HP-SEE
Programming with MPI & OpenMP

www.hp-see.eu

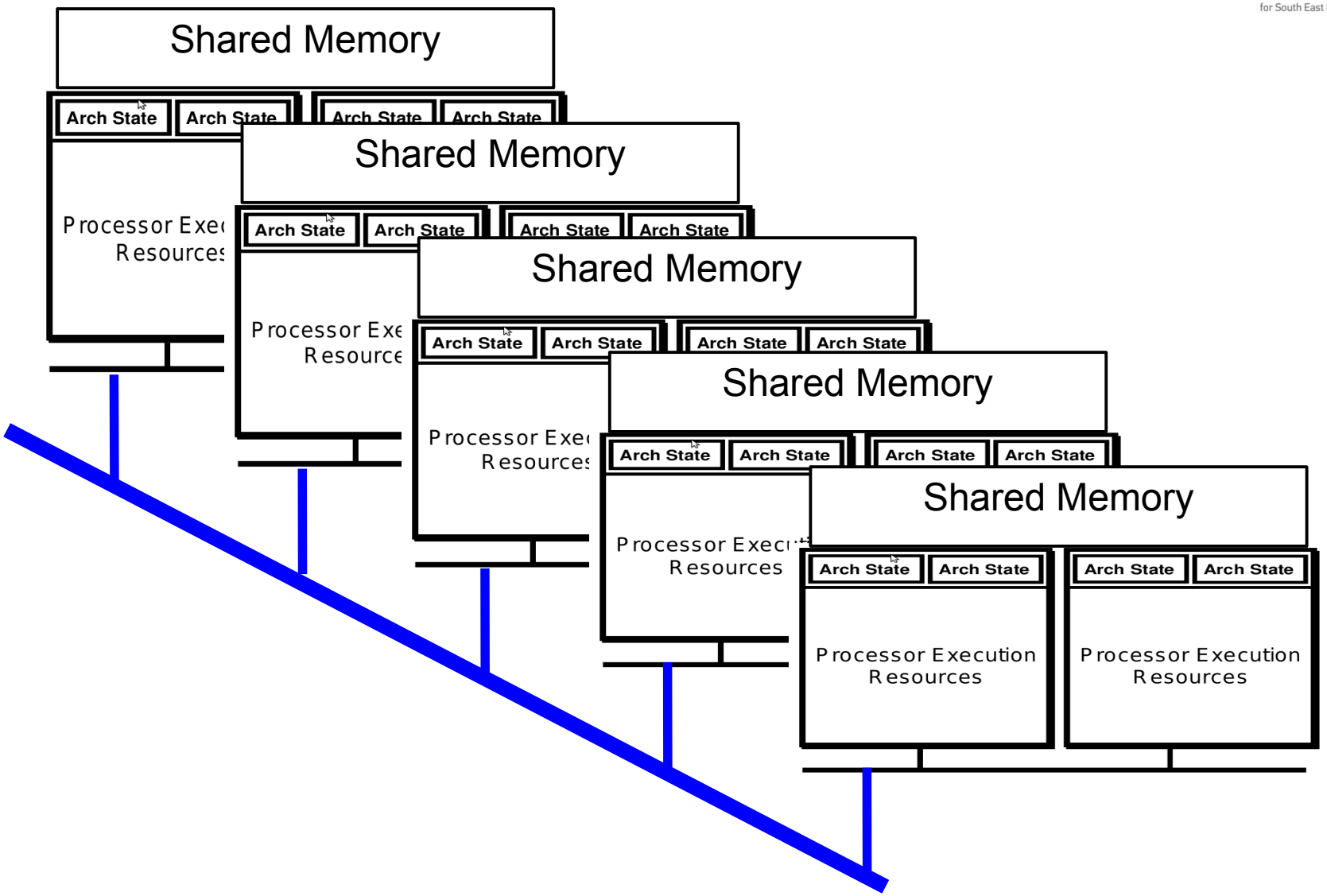
HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

