

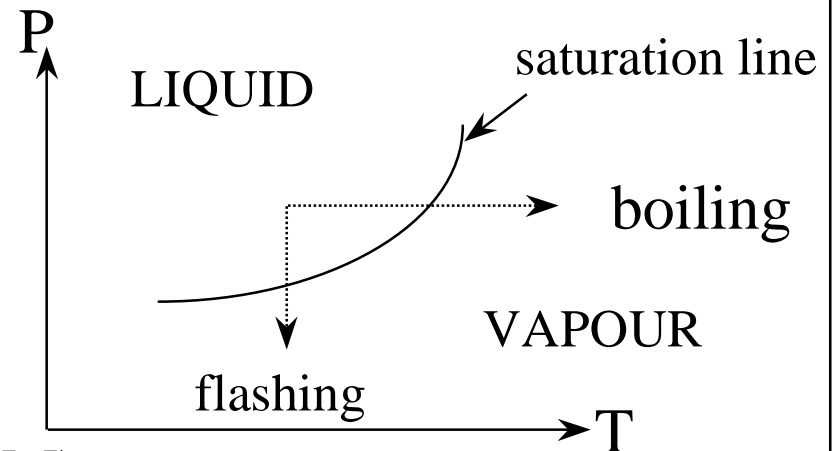
# Comparing Various Parallel Programming Models Applied to a Diphasic Compressible Fluid Mechanics Code

Jean-Yves Berthou, Eric Faucher, Eric Fayolle, Laurent Sciffet

Electricité de France  
Research and Development division

# Introduction

- Nuclear plants use numerous steam generator safety valves
- **Flashing phenomenon**: vaporisation of a liquid due to depressurization
- Z. Bilicki and co-authors proposed the homogeneous relaxation model (HRM)



## Applications

- Nuclear industry : Loss of Coolant Accident (LOCA)
  - Depressurization of a vessel full of subcooled liquid
- Propagation of pressure and void fraction waves

# Introduction

## HOMOGENEOUS RELAXATION MODEL

### Equations

Three conservation laws for the mixture

+ Mass balance for the vapour

$$\frac{\partial}{\partial t}(\rho A) + \frac{\partial}{\partial z}(\rho A u) = 0$$

$$\frac{\partial}{\partial t}(\rho A u) + \frac{\partial}{\partial z}(\rho A u^2) + A \frac{\partial}{\partial z} P = -A f$$

$$\frac{\partial}{\partial t}(A E) + \frac{\partial}{\partial z}[A u(E + P)] = 0$$

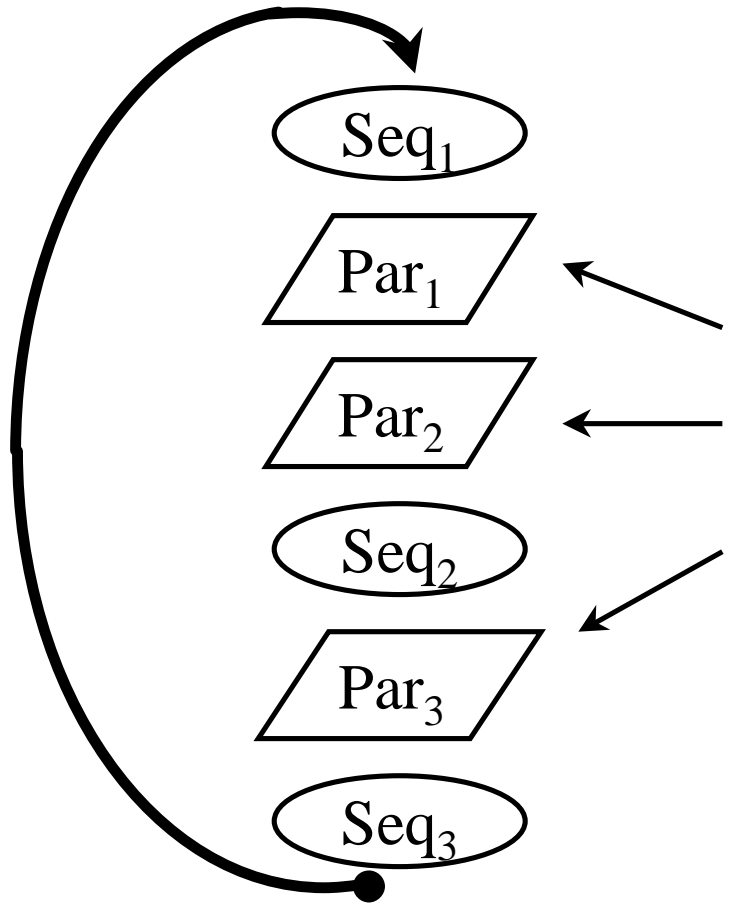
$$\frac{\partial}{\partial t}(\rho A x) + \frac{\partial}{\partial z}(\rho A u x) = A \Gamma$$

