Using a Parallel CFD Code for Evaluation of Clusters and MPPs

Oleg Bessonov
Institute for Problems in Mechanics, Moscow, Russia bess@ipmnet.ru

Dominique Fougère, Bernard Roux
Laboratoire de Modélisation et Simulation Numérique en Mécanique (L3M), Marseille, France {fougere,broux}@l3m.univ-mrs.fr

In this talk we present methods of investigation of several components of the cluster performance, along with the results of this investigation.

- Numerical method for incompressible flow in a cylinder.
- Parallelization for distributed memory computers (up to 16 processors).
- Extension of the parallelization method and techniques.
- Analysis of components of the cluster performance.
- Evaluation and comparative analysis of parallel performance.
- Optimal choice of inexpensive PC cluster.
- Conclusion.

Numerical method for incompressible flow in a cylinder

3D non-stationary Navier-Stokes equation in Boussinesq approximation for incompressible viscous flows in a cylindrical domain (φ, z, r) :

$$\vec{\nabla} \cdot \vec{V} = 0$$

$$\frac{\partial \vec{V}}{\partial t} + \vec{\nabla} \cdot \left(\vec{V}\vec{V}\right) = -\vec{\nabla} p + \nabla^2 \vec{V} - \mathrm{Gr}\, \vec{n}\, \theta$$

$$\frac{\partial \theta}{\partial t} + \vec{\nabla} \cdot \left(\vec{V}\theta\right) = \frac{1}{\mathrm{Pr}} \nabla^2 \theta$$

$$\varphi = 0 \,..\, 2\pi, \;\; z = 0 \,..\, L, \;\; r = 0 \,..\, R$$

Solution method:

- **velocity-pressure formulation $(\vec{V}, p)$;**
- **2-nd order FVM discretization on uniform staggered grids;**
- **decoupled solution of $\vec{V}$, p and θ equations (Fractional step method);**
- **time integration with the implicit treatment of the most critical terms;**
- **Fourier method for pressure Poisson equation:**

  FFT(φ), FFT(z), 3-diag solver(r), FFT(z), FFT(φ)

Parallelization for distributed memory computers

Modern distributed memory parallel computers are characterized by:

- **very high computational potential – e.g. 300 MFLOPS sustained;**
- **relatively slow communication speed of interconnection networks – e.g. 20 MWords/s.**

$\Longrightarrow$ **Numerical methods and parallelization algorithms with less data exchanges are needed – e.g. with $O(N^2)$ communications vs. $O(N^3)$ computations.**

Example: FDM/FVM methods for 3D regular domains:

- **substantial fraction of "explicit" time integration codes – no data exchanges;**
- **implicit (ADI) parts with 3-diagonal linear systems – few data exchanges;**
- **pressure Poisson solver with FFT and 3-diagonal systems – full data exchange (for FFT).**

$\Longrightarrow$ **Parallelizable Poisson solver is needed.**

Plan of the section:

- **Parallelization of the algorithm (SPMD + message passing);**
- **New method for solving Poisson equation;**
- **Some technological aspects of parallelization.**

Parallelization of the algorithm

Splitting in 1 or 2 directions (variants: 1×1, 2×1, 4×1, 4×2, 4×4).

