical coils I_{hc}. The plasma volume decreases and the average magnetic energy increases with increasing helical coil current.

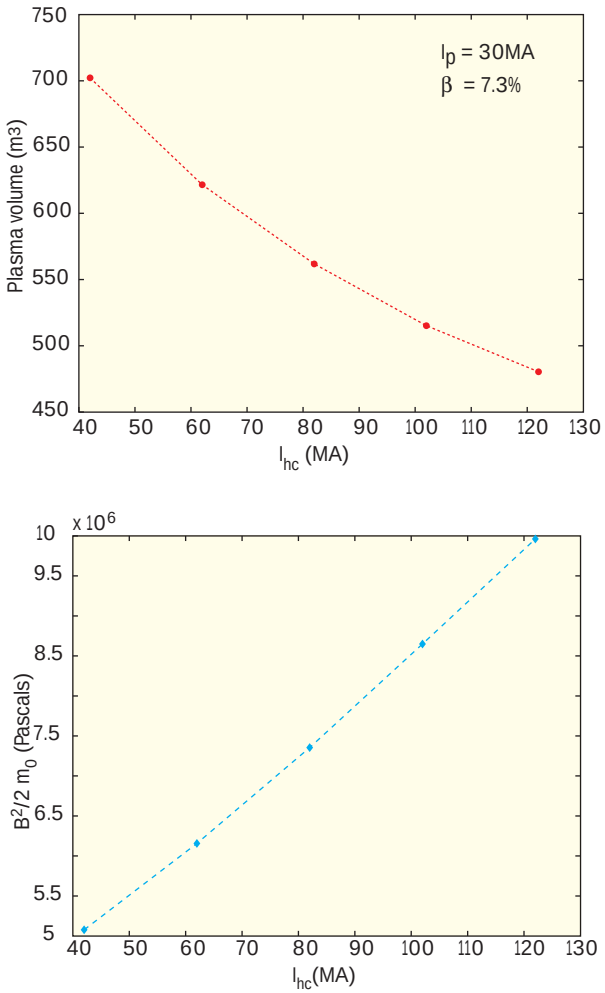


Fig.2 – The plasma volume in m^3 (top) and the average magnetic energy density in **Pascals** (bottom) as a function of the helical coil current I_{hc} in MA of the sequence of equilibria explored with toroidal plasma current I_p =-30 MA and β =7.3%

MAGNETOHYDRODYNAMIC STABILITY

We apply the local and global modules of the 3D ideal MHD stability code TERPSICHORE [5] to investigate Mercier, ballooning, internal and external kink modes of the Sphellamak sequence of equilibrium configurations. The TERPSICHORE code performs a coordinate transformation of the VMEC equilibria to Boozer magnetic coordinates [6]. These coordinates facilitate the evaluation of stability properties because the magnetic field lines become straight which simplifies the inversion of the $\boldsymbol{B} \cdot \nabla$ operator and because the parallel current density, which is an important source of free energy for instabilities, can be more efficiently calculated.

The local stability modules of TERPSICHORE determine the Mercier criterion and the Fourier coefficients of the driving and stabilising terms that determine ballooning stability. Mercier modes are instabilities that are very localised

about a magnetic surface but have extended structures along the magnetic field lines. Ballooning modes form structures that are localised along magnetic field lines typically in the region where the directions of the magnetic field line curvature and the pressure gradient become aligned, which is usually radially away from the major axis. The ballooning coefficients are reconstructed along the magnetic field lines and a shooting method is applied to evaluate the eigenvalue of the second order ordinary differential equation that describes ballooning stability. The Mercier criterion and the ballooning eigenvalue profiles as a function of the radial variable s are presented for the two limiting configurations of the sequence in Fig. 3.

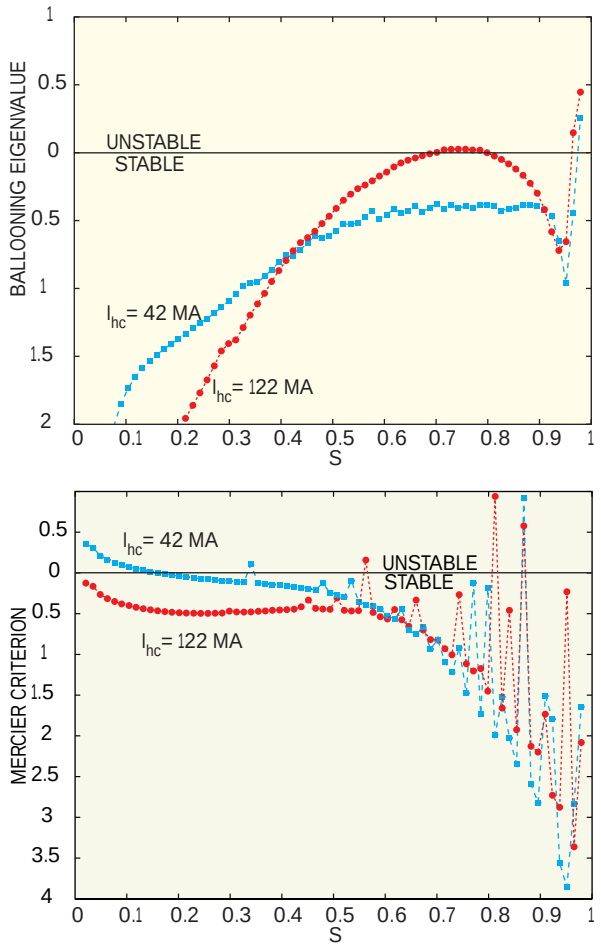


Fig. 3 – The ballooning eigenvalue (top) and Mercier criterion (bottom) profiles for the two limiting configurations of the sequence of equilibria explored having I_{hc} = 42 MA and I_{hc} = 122 MA, respectively

Ignoring the very edge of the plasma where a slight flattening of the pressure profile will stabilise the ballooning eigenvalues, we observe that the Mercier criterion predicts unstable conditions for the system with helical coil current I_{hc} = 42 MA in the central region of the plasma but is otherwise stable to ballooning modes. On the other hand, the configuration with I_{hc} = 122 MA is Mercier stable but becomes weakly unstable in the region around $s \approx 0.75$ (at 3/4 of the plasma volume). The intermediate configurations of the sequence are all stable. The spikes observed in the Mercier criterion are not considered as indicative of instability but rather of conditions whereupon the pressure gradient drives the formation of magnetic islands. Though this technically violates the condition of magnetic surface nestedness, these spikes are sufficiently localised and nonoverlapping that the assumption that guides the application of the VMEC code remains valid under these circumstances.

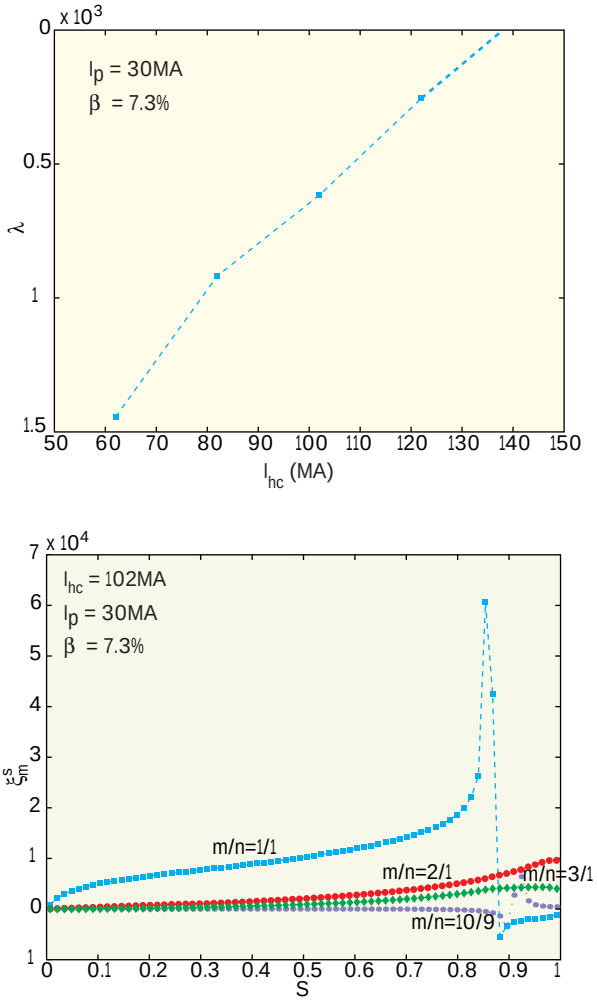


Fig. 4 – The global eigenvalue λ corresponding to the n=–1 family of instabilities as a function of the helical coil current I_{hc} of the sequence of Sphellamak equilibria investigated (top). The marginal point that is extrapolated has I_{hc}≈138 MA. The 4 leading Fourier components of the mode structure corresponding to the n=–1 instability family as a function of the radial variable s for the I_{hc}=102 MA case of the sequence explored (bottom)

The global stability modules of the TERPSICHORE code serve to calculate unstable eigenvalues associated with current driven instabilities and pressure driven ballooning/interchange modes. Global mode structures are anticipated to have more deleterious consequences than local modes because they can produce vortices that connect the central region of the plasma to the edge and as a result destroy the

plasma column. In Fig. 4, we show the unstable eigenvalue with respect to the family of unstable $n = -1$ global eigenstructures as a functon of I_{hc}. The sequence we investigate is unstable in the range 42 MA < I_{hc} < 122 MA, but from extrapolation we can predict a nearby case that is marginally stable at $I_{hc} \approx 138$ MA. We also present in this Figure the 4 dominant Fourier amplitudes of the radial component of the displacement vector for the case obtained with I_{hc} = 102 MA. The internal region of the plasma is dominated by the $m = 1, n = -1$ term, where m is the poloidal mode number and $n = -1$ is the toroidal mode number. However, near the edge of the plasma, the m = 2,3 terms become important because the m = 1 term becomes nearly vanishing there. The perturbed magnetic field is related to the displacement vector through the equation $\delta \boldsymbol{B} = \Delta \times (\xi \times \boldsymbol{B})$ from which we obtain $\sqrt{g}\delta B^s = \sqrt{g}B \bullet \nabla\xi^s$, where $\sqrt{g}$ is the Jacobian, and δB^s and ξ^s are the radial components of the perturbed magnetic field and the displacement vector, respectively. The quantity $\sqrt{g}\delta B^s$ is displayed an a flux surface very near the edge of the plasma for the I_{hc} = 102 MA case which confirms that a combination of m/n=2/–1 and m/n= 3/–1 components form the eigenstructure. It is relevant because magnetic probes external to the plasma can be used to detect structures of this type.