+ EOS :  $E = \rho e(\rho, P, x) + \frac{1}{2} \rho u^2$

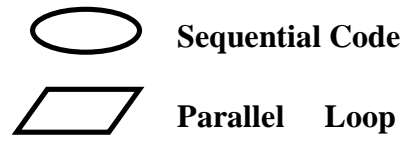
# Parallelization Principles of HRM1D

- 8000 lines f77 code

The time step loop performs an iterative computation



Block-cyclically  
Distributed  
Iteration space



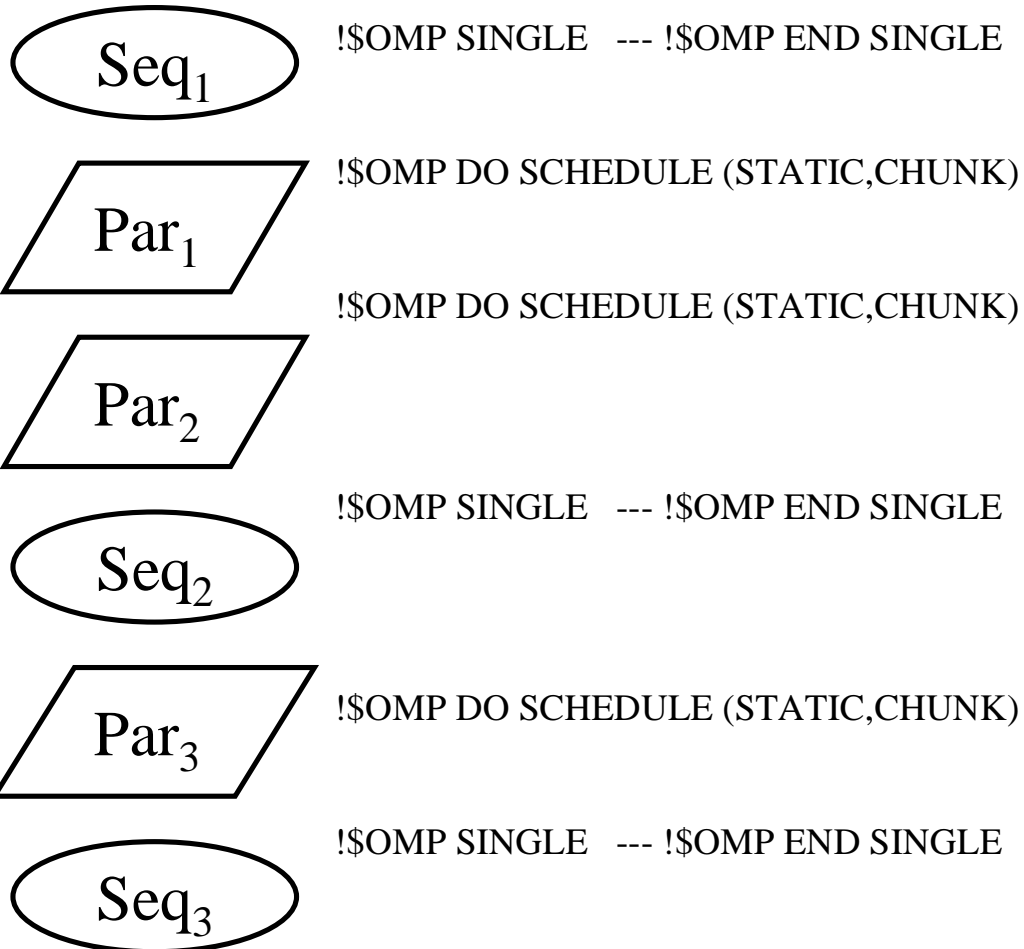
# Target platforms

Platforms (system)	Processor /MHz	NCPUS	Peak Perf. Mflops/PE	Memory (MB)	Peak Bandwidth (MB/s)	Owner
<b>SGI/Cray T3E-750</b> (Unicos/mk 2.0.4.55)	DEC 21164 /375 MHz	300	750	128 /PE	600	CEA Civil
<b>Compaq SC 232</b> (OSF1 V5.0)	DEC 21264 /667 MHz	232	1200	4000/PE	200	CEA Civil
<b>HP9000/800</b> (HP-UX B.11.00)	PA-8200 /240 MHz	16	960	2000	950	EDF-R&D
<b>Compaq Proliant 6000</b> (Linux2.2.9)	PentiumPro /200 MHz	4	200	1000	264	EDF-R&D
<b>SGI Origin2000</b> (Irix IRIX64 6.5 )	R10000 /195 MHz	32	390	512/2PE	1200	Strasbourg Univ.
<b>Cluster of PC</b> (Linux 2.2.1)	BiPentium III /800 MHz	16	800	128/PE	152	EDF-R&D

The PC cluster is equipped with MYRINET network devices

# OpenMP Implementation

**EXTERNAL** OMP\_GET\_THREAD\_NUM, OMP\_GET\_NUM\_THREADS,  
OMP\_GET\_NUM\_PROCS, OMP\_SET\_DYNAMIC, OMP\_SET\_NESTED



# Parallelization of the 102 loop

```
!$OMP DO SCHEDULE (STATIC,50)
```

```
DO 102 I=1,NPTOT
```

```
IF ((CON(I).LE.0.D0).OR.(CON(I).GE.1.D0)) THEN
```

```
    GAMMAV(I)=GAM_PV(P(I),TAU(I),TLIQ(I))
```

```
ELSE
```

```
    GAMMAV(I)=GAM_PVX(P(I),TAU(I),CON(I),
```

```
    *          TEMPSAT(I),TLIQ(I),VGSAT(I))
```

```
ENDIF
```

```
102 CONTINUE
```

```
!$OMP END DO
```

# OpenMP Implementation

## Characteristics :

- Interprocedural dependence analysis
- Development of a prototype : tests of various policies of work distribution (STATIC CYCLIC distribution retained)
- Two weeks of development
- 30 OpenMP directives + few routine calls inserted
- Various compiler options tested
- Weak expertise needed

# OpenMP Platforms

Compilers and compilation options used :

Platform	Compiler	Options
<b>HP9000/800 Class V</b>	guidef90 3.6, KuckAssociate	+O3 +Odataprefetch +Oinline
<b>Compaq SC 232</b>	Compaq Fortran V5.3-915	-O5 -arch ev6 -tune ev6 -omp
<b>Compaq Proliant 6000</b>	pgf90 3.04, Portland Group Inc.	-Minfo=all -mp -O2
<b>SGI Origin2000</b>	f90 MIPSpro: 7.2.1	-mp -Ofast

# High Performance Fortran Implementation

Seq<sub>1</sub>

!HPF\$ TEMPLATE TPL(0:NPTOTP+1)

!HPF\$ ALIGN WITH TPL(1:NPTOTP) :: GAMMAV

!HPF\$ DISTRIBUTE TPL(BLOCK)

Par<sub>1</sub>

!HPF\$ INDEPENDENT, NEW(EINT,DT2,I)

Par<sub>2</sub>

!HPF\$ INDEPENDENT,  
NEW(I,USTA,CONCEN,TAUSTA,ESTA,PSTA)

Seq<sub>2</sub>

Par<sub>3</sub>

!HPF\$ INDEPENDENT, NEW(I)