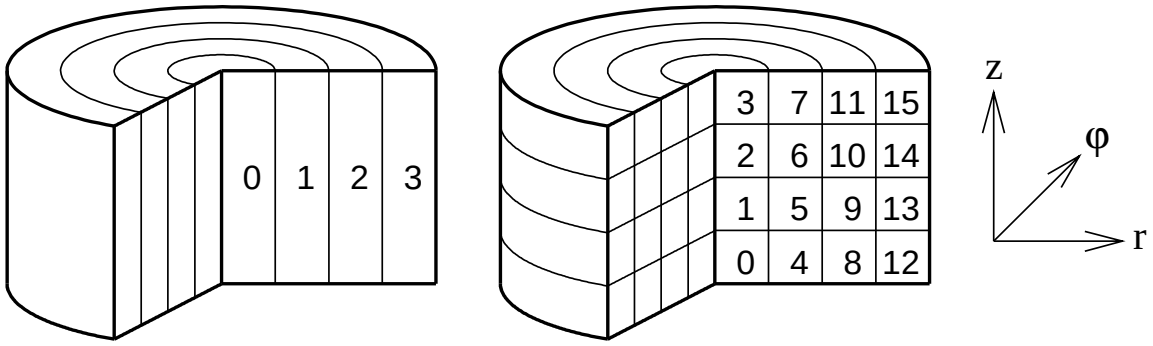


Figure 1: 1D and 2D decompositions of a computational domain

1-dimensional splitting:

- **all computations within a plane (φ, z) – independent, exchanges are needed only between timesteps;**
- **3-diagonal sweeps in the direction r - cannot be parallelized directly, need special algorithm (2-way parallel partition method):**

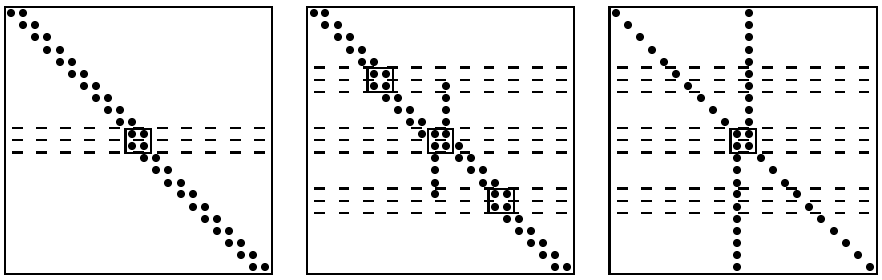
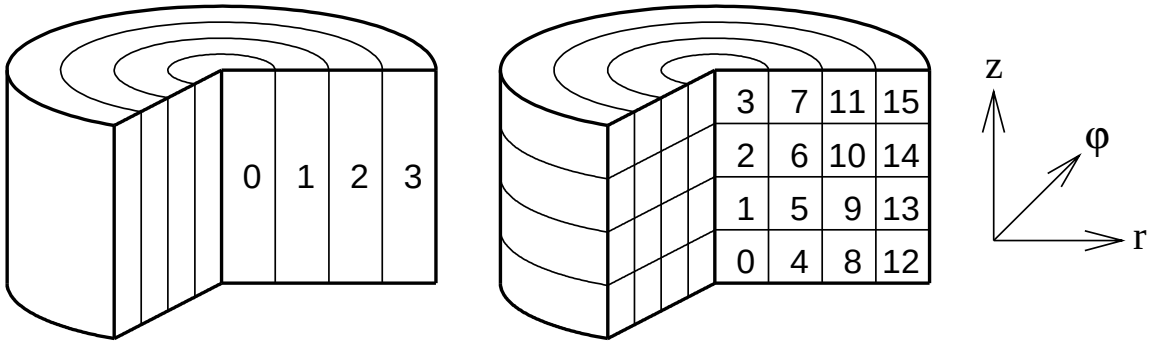


Figure 2: Illustration of the 2-way method of parallelization for 2 and 4 proc.

$\Longrightarrow$ **As a result, the parallelized numerical method is algebraically identical to the sequential one and demonstrates good parallelization efficiency. However, the increased complexity of solving 3-diagonal systems limits the number of processors by 4, at most 8.**

2-dimensional splitting:



1D and 2D decompositions of a computational domain

- **3-diagonal sweeps in the directions r and z – as above;**
- **FFT in the direction z – cannot be efficiently parallelized, needs full data exchange (blocked transposition):**

