Published in *Lecture Notes in Computer Science n°* 1697. page 341-348. Springer editor, 1999

Using PMD to Parallel-Solve Large-Scale Navier-Stokes Equations. Performance Analysis on SGI/CRAY-T3E Machine

Jalel Chergui

Institut du Développement et des
Ressources en Informatique Scientifique.
CNRS/IDRIS, Bât. 506, BP167
F-91406 Orsay cedex, France
Jalel.Chergui@idris.fr

Abstract. PMD (Parallel Multi-domain Decomposition) is an MPI based Fortran 90 module which objective is to parallel-solve positive definite linear elliptic second order equations. It has been used to solve unsteady Navier-Stokes equations in order to simulate an axisymmetric incompressible viscous fluid flow inside a centrifugal pump. In this paper, we will present a brief description of the implementation and will discuss the performance measurements obtained on the SGI/CRAY T3E parallel machine.

1 Introduction

PMD [CHE 98] is an MPI [GRO 96] based Fortran 90 module which was developed at IDRIS[1] in cooporation with LIMSI[2]. It's a public domain software[3] which objective is to parallel-solve elliptic linear second order equations. This arises the question: How PMD can be used to solve parabolic non-linear time-dependent systems as the unsteady Navier-Stokes equations ?
The answer, which will be discussed in this paper, consists in applying a classical time-discretization procedure which transforms the continuous time-dependent parabolic non-linear equations into a time-discretized elliptic linear systems. In turn, this arises the choice of the time-discretization scheme which has to be consistent with the continuous equations and to ensure an acceptable accuracy when advancing the flow in time. In the next paragraphs, we will review the Navier-Stokes equations and establish the time-procedure used in this study to discretize these equations.

1.1 Mathematical model

We shall assume an axisymmetric flow of a newtonian incompressible viscous fluid. The Navier-Stokes equations are formulated in terms of stream function ψ, azimuthal component ω of the vorticity and tangential component v of the velocity. This leads to the following set of equations formulated in a cylindrical coordinate system $(r, z) \in [R_{in}, R_{out}] \times [0, 1]$ (figure 1):

[1] Institut du Développement et des Ressources en Informatique Scientifique, CNRS/IDRIS, Bât. 506, BP167, F-91406 Orsay cedex, France.
<http://www.idris.fr>

[2] Laboratoire d'Informatique et de Mécanique pour les Sciences de l'Ingénieur, CNRS/LIMSI, UPR 3251, BP133, F-91406 Orsay cedex, France.
<http://www.limsi.fr>

[3] PMD can be retrieved following this link:
<http://www.idris.fr/data/publications/PMD/PMD.html>

$$\overbrace{\frac{\partial \omega}{\partial t} - \frac{1}{R_e}\left(\nabla^2 - \frac{I}{r^2}\right)\omega}^{\text{Diffusive terms}} = \underbrace{-\frac{\partial(u\omega)}{\partial r} - \frac{\partial(w\omega)}{\partial z} + \frac{1}{r}\frac{\partial v^2}{\partial z}}_{\text{Convective terms}} \tag{1}$$

$$\overbrace{\frac{\partial v}{\partial t} - \frac{1}{R_e}\left(\nabla^2 - \frac{I}{r^2}\right)v}^{\text{Diffusive terms}} = \underbrace{-\frac{\partial(u\ v)}{\partial r} - \frac{\partial(wv)}{\partial z} - \frac{2\ u\ v}{r}}_{\text{Convective terms}} \tag{2}$$

$$\left(\frac{\partial^2}{\partial z^2} + r\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\right)\right)\psi = r\omega \tag{3}$$

Where u and w are respectively the radial and axial components of the velocity. ∇^2 and I are the Laplace and the Identity operators respectively. $R_e = \frac{\Omega H^2}{\nu}$ is the Reynolds number (based on the angular velocity Ω of the rotor, on the distance H between the two disks and on the kinetic viscosity ν) which is the only free dimensionless parameter in this problem.
All physical boundary conditions are of DIRICHLET type and stem from the adherence of the fluid particles to the walls.