Seq<sub>3</sub>

# Parallelization of the 102 loop

```
!HPF$ DISTRIBUTE (BLOCK) :: CON,GAMMAV,P,TLIQ
```

.....

```
!HPF$ INDEPENDENT, NEW(I)
```

```
DO 102 I=1,NPTOT
```

```
IF ((CON(I).LE.0.D0).OR.(CON(I).GE.1.D0)) THEN
```

```
    GAMMAV(I)=GAM_PV(P(I),TAU(I),TLIQ(I))
```

```
ELSE
```

```
    GAMMAV(I)=GAM_PVX(P(I),TAU(I),CON(I),
```

```
    *      TEMPSAT(I),TLIQ(I),VGSAT(I))
```

```
ENDIF
```

```
102 CONTINUE
```

# High Performance Fortran Implementation

## Characteristics :

- Performance => Block DISTRIBUTION instead of CYCLIC DISTRIBUTION  
32 T3E processors, CYCLIC/BLOCK : 30.1s/19.3 s
- f90 interface writing (150 lines)
- 20 HPF directives
- Developement of a prototype
- Two weeks of development
- High expertise needed

# HPF Platforms

Compilers and compilation options used :

Platform	Compiler	Options
<b>Cray T3E-750</b>	Pghpf 2.4, Portland Group Inc	-O2
<b>HP9000/800 ClassV</b>	Pghpf 2.4, Portland Group Inc	-O2
<b>Compaq Proliant 6000</b>	Pghpf 3.0, Portland Group Inc.	-O2 -smp
<b>SGI Origin2000</b>	Pghpf 2.4, Portland Group Inc	-O2

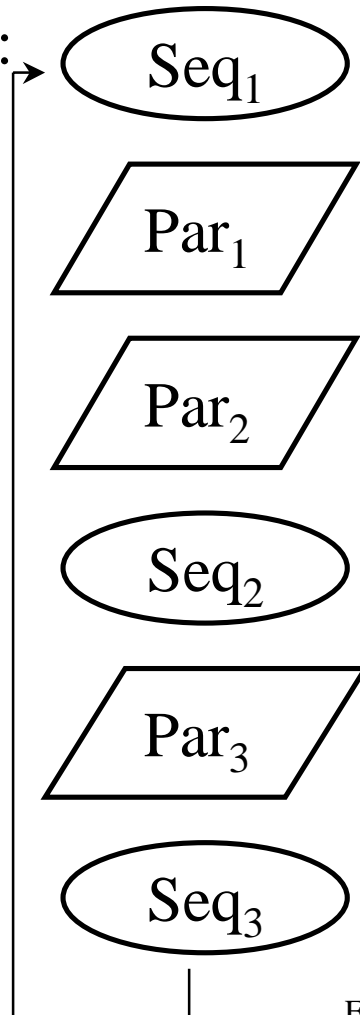
- T3E & O2K performances are better without the -smp option under pghpf 2.4
- We have been unable to use the -smp option with the HP class V

# Message Passing Interface Implementation version3

## Characteristics :

- 3 versions
- 2 months of development

$t=t+dt$



### Seq<sub>1</sub> (dt computation )

Process<sub>0</sub> broadcasts computed time step dt. dt depends on arrays computed during Par3 step.

### Par<sub>1</sub>

each process sends all the left values of all the blocks it owns within five distributed arrays to its predecessor

### Par<sub>2</sub>

No communication

### Seq<sub>2</sub> (Bound Computation)

each process needs the left elements of all the blocks its successor owns

### Par<sub>3</sub>

Processes asynchronously send their blocks to the master which computes the mass residu (non-blocking communications)

### Seq<sub>3</sub>

EWOMP 2000, September 2000

# MPI Platforms

## Characteristics :

- Three versions developed
- Two months of development

## Compilers and compilation options used :

Platform	MPI	Compiler	Options
<b>Cray T3E-750</b>	Cray	cf90 3.0.2.3	+O3
<b>Compaq SC 232</b>	Compaq (based on mpich 1.1.1)	Compaq Fortran V5.3-915	-O5 -arch ev6 - tune ev6
<b>HP9000/800 ClassV</b>	Hp	HP f90 v2.2	+O3
<b>Compaq Proliant 6000</b>	Mpich 1.1.2p	Pgf90 3.04, Portland Group Inc.	-O2
<b>SGI Origin2000</b>	Sgi	f90 MIPSpro: 7.2.1	-Ofast
<b>Cluster of 8 PIII 800</b>	Mpich-BIP 0.99b	pgf90 3.1-2	-O2

# Parallelization of the 102 loop

```
DO J=1,MYNBBLOCK
```

```
DO 102 I= LOOPBEG(J),LOOPEND(J)
```

```
IF ((CON(I).LE.0.D0).OR.(CON(I).GE.1.D0)) THEN
```

```
GAMMAV(I)=GAM_PV(P(I),TAU(I),TLIQ(I))
```

```
ELSE
```

```
GAMMAV(I)=GAM_PVX(P(I),TAU(I),CON(I),
```

```
*      TEMPSAT(I),TLIQ(I),VGSAT(I))
```

```
ENDIF
```

```
102 CONTINUE
```

```
ENDDO
```