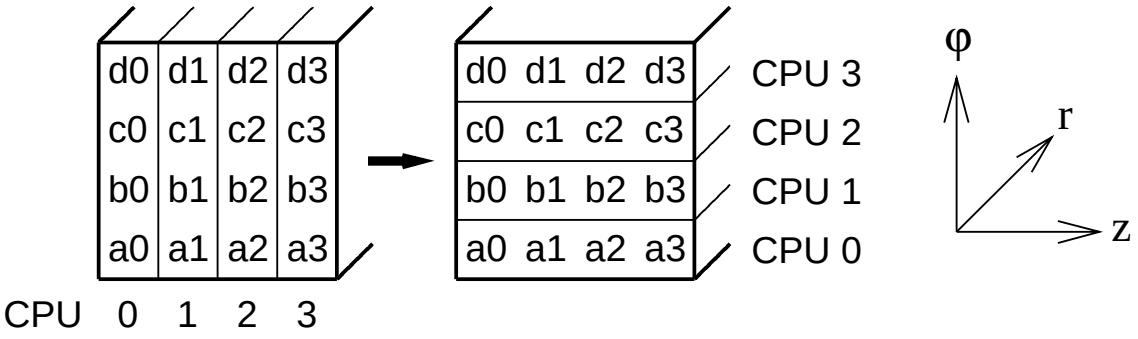


Figure 3: Blocked transposition for parallelization of FFT in the direction z

Paralllelized procedure for the Fourier method:

FFT(φ), transposition, FFT(z), 3-diag(r), FFT(z), transposition, FFT(φ)

$\Longrightarrow$ **Exchange of full 3D data arrays is required**
(for other steps of the algorithm – only 2D boundary planes).

New method for solving Poisson equation

FACR (Fourier analysis with cyclic reduction) method for 3D linear system:

- **FFT(φ);**
- **cyclic reduction of resulting 2D linear systems;**
- **FFT(z);**
- **solving 3-diag(r);** $\Longleftarrow$ **for reduced linear system**
- **FFT(z);**
- **backsubstitution for cyclic reduction;**
- **FFT(φ);**

Method of cyclic reduction:

$$\begin{aligned} x_{i-2} + A\,x_{i-1} + \quad x_i \qquad\qquad\quad &= y_{i-1} && \times 1 \\ x_{i-1} + A\,x_i + \quad x_{i+1} \qquad &= y_i && \times (-A) \\ x_i + A\,x_{i+1} + x_{i+2} \quad &= y_{i+1} && \times 1 \end{aligned}$$

Resulting linear system:

$$x_{i-2} + (2 - A^2)\,x_i + x_{i+2} = y_{i-1} - A\,y_i + y_{i+1}$$

- **Substitutions: $A^{(1)} = 2 - A^2$ and $y_i^{(1)} = y_{i-1} - A\,y_i + y_{i+1}$ to obtain a reduced system of the same type and to employ the cyclic reduction procedure again.**
- **A is a 3-diagonal matrix in our case.**
- **$A^{(1)}$, $A^{(2)}$ ... are no more 3-diagonal, but can be decomposed into 3-diagonal factors.**
- **For 2-step cyclic reduction scheme – 4-fold reduction of matrix size (and amount of necessary data exchanges).**

$\Longrightarrow$ **The resulting amount of transmissions is now on the reasonable level and doesn't influence so much the efficiency of parallelization.**

Some technological aspects of parallelization

- **Alternating numbering scheme for data elements (grid points) in order to simplify the code flow:**

Figure 4: Standard (left) and alternating (right) numbering schemes

- **Preliminary computation of LU matrix decompositions (for constant coefficient matrices);**
- **Communication library–independent approach – all library-specific calls are encapsulated within intermediate data exchange routines. As a result, a parallel application program becomes system-independent.**
  - **more efficient options exists (SHMEM, GM, MPL . . .),**
  - **lack of MPI implementation or incompatible implementation,**
  - **specific hardware or firmware requirements (block size limitations, regulating duplex mode of transmission, other optimizations),**
  - **renumbering (remap) of allocated processor nodes is needed (useful for SMP-node machines, 2D-grids etc).**

