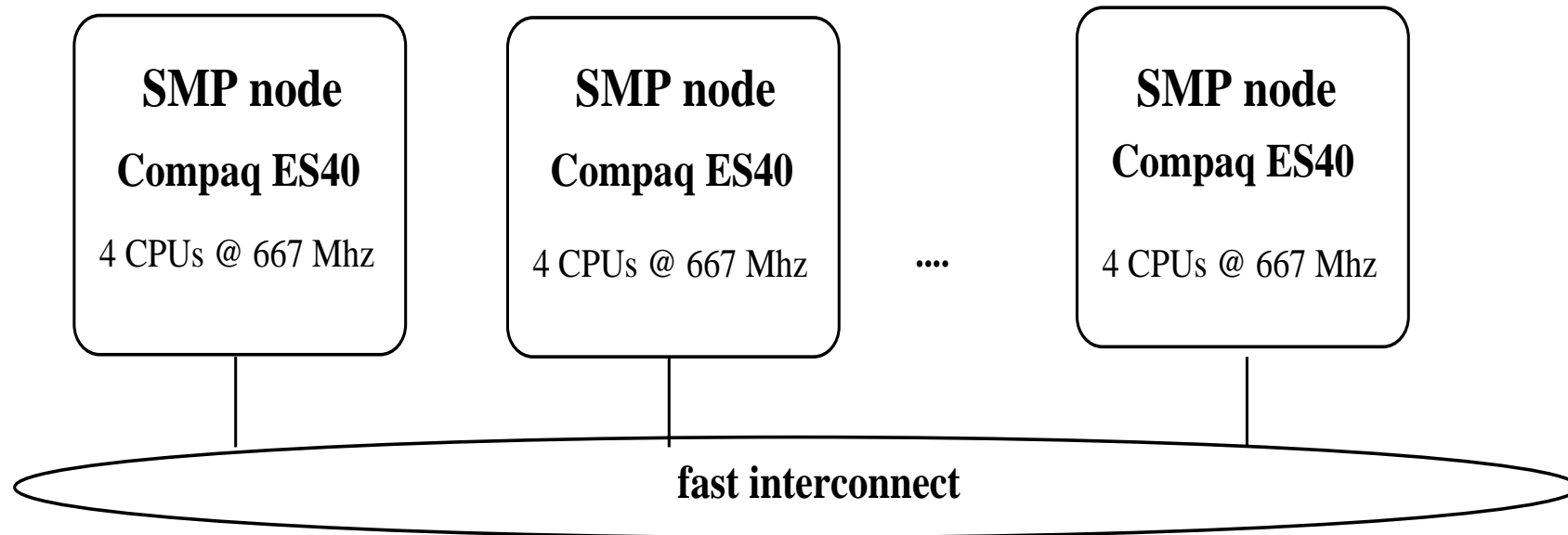


OPENMIP AND MPII
PROGRAMMING WITH A
CG ALGORITHM

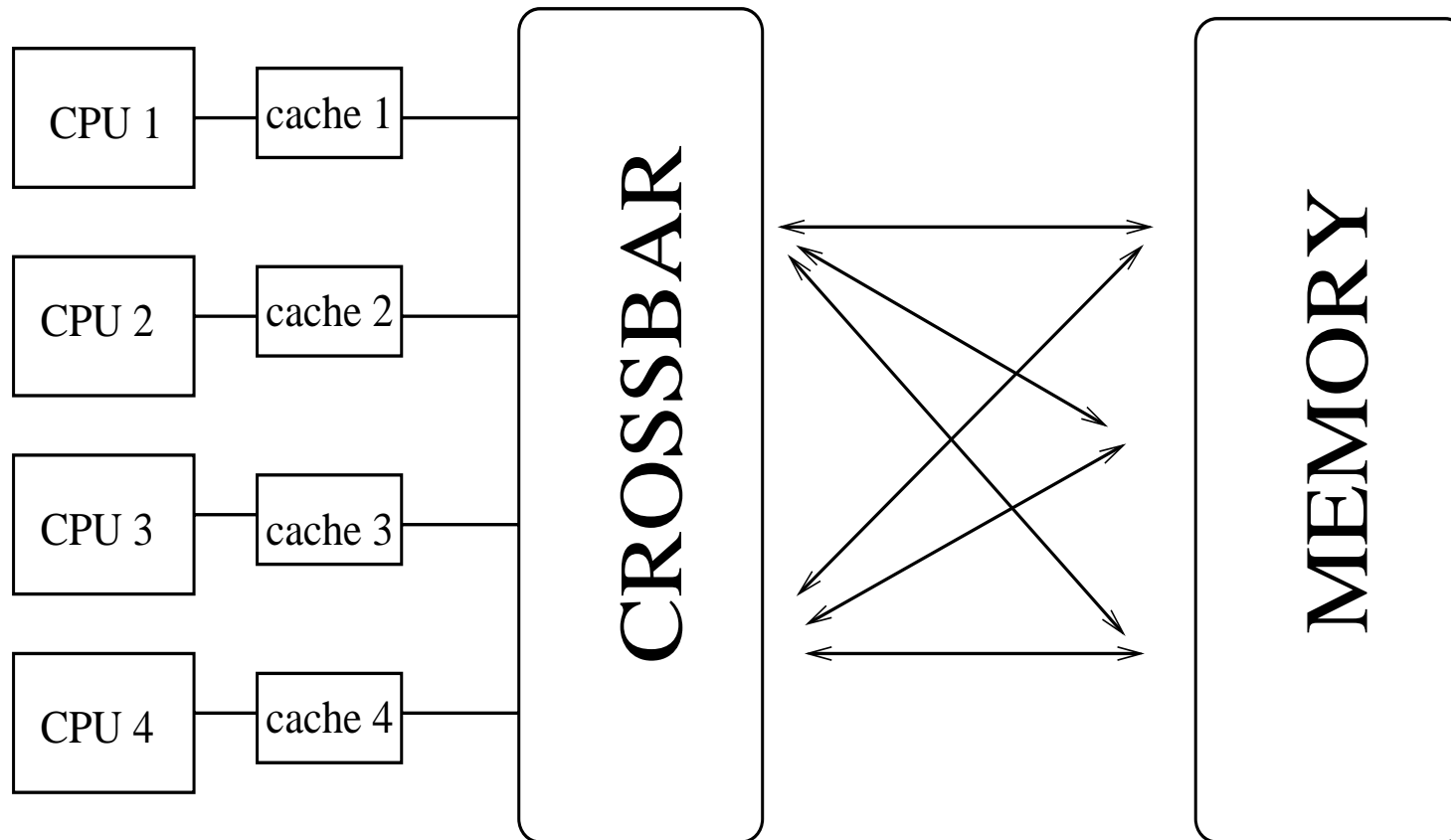
P. Kloos, P. Blaise, F. Mathey

CEA

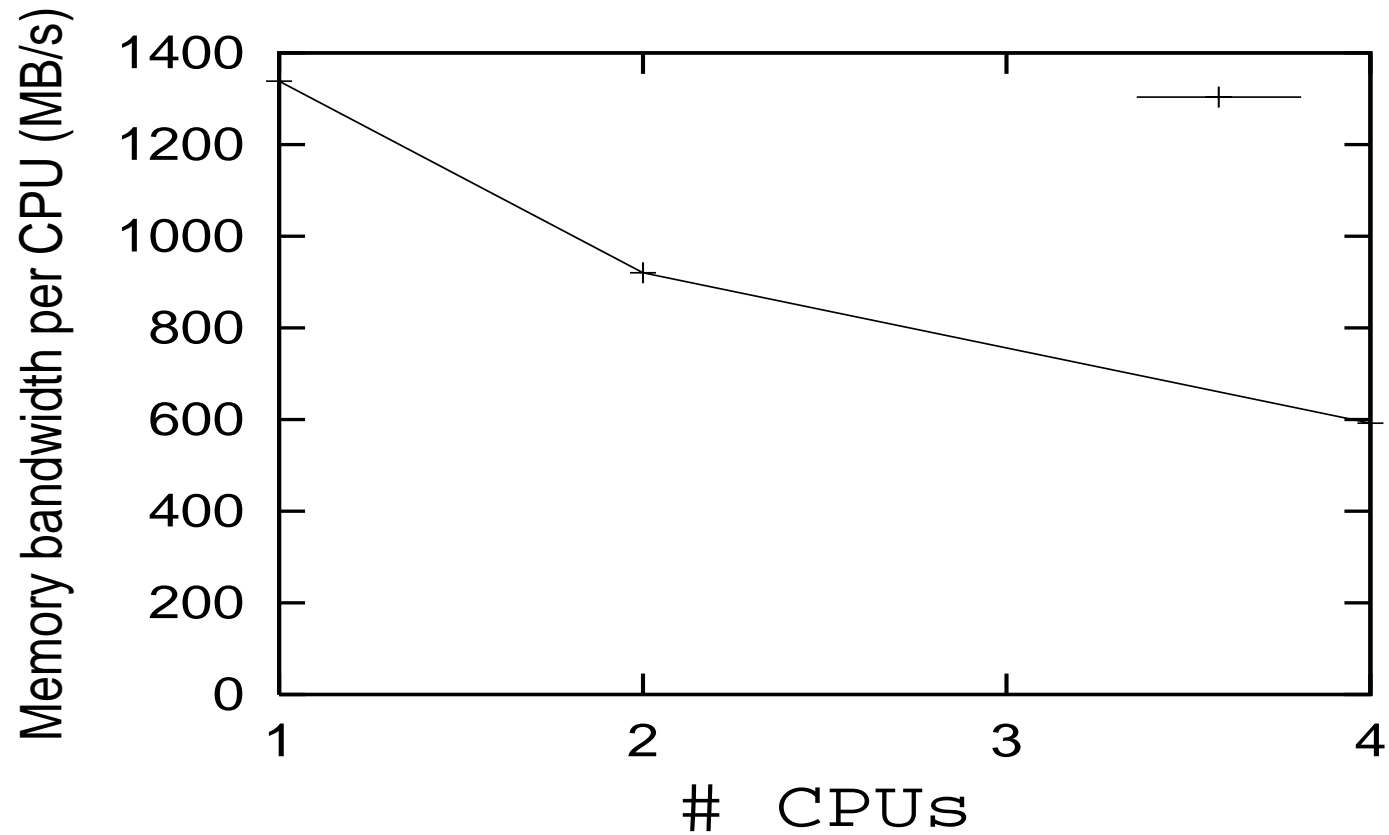
Compaq SC : A cluster of SMP nodes



A Compaq ES40 node