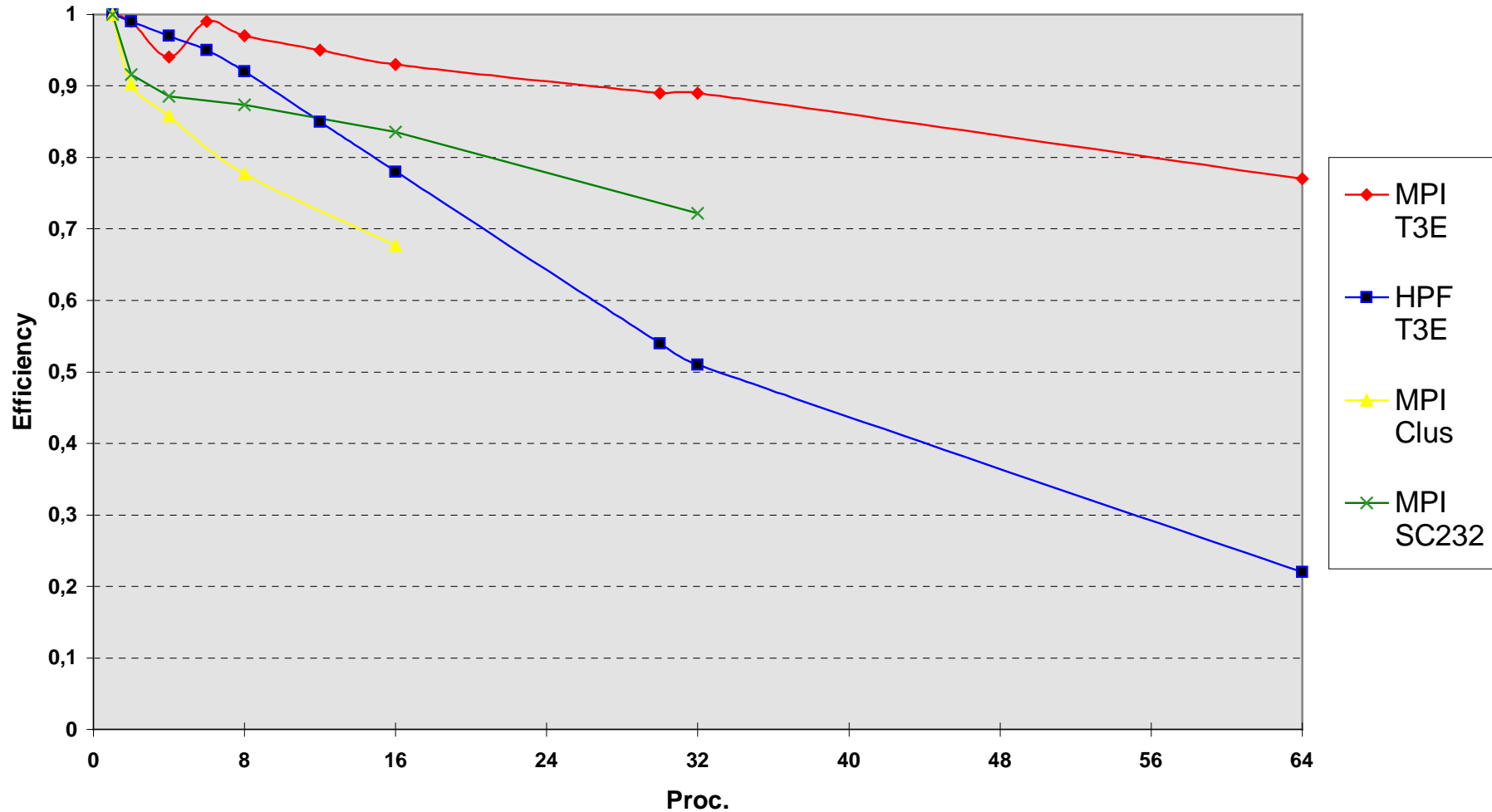
# Experimental Results

Versions/NCPUS	1	2	4	6	8	12	16	32	64
<b>Measurement performed on the T3E</b>									
MPI	309.1	156.2 0.99	81.9 0.94	52.2 0.99	40.0 0.97	27.1 0.95	20.7 0.93	10.9 0.89	6.3 0.77
HPF	317.0	159.3 0.99	81.5 0.97	55.8 0.95	43.1 0.92	30.9 0.85	25.3 0.78	19.3 0.51	22.6 0.22
<b>Measurement performed on the Compaq SC232</b>									
MPI	52.4	28.6 0.92	14.8 0.89	-	7.5 0.88	-	3.92 0.83	2.27 0.72	
OpenMP	54.2	27.0 1	13.6 0.99	-	-	-	-	-	-
<b>Measurement performed on the Cluster de Pcs</b>									
MPI	147.3	81.6 0.90	42.94 0.86		23.7 0.78		13.6 0.68	-	-
<b>Measurement performed on the HP class V</b>									
MPI	157.8	80.8 0.98	41.2 0.96	28.4 0.93	22.1 0.89	15.7 0.84	13.7 0.72	-	-
HPF	154.8	79.3 0.98	41.9 0.92	30.8 0.84	24.8 0.78			-	-
OpenMP	159.7	80.8 0.99	40.8 0.98	27.7 0.96	20.9 0.95	14.5 0.92	11.9 0.84	-	-
<b>Measurement performed on the SGI Origin2000</b>									
MPI	221.5	113.0 0.98	56.2 0.98	38.4 0.96	29.2 0.95	21.0 0.88	16.2 0.85	9.1 0.76	-
HPF	211.1	106.8 0.99	55.1 0.96	38.6 0.91	30.9 0.85	25.1 0.70	22.4 0.59	25.4 0.26	-
OpenMP	218.6	120.1 0.91	59.6 0.92	41.6 0.88	31.4 0.87	21.89 0.83	17.3 0.79	10.7 0.64	-
<b>Measurement performed on the Compaq Proliant 6000</b>									
MPI	410.4	227.5 0.90	117.3 0.87	-	-	-	-	-	-
HPF	437.8	220.9 0.99	116.6 0.94	-	-	-	-	-	-
OpenMP	431.2	217.6 0.99	115.3 0.93	-	-	-	-	-	-