$\Longrightarrow$ **The intermediate communication routines have been adapted to the following protocols:**

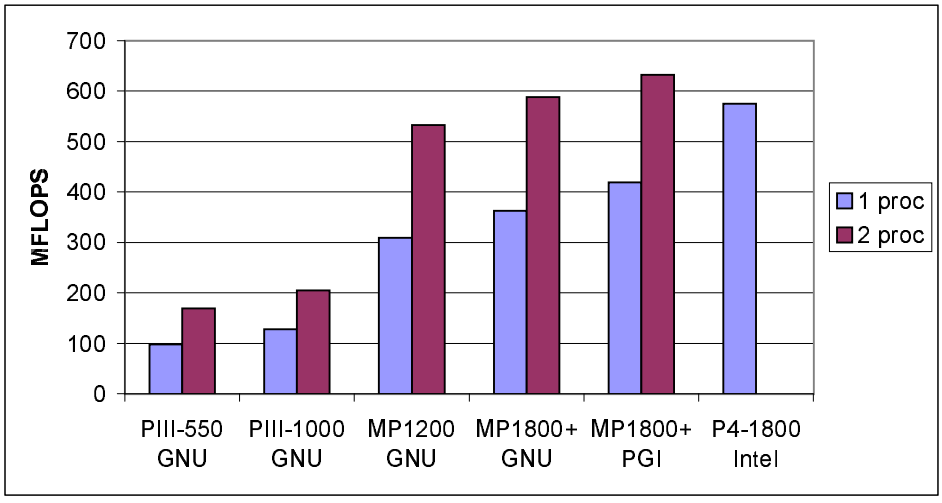
- **NX (Intel i860),**
- **Parix (Parsytec),**
- **PVM,**
- **MPL (IBM SP2),**
- **SHMEM (Cray T3E, SGI),**
- **MPI in different incompatible implementations.**

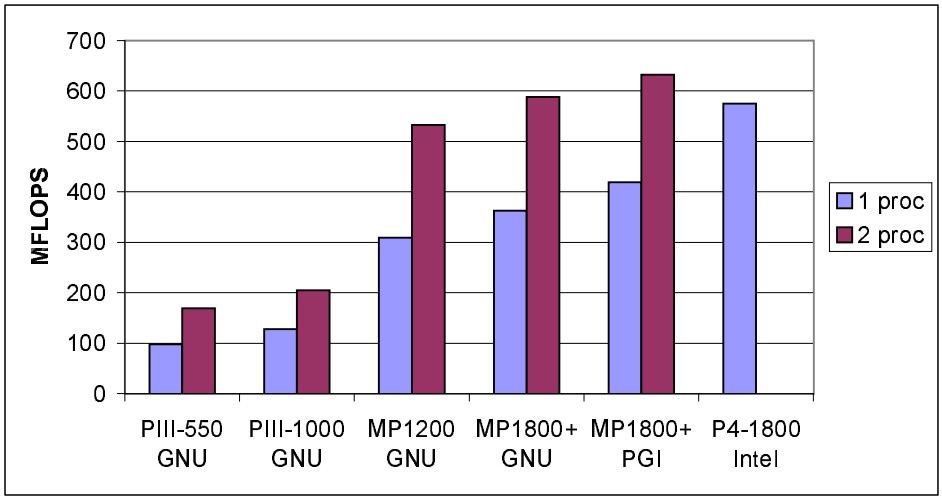
Analysis of components of the cluster performance