Memory bandwidth per CPU on a node



Conjugate Gradient algorithm

1 matrix vector product (95% of CPU time), 5
vector–vector products

while ($\|r\| < \epsilon$)

$$\beta = \|r\|^2$$

$$\alpha = \frac{\beta}{p^t A p}$$

$$v = v + \alpha p$$

$$r = r - \alpha A p$$

$$\beta = \frac{\|r\|^2}{\beta}$$

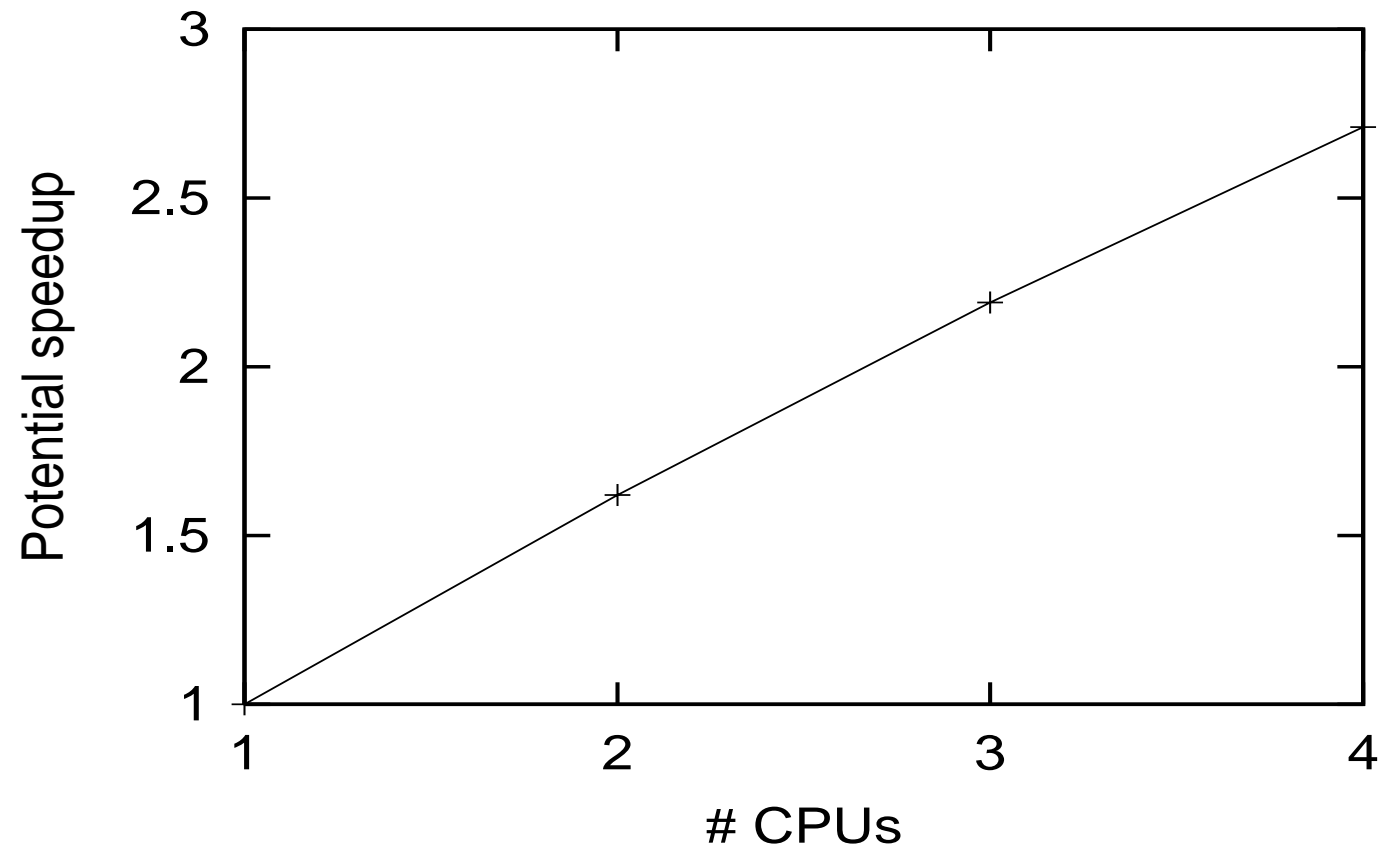
$$p = r + \beta p$$

Potential speedup

- Complexity of CG : $O(n^2)$
- On each CPU : $O(n^2/p)$, equivalent to a problem of size n/\sqrt{p}
- « Potential speedup » to take into account memory contention :