# Experimental Results :

## Comparing HRM1D MPI & HPF versions on MPP

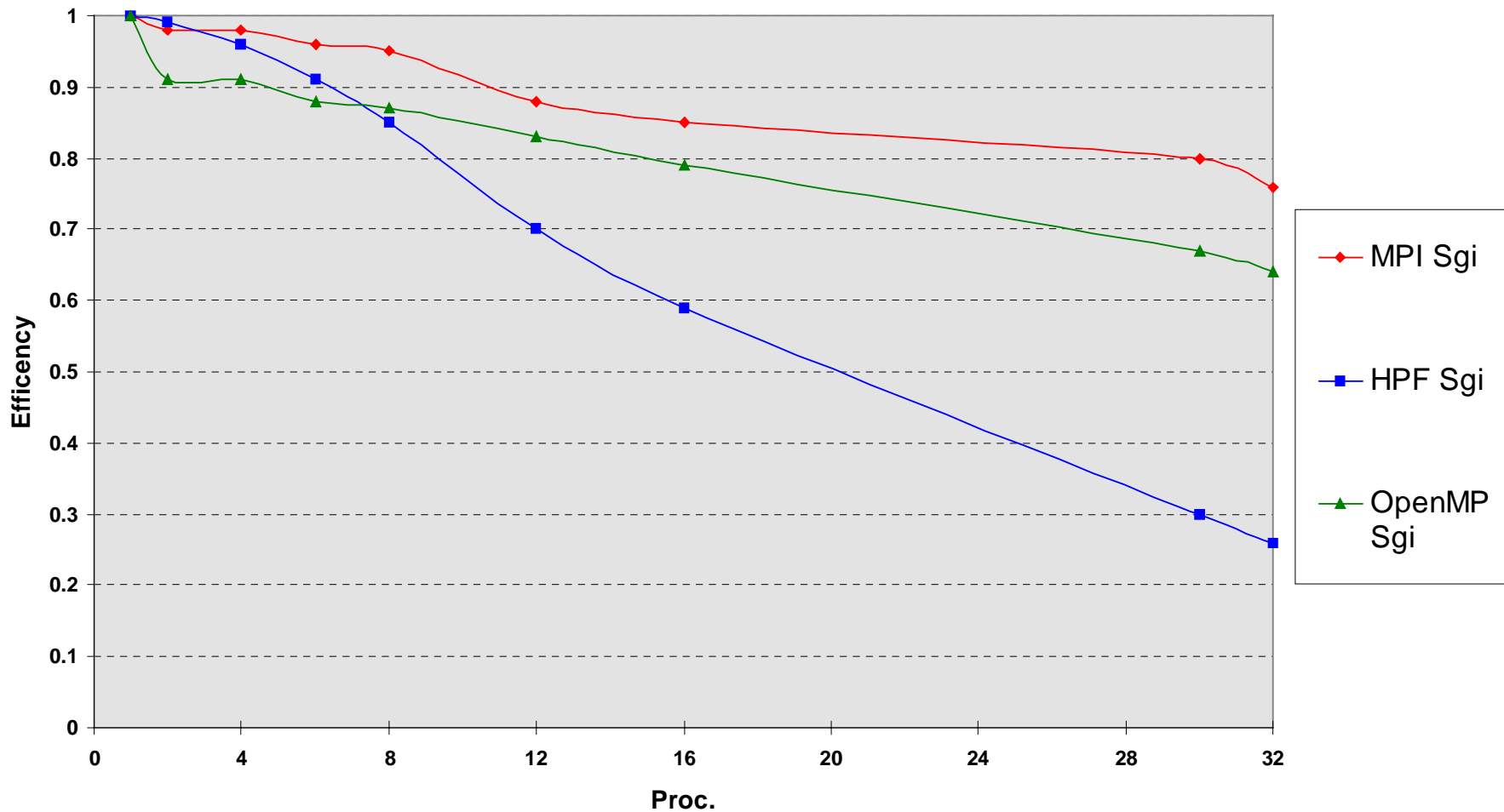
### HRM1D Efficiencies (T3E, SC232, Cluster)



# Experimental Results :

## Comparing MPI,HPF,OpenMP versions on SGI 2K

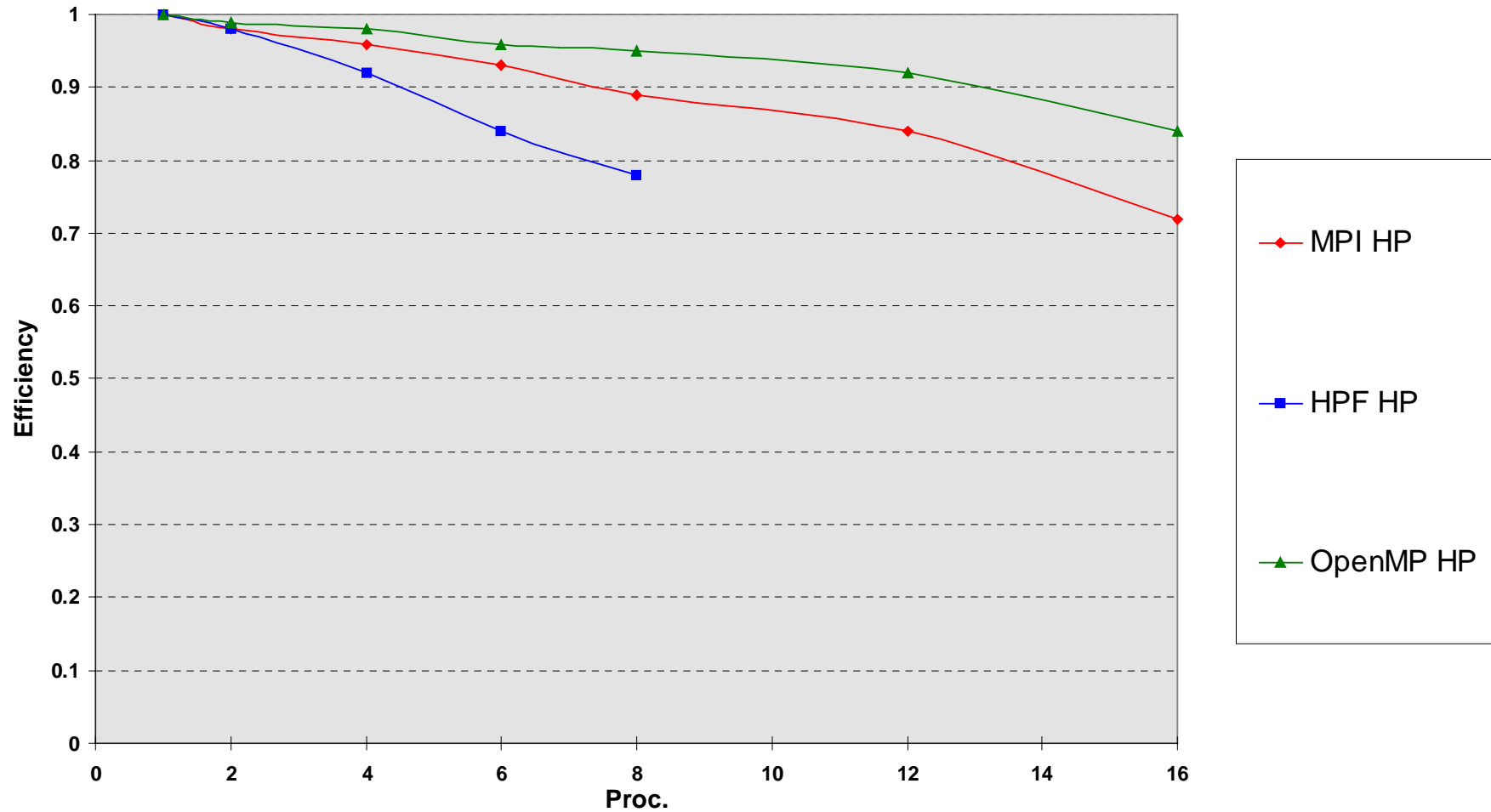
### HRM1D Efficiencies (SGI)