Single-processor performance

- **3D CFD code of the problem size 70 MB (grid size $128 \times 64 \times 92$) is employed as a benchmark, that correlates well with the SPECfp2000.**
- **MFLOPS rate (for 64-bit arithmetic) achieved by this code is used as a performance indicator.**
- **All measurements on SMP nodes are performed in single-program and multiple-program runs (in order to account shared memory conflicts).**
- **Different compilers are used, with the best compiler options.**

processor	compiler	MHz	1-prog	2-prog	ratio
Pentium-4-1800	Intel	1800	575.0	–	–
Athlon-MP1800+	PGI	1525	419.5	316.2	75.4 %
Athlon-MP1800+	GNU	1525	362.7	294.1	81.1 %
Athlon-MP1200	GNU	1200	309.2	266.6	86.2 %
Pentium-III-1000	GNU	1000	128.1	102.4	80.0 %
Pentium-III-550	GNU	550	97.5	84.5	86.7 %





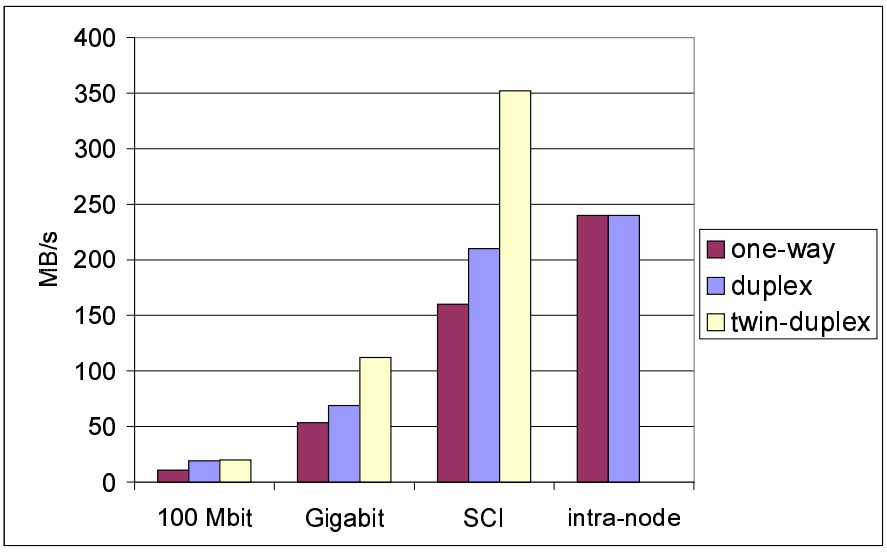
⟹ Results and conclusions:

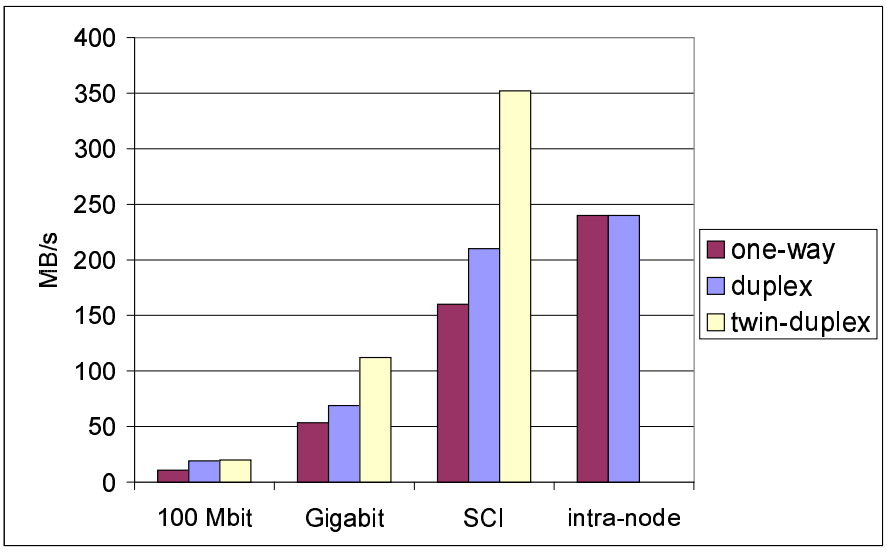
- **Performance of dual-program runs is degraded on faster configurations (due to the memory throughput saturation).**
- **Dual Pentium-III server platforms have demonstrated disappointing performance level (being considered as candidates in Sept. 2001).**
- **P4 is generally faster than Athlon (partly due to SSE2), but its performance depends sharply on many factors.**
- **P4 1U-platforms (single or dual) were not available at the moment (Sept. 2001), now they are more expensive than Athlon 1U-platforms.**