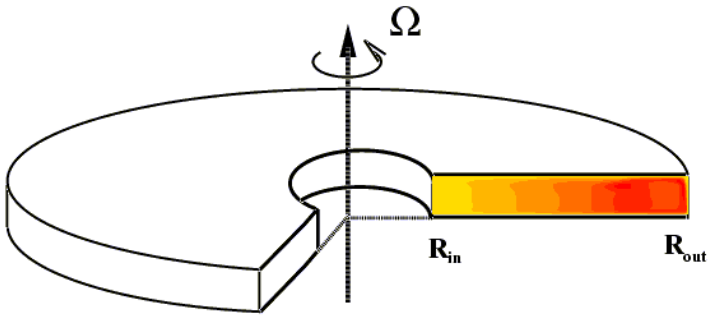

Fig. 1: The computational domain. The interdisk domain shows the mean stream function field at $R_e = 5000$, $P_r = 0.72$, $R_{in} = 2$, $R_{out} = 8$ and $\frac{H}{R_{out}} = 0.125$. Courtesy of RÉMI JACQUES, UMR6600, Université de Technologie de Compiègne, BP 20529, 60200 Compiègne, France

1.2 Numerical model

The equations 1 and 2 have been discretized using a semi-implicit second order finite difference scheme where the convective terms are discretized using an ADAMS-BASHFORTH procedure while the CRANK-NICOLSON scheme has been applied to discretize the diffusive terms. In order to clarify this point, let's consider the following advection-diffusion equation:

$$\frac{\partial f}{\partial t} + \underbrace{V\ \nabla f}_{\text{convective term}} = \overbrace{\nabla^2 f}^{\text{diffusive term}} \tag{4}$$

where f is any scalar field and V is the transport velocity. Let's denote by Δt the timestep and by n the index of the timestep. ADAMS-BASHFORTH time-discretization is applied on the convective term while the diffusive term is discretized using the CRANK-NICOLSON scheme. This procedure leads to the following second order time-discretized equation:

$$\frac{f^{n+1}-f^n}{\Delta t}+\frac{3}{2}\left(V\ \nabla f\right)^n-\frac{1}{2}\left(V\ \nabla f\right)^{n-1}=\frac{1}{2}\left(\nabla^2 f^{n+1}+\nabla^2 f^n\right) \tag{5}$$

The previous equation leads to the well-known HELMHOLTZ problem which reads:

$$\left(\sigma I-\nabla^2\right) f^{n+1}=S_f^n \tag{6}$$

where S_f^n is the source term which includes all known quantities computed at the timestep n while seeking the solution at the timestep $n+1$.
This discretization procedure, applied to the equations 1 and 2, yields a set of coupled elliptic linear equations:

$$\left(\left(\sigma+\frac{1}{r^2}\right)I-\nabla^2\right)\omega^{n+1}=S^n_{\omega,v,\psi} \tag{7}$$

$$\left(\left(\sigma+\frac{1}{r^2}\right)I-\nabla^2\right)v^{n+1}=S^n_{v,\psi} \tag{8}$$

$$\left(\frac{\partial^2}{\partial z^2}+r\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}\right)\right)\psi^{n+1}=r\omega^{n+1} \tag{9}$$

where $\sigma=\frac{2R_e}{\Delta t}$. At each timestep $n+1$, two HELMHOLTZ problems (7, 8) have to be solved together with the stream-function equation (9).