# Experimental Results :

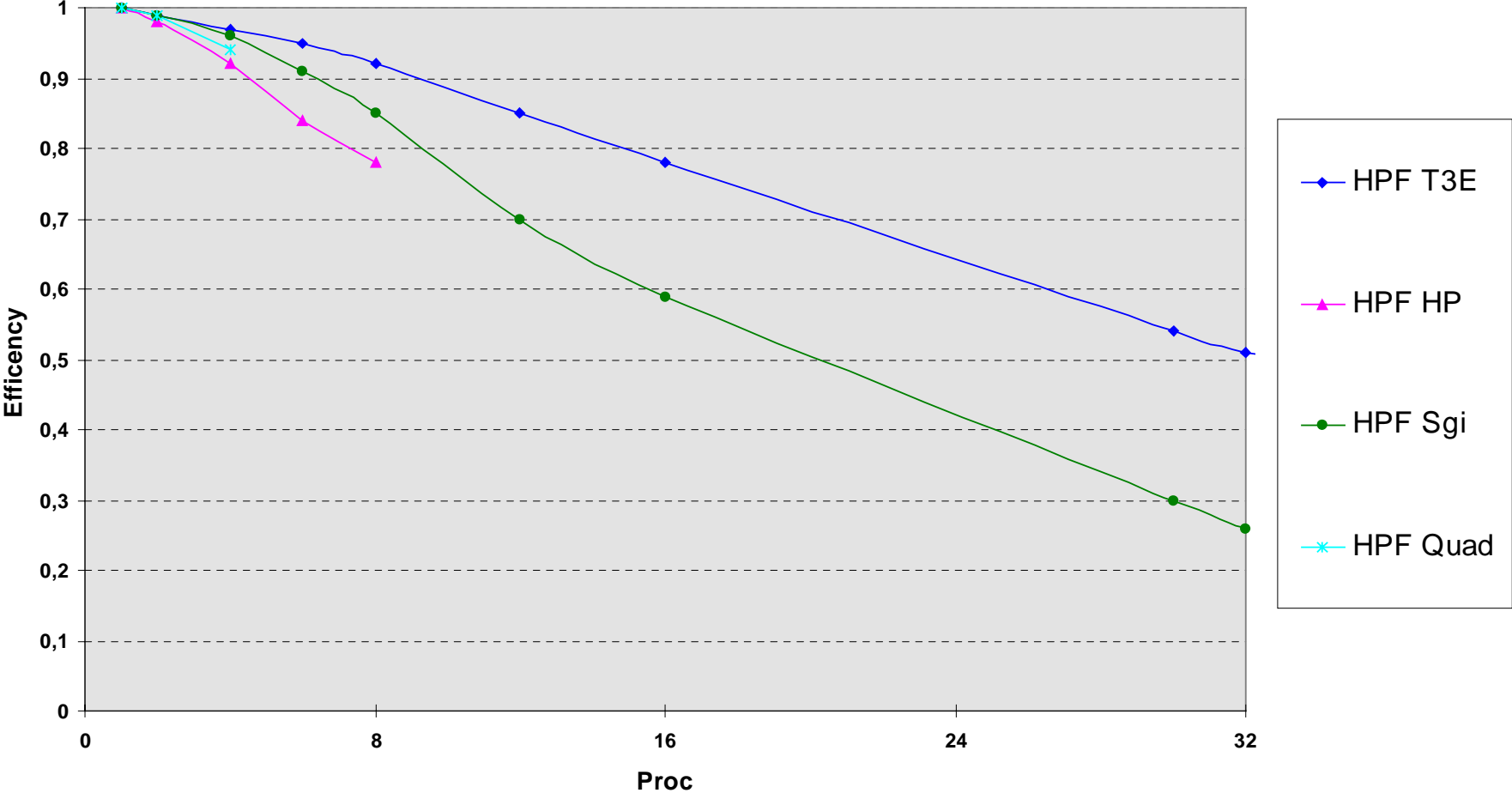
## Comparing MPI,HPF,OpenMP versions on HP class V

HRM1D Efficiencies (HP)