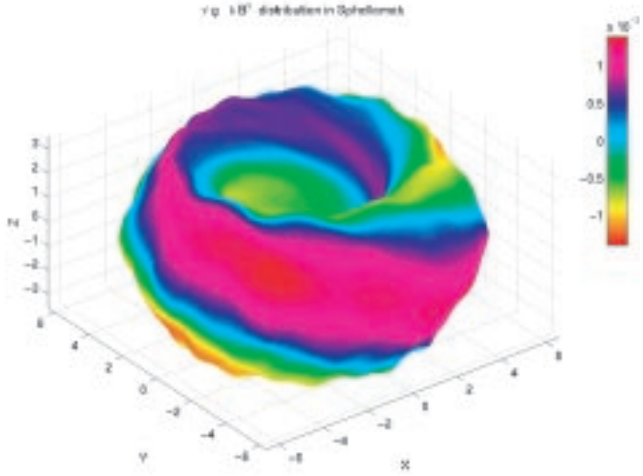


Fig. 5 – The distribution of $\sqrt{g}\delta B^s$ on a flux surface very near the edge of the plasma for the I_{hc} =102MA case of the sequence of Sphellamak reactor equilibria explored

THE MAGNETIC FIELD STRUCTURE

In typical Tokamak axisymmetric systems, the magnetic B-field strength is inversely proportional to the distance from the major axis. In classical Stellarators, the B-field strength becomes 3D and the magnetic drifts of trapped energetic particles cause them to quickly escape the plasma. In the Sphellamak device, the strong toroidal plasma current plays a critical rôle in generating the magnetic fields. With peaked toroidal current profiles, the magnetic field structure can acquire properties of a maximum-B system which are characterised by the B-field strength developing a local maximum in the central region of the plasma. The sequence of configurations we have investigated do in fact satisfy these conditions. The distribution of the modulus of B^2 (mod–B^2) on three different cross sections of the plasma (corresponding to the toroidal angles ϕ = 0, $\pi/20$ and $\pi/10$) are displayed in Fig. 6 for the configurations obtained with I_{hc} = 42 MA and I_{hc} = 122 MA. The mod–B^2 distribution in the vicinity of the magnetic axis becomes closely aligned with the magnetic flux surfaces. This corresponds to a nearly isodynamic system [7] which has very favourable implications for confinement because the particle drifts remain confined within the flux surface [8].

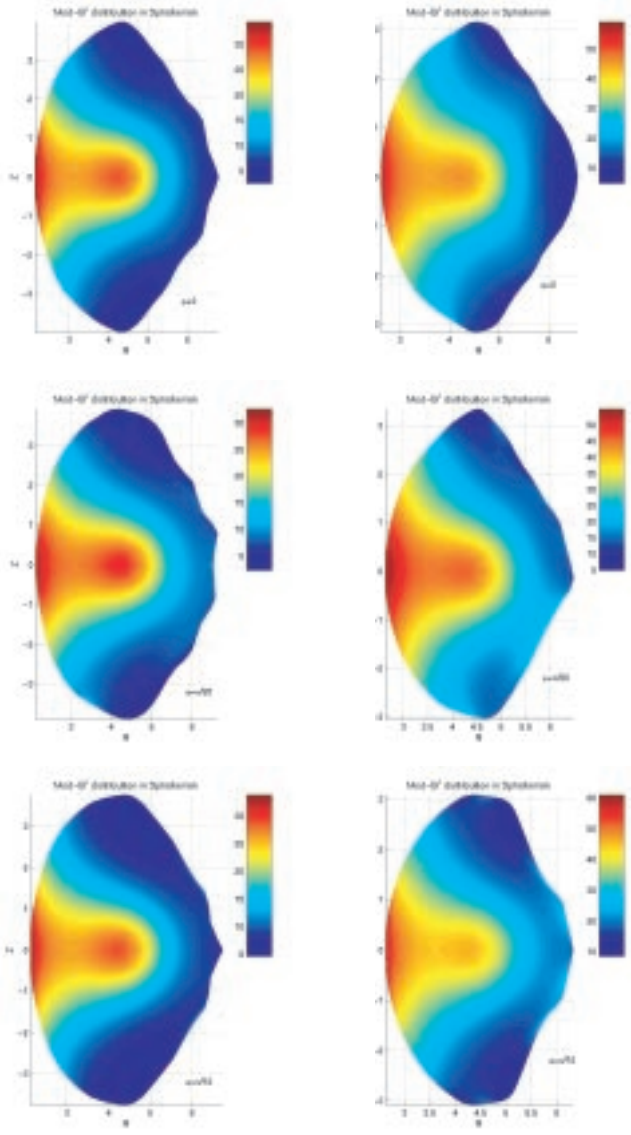


Fig. 6 – The B^2 distribution on cross sections at the beginning of a field period ϕ =0 (top), at one quarter of a field period $\phi = \pi/20$ (middle) and at half period $\phi = \pi/10$ (bottom) for the two limiting configurations of the sequence of Sphellamak equilibria explored with I_{hc}= 42 MA (left) and I_{hc}=122 MA (right), respectively in the Boozer magnetic coordinate frame

α-PARTICLE GUIDING CENTRE ORBITS

The motion of electrically charged particles in a magnetic field is characterised by rapid gyration about a *B*-field line and streaming along it, but also by a slower drift across the field lines due to the inhomogeneity in *B*. In magnetic confinement systems where the typical scale length of the gyro-orbit is much smaller than characteristic scale lengths of the device, the guiding centre approximation, which averages over the gyromotion, can be invoked. This is the case of the Sphellamak equilibria we are considering. Thus the α-particle guiding centres rather than the exact orbits are followed. This reduces the problem to a much more tractable undertaking. The guiding centre motion can be described through a Hamiltonian formalism in which the Boozer magnetic coordinates constitute a canonical frame subject to the condition that the magnetic field is static with nested surfaces (and that time dependent general perturbations can be represented as the curl of a scalar function times the equilibrium magnetic field) [9]. Furthermore, the guiding centre orbit equations expressed in these coordinates depend on quantities that are constant on each flux surface and on the magnitude of *B*. Therefore all of the geometric effects of shaping on the orbits manifest themselves only and exclusively through the structure of *B*. It thus becomes possible to conceive of confinement systems with strong 3D geometry but where *B* becomes independent of one or possibly both of the angular variables.

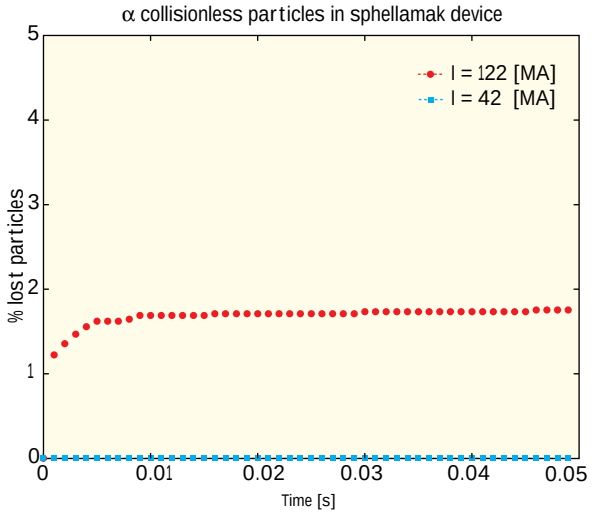


Fig. 7 – The fraction (in percent) of α-particle orbits lost in 0.05s (half of a 3.5MeV α-particle slowing down time) for the two limiting configurations of the Sphellamak reactor sequence explored with I_{hc}=42 MA (in blue) and with I_{hc}=122 MA (in red), respectively

We follow the trajectories of 4500 α-particle guiding centres that are *born* on the flux surface $s \approx 0.25$ (at 1/4 the plasma volume) with a random distribution of pitch angle (the ratio of the parallel to the total velocity), poloidal and toroidal angles using the VENUS code [10]. These trajectories are followed for up to 0.05 s which corresponds to half of a slowing down time for a 3.5 MeV α-particle. We monitor the guiding centre orbits that reach the last closed magnetic flux surface and consider these as lost. The fraction of α-orbits that are lost are shown in Fig. 7 for the two limiting configurations of the sequence examined. We observe that the α-particle confinement is virtually perfect for I_{hc} = 42 MA, while less then 2% of the orbits are lost for the I_{hc} = 122 MA case. For this case, the α loss is referred to as prompt and corresponds to particles that drift rapidly out of the device. To understand these results, we refer back to Figs 2 and 6 where we notice that the local maxima of B^2 increases with I_{hc} (B^2 ~ $30T^2 \rightarrow 45T^2$), while the plasma volume decreases from $702m^3 \rightarrow 480m^3$. Though the distribution of B^2 appears similar for the two limiting configurations of the sequence, the I_{hc} = 122 MA case appears to be slightly more 3D than the I_{hc} = 42 MA case. The combination of smaller volume and higher 3D shaping may be just sufficient enough to overcome the larger magnitude of *B* to cause the slight deterioration of α confinement for the I_{hc} = 122 MA case. Nevertheless, this level of loss would not compromise the viability of $I_{hc} \geq 122$ MA configurations as power producing reactor devices.

COMPUTATIONAL ISSUES

Five programmes have been employed in the computations presented. The COIL.SPHELL package evaluates the vacuum magnetic fields from the currents in the external coil segments using the Biot-Savart law. The VMEC code numerically determines 3D equilibria. It is based on an accelerated preconditioned spectral energy minimisation scheme. The stability code TERPSICHORE evaluates local and global MHD stability properties. For the global modes, it solves a special block pentadiagonal matrix eigenvalue equation using an inverse vector iteration technique to determine the eigenvalue and the eigenvector. The VVBAL code uses a shooting method to solve the ballooning mode eigenvalue. The VENUS code solves the guiding centre orbit equations using an initial value method that combines a 4th order Runge-Kutta solver with a 2nd order version. The time step is typically 10^{-8}s.

The VMEC and TERPSICHORE codes have long vector lengths for which the use of vector/parallel supercomputers is most appropriate. Typical VMEC runs require less than 100Mbytes and take ~ 1000s CPU time while TERPSICHORE runs require 0.5 to 1.5 Gbytes and take ~100-200 s CPU time on a NEC/SX4 platform. The VENUS code is most effectively run on a massively parallel system as each guiding centre particle can be assigned to a different processor. A typical run consists of 4500 particles on 16 processors which takes about 1.5 hours on an ORIGIN 2000 machine.