Performance of inter- and intra-node communications

- Transferring large arrays of data (32 – 64 KB).
- Two sorts of test programs: ”MPI Performance Test Suite” and custom.
- The latter employs more realistic method when arrays to be transferred are shifted in memory at every iteration in order to avoid misleading cache effects (their results are therefore sometimes lower).
- One-way, duplex and twin-duplex modes of transfer.

program	mode of transfer	intra-node (shared memory)	inter-node Dolphin SCI	inter-node Gigabit Ethernet	inter-node 100 Mbit Ethernet
”transf1”	one-way	500–600	200	56	10.8
”transf2”	duplex	140	120	36	9–10
custom	one-way	240	160	53.5	10.8
custom	duplex	120	105	34.5	9.5
custom	twin-duplex	–	88	28	–





⟹ Results and conclusions:

- **Gigabit Ethernet: twin duplex max.** $4 \times 28.0 = 112$ **MB/s, i.e. 45 % of the** $2 \times 1000/8 = 250$ **MB/s peak value. Possible reasons: poor implementation of software levels, the nature of Gigabit Ethernet protocol.**
- **Dolphin/Scali/SCI: twin duplex max.** $4 \times 88 = 350$ **MB/s, about 65 % of the peak rate of the PCI64/66 port.**
- **Intra-node: not efficient as they could be. Possible reasons: transmission of data through the shared memory (that is relatively slow), poor software implementation.**

Comparison of different parallel computers

MFLOPS — measured in multiple-program runs (problem size 70 MB)

MB/s — intra-node / inter-node multiple-duplex – single-duplex transfers

Table 1: Characteristics of the analyzed parallel computers

parallel platform and interconnect	CPUs per node	CPU cache size	theor. peak MFLOPS	real code MFLOPS	comm. library	comm. duplex MB/s	ratio MB/s to MFLOPS
IBM SP4-1300 Colony switch	32	1.4M	5200	491	MPL	235–550 –	0.48–1.12 –
IBM SP3-375 Colony switch	16	8M	1500	297	MPL	80–175 –	0.27–0.59 –
AMD Athlon-MP Dolphin/SCI	2	256K	3050	308	MPI	120 88–105	0.39 0.29–0.34
AMD Athlon-MP Ethernet1000	2	256K	3050	308	MPI	120 28–35	0.39 0.09–0.11
Alpha 21264-667 Myrinet	2	4M	1333	347	MPI	90 41–73	0.26 0.12–0.20
Intel PIII-550 2×Ethernet100	2	512K	550	84.5	MPI	39 5.8–10	0.46 0.07–0.12

Two benchmarks for evaluating the efficiency of parallelization:

- **fixed size problem – 70 MB total ($128 \times 64 \times 92$),**
- **scalable problem – 70 MB per processor (up to $256 \times 256 \times 184$).**

Table 2: Parallelization efficiency (%) for the fixed and scalable problems

parallel platform	fixed size problem				scalable problem			
	2	4	8	16	2	4	8	16
IBM SP4-1300	102.4	102.3	106.1	98.0	99.6	92.2	88.8	83.7
IBM SP3-375	98.9	98.0	105.5	102.3	96.2	86.4	79.6	71.5
AMD cluster SCI	98.0	89.6	104.0	–	96.3	90.3	85.9	–
AMD cluster Gig	96.9	83.5	88.2	73.2	95.2	88.2	80.4	70.8
Alpha cluster	91.8	82.4	83.5	75.2	90.5	82.3	82.2	71.8
Intel cluster	89.0	82.0	78.6	66.4	90.8	86.2	78.8	74.5