# Experimental Results : Comparing HPF on various platforms

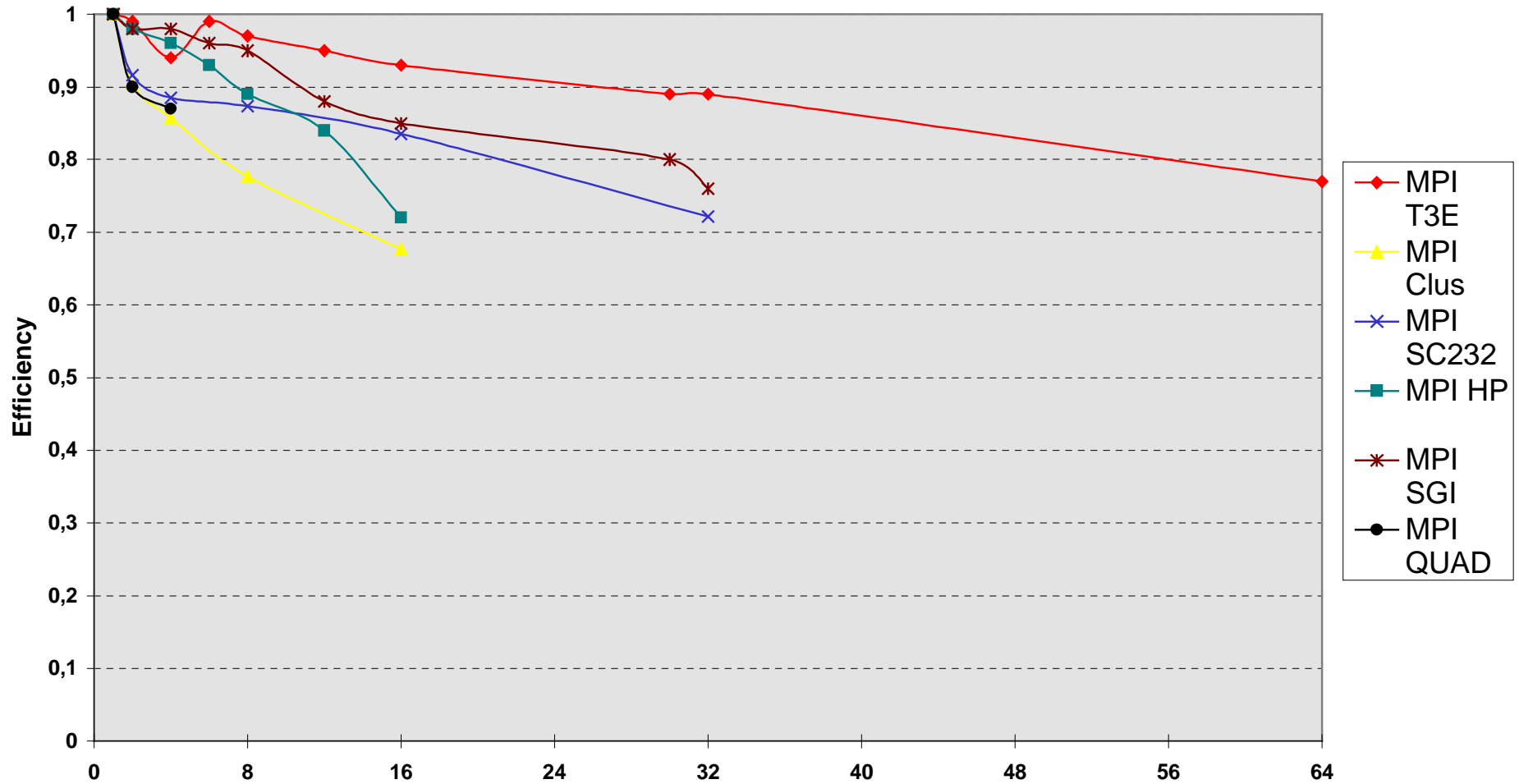
HPF Version : HRM1D Efficiencies



# Experimental Results :

## Comparing MPI on various plateforms

MPI version : HRM1D Efficiencies

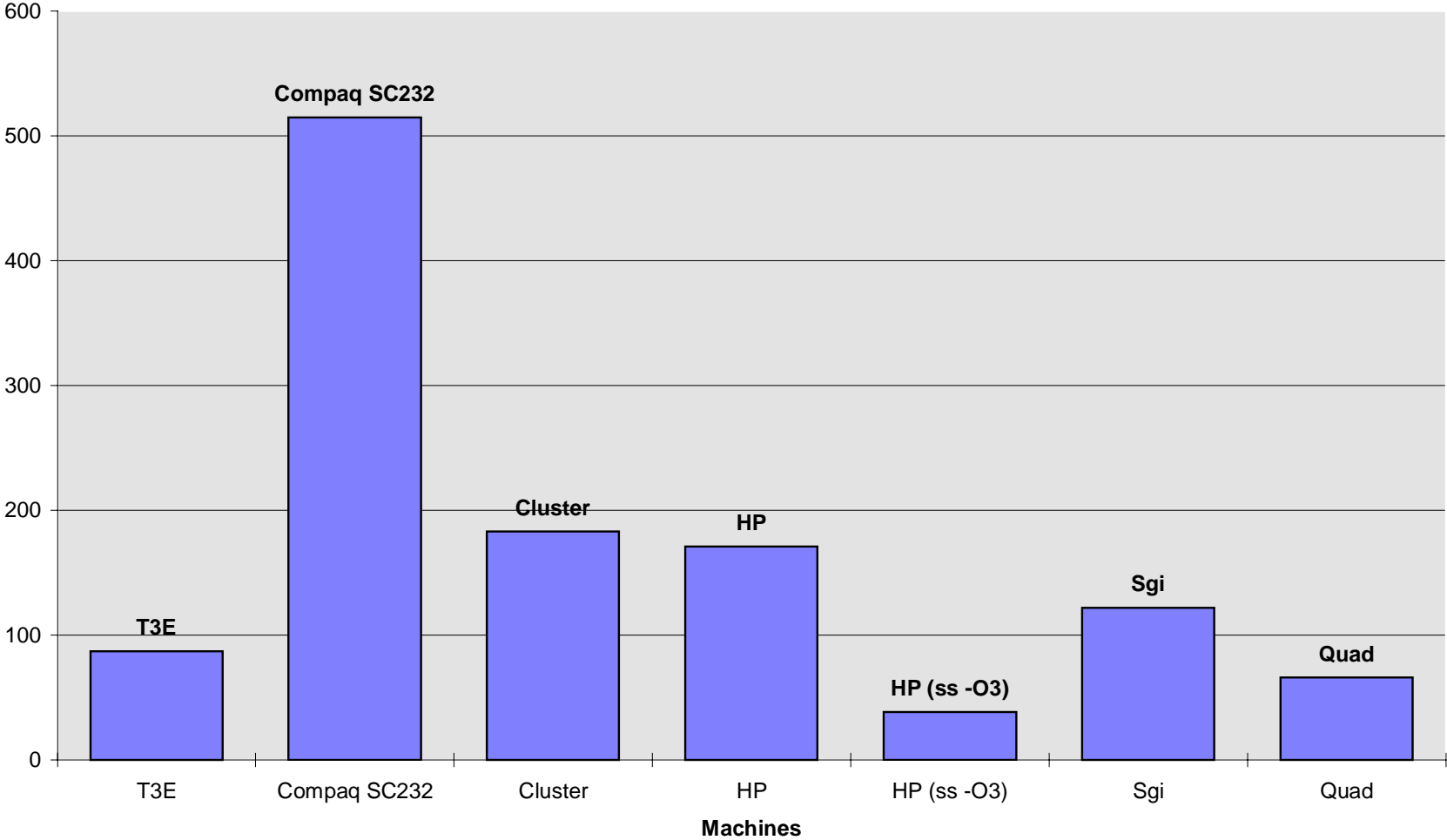


Proc.  
EWOMP 2000, September 2000

# Experimental Results :

## Peak Performances

Mflops (1 proc)



# Real case : 100000 elements, 10000 time step

## Mesures effectuées sur Cray T3E

Versions/ NCPUS	1	2	4	6	8	12	16	32	64	128
MPI	154462		42903 0.90		21676 0.89		11055 0.87	5745	3261 0.74	2173 0.66
HPF			45613				16870 0.68		13635 0.20	-

## Mesures effectuées sur HP Class V

MPI	80904	43611 0.93	22539 0.90	15609 0.86	11505 0.88	7962 0.84	6480 0.78			
HPF	78663	41573 0.95	23010 0.86	17356 0.75	14271 0.69	12044 0.55	11809 0.42			
OpenMP	81291	41511 0.98	21123 0.96	.	11247 0.90	7886 0.86	6384 0.80			

On Cray T3E : 1 proc. = 43 hours, 64 proc. = 54 minutes,  
128 proc. = 34 minutes

# Relevance of the CYCLIC distribution

Increase of the HRM1D execution time with a block distribution compared to a cyclic distribution (100000 elements, 10000 time steps):

<b>NCPUS</b>	<b>4</b>	<b>8</b>	<b>16</b>	<b>32</b>	<b>64</b>
Cray T3E / version MPI			33%	50%	49%
HP Class V / version MPI			22%		
HP Class V / version OpenMP	6%	12 %	28 %		

# HRM1D parallelization conclusions

	<b>Development Time</b>	<b>Efficiency/ Scalability</b>	<b>Technicality</b>	<b>Portability</b>	<b>Readability/ Maintainability</b>
<b>MPI</b>	2 Months	80% up to 64 processors	moderate	Optimizations are MPI implementation and machine dependent	Bad
<b>HPF</b>	2 Weeks	80% up to 8 processors	High	Optimizations are compiler dependent	Good
<b>OPENMP</b>	2 Weeks	80% up to 16 processors	Low	Limited to shared memory machines	Good

# HRM1D parallelization conclusions

Viability of the HRM1D parallelization : proved

What is the best parallelization strategy for HRM1D code?

- When large speed-up targeted : MPI
- When moderate speed-up targeted :
  - Shared memory machine : OpenMP
  - Distributed memory machine : HPF

Cluster of PCs may be an inexpensive solution for distributed computing

# ECOSS PARALLELIZATION

- 11 000 lines FORTRAN 90 code
- 20 000 1D mesh, 2.5 s physical phenomenon : **100 days** on SUN Sparc Ultra60 (years on 2D meshes) => large speed-up targeted => message passing version
- ECOSS parallelization :