All graphics (1D → 3D) are undertaken in a post-process procedure on a PC utilising mostly MATLAB but sometimes BASPL routines. The Sphellamak coil system is designed with a MATLAB programme.

SUMMARY, CONCLUSIONS AND DISCUSSION

The ideal MHD stability and the α particle confinement properties of a Sphellamak reactor concept have been investigated. We have concentrated on a sequence of configurations where we have varied the current I_{hc} in the helical coils keeping a peaked toroidal plasma current at –30 MA and a volume average β = 7.3% fixed. The magnetic field strength B increases and the plasma volume decreases with raising I_{hc}. The local stability criteria improve with increasing I_{hc} in the inner half of the plasma volume and deteriorate in the outer half. The global ideal MHD kink mode imposed by the n=–1 family of instabilities becomes less unstable as I_{hc} is raised predicting a marginal point for the case I_{hc} ~ 138 MA. The magnetic field structure displays local maxima of B in the central region throughout the range of configurations explored. These maximum-B equilibria have closed B-contours that become aligned with the magnetic flux surfaces. This has a very favourable impact on the confinement of α-particles as the magnetic drifts out of the flux surfaces are weak as confirmed by the computation of a very small α-guiding centre orbit loss fraction of less than 2%.

Although the set of Sphellamak configurations we have presented appears to constitute a very attractive example for a reactor system, further improvements in the ideal MHD stability properties are still required to guarantee a robust margin of stable operation as this sequence we have identified is not sufficiently satisfactory.

The very satisfactory α-particle confinement in the Sphellamak reactor configurations can be attributed to the maximum-B properties of the magnetic field structure around the centre of the plasma. This structure is realised with a peaked toroidal plasma current. One of the principal technical challenges beyond the scope of this article is how to generate and sustain such type of current profile and magnitude of current in the absence of a central column.

The satisfactory resolution of MHD stability and α-particle confinement is paramount in the evaluation of a viable reactor concept. However, as confined plasmas are not in general quiescent, other physics issues must eventually also be considered. Specifically, we have not addressed the issue of confinement of thermal particles in the background of a turbulent plasma that determines the plasma energy confinement that must approach τ ~ 1s in a reactor. These turbulent fields may also unfavourably impact the α-particle confinement.

ACKNOWLEDGMENTS

This research was partially sponsored by the Fonds National Suisse de la Recherche Scientifique. We thank Dr. S.P. Hirshman for use of the VMEC equilibrium code. The numerical calculations on equilibrium and stability presented in this paper were performed on the NEC-SX4 computer at the Centro Svizzero di Calcolo Scientifico, Manno, Switzerland. The guiding centre orbit calculations were undertaken on the ORIGIN-2000 massively parallel computer at SIC-EPFL.

REFERENCES

[1] J. Nührenberg, W. Lotz and S. Gori, *Quasi-axisymmetric Tokamaks*, in Proc. Joint Varenna-Lausanne Int. Workshop on Theory of Fusion Plasmas, Editrice Compositori, Bologna (1994) 3-12.

[2] S. Gori and J. Nührenberg, *Quasi-isodynamic stellarators with magnetic well and positive shear*, in Proc. Joint Varenna-Lausanne Int. Workshop on Theory of Fusion Plasmas, Editrice Compositori, Bologna (1998) 473-480.

[3] W. A. Cooper, J. M. Antonietti and T. N. Todd, *A Paramagnetic Nearly Isodynamic Compact Magnetic Confinement System* in Proc. 17th IAEA Fusion Energy Conf., Yokohama, Japan, IAEA-CN-69/EX4/1(R) (1998).

[4] S. P. Hirshman, W. I. Van Rij and P. Merkel, *Three-dimensional free boundary calculations using a spectral Green's function method*, Computer Physics Communication **43** (1986) 143-155.

[5] D. V. Anderson, W. A. Cooper, R. Gruber, S. Merazzi and U. Schwenn, *Methods for the efficient calculation of the (MHD) magnetohydrodynamic stability properties of magnetically confined fusion plasmas*, The International Journal of Supercomputer Applications **4** (1990) 34-47.

[6] A. H. Boozer, *Establishment of magnetic coordinates for a given magnetic field*, Physics of Fluids **25** (1982) 520-521.

[7] D. Palumbo, Il Nuovo Cimento X **53B** (1968) 507.

[8] L. S. Hall and B. McNamara, *Three-dimensinal equilibrium of the anisotropic, finite-pressure guiding-center plasma: Theory of the magnetic plasma*, Physics of Fluids **18** (1975) 552-565.

[9] R. B. White and M. S. Chance, *Hamiltonian guiding center drift orbit calculation for plasmas of arbitrary cross section*, Physics of Fluids **27** (1984) 2455-2467.

[10] O. Fischer, W. A. Cooper and L. Villard, *Magnetic topology and guiding centre drift orbits in a reversed shear tokamak*, Nuclear Fusion **40** (2000) 1453-1462. ■

HPC in Computational Structural Mechanics

Pieter Volgers, EPFL DGC-IMAC

En mécanique des structures, le calcul haute performance n'est pas aussi commun que par exemple en aérodynamique. Les problèmes étudiés jusqu'à présent étaient généralement suffisamment petits pour être traités sur un simple PC voire une grosse station de travail. C'est seulement récemment que l'intérêt s'est porté à la simulation numérique de très grands problèmes. L'industrie automobile arrive en tête avec des besoins de simulation à grande échelle de voiture en situation d'accident. Pour ces nouveaux problèmes, les micro-ordinateurs ne suffisent plus et il est nécessaire de se tourner vers les super-ordinateurs. Cet article aborde quelques-uns des aspects du calcul haute performance en mécanique des structures et plus particulièrement des nouvelles méthodes développées dans ce cadre.

High-performance computing in structural mechanics is not as common as is it in e.g. aerodynamics. Most of the problems studied so far are sufficiently small that the ever faster workstations and PCs were usually powerful enough to solve the problem. Only recently the interest in large problems, too big and/or too complex to solve on the desktop, require the use of super-computers for structural mechanic problems. The automotive industry is leading the way with large scale crash simulations. In this article we will discuss some of the aspects of these problems and the recently developed methods to deal with them.

Introduction

In computational structural mechanics, the use of numerical techniques, especially the finite element method, has greatly improved the predictive nature of the analysis of structures. The finite element method, originating from civil and aeronautical engineering, has found its way into all structural analysis fields. In contrast to other numerical simulation fields, the use of high-performance computers to solve the problems has not taken a great flight. There are multiple reasons for this to be found: Firstly, most problems to be solved could be dealt with using workstations. The increase of desktop computational power kept trend with the increasing demand of the structural engineer. Secondly, the nature of the unstructured method, combined with the use of shell theory (which assumes one dimension to be much smaller than the other two) results in linear systems with very bad conditions for iterative solvers. With the structural mechanics finite element programs being therefore limited to direct solvers (LU decomposition), which does not vectorize well, and no general parallel direct solver available, the engineer was limited to desktop computing. And finally, a study of the problems to be solved in numerical structural mechanics shows that there are basically two type of problems: Relatively small ones, which are, especially with todays powerful processors, solvable on the desktop. And very large problems, which require an enormous amount of computational power and/or memory.

With new programming techniques and new direct parallel solvers available, computational structural mechanics can benefit from the use of high performance computing. At the Institute of Stress Analysis and Measurement (IMAC) of the Civil Engineering Department at EPFL new methods have been evaluated for the use of large scale simulations in mechanical engineering. They involve different techniques for two types of problems: Static and dynamic. In this article we will describe one of them: The use of parallel computing for dynamic simulations. This is the kind of problem which requires massive computation power to solve a problem in a reasonable amount of time. Section **Explicit dynamic simulations** gives a brief overview of the explicit finite element program and its parallel implementation. Section **Performance on Swiss-T1** shows the results of the benchmarks performed on the Swiss-TI machine here at the EPFL.

Explicit dynamic simulations

Explicit finite element programs are generally used for the simulation of highly non-linear dynamic phenomena occurring in a relatively small time interval. The method is most widely used for the simulation of impact in automotive and aerospace industry, as well as for sheet metal forming simulations. Due to the restriction given by the small time step, forced by the explicit time integration method described in subsection **Background** and the very complex element computations due to the non-linearity, this method does require extensive computing power. As an example: Standard crash analysis for the design of a new car requires the multi-processor computing power of a NEC SX-5 to be solved overnight. The computational costs and the need to finish the simulation in approximately 10 hours limit the possible accuracy of the simulation. Current commercial versions of explicit finite element programs run only on shared memory machines and their scalability is poor. This required the design of a program structure which allows for easy implementation of parallelisation for distributed memory computers with good scalability. This was the basis of a thesis [3] at the IMAC-DGC.

BACKGROUND

Considering the (dynamic) equilibrium of a general continuous body in space, using the principle of virtual work and the standard finite element discretisation, we can write the discretised equilibrium equations as:

$$M\ddot{\mathbf{U}} + \mathbf{R} = \mathbf{F} \qquad (1)$$

where M is the mass matrix, U the vector of unknown displacements, R the vector of internal forces, F the external load vector and $\ddot{\mathbf{U}} = \partial^2\mathbf{U}/\partial t^2$. These equations can be integrated in time by means of an explicit predictorcorrector method:

$$\ddot{\mathbf{U}}_i = M^{-1}(\mathbf{F}_i - \mathbf{R}_i) \qquad (2)$$
$$\dot{\mathbf{U}}_{i+½} = \dot{\mathbf{U}}_{i-½} + \Delta_{ti}\,\ddot{\mathbf{U}}_i \qquad (3)$$
$$\mathbf{U}_{i+1} = \mathrm{Ui} + \Delta_{ti+½} + ½ + \Delta_t^2{}_i\ddot{\mathbf{U}}_i \qquad (4)$$

Where $\dot{U}_{i-H}$ is the predictor for the velocities. The predictor expressions are integrated in the total formulation. When using a lumped mass (diagonal) mass matrix the above equations are fully decoupled and easily solved in parallel.