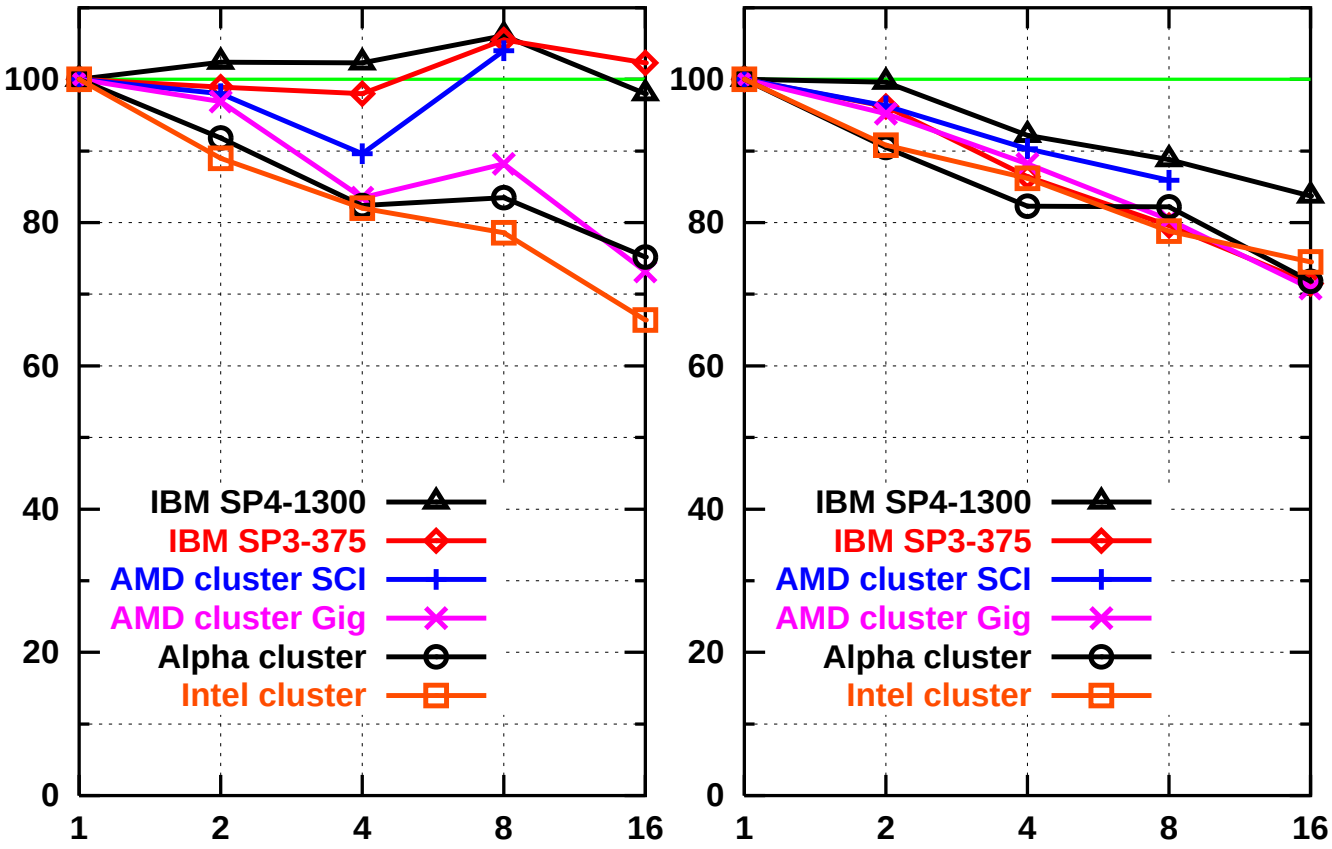


Figure 5: Efficiency (%) for the fixed (left) and scalable (right) problems

- **Fixed-size problem (70 MB total):**
  - **The correlation of the parallelization efficiency with the communication-to-computation speed ratio is clearly visible.**
  - **IPM SP4, IBM SP3, AMD-SCI: better efficiency (due to much higher communication-to-computation speed ratio), superlinear speed-up (due to cache effects, multilevel for IBM SP4).**
  - **AMD-Gigabit, Alpha, Intel: very close behaviour (since their ratios are not much different), some positive cache effects for 8+ CPUs.**
- **Scalable problem (70 MB per CPU):**
  - **The correlation of the parallelization efficiency with the communication-to-computation speed ratio is less clear, partly because the bigger problem is less sensitive to this ratio.**
  - **IPM SP4 demonstrates the best efficiency (due to much higher communication-to-computation speed ratio).**
  - **Athlon, Alpha suffer from differences in processor node's speeds (2–3 % or more).**
  - **IBM SP3 is supposed to suffer from multi-user environment (process migration ?).**
  - **The effect is partly masked by the algorithmic overhead of parallelization.**

$\Longrightarrow$ **In general – there is a reasonable correlation between communication-to-computation speed ratio and parallelization efficiency.**

Gigabit Ethernet PC clusters – demonstrate comparable level of parallelization efficiency due to better intra-node exchanges. With Dolphin/SCI interface, PC clusters would become a good and inexpensive alternative to RISC machines.

Description of the AMD/Linagora bi-Athlon cluster at L3M

Hardware: **25 bi-processor nodes with Athlon MP1800+ CPUs (1 master node (2 CPU) and 24 computational nodes (48 CPUs)); all 1U dual-CPU nodes and communication hardware are mounted in the 19” rack**

- **19 motherboards TYAN K7 (AMD 760MP chipset) connected to the Gigabit Ethernet switch**
- **6 motherboards TYAN K7X (AMD 760MPX chipset) interconnected into the Dolphin/SCI mesh**