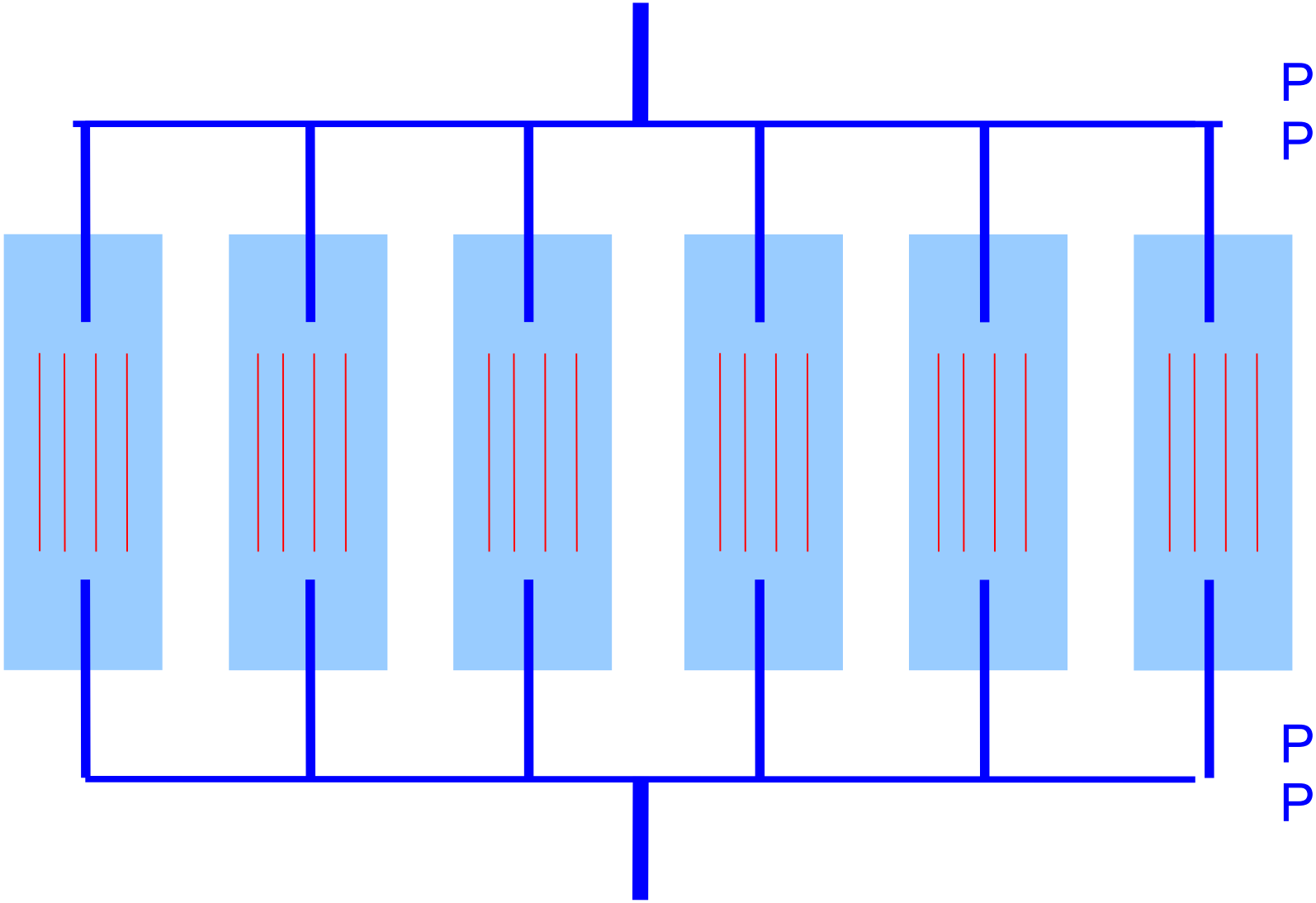
N. Frasheri, B. Cico

The HP-SEE initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract no. 261499