PARALLEL IMPLEMENTATION

Although the time-integration equations are coupled, the computation of the internal forces is done on the element level, and therefore not decoupled. In order to minimize inter-process communication, domain decomposition is used. The decomposed domains are then distributed over the number of processors available. However, in the Fortran based existing (commercial) programs, this is not readily available. Due to the static nature of the Fortran 77 programming language, the implementation in existing codes is complicated and time consuming. Therefore a new program framework has been designed, based on object-oriented programming principles and implemented in C++. This structure treats the subdomains as independent objects. Communication between the subdomains is implemented in a top-level base class. This has the advantage that the communication is completely separated from the finite element computation and transparent to the programmer. Apart from the implementation of some very specific features (like the initial implementation of contact), the programmer is not bothered with the notion of multiple subdomains. This also makes the actual implementation independent of other parts of the program, as everything is concentrated in the communication class.

PERFORMANCE ON SWISS-T1

A simple benchmark problem has been chosen to evaluate the performance of the program on the distributed memory cluster Swiss-T1. This cluster allows the communication to be over standard Fast Ethernet as well as a custom-build network called TNet. The benchmark problem, an elastic free vibration problem, consists of a beam clamped on one side. The model consists of 64000 elements or approximately 222000 degrees of freedom. The domain composition was done along the length of the beam, as shown in Fig. 1. The chosen domain decomposition resulted in a constant amount of communication for each integration cycle, which allowed for a theoretical performance analysis. This was done for both the Fast Ethernet and TNet and compared with actual measured performance.

Fig. 1 – Benchmark for explicit program

THEORETICAL ANALYSIS

Amdahl's law [1] gives us the theoretical maximum speedup of a parallel program, given that all processors are the same. Including the time spent in communication, we can write for the speedup Sp of a program on P processors:

$$\frac{1}{\mathbf{S_p}} = \frac{1}{\mathbf{P}} + \frac{1}{\mathbf{r_p}} \qquad (5)$$

where we introduce the computation to communication ratio $\mathbf{r_p}$:

$$\mathbf{r_p} = \frac{T_s}{Tcomm_p} \qquad (6)$$

Using Fast Ethernet, we have a latency of approximately 500μs and a bandwidth of 10 MB/s. TNet gives us a latency of 20μs and a bandwidth of 50 MB/s. With the measured serial time per cycle of the benchmark problem T_s = 0.43s we get for the two communication ratios:

$$r_{pEthernet} = 80 \qquad (7)$$
$$r_{pTNet} = 470 \qquad (8)$$

This allows us to compute the theoretical parallel performance for a given number of processors.

RESULTS

P	Fast Ethernet		T-Net	
	S_p measured	S_p computed	S_p measured	S_p computed
2	1.92	1.95	1.98	1.99
4	3.64	3.81	4.04	3.97
8	6.84	7.27	8.53	7.8
16	12.1	13.3	–	15.5
32	19.6	22.8	–	30.0
64	–	35.6	–	56.3

Table 1 – Measured and computed speedup numbers

The results of the benchmark with the explicit finite element code are shown in Table l and Figs 2 and 3. Comparing the theoretical results with the measured performance shows a good coherence between the two. It

also demonstrates the advantage of the use of TNet. This is best expressed in the theoretical maximum speedup for an infinite number of processors: For Fast Ethernet this is limited to 80, while for TNet this is 470. This clearly shows the possibilities of parallel computing for explicit finite element simulations, given a suitable program structure.

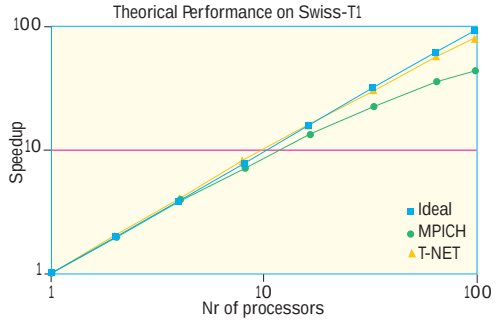


Fig. 2 – Theorical speedup benchmark on Swiss-T1

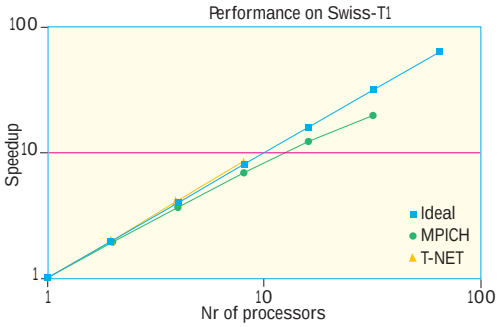


Fig. 3 – Mesured speedup benchmark on Swiss-T1

GELATINE IMPACT

Finally, we will show here one of the applications for the explicit finite element code which requires substantial computational resources. It concerns the simulation of an impact test, performed at the University of Oxford [2]. A composite plate, clamped on both sides, is impacted by a gelatine cylinder, as shown in Fig. 4. This simulation takes several hours to more than one day (depending on the chosen material parameters and mesh density) on a standard workstation, so parallel computing is needed in order to perform this analysis in a reasonable amount of time.

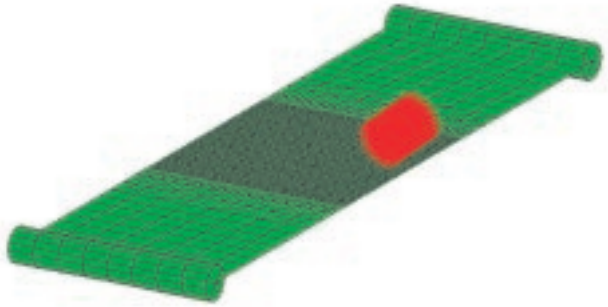


Fig. 4 – Gelatine impacting composite plate – initial solution

Fig. 5 shows the effect of the gelatine impacting the plate. The two main problems of this kind of simulations are clearly the behaviour of the gelatine and the composite damage material model. The destructive effect of the gelatine impact is shown in Figures 6 and 7. They show reasonable correlation between the simulation and actual test. Improvement of the material parameters should increase the predictive behaviour of the numerical simulation.

Fig. 5 – Gelatine impacting plate (deformation of plate not shown for clarity)

IMPACT FIGURES

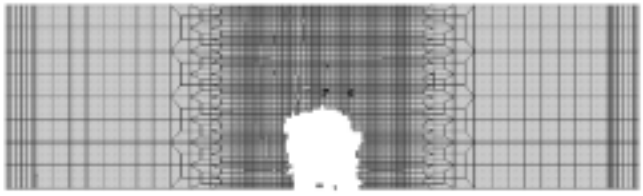


Fig. 6 – Simulation result: damage on plate at 390 m/s

Fig. 7 – Test result: damage on plate at 390 m/s

ACKNOWLEDGEMENTS

Part of the work described in this article has been performed for the CTI project Swiss-Tx. The funding of the CTI is duly acknowledged. The author would also like to thank SMR Corp., Bienne, Switzerland, for the use and support of the finite element program B2000 and the data management system MemCom.

REFERENCES

[1] Almasi G. S., and Gottlieb, *A., Highly Parallel Computing*, Benjaming/Cummings Publishing Company, Inc., Redwood City, California, 1994.

[2] Harding J., and Petrinic N., Documentation on oxford highspeed structure tests, Tech. rep., University of Oxford, Department of Engineering Science, l999.

[3] Volgers P., *High-Performance Explicit Transient Strucutral Analysis*, PhD thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2000. Thèse no. 2200

Flow Simulations on High Performance Computers using the NSMB flow solver

Jan B. Vos, EPFL-DGM Fluid Mechanics Laboratory and CFS Engineering SA

On se propose de présenter une vue d'ensemble du logiciel de calcul d'écoulements NSMB développé dans le cadre d'un consortium industriel et académique. Les objectifs de ce consortium sont de développer et d'améliorer un logiciel commun pour le calcul d'écoulements afin de promouvoir dans l'industrie l'utilisation de la simulation numérique des équations de Navier-Stokes. Pour l'industrie, l'un des goulots d'étranglement pour la résolution numérique de ces équations est le coût élevé de ce type de simulation.

On présente des résultats de benchmarks du logiciel NSMB pour 2 cas test effectués sur différents types de super-ordinateurs, incluant le NEC SX5 de Manno et le Swiss T1 de l'EPFL.

An overview of the NSMB flow solver wihich is developed in an academic-industrial consortium is given. The objectives of this consortium are to develop and improve a common flow solver called NSMB in order to advance the use of Navier Stokes simulations in industry. One of the bottlenecks of the use of Navier Stokes simulations in industry is the high costs of these simulations. Benchmark results of NSMB for 2 test cases using different computers including the NEC SX5 at the CSCS and the Swiss T1 at the EPFL, are given.

Introduction

Computational Fluid Dynamics (CFD) is used in industry since the 1960's. In the early days of CFD [1], solving the system of partial differential equations describing viscous fluid flows (the Navier Stokes equations) was not possible, and dramatic simplifications to these equations were made to obtain a system of equations which could be solved numerically in an acceptable turn around time. CFD was mainly used to complement experimental or full scale testing.

With the advent of vector supercomputers (Cray 1, Cyber 205) by the end of the 1970's, followed by (massively) parallel computers by the end of 1980's, combined with the progress in development of efficient numerical methods and the progress in physical modeling, the use of CFD in industry has grown considerably. Today the Reynolds Averaged Navier Stokes (RANS) equations are solved on a routine base, and CFD is used in the early design phases of many industrial products. In some industries, CFD has even almost replaced more time consuming and expensive experimental testing.

An example to illustrate this progress is the simulation of the flow over the full aircraft configuration AS28G (see Section *AS28G Full Aircraft*). In 1995, this simulation costed on a Cray J916 computer about 250 CPU hours when running on 1 processor, which in 1996 was reduced to 50 CPU hours when using 8 processors, and which was further reduced in 1997 to about 9 CPU hours by using a more efficient numerical scheme.

Despite the enormous progress in the last 40 years, CFD has still limitations:

- CAD cleaning and grid generation are time consuming, and require specialist skills;
- the need of fine grids to resolve correctly boundary layers results in substantial memory and CPU time requirements;
- the need of models to simulate turbulent flows introduces an uncertainty in the results, which is difficult to quantify. Complex turbulence models could reduce this uncertainty and improve the quality of the predictions, at the expensive of increased costs of the simulation.

To further advance the use of Navier Stokes simulations in industry towards a *virtual design office* progress is needed in the following areas[2]:

- more accurate physical models:
  - turbulence and transition;
  - chemistry and two phase flow models;
  - interaction with other physical phenomena (fluid-structure, fluid heat transfer, etc);
- more accurate and robust numerical algorithms:
  - to get a faster convergence;
  - to cluster grid points in regions of interest;
- better implementation and higher performance on vector/parallel computers.