$$\frac{\text{time for one } O(n^2) \text{ problem}}{\text{time for } p \text{ sequential } O(n^2/p) \text{ problems}}$$

Potential speedup



Loop-level and SPMD programming

- Loop-level :

```
!$OMP PARALLEL PRIVATE(i) SHARED(a,b,n)
```

```
!$OMP DO
```

```
DO i=1,n
```

```
  a(i) = a(i) + b(i)
```

```
ENDDO
```

```
!$OMP ENDDO
```

```
.....
```

```
!$OMP END PARALLEL DO
```

Loop-level and SPMD OpenMP

- SPMD programming :

```
!$OMP PARALLEL PRIVATE(start,end,i)
```

```
!$OMP+ SHARED(a,b)
```

```
nb_threads = omp_get_num_threads()
```

```
thread_num = omp_get_thread_num()
```

```
start = n * thread_num / nb_threads + 1
```

```
end = n * (thread_num+1) / nb_threads
```

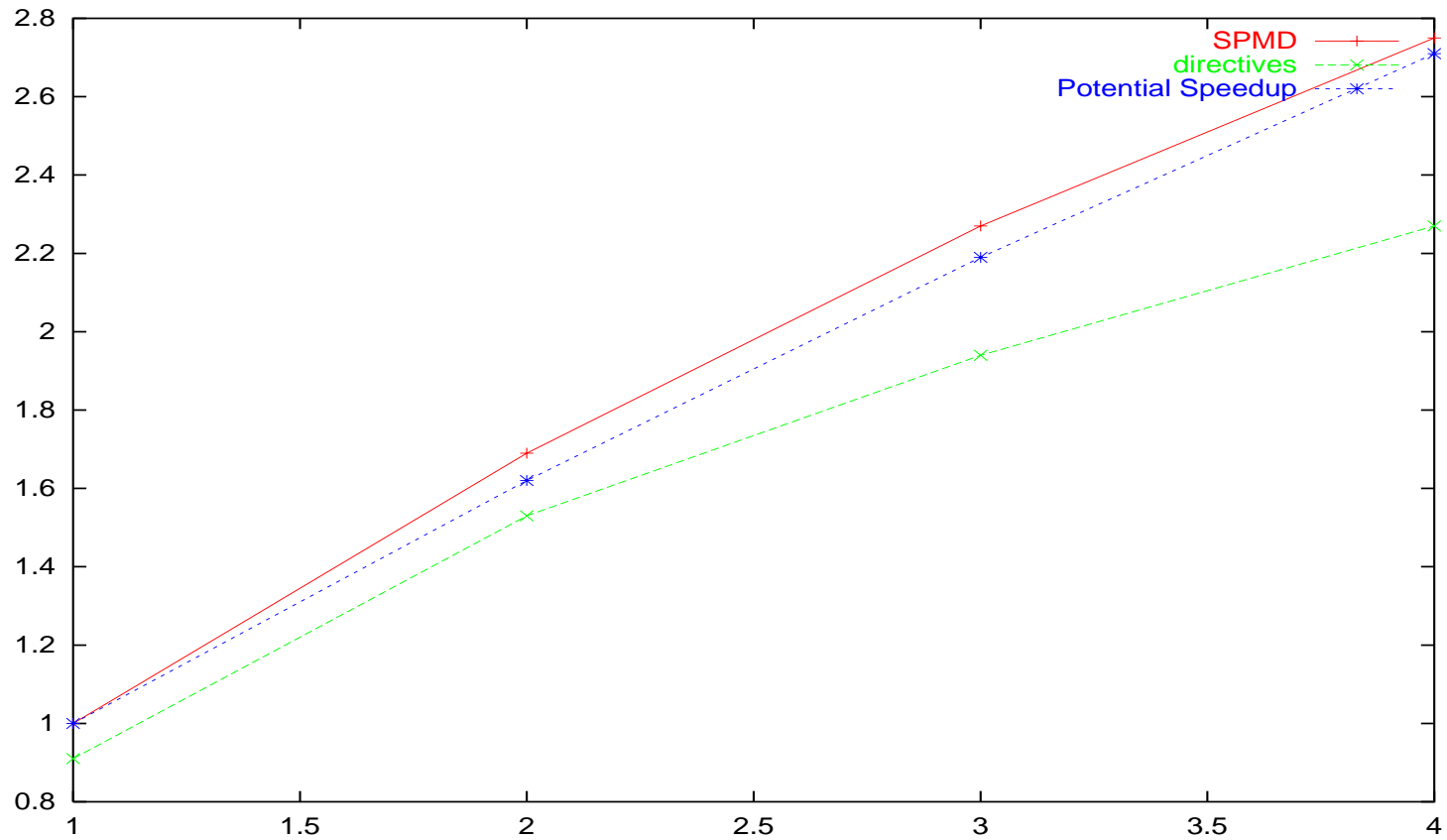
```
do i = start, end
```

```
  a(i) = a(i) + b(i)
```

```
enddo
```

```
!$OMP END PARALLEL
```