Typical HPC Architecture



Combined MPI & OpenMP



MPI&OpenMP Source Code

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef struct
{ double thred; double start; double  stop;
} metad; metad   * array, procs;
int main(int argc, char **argv)
{   int rank,size,noel,nopr,ntrd,i,j,k;
  double  sinsin, startime, stoptime;
  long int li, Nruntime;   Nruntime=9999999;
  nopr = 1; ntrd = 1; noel = 1;
  if (argc>1) nopr = atoi(argv[1]); // processes
  if (argc>2) ntrd = atoi(argv[2]); // threads
  if (argc>3) noel = atoi(argv[3]); // base iteration
  MPI_Init(&argc, &argv); // init MPI
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
// print inputs & take time in mother process
  if (rank==0)
  {  startime = MPI_Wtime();
     array = malloc(nopr*sizeof(metad));
     for (i=0; i<nopr; i++)
     {  array[i].thred=0; array[i].start=0; array[i].stop =0;
  } }
MPI_Scatter(&array[rank],3,MPI_DOUBLE,&procs,3,MPI_DOU
    BLE,0,MPI_COMM_WORLD);
// run processes inner loops
  procs.start = MPI_Wtime();
#pragma omp parallel for num_threads(ntrd) private(j, k, sinsin)
  for (j=0; j<16*noel/nopr; j++)
          for (k=0; k<Nruntime; k++)
          {  sinsin=sin(k);   }
// end of #pragma