Since 1992, a joint project between academic and industrial partners is being carried out to work on the above mentioned topics in a common flow solver called NSMB (Navier Stokes Multi Block). Today, the NSMB consortium is composed of 3 Universities (ENSAM in Paris, EPFL in Lausanne, KTH in Stockholm), one research establishment (CERFACS in Toulouse) and three industrial partners (Aerospatiale-Matra in Paris and Toulouse, CFS Engineering in Lausanne, and SAAB Aerospace in Linköping).

This paper first gives an overview of the NSMB flow solver, which is followed by the presentation of benchmarks results of NSMB on different computer platforms, including the NEC SX5 and the Swiss T1.

NSMB OVERVIEW

DESIGN CHOICES AND PROGRAM STRUCTURE

NSMB was initially developed at EPFL in 1991. One of the principal design choices was to use structured multi block grids. Structured grids were adopted for their higher precision in boundary layers, and their simpler data structures compared to unstructured grids. The multi block approach was adopted for two reasons, first to facilitate the mesh generation for complex geometries, and second to permit an easy use of parallel computers by solving the equations in different blocks in parallel.

NSMB was developed on top of a data base system, called MemCom [1]. MemCom is an object oriented data management system for memory and memory-to-disk data handling. The main advantage of using a data base system is that for large scale multi block flow simulations (i.e. more than 100 blocks and over 1 Million grid points), access to the independent blocks is extremely fast, and almost independent of the block number. From the user point of view, the data base file appears as a single UNIX file, hence all the information related to the simulation can be directly saved on an archival system without possible loss of information.

NSMB is written in Fortran 77, using a Dynamic Memory Manager (DMM) included in the MemCom library to allocate at run time the necessary storage of the arrays in NSMB. When running on distributed memory computers, the DMM allocates the memory on each node of the computer. Dynamic memory allocation is since a few years available in the Fortran 90 programming language, and it was one of the innovative features of NSMB at the time NSMB was designed.

NUMERICAL MODELLING

The Navier Stokes equations describe the conservation of mass, momentum and energy. They are discretized in space using the finite volume method, which was especially designed for the discretization of conservation equations. NSMB includes a variety of schemes to approximate the inviscid fluxes at the face of a finite control volume[2]. Among them are the classical second order central scheme with added artificial dissipation, second and third order upwind schemes (the Roe scheme, schemes of the AUSM family), and higher order schemes as a fourth order central scheme and a fifth order WENO scheme.

After space discretization, the Navier Stokes equations can be written as a system of coupled ordinary differential equations in time. They are integrated in time using the explicit Runge Kutta scheme, a scheme widely used in CFD for its simplicity and good stability properties, or the LU-SGS semi implicit scheme. Convergence acceleration procedures as local time stepping, implicit residual smoothing, multi grid and preconditioning can be used for steady state calculations, while the dual-time stepping technique is available for unsteady calculations [4].

TURBULENCE MODELLING

One of the major sources of uncertainty in CFD is the necessity to adopt a turbulence model when simulating turbulent flows. Although it is believed that the Navier Stokes equations can describe turbulence, the time and length scales encountered in turbulence are several orders of magnitude smaller than the time and length scales characteristic for practical applications. As example, the grid needed to capture all the turbulence scales in the flow around a full aircraft would require about 10^{15} grid cells, which is beyond the capacity of any computer in the coming decades. Moreover, one is not always interested in the instantaneous values of the flow variables.

Statistical methods are therefore used to describe turbulent flows, and each flow variable in the conservation equations is decomposed into an average and a fluctuating part. The resulting equations are then averaged, and terms involving products of fluctuating variables are modelled using a turbulence closure model. Turbulence modelling has been studied for more than 100 years, and several well tested turbulence models are available today. NSMB includes algebraic turbulence models as the Baldwin-Lomax and the Granville model, the Spalart-Allmaras 1-equation model, which is very popular in aerospace, and the $k-\varepsilon$ and $k-\omega$ two equation models. Algebraic turbulence models do not require the solution of an additional conservation equation, while the 1 and 2 equation turbulence models require the solution of respectively 1 and 2 additional partial differential equations.

ALE FORMULATION

NSMB includes the possibility to solve the equations on moving grids using the ALE (Arbitrary Lagrangian-Eulerian)[5] approach. This possibility has several applications:

- to solve the Navier Stokes equations in a rotating frame of reference;
- to compute the dynamic aerodynamic coefficients, needed to derive the control laws of aircraft;
- buffeting and flutter simulations;
- coupled fluid - structure simulations, in which the movement from the grid is coming from the deformation of the structure.

PARALLEL COMPUTING

The NSMB code was parallelized in the EC ESPRIT III project Parallel Aero, which was part of EUROPORT 1, and finished in 1996 [6]. Two design choices were made in the development of the parallel version of NSMB: first the domain partitioning is executed before the execution of parallel NSMB using the MB-Split decomposition tool, and second the parallel implementation was based on the master-slave paradigm using PVM or Parmacs. In the frame of the CSCS/SCSC-NEC Joint Program in Application Porting and Development, this latter design choice was changed to the SPMD paradigm using MPI [7].

Domain Partitioning Tool MB-Split

One of the most important functionalities of a domain decomposition tool is to distribute the blocks on the parallel computer such that a good load balance is obtained between the different processors. The complexity of the domain decomposition process, especially for large number of blocks, requires a tool which automatically generates and updates boundary conditions when splitting a block.

The MB-Split domain decomposition tool [8] was developed at KTH-Stockholm during the Parallel Aero project. It is a programme written in C++, which reads a MemCom data base with an arbitrary number of blocks, and generates a new MemCom data base with a new number of blocks such that a balanced calculation is obtained on a parallel computer. MB-Split automatically corrects the boundary conditions on blocks which are split, and generates the boundary conditions on newly created blocks. Blocks are split using either recursive edge bi-section, or greedy load balancing. MB-Split can also split a solution saved in the MemCom data base, and if required, MB-Split can merge a splitted data base, with solution, back to the original data base.

Single Processor Optimization

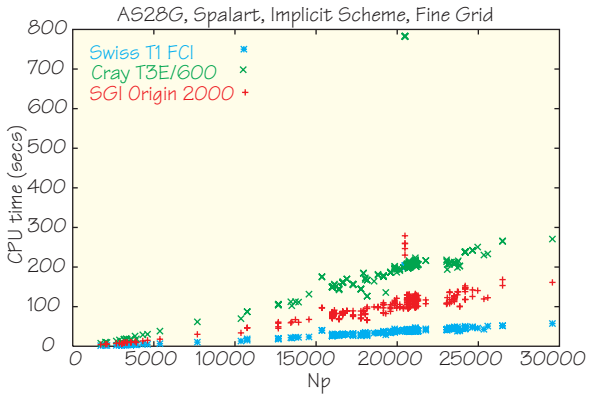


Fig. 1 – CPU time as function to the number of points in a block, AS28G Full Aircraft, 290 block grid

NSMB was designed for running on vector computers, and the most time consuming routines are written to have vectorizable do-loops over all grid points in a block. This yields an excellent performance on vector computers, reaching about 40 to 50% of the peak performance. On RISC architectures the performance of NSMB is much less, reaching between 5 and 15% of the peak performance. There are several reasons for this. First, when using do-loops over all grid points, unnecessary work is carried out in grid cells used for imposing the boundary conditions. This additional work is not a problem on vector computers, where the gain by using long vectors largely compensates this overhead. This is not the case on RISC architectures, and for this reason most time consuming routines in NSMB are now available as VECTOR and RISC version. The second reason for the poor performance of NSMB on RISC architectures are memory and cache contention problems because data loaded in cache cannot be re-used. When the problem size is reduced (i.e. when smaller blocks are used), the performance is increased because all the data can fit into cache. A typical example of cache problems is shown in Fig. 1, which shows the CPU time as function of the block size for the AS28G test case on 3 different RISC architectures.

As can be seen, the CPU time increases almost linearly with the block size, except for a few blocks which have a size of around 20000 cells. For these blocks, the CPU time is suddenly about 4 to 5 times larger. The size of these blocks is 32 x 32 x 20, a multiple of 1024 (1K). Since the cache memory is a multiple of 1024, this performance degradation can be attributed to cache misses. Data required for calculations do not fit into cache, and need to be reloaded each time it is used. Since reloading data into cache requires more clock cycles than accessing data in cache, the CPU time is increased.

On-going Developments

All the partners of the NSMB consortium work on various aspects of NSMB. Here a non-exhaustive list of items partners of NSMB are working on is given:

- Explicit Algebraic Reynolds Stress Models (EARSM) turbulence models seem to be very promising since they remove basic short comings of 2 equation turbulence models while only marginally increasing the computational costs;
- Large Eddy Simulation (LES) is another promising technique to simulate turbulent flows. The large scales of turbulence are resolved using the unsteady Navier Stokes equations, while the small scales are modelled using subgrid scale models. LES simulations are unsteady, hence the computational costs are increased considerably compared to solving the RANS equation;
- Coupling with the Maxwell equations to simulate the flow in a plasma torche;
- Automatic Mesh Refinement. Work is underway on building an adaptative mesh environment based on the subblock refinement technique. A coarse multi block mesh is adapted by adding finer sub-blocks in regions which need a better resolution;
- Non-equilibrium chemistry for hypersonic flows. This development is needed for applications involving future re-entry space vehicle;
- Higher order numerical schemes. The main motivation of this development to increase the precision of the simulation, which may lead to the use of coarser grids. Also higher order schemes are needed to simulate the vortex flows in the wake behind aircraft;
- Development of a RISC optimized version, in which the declaration of arrays in NSMB is changed to permit a better use of the cache;
- Coupling with heat transfer and structural mechanics codes;
- Object Oriented Programming languagues. At present, NSMB has become a complex piece of software, and has reached the limits of what can be done using the Fortran 77 programming language. A project is underway to

build a prototype of NSMB using the object oriented features of Fortran 90.

BENCHMARK SIMULATIONS

Benchmark calculations using NSMB version 5.02 were made in August 2000 using 2 test cases from the Parallel Aero project:

- A-Airfoil. The A-Airfoil is a 2D test case, and the grid can be split into equal sized blocks. It is a good test case to assess the network of a computer;
- AS28G Full Aircraft Configuration, which is representative of an industrial test case.

THE COMPUTERS USED

The benchmark calculation were made on computers at the Swiss Federal Institute of Technology in Lausanne (EPFL), the Swiss Federal Institute of Technology in Zürich (ETHZ), the Swiss Center for Scientific Computing (CSCS), the University of Linköping (LIU), and at CFS Engineering.