  - First step : OpenMP version => check parallelisation analysis, test parallelization relevance, experiment various work distributions
  - Second step : MPI version
- Target machines : SC COMPAQ 232, cluster of PC

# ECOSS PARALLELIZATION

- Minimize MPI programming cost => MPI block distribution => lower performance

Versions/ NCPUS	1	2	4	8	16
<b>Measurement performed on the Compaq SC232</b>					
MPI	4253	2238 0.95	1407 0.76	801 0.66	650 0.41
<b>Measurement performed on the Cluster de Pcs</b>					
MPI	14995	8244 0.91	5279 0.71	2916 0.64	1756 0.53
<b>Measurements performed on the HP class V machine</b>					
MPI	16884	10478 0.81	6378 0.66	3393 0.62	2144 0.49
<b>Measurements performed on the SGI Origin2000 machine</b>					
MPI	12778	6700 0.95	4261 0.75	2414 0.66	1762 0.45

**20 000 elements, 0.01 s of simulation time**

# What is the best parallelization strategy?

When **large speed-up** is targeted ( $>32$ ) :

- First develop a OpenMP version
  - Easily and quickly developed
  - Evaluate the parallelization relevance
  - Test easily various work distribution policies
- Second, develop a HPF version
  - Quickly developed with HPF expertise (using OpenMP parallelization analysis)
- If HPF version inefficient, develop a MPI (or PVM) version : using HPF parallelization analysis

# What is the best parallelization strategy?

When **moderate speed-up** targeted :

→ Shared memory machine : OpenMP

→ Distributed memory machine : same strategy as for large speed-up (OpenMP -> HPF-> MPI)

Panel Session

**OpenMP after Fortran 2.0**

Jean-Yves Berthou

Electricité de France

Research and Development division

# OpenMP is good for you!

Complete API for programming shared memory machines

Easy to learn (and to teach)

Easy and quick to write

Easy to test various work distribution policies

Seems really portable

Compilers seems efficient

Debuggers and profilers available

## But (1) ...

- Performance optimization : take care of non local memory accesses (as for HPF programs)

➤ Competition between the work distribution granularity and the memory accesses. As an example, does such a loop has to be parallelized ?

```
!$OMP DO SCHEDULE (STATIC,CHUNK)
DO I=1,NPTOT
    AIREXG(I)=AIRNP1(I)
ENDDO
!$OMP END DO
```

➤ Do we have to write OpenMP programs as MPI programs ?  
=> privatize most data

## But (2) ...

- We were recently confronted to several **compiler bugs** (KAI guidef90, SGI f90 MIPSPro) : very time consuming
- Mixing MPI and OpenMP : what about portability !