2 Application development

The objective was to develop a message passing parallel application in a short timeframe (2 days) that efficently solves the problem defined by equations 1, 2 and 3. The `PMD` module (release 1.0.2) has been used since it offers all the building box to quickly parallel-solve such a problem. For each operator, `PMD` applies a domain decomposition based on the Schur complement ([QUA 90]) to build the Schur matrix and to solve the subdomain interface problem. Figure 2 summarizes the `PMD` routines calling sequence to reach the final solution.
As we notice, the system defined by equations 7, 8 and 9 is composed of two elliptic HELMHOLTZ problems and one elliptic Stream-Function equation. Moreover, if we compare equations 7 and 8, we notice that both equations exhibit the same HELMHOLTZ operator $\left(\sigma+\frac{1}{r^2}\right)I-\nabla^2$. Hence, two Schur matrices have to be build (figure 2: lines 24, 25). The former corresponds to the HELMHOLTZ operator and the latter to the Stream-Function one. During this step, each process solves a homogeneous time-independent problem to build its own block of the Schur matrix which is then LU-factored (figure 2: lines 29, 30). Conversely, in the time-loop, each process seeks time-dependent final local solutions of ω, v and ψ fields for which we parallel-solve three equations (figure 2: lines 37-39) at each timestep iteration.
Furthermore, standard second order centered finite differences have been used for both first and second order spatial derivatives. Subsequently, each process solves a local problem of size $(N_r \times N_z)$ using fast direct solvers. These methods are documented as FOURIER-TOEPLITZ or FOURIER-tridiagonal methods [LAS 87]. In our case, the procedure mainly consists in locally performing a diagonalization in the z-direction using FFT routines and in performing a finite differences method in the r-direction.

```
1  PROGRAM NS
2    USE PMD
3  !... Declare User and MPI type objects
4    ...
5  !... Extended PMD communicator
6    TYPE(PMD_Comm_2D)  :: Comm
7  !... PMD datatype to handel the Schur matrices associated
8  !... to the Helmholtz and to the Stream-Function operators
9    TYPE(PMD_R8_Schur) :: H, SF
10 !... PMD datatype to handel the LU factorization
11 !... of the Schur matrices
12   TYPE(PMD_R8_GELU)  :: LU_H, LU_SF
13
14 !... Begin PMD
15   CALL MPI_INIT(...)
16   CALL PMD_Init_1DD(..., Comm)
17
18 !... Set the local Helmholtz and Stream-Function
19 !... operator matrices and initial conditions
20   ...
21
22 !... Build the Schur matrices of the Helmholtz
23 !... and of the stream-function operators
24   CALL PMD_Schur_1DD(Comm, ..., H)
25   CALL PMD_Schur_1DD(Comm, ..., SF)
26
27 !... LU-Factor the Schur matrices of the Helmholtz
28 !... and of the stream-function operators
29   CALL PMD_Schur_Factor_1DD(Comm, H, LU_H)
30   CALL PMD_Schur_Factor_1DD(Comm, SF, LU_SF)
31
32 !... Start timestep loop
33   DO
34 !... Set time-dependent boundary conditions
35     ...
36 !... Solve the final problem for ω, v and ψ fields
37     CALL PMD_Solve_1DD(Comm, ..., LU_H, ..., omega)
38     CALL PMD_Solve_1DD(Comm, ..., LU_H, ..., v)
39     CALL PMD_Solve_1DD(Comm, ..., LU_SF, ..., psi)
40
41 !... print the results
42     IF(convergence) EXIT
43   END DO
44
45 !... End PMD
46   CALL PMD_End_1DD(Comm)
47   CALL MPI_FINALIZE(...)
48 END PROGRAM NS
```

Fig. 2: Fortran code to parallel-solve the Navier-Stokes equations using `PMD`.

#Processors	256
Processor	Dec Alpha EV5
Processor memory	128 MB
Memory data cache	8 KB + 96 KB two level caches
Processor peak performance	600 MFlops/sec
Interconnexion network	Bidirectional 3D Torus: 600 MB/sec
Operating system	Unicos/mk 2.0
MPI Library	MPT (Version 1.2)
Compiler	f90 (Version 3.1)
Optimization options	-O3,unroll2,pipeline3

Table 1: Hardware and software characteristics of the used T3E machine.

3 Performance

Timing measurements and performance rates of the application will be presented in terms of sustained 64 bit floating point operations per second (Mflops/sec) on the SGI/CRAY T3E-600 parallel machine whose charateristics and compiler options are summarized in Table 1. The instrumentation has been performed with `MPI_WTIME` function and with the SGI/CRAY `apprentice` tool. Nevertheless, the total elapsed time returned by `MPI_WTIME` has always been compared to the **user processors connect time** reported by the `acctcom` CRAY command. This is in order to avoid instrumented cases corresponding to situations in which processes are held or checkpointed by the system job scheduler during the execution. Actually, in such situation, the elapsed times returned by the `MPI_WTIME` function are over estimated with respect to the connect time.