Characteristics of the computers used

Table 1 summarizes the characteristics of the different computers used for the benchmark calculations. The table also includes the estimated single processor performance of NSMB. On vector architectures as the NEC SX4 and SX5, the NSMB performance can reach between 40 and 50% of the peak performance. On RISC architectures, NSMB reaches around 16% of the peak performance on the SGI Origin 2000 and the Swiss T1, but it is only 10% on the PC Cluster, and even lower on the Cray T3E. As can be seen from Table 1, both the Swiss T1 and the NEC SX5 were installed in the year 2000, and have a theoretical peak performance of 64 Gflops. Almost all benchmark calculations were run during production time (loaded systems), but on most computers queues were available to ensure that the allocated resources were not shared with other processes.

Comments from a user viewpoint

In this section, some remarks from the user point of view on using the different computers are given. The remarks reflect the personal opinion of the author. No remarks could be given for the Cray SV1, since the jobs were not run by the author himself.

Most people needing large computer resources have accounts on several computer systems. For production runs, they in general use the computer which gives them the most results per elapsed time. For testing and debugging the code, they use the computer with the best debugger and the fastest compiler.

A user of different computers expects (or hopes) that he may use the same user name on all computers, and that he finds on each computer the same environment (shell, job scheduling software). Errors are made when commands are just slightly different from one computer to the other. A typical example is that for the Cray J90 or SV1, you need to use the command `mpirun -nt 2 nsmb5.02.mpi` to run nsmb on 2 processors, while on all other computers it is `mpirun -np 2 nsmb5.02.mpi`. However the command `mpirun -np 2 nsmb5.02.mpi` is accepted on the Cray J90 but should not be used for shared memory parallelism.

In the following a list of observations is given

- **Compilation of NSMB**
  Fast on the Swiss T1, SGI Origin 2000 and the PC Cluster (around 5 minutes), slow on Cray and NEC computers (around 2 hours).
- **Environment**
  CSCS has still the policy to give users not the user name they have on other platforms. tcsh was available on all computers except the Origin 2000 at EPFL.

	Cray T3E	Cray J90	Cray SV1	NEC SX4	NEC SX5	SGI Origin	PC 500Mhz	Swiss T1
	LIU	EPFL	ETHZ	CSCS	CSCS	EPFL	CFS	EPFL
Year	1995	1995	1998	1996	2000	1996	1999	2000
# of nodes	256	8	16	12	8	80	6	64
Shared memory (Gb)		4	16	8	64	(20) NUMA		
Memory per node (MB)	128/256					(256)	384	512
Cache (Kb)	8					32	128	64
Node peak performance (Mflops)	600	200	1000	2000	8000	390	500	1000
Installed peak performance (Gflops)	153.6	1.6	16.0	24.0	64.0	31.2	3.0	64.0
Node NSMB performance (Mflops)	25-40	60-70	190-220	500-1100	1500-3500	50-70	40-50	90-160

Table 1 – The computers used in the NSMB benchmarks

Job execution

- Timing results on the SGI Origin 2000 showed variations up to 50%, only the best results are shown;
- Results on the NEC SX4, NEC SX5 and Cray SV1 were the easiest obtained;
- The Swiss T1 is still a somewhat experimental computer. Several jobs had to be re-run since another job blocked one of the nodes, job scheduling software not optimal yet, not possible to run on 1 node with FCI, number of nodes should be a power of 2. Impossible to use MPICH with more than 16 nodes. Communication problems appeared using MPICH for the AS28G test case, and only a few steps could be made. For the AS28G testcase on 8 processors, it happened that one of the nodes started to swap due to lack of available memory, leading to very long elapsed time of the job;
- Large waiting time to get results on 64 nodes on the T3E due to the heavy load on this machine.

A-AIRFOIL

The A-Airfoil is a well known test case for testing turbulence models for separated flows which occur near the trailing edge at high angles of attack. A view of the grid used for the calculations is given in Fig. 2.

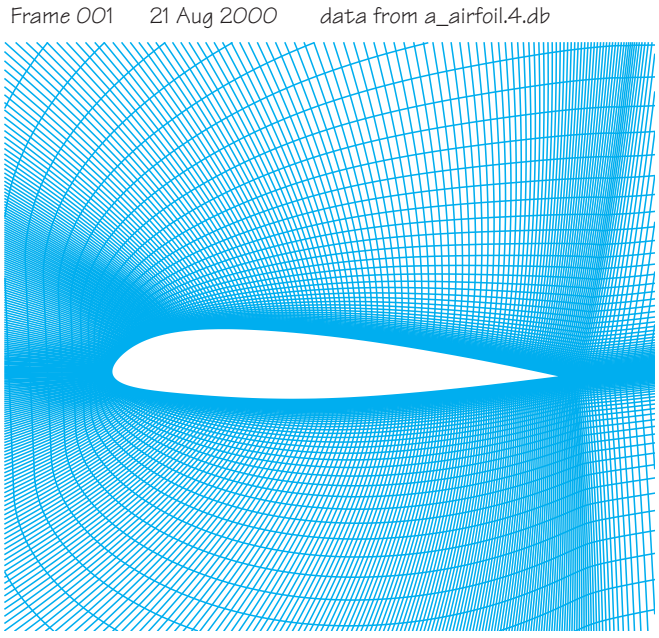


Fig. 2 – Grid A-airfoil

The flow is turbulent, and the calculations were made using the Baldwin-Lomax algebraic turbulence model. A grid of 512 x 128 grid points was used, which was split into 8 blocks for computations on vector computers, and 64 blocks for computations on RISC based architectures. The central scheme using artificial dissipation was used for the space discretization, and the equations were integrated in time using the LU-SGS semi-implicit scheme. Only 100 time steps were made for the benchmark calculations.

Figs. 3a and 3b show the speedup curves up to respectively 8 and 64 processors.

As can be seen from Fig. 3a, a good speed-up up to 8 processors has been obtained on all computers except the Swiss T1 using MPICH. It should be remarked that no parallel queues were available on the Cray SV1, and for the NEC SX5 running on 8 processors, the timing results are influenced by interactive users and the operating system. This explains the somewhat lower speedup using 8 processors on these 2 computers.

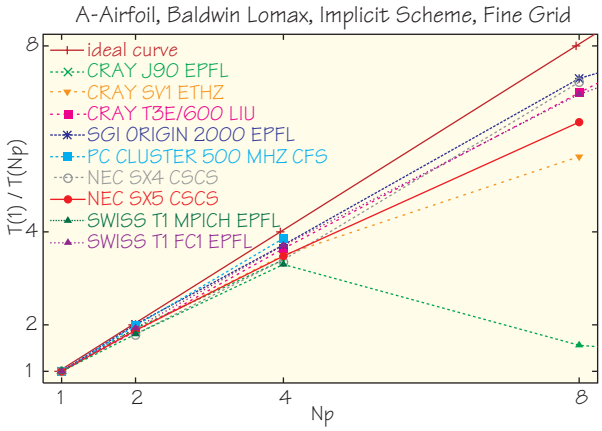


Fig. 3a – Speedup curves A-Airfoil calculations

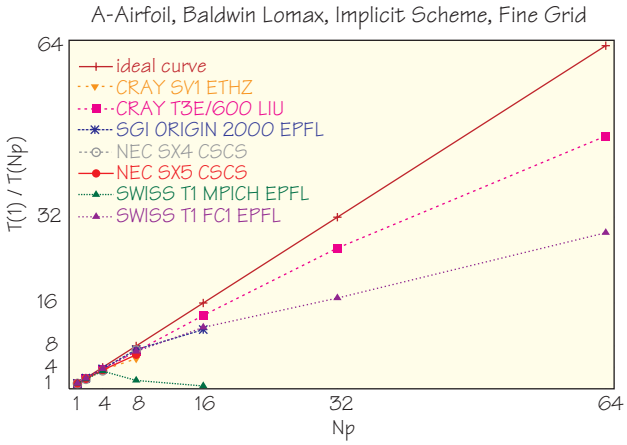


Fig. 3b – Speedup curves A-Airfoil calculations

Looking to the speedup curve up to 64 processors, it can be seen that the Cray T3E still yields a good speedup, while on the Swiss T1 the speedup levels off after 16 processors. The reason for this is twofold. First, the Swiss T1 is about 4 times faster than the Cray T3E in the computing part, hence the ratio time spent in the calculation to time spent in communication is better on the Cray T3E. Second, the network of the T3E is faster when using more than 8 processors. This is illustrated in Fig. 4, which shows the elapsed time per timestep, and the total communication time (including processor synchronization) for the calculations made.

Fig. 4b clearly shows that the time spent in the communication is reduced when using more processors on

the Cray T3E. This is to be expected, since the blocks are distributed over more processors, and the amount of communication for each processor is therefore decreasing. On the Swiss T1 using FCI, one can observe that the time spent in communication is slightly decreasing when going from 2 to 8 processors, but then suddenly increases, remains constant, and then decreases slightly. It should be remarked that the time spent in communication is lower on the Swiss T1 than on the Cray T3E up to 8 processors.

When looking to the elapsed time per time step, Fig. 4a, it can be seen that the results of the Swiss T1 are close to those of the Cray SV1 up to 8 processors. The results of the Cray J90, SGI Origin 2000, Cray T3E and the PC Cluster are also very close.

The NEC SX5 is only about 2 times faster than the NEC SX4 for this test case. The NEC SX5 is a less balanced computer than the NEC SX4, requiring long vectors to approach the peak performance. Since the block size (hence vector length) for the a-airfoil test case is rather small, the performance on the SX5 for this test case is rather poor. Fig. 4a shows that the elapsed time for the 64 processor Swiss T1 using FCI approaches the elapsed time on the 8 processor NEC SX5.