  procs.thred= rank;   procs.stop = MPI_Wtime();
 MPI_Gather(&procs,3,MPI_DOUBLE,array,3,MPI_DOUBLE,0,
    MPI_COMM_WORLD);
// print process runtime vector in mother process
  if(rank==0)
  { for (i=0; i<nopr; i++)
          printf("Iter %d process %f Runtime %f \n",
          i, array[i].thred, array[i].stop - array[i].start);
      stoptime = MPI_Wtime();
      printf("time = %f \n",stoptime-startime);
  } MPI_Finalize(); // close MPI
  return 0; }
```

Comments on the Source

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

MPI processes main loop is automatic

Within processes the internal loop is split in threads using PRAGMA directive

Internal loop upper bound multiplied by *noel* to increase iterations > processes * threads

Preparation of Test

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Compilation

build.sh = mpicc -fopenmp $1.c /usr/lib/libm.a -o $1

Script

run.sh = (/usr/bin/time mpirun -np $2 ./$1 $2 $3 $4)

Execution

./run.sh test <process> <threads> <iterations>

Tested in hardware

Dell Inspiron ~ 1 processor 2 cores

Run Example

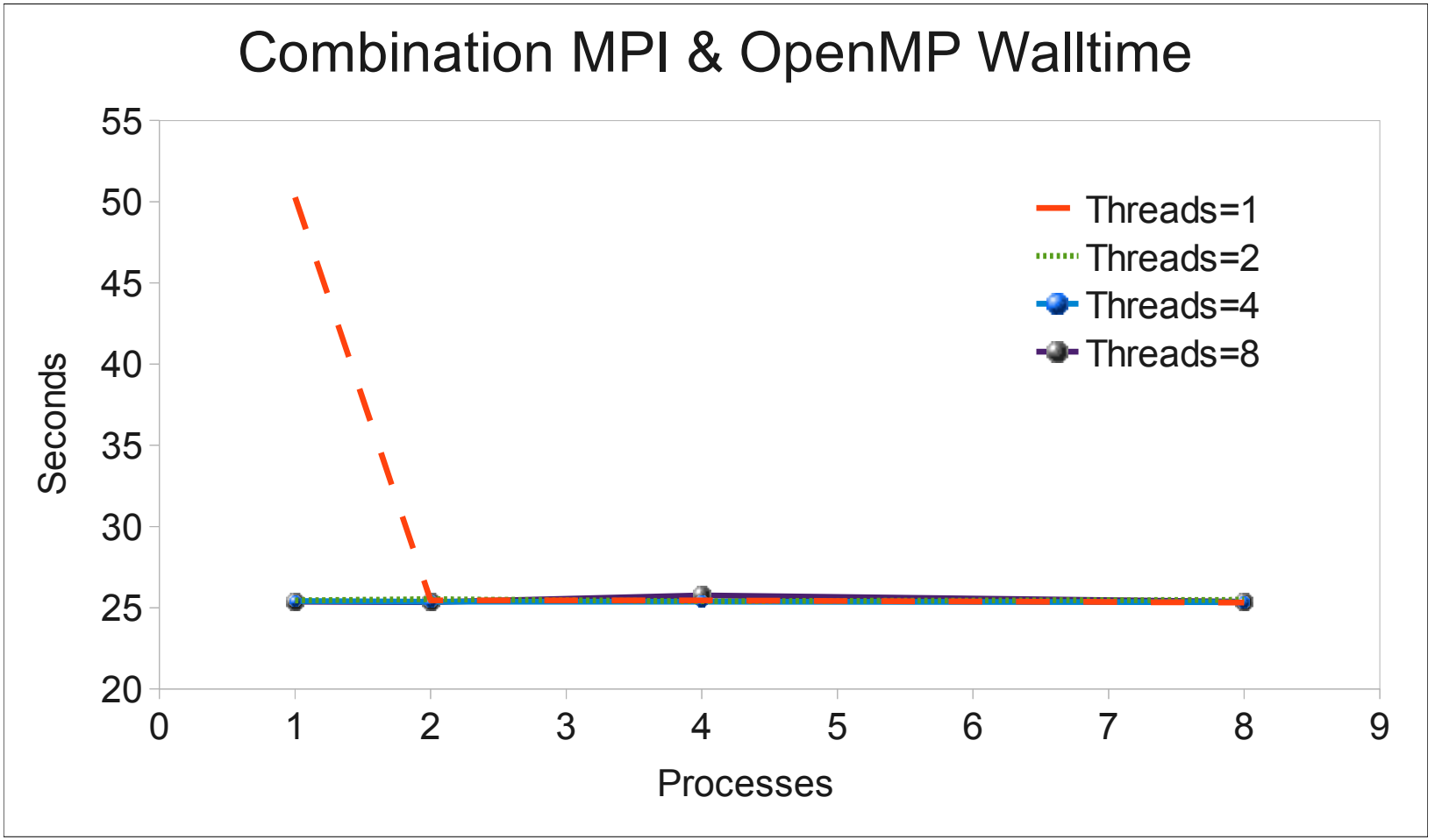
HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

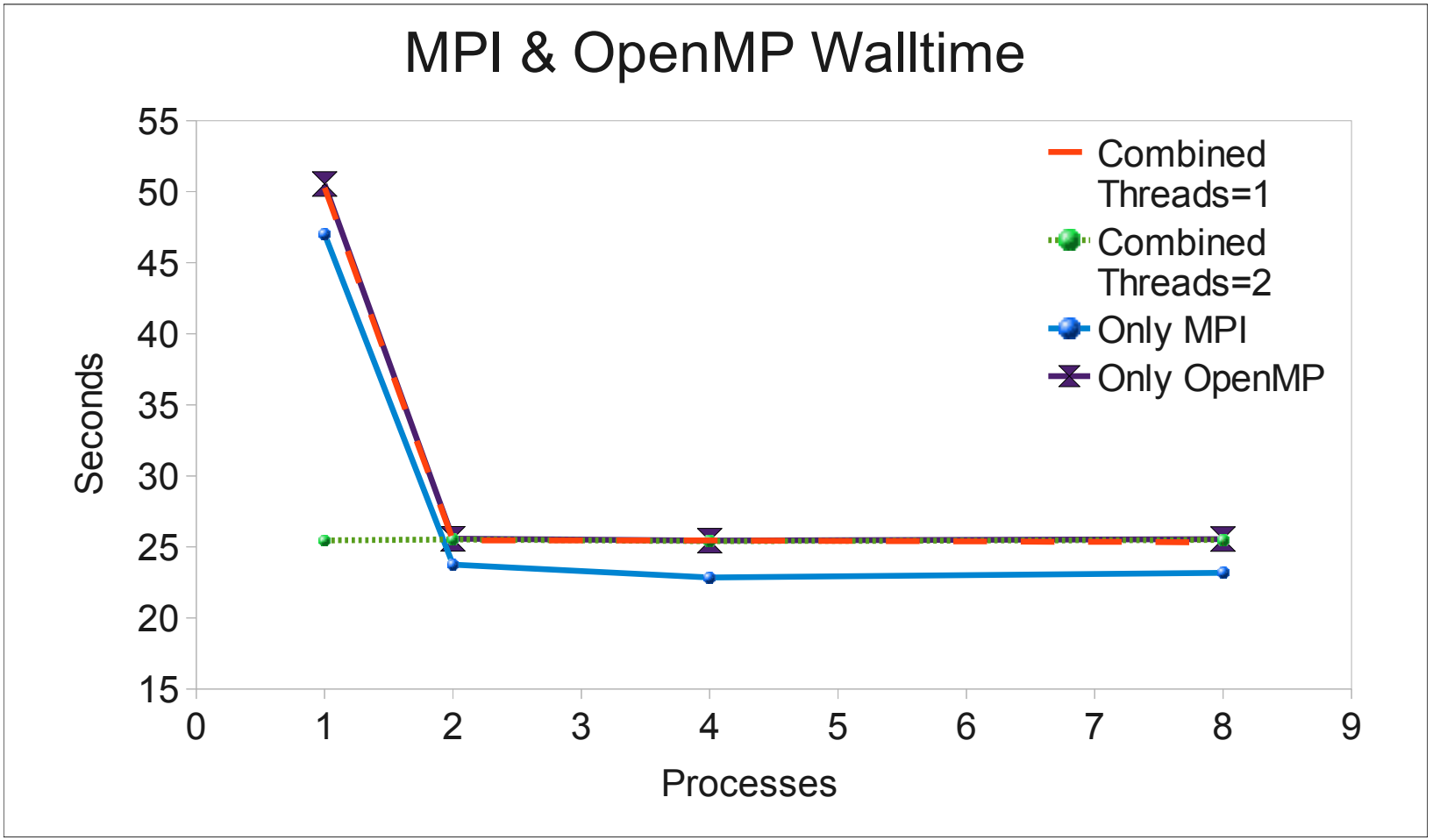