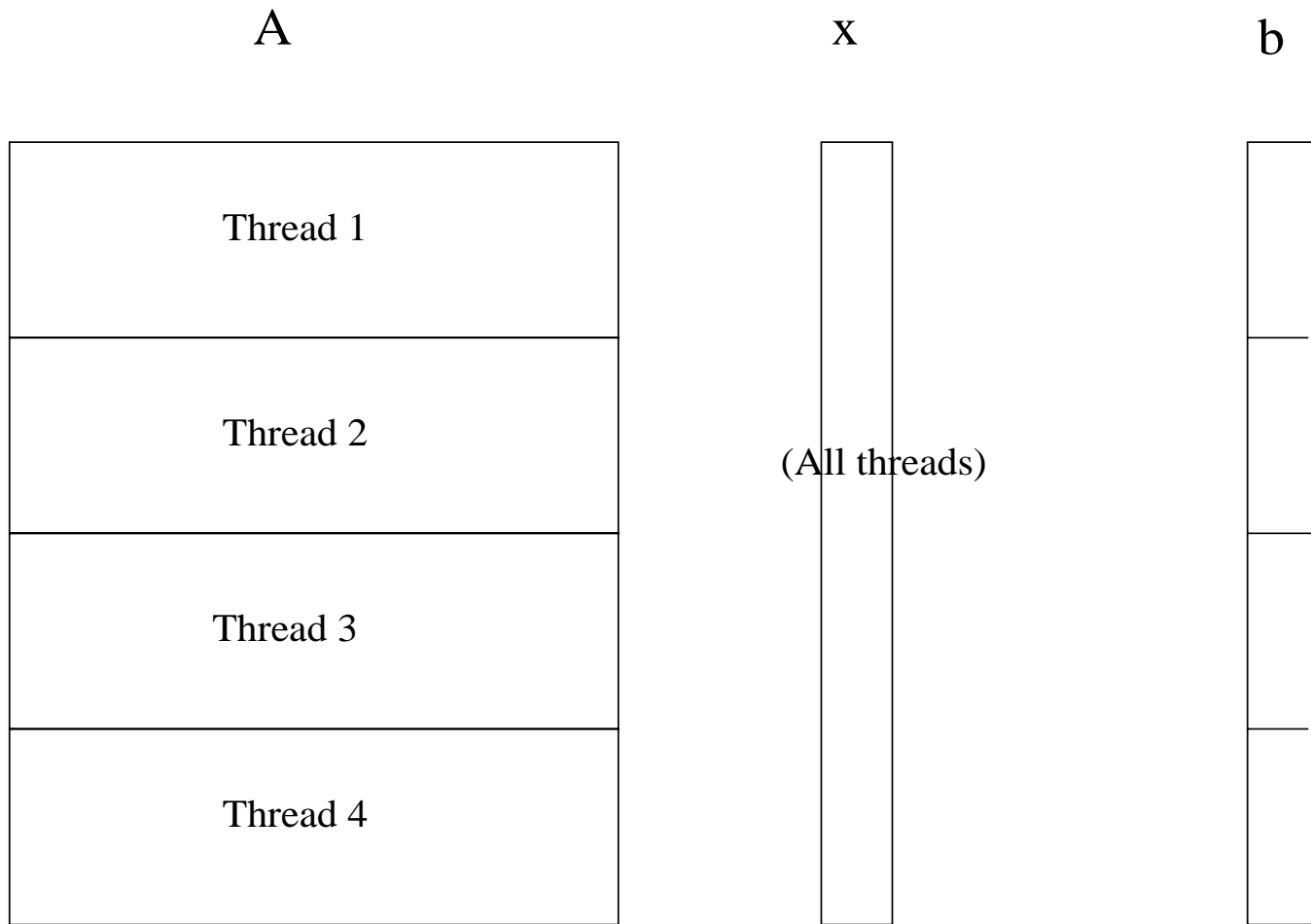
Loop level and SPMD OpenMP



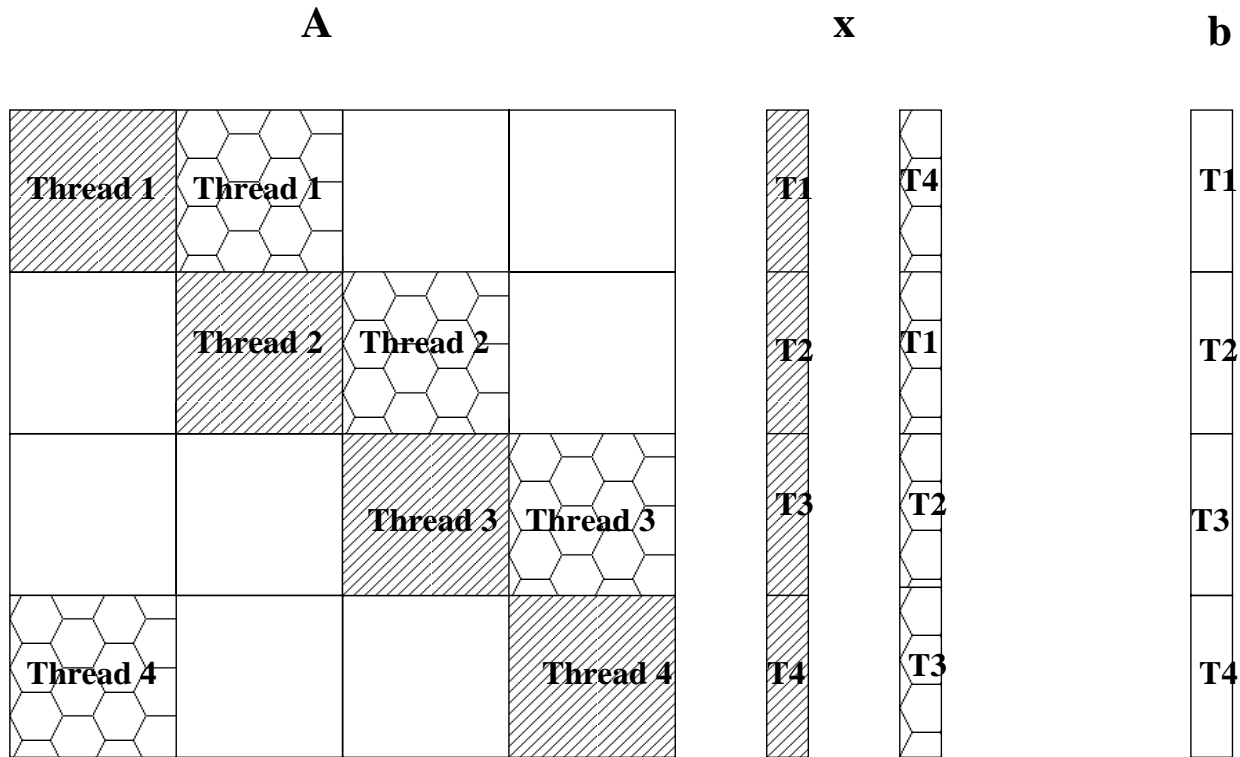
Loop level vs SPMD OpenMP

- Loop–level OpenMP is easy to implement, but less scalable :
 - some compiler optimizations are disabled
 - more synchronisations ?
 - cache misses ?
 - data decomposition is recalculated ?