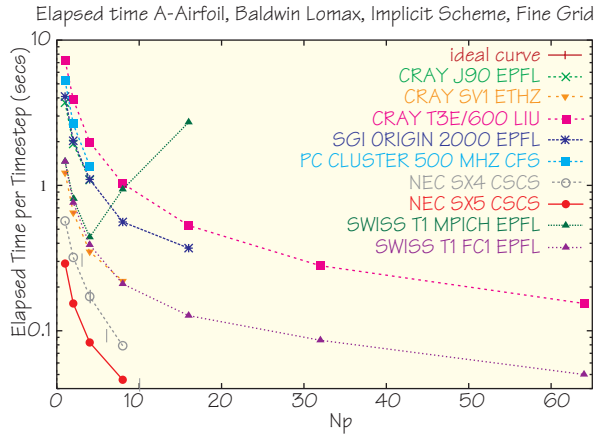


Fig. 4a – Elapsed and Communication Time A-Airfoil calculations

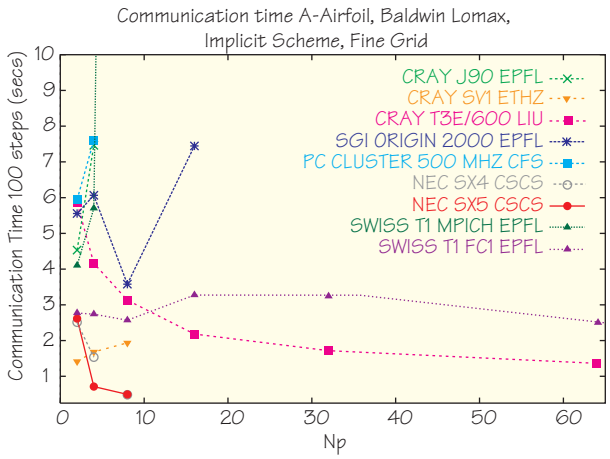


Fig. 4b – Communication Time A-airfoil calculations

Fig. 5 shows the Mach number contours over the A-airfoil. At the trailing edge on the leeward side of the airfoil, the separated area is clearly visible.

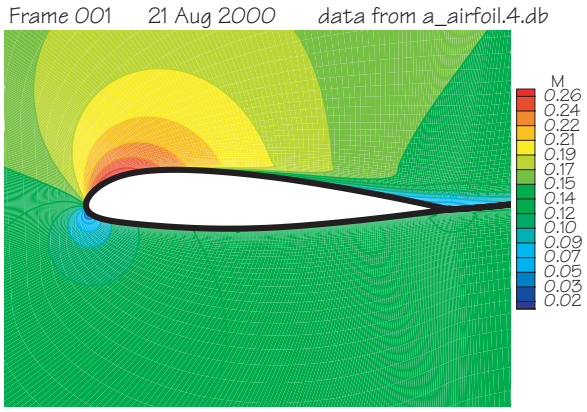


Fig. 5 – Mach number contours A-airfoil, $M_\infty = 0{,}15$, $\alpha = 7°$, $Re/m = 2.1\ 10^6$

AS28G FULL AIRCRAFT

The AS28G is a wing-body-pylon-nacelle generic aircraft, which has been extensively tested in a windtunnel. The configuration is representative of an aircraft in cruise conditions, and one of the principal interests of this test case is the engine integration. Fig. 6 shows the surface grid of the AS28G.

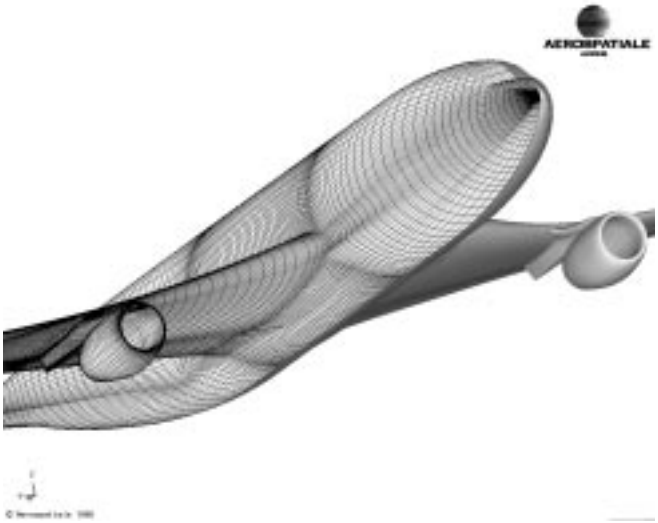


Fig. 6 – Surface grid AS28G

The grid for this test case is composed of 62 blocks, and contains in total 3.5 Million grid points. The largest block has about 300'000 points, the smallest 729 points. The flow is turbulent, and the benchmark calculations were made using the 1-equation Spalart-Allmaras turbulence model. The calculation required about 2.0 Gbytes of memory when running on 1 processor, substantially more when running in parallel due to the allocation of send buffers and the fact that temporary storage needs to be allocated on each processor. For example, the calculation required 3.6 Gbytes

of memory when running on 8 processors on the NEC SX4. The original 62 block grid was used on the Cray SV1, the NEC SX4 and the NEC SX5. A 290 block grid was used on the Origin 2000, the Cray T3E and the Swiss T1. Fig. 7 shows the distribution of these 290 block over 32 processors.

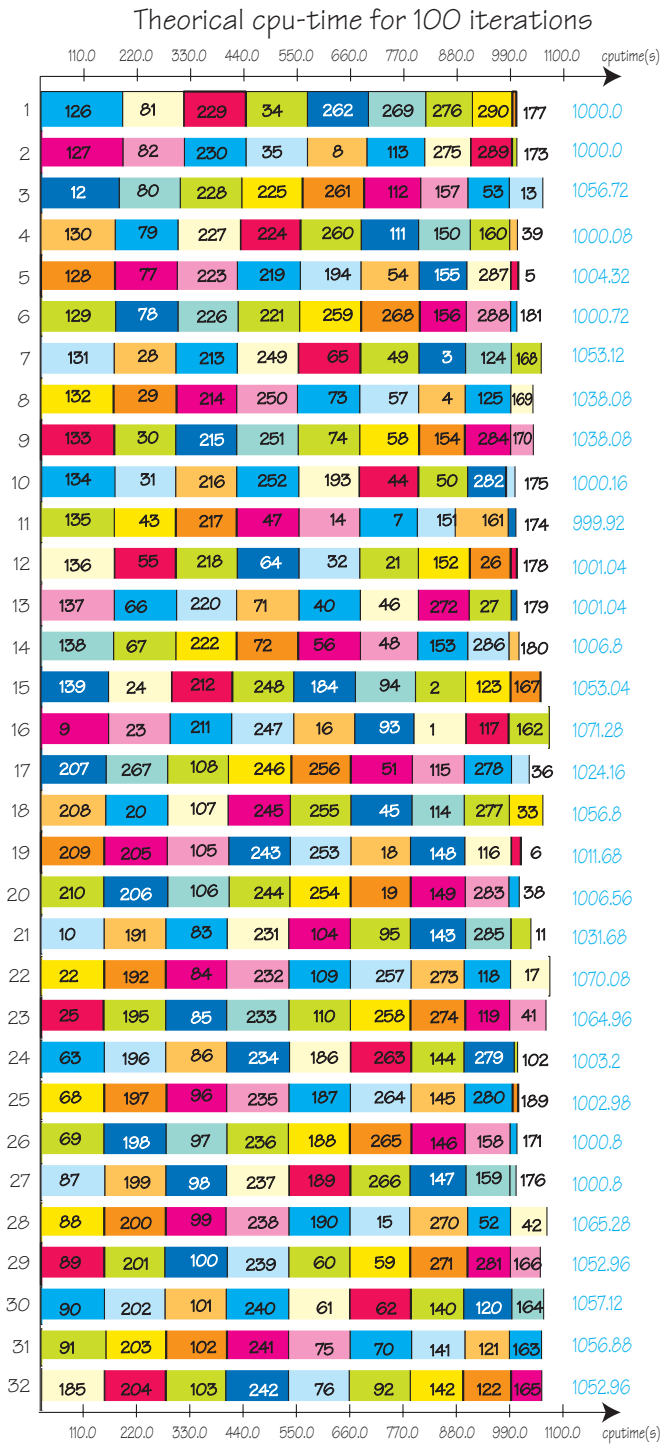


Fig. 7 – Distribution 290 block grid AS28G over 32 Processors

Other block decompositions were used on 16 and 32 processors, but in all cases, the elapsed time obtained with the 290 block grid was the smallest. The reason for this is the better use of the cache when using small blocks, see Section *Single Processor Optimization*.

Fig. 8 shows the speed-up curves for the AS28G calculations. It should be remarked that it was not possible to run on a single processor on the SGI Origin 2000, Cray T3E and the Swiss T1. The speed-up results on these computers are normalized with the result on the lowest number of processors which could be used.

Fig. 8a shows the speed-up curves up to 8 processors. As can be seen, a super linear speed-up was obtained on the SGI Origin 2000, which is probably due to the bad performance of NSMB for this test case on 2 processors. The reason for this bad performance is the large amount of memory needed, coupled with the NUMA memory architecture of the SGI 2000. The Cray SV1 also showed a super linear speedup for 2 processors for an unknown reason. As for the A-airfoil, the SX5 results on 8 processors are influenced by interactive users and the operating system.

Fig. 8b shows the speed-up curves up to 64 processors. With MPICH on the Swiss T1, only 8 and 16 processors could be used. The results on 8 processors was used for the normalization, and since this computation was rather slow, a super linear speed up is obtained when using 16 processors. A close to linear speedup is obtained on the Cray T3E and the Swiss T1 between 16 and 64 processors.

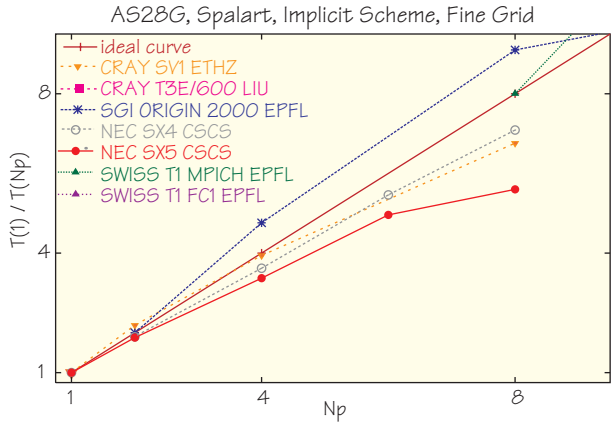


Fig. 8a – Speedup curves AS28G calculations

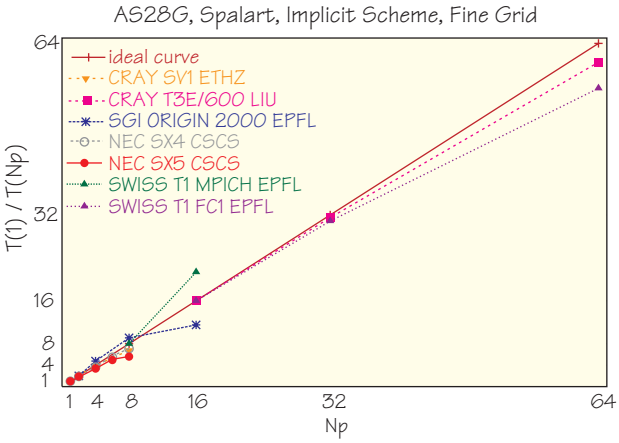


Fig. 8b – Speedup curves AS28G calculations

Fig. 9a shows the elapsed time per time step. As can be seen, the NEC SX5 is the fastest computer, and the elapsed time on 4 processors of this machine is the same as 64 processors on the Swiss T1 using FCI. The Cray T3E is about 4 times slower than the Swiss T1 using FCI. One can observe that when using 16 processors, the elapsed times on the Cray T3E, SGI Origin 2000 and the Swiss T1 using MPICH are very close.