```
$ ./run.sh testime 8 4 4
input nopr=8 ntrd=4 noel=4
Iter 0 process 0.000000 Runtime 25.628089
Iter 1 process 1.000000 Runtime 25.627043
Iter 2 process 2.000000 Runtime 25.142232
Iter 3 process 3.000000 Runtime 25.625365
Iter 4 process 4.000000 Runtime 25.162345
Iter 5 process 5.000000 Runtime 25.624585
Iter 6 process 6.000000 Runtime 25.157813
Iter 7 process 7.000000 Runtime 25.598504
time = 25.628342
```

```
49.35user 0.07system 0:26.74elapsed 184%CPU
(0avgtext+0avgdata 16192maxresident)k
0inputs+5424outputs (27major+11367minor)pagefaults 0swaps
```

Test Walltime Chart



Compared with OpenMP / MPI



Comments on Performance

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

Performance ~ number of cores

Difference when

- Processes=1 and threads=1
- Processes>1 or threads>1

in single processor with two cores

Performance dominated from OpenMP

Question – how the Intel architecture is exploited ???

Conclusions on MPI+OpenMP

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Effective when running in

- Multiprocessor system with multicore & shared memory processors
- Distributed cluster / grid of multicore systems

Issue => balancing between MIP ~ OpenMP