3.1 Global performance analysis

Table 2 shows the elapsed time and the global performance rate on 64 processors of the T3E. The most cost effective part of the application is the resolution step during which we performed 5000 timestep iterations. Especially noteworthy is the time induced by the MPI communication calls which represents almost 21% of the total elapsed time on 64 processors. This percentage includes, however, MPI point-to-point and collective communication times. Since MPI collective communications are blocking routines, this measured time includes the overhead induced by all local MPI synchronization and internal buffer management operations.

Time to Build the Schur Matrices	180. sec.
Time to LU-Factor the Schur Matrices	90. sec.
Time to Solve the Problem: 5000 timestep iterations	6600. sec.
Total Elapsed Time	6939. sec.
Total MPI Communication Time	1470. sec.
Global Performance Rate on 64 processors	2.7 GFlops/sec.

Table 2: Elapsed time and performance rate on 64 processors. Mesh size in each subdomain: $N_r = 129$, $N_z = 81$. Timestep: $\Delta t = 0.00008$. Reynolds Number: $R_e = 1000$. Radius of inner and outer cylinders: $R_{int} = 1$, $R_{out} = 11$.

More details are obtained using the CRAY `apprentice` performance tool. Figure 3 represents a snapshot section of the `apprentice` window. It shows the global cost of the `MPI` communication overhead (dark coloured bar) with respect to the cost of the parallel

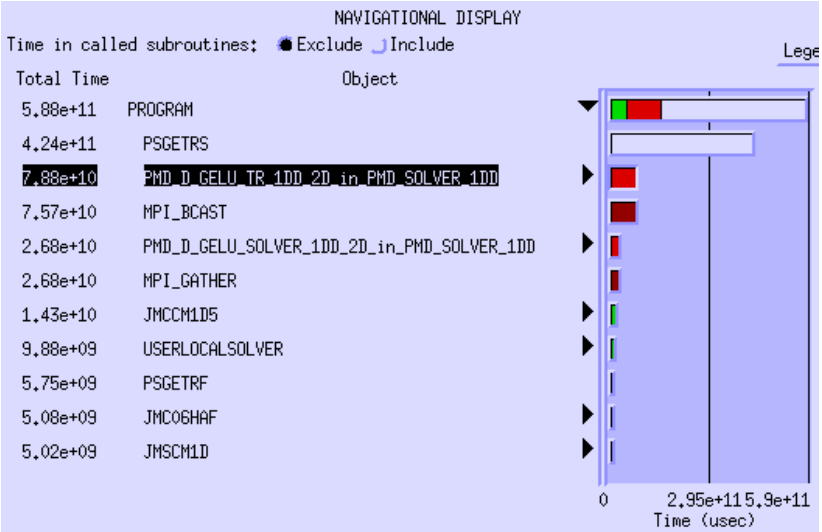

Fig. 3: Apprentice window section. Details of the global parallel work. Dark coloured bars mostly represent MPI communication overheads. Processors number: 64. Mesh size in each subdomain: $N_r = 129$, $N_z = 81$. Timestep: $\Delta t = 0.00008$. Reynolds Number: $R_e = 1000$. Radius of inner and outer cylinders: $R_{int} = 1$, $R_{out} = 11$.

work which, as we can see, is induced mainly by the `ScaLAPACK` routine `PSGETRS` called inside the `PMD` module.

3.2 Scalability

In this case study, the subdomain mesh size is maintained constant for any processors number. The ideal scalable parallel application would be the one with no inter-process communications. Such kind of application should return a constant total elapsed execution time per meshpoint and per timestep iteration for any processors number. However, in a realistic context of a parallel application, the communication rate may has some influence on the scalability of the code beyond a critical number of processors.
In fact, the application scales quiet well till 64 processors (figure 4). Beyond this limit, the number of processors being larger, the communication rate has greater incidence over the computational part. Especially noteworthy is that till 16 processors the elapsed time rather decreases. This has to be related to a better reuse of the memory data cache at the ScaLAPACK routine level [CHER 97] since the Schur matrices have been block-distributed among the processors. Hence beyond 16 processors, the block size of the Schur matrices seems to be better appropriate.
Also, it has to be noted (figure 4) that the local memory allocation size grows linearly as the processors number increases. Hence on 100 processors the amount of allocated memory is only 2.7 times larger than on 25 processors, though the global mesh size is 4 times larger on 100 processors.