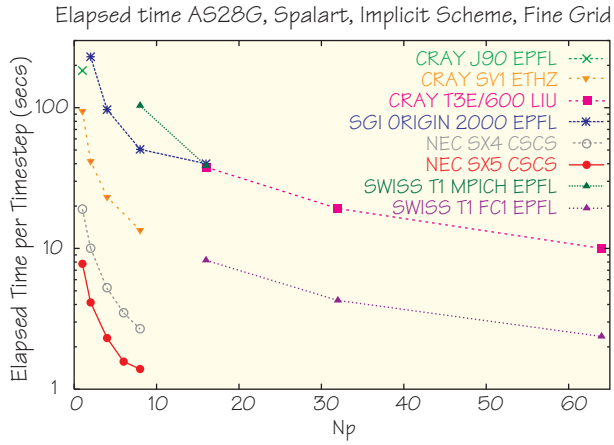


Fig. 9a – Elapsed Time AS28G calculations

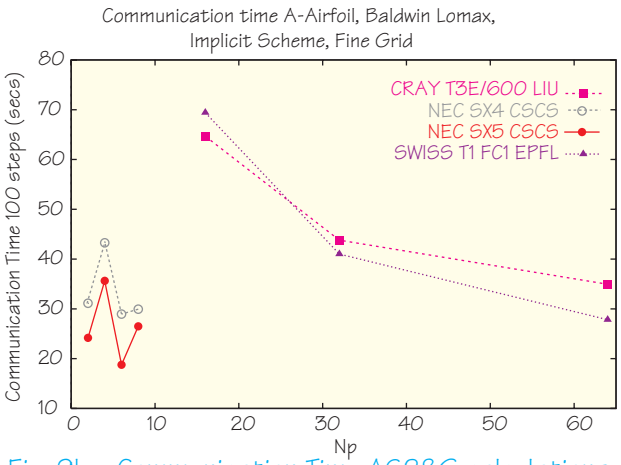


Fig. 9b – Communication Time AS28G calculations

Fig. 9b shows the time spent in the communication (and synchronization) for the NEC SX4 and SX5, the Cray T3E and the Swiss T1 using FCI. It can be seen that communication time on the NEC SX4 and SX5 follow the same zig-zag pattern. The run using 4 processors is probably slightly less balanced than using 2 and 6 processors, which increases the time needed for synchronization. Comparing Cray T3E and the Swiss T1 shows that the time spent in communication and synchronization is close on both machines.

Fig. 10 shows the surface pressure and the stream lines around the pylon-nacelle. The influence of the nacelle on the flow is clearly visible in the stream lines, and on the surface pressure.

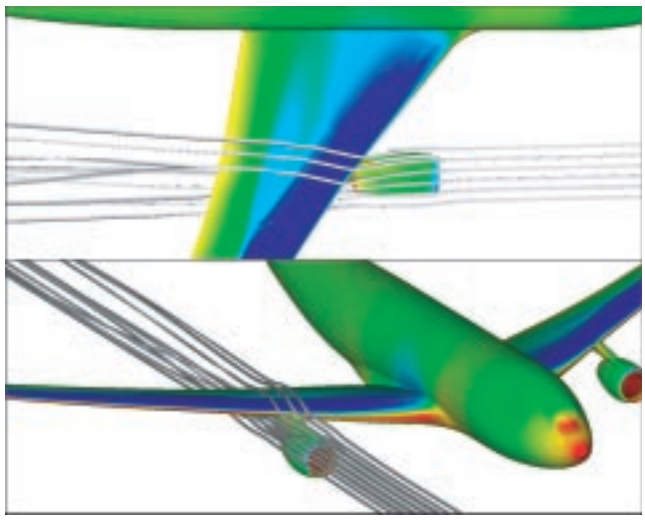


Fig. 10 – Stream Lines and Surface Pressure AS28G

CONCLUSIONS

A short overview of the NSMB flow solver was given. In the 9 years since the first version, NSMB has grown to a complex piece of software which is used in industry to simulate a wide variety of different flows, including flows over aircraft (AS28G, Airbus Aircraft, FA-18 of the Swiss Army), re-entry space vehicles (ARD and Soyuz capsule, EXTV re-entry vehicle) to internal flow problems as air intakes and flows in compressors. Development of NSMB is an ongoing effort at all the partners involved, with as objective to improve the prediction capabilities of NSMB, and to reduce the costs of Navier Stokes simulations.

NSMB was ported to different computer platforms, and benchmark calculations were made for two test cases. The major objective of these benchmark calculations was to assess the performance of NSMB on the NEC SX5 and the Swiss T1 computers. Both computers were installed this year, and both have a installed peak performance of 64 Gflops.

From the benchmark results, the following conclusions can be drawn:

- the use of MPICH is not recommended on the Swiss T1;
- distributed memory massively parallel computers require a fast network to obtain a good performance;
- the FCI network on the Swiss T1 is fast up to 8 processors. For larger number of processors the network on the Cray T3E (a computer which is 5 years old !) is at least as fast if not faster than FCI;
- the Compaq Alpha chips on the Swiss T1 are fast, and a good single processor efficiency for NSMB was obtained. This efficiency can be further improved in the future with a RISC optimized NSMB version;

- the NEC SX5 is a less balanced computer than the NEC SX4, resulting in a reduction of the single processor efficiency. For NSMB, the NEC SX5 is only about 2 to 2.5 times faster than the NEC SX4;
- the elapsed time on the NEC SX5 was the smallest for both benchmark testcases.

From a user point of view, it is remarked that the Swiss T1 is still a somewhat experimental computer, and improvements need to be made to the job-scheduling software to make it a production system. The NEC SX5 is a production computer.

One of the on-going discussions in the High Performance Computing world is which type of computer yields the fastest performance: distributed memory massively parallel RISC architectures or shared memory (moderately) parallel-vector architectures. The Swiss T1 belongs to the first type of computers, the NEC SX5 to the second. It is concluded that for NSMB, shared memory parallel-vector architectures yields the fastest results. However, the difference between the two types of architectures is closing rapidly.

As final conclusion, it is remarked that computer vendors should strive to build well balanced computers. For distributed memory computers it is the network and cache memory which are primordial, for shared memory parallel-vector architectures, it is the memory access and memory bandwidth which have a large influence on the processor efficiency.

ACKNOWLEDGEMENTS

Aerospatiale Avions is acknowledged for providing the grids for the A-Airfoil and for the AS28G.

The people from the Central Computing Facility at EPFL are acknowledged for providing their support and help to run NSMB on the EPFL Computer facilities.

Olivier Byrde from Cray Research is acknowledged for running the test cases on the Cray SV1 at ETHZ.

Jean Favre from CSCS is acknowledged for providing the figure of the result of the AS28G calculation.

The people from the operations group at CSCS are acknowledged for allowing me to use 8 processors on the NEC SX5.

The CTI/KTI is acknowledged for financing the Swiss Tx project at EPFL.

REFERENCES

[1] Raj, P., *CFD at a Crossroads: An Industry Perspective*, In: Frontiers of Computational Fluid Dynamics 1997, Eds. D.A. Caughey and M.M. Hafez.

[2] Vos, J.B., Rizzi, A., Corjon, A., Chaput, E. and Soinne E., *Recent Advances in Aerodynamics inside the NSMB (Navier Stokes Multi Block) consortium*. AIAA Paper 98-0225, 1998.

[3] Merazzi, S., *MemCom, An Integrated Memory and Data Management System* - MemCom User Manual Version 6.0, SMR TR-5060, March 1991, SMR Corporation, P.O. Box 41, CH-2500 Bienne.

[4] Vos, J.B. , Leyland, Van Kemenade, V. , Gacherieu, C., Duquesne, N., Lotstedt, P., Weber, C., Ytterström, A., and Saint Requier, C.NSMB Handbook Version 4.5, 1998.

[5] Hirt, C., Amsden, A. and Cook, J., *An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds*, Journal of Computational Physics, Vol. 14, 1974, pp. 227-253.

[6] Vos, J.B.,Van Kemenade, V., Ytterström, A. and Rizzi, A.W., *Parallel NSMB: An Industrialized Aerospace Code for Complete Aircraft Simulations*. In: Proceedings of Parallel CFD Conference 1996, Eds. P. Schiano et al., North Holland, 1997.

[7] Vos, J.B., Haberhauer, S. and Ytterström, A. Industrial Flow Simulations using Different Parallel Architectures. In: Proceedings Parallel CFD Conference 1997, Eds. D. Emerson et al., North Holland, 1998.

[8] Ytterström, A., *MB-Split A structured mesh partitioning tool for load balancing on MIMD-Computers*, Proceedings of the Second ECCOMAS Conference on Numerical Methods in Engineering, September 1996, Paris, France, pp. 803-809.

NOTE FROM THE EDITOR

As mentioned by the author of the article above, the EPFL Swiss T1 was and is still an experimental machine. Indeed, work is being done to improve the communication library and the job scheduling system to make the T1 an efficient production machine. ■

RÉDACTEUR EN CHEF	*Trach-Minh Tran, SIC-EPFL*	EDITOR
MISE EN PAGE ET GRAPHISME	*Appoline Raposo de Barbosa, SIC-EPFL*	TEXT PROCESSING AND LAYOUT
ADRESSE	*Service informatique central EPFL* *MA-Ecublens Case Postale 121* *CH - 1015 Lausanne*	ADDRESS
TÉLÉPHONE	*(021) 693 22 11*	PHONE
TÉLÉCOPIE	*(021) 693 22 20*	FAX
ADRESSE ÉLECTRONIQUE	*trach-minh.tran@epfl.ch*	E-MAIL
ADRESSE WEB	*sic.epfl.ch/publications*	WEB LOCATION