- **Processors: 1.53 GHz, L2 cache 256 KB**
- **Memory: 1 GB per node, DDR266**
- **Gigabit Ethernet switch HP1000 + 100Mbit switch**
- **26 hard disks IDE 20 GB**

Software: **Cluster software Alinka and SCALI**

- **Cluster administration software for Linux RedHat 7.1: Alinka Raisin (Gigabit subcluster), Scali software (SCI subcluster)**
- **NFS, SSH, PVFS (on test)**
- **Compilers Portland Group: pgf77, pgHPF, pgcc, pgprof, pgdbg**
- **TotalView debugger**
- **LAM, MPIch, PVM (100Mbit, Gigabit), LAM (SCI)**
- **Fluent, M-Implicit, FieldView**

Conclusion

We illustrate that a parallel CFD code can be successfully used as an adequate and sensitive measurement tool for evaluating parallel computers.

The presented method allows to parallelize 3D CFD codes for simulation of incompressible flows in regular domains:

- **ensures a reasonable level of parallelization efficiency (despite its partially implicit nature and relatively low communication speed of modern computers' interconnects),**
- **follows SPMD model and can be easily adapted to different architectures,**
- **can be used for analyzing parallel computer performance in order to reveal their important characteristics.**

Acknowledgements: **The work was supported by the program "Réseau de coopération universitaire et scientifique Franco-Germano-Russe" of the French Ministry of National Education, and by the Russian Foundation for Basic Research (grants RFBR-01-01-00745 and RFBR-02-01-00210).**

The evaluation of performance of AMD-SCI cluster was performed with the help of Linagora SA (France) and its partners AMD, Dolphin and Scali.

The access to other parallel computers was given by CINES, France, and JSCC (Joint SuperComputer Center), Russia.

This presentation at IPDPS 2003 has become possible due to the support of ACI GRID (Action Concertée Iniciative – Globalisation des Ressources Informatiques et des Données) under the French Ministry of Research.


bess@ipmnet.ru