4 Conclusion

We used `PMD` release 1.0.2 to develop a message passing parallel code within a short time-frame to solve the Navier-Stokes equations. The application simulates an axisymmetric incompressible viscous fluid flow inside a centrifugal pump. It has run on the SGI/CRAY T3E-600 parallel machine at IDRIS and has delivered a total performance rate up to 6 GFlops/second on 144 processors. Performance and timing analysis have shown good scalability till 64 processors beyond which the MPI communication overhead has taken larger influence with respect to the processor computational load.

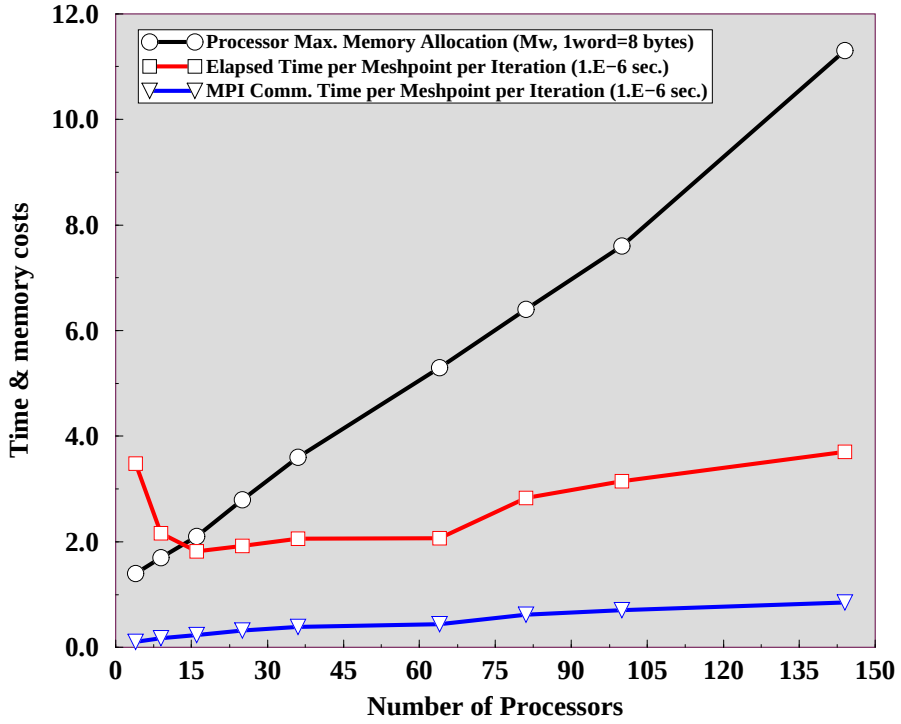

Fig. 4: Scalability of the application on the T3E-600. Mesh size in each subdomain: $N_r = 129$, $N_z = 81$. Timestep: $\Delta t \in [0.0001, 0.00008]$. Reynolds Number: $R_e = 1000$. Radius of inner and outer cylinders: $R_{int} = 1$, $R_{out} = 11$.

References

[CHE 98] J. Chergui & al., `PMD`: A Parallel Fortran 90 Module to Solve Elliptic Linear Second Order Equations. *Calculateurs Parallèles*, Hermes publisher, 75004 Paris, France. Vol. 10, N^o 6, Dec. 1998.

[CHER 97] J. Chergui, Performance ratio analysis between CRAY T3E and T3D machines (in French). *Calculateurs Parallèles*, Hermes publisher, 75004 Paris, France. Vol. 9. No 4. December 1997.

[GRO 96] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Safir, M. Snir, *MPI: The Complete Reference*. Volume 1, The MPI Core. The MIT Press Cambridge, Massachusetts London, England. 1998.

[JAC 98] R. Jacques, P. Le Quéré, O. Daube, Parallel Numerical Simulation of Turbulent Flows in Closed Rotor-Stator Cavities. Notes on Numerical Fluid Mechanics, Vieweg Publisher. 1998

[LAS 87] P. Lascaux, R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*. Tome2, éd. Masson, Paris, France. 1987.

[QUA 90] A. Quarteroni, Domain Decomposition Method for the Numerical Solution of Partial Differential Equations. UMSI 90/246. University of Minnesota Supercomputer Institute. Research report. December 1990.