Data decomposition



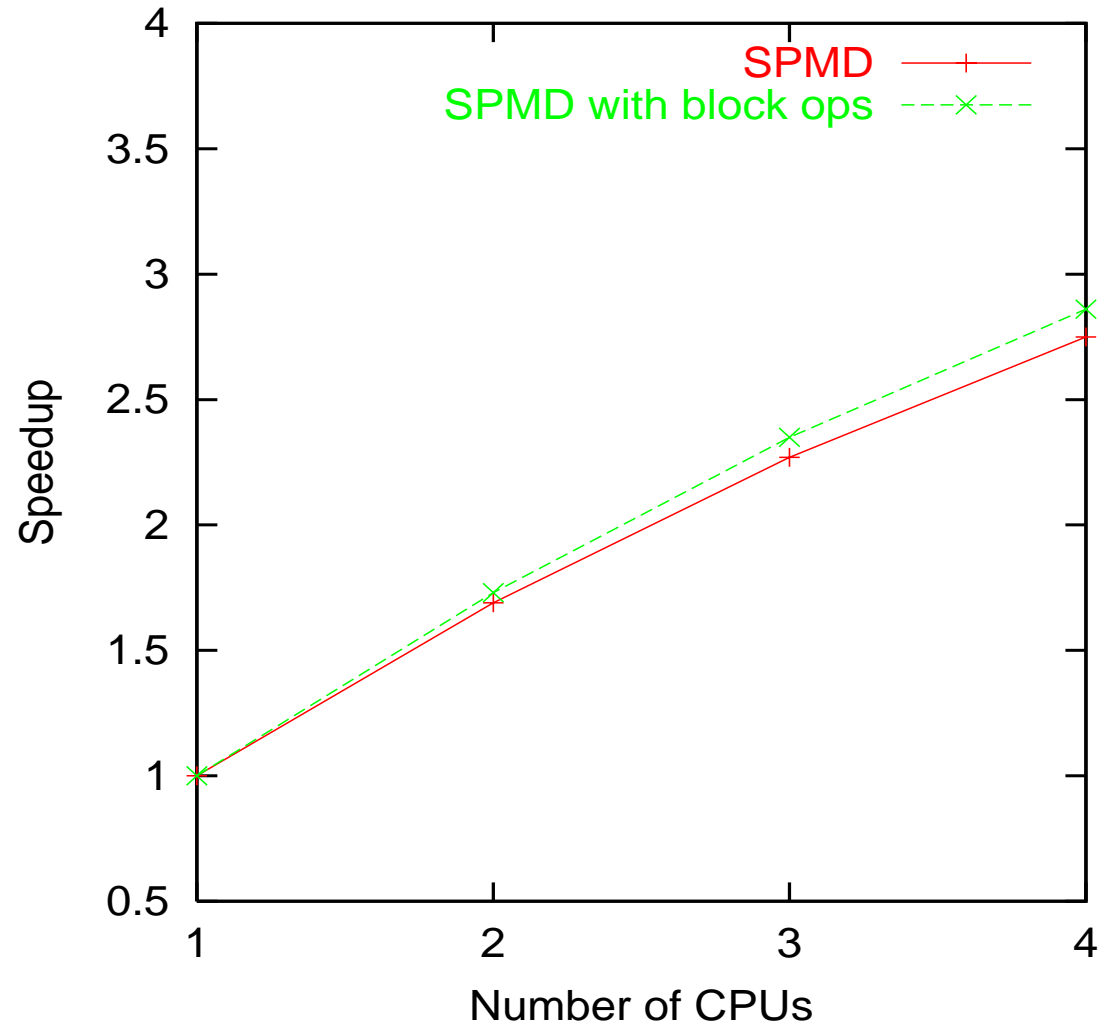
Block data decomposition



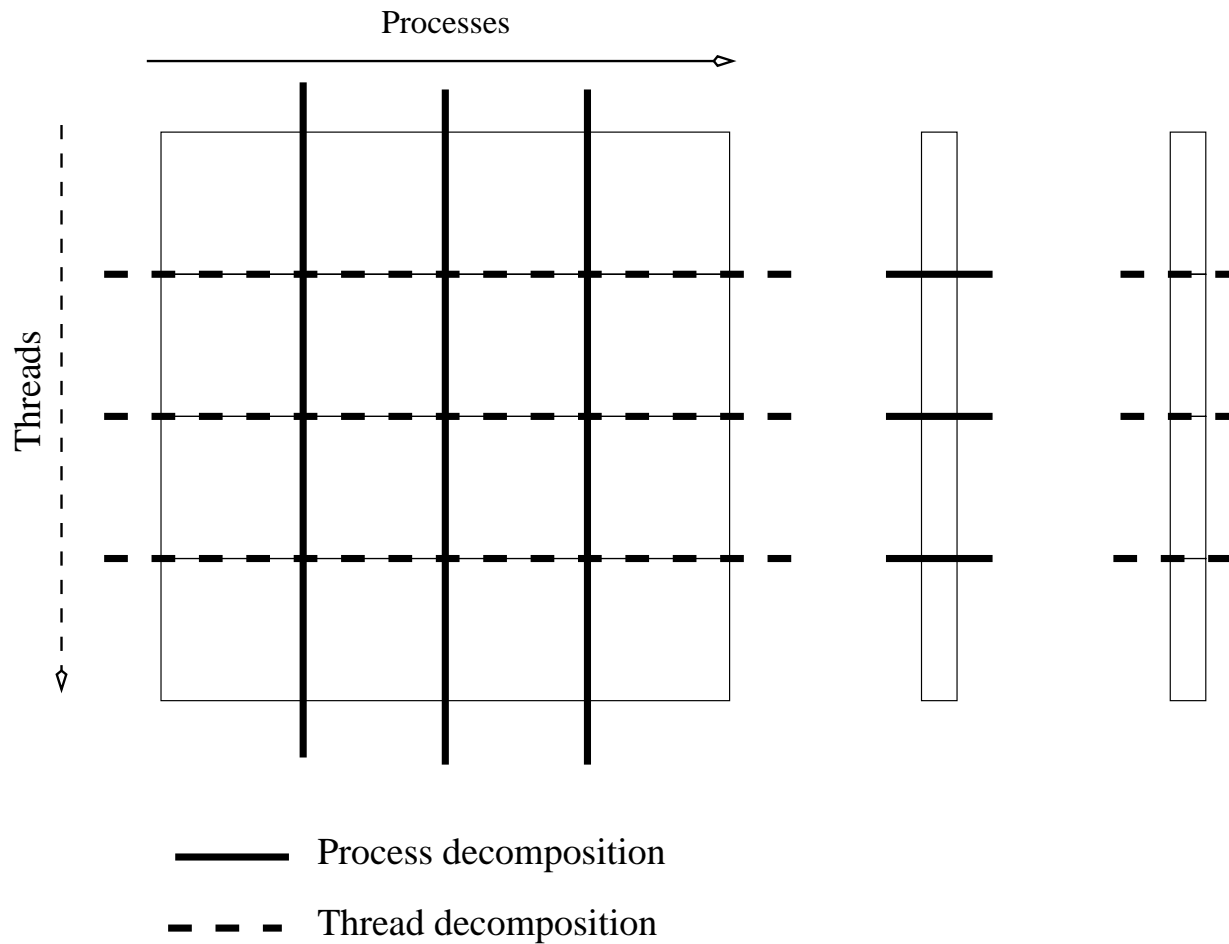
 **Step 1**

 **Step 2**

Block data decomposition



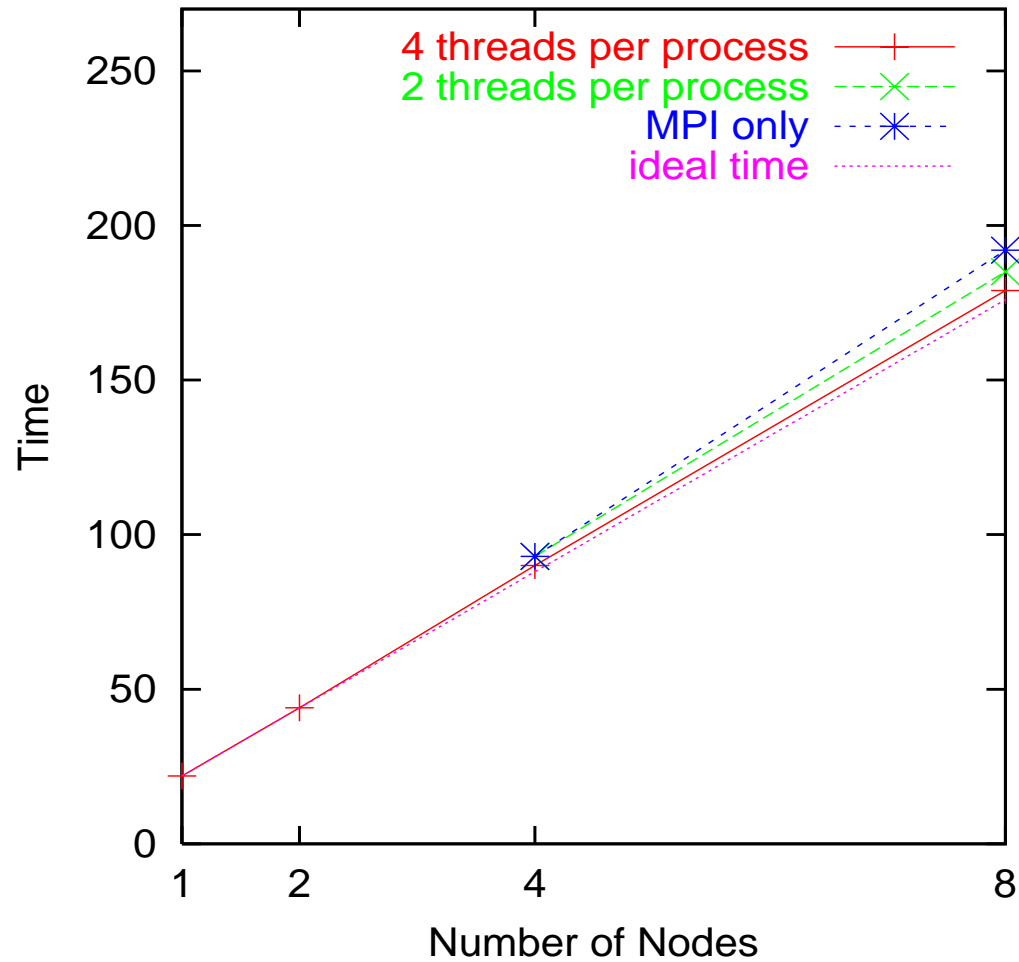
MPI/OpenMP implementation : matrix vector product



MPI/OpenMP implementation : Tests

- The problem size is increased linearly with the number of nodes
- Complexity on each node :
 $O((n \times \text{nodes})^2 / \text{nodes}) = O(n^2 \times \text{nodes})$

MPI/OpenMP implementation : results



Conclusions

- Scalability problem with loop level OpenMP. Need for a directive that keeps the same data distribution across several loops ?
- MPI/OpenMP implementation performs better from 8 nodes.