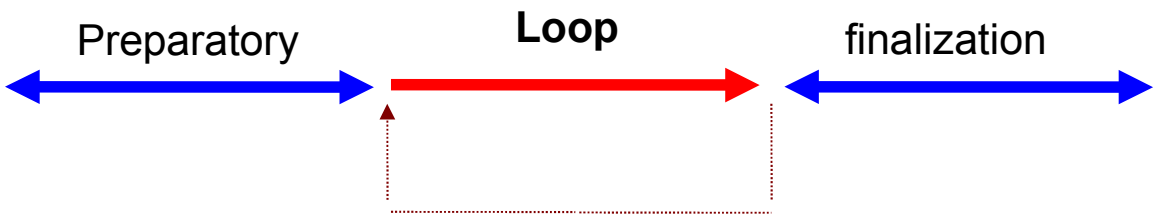
HP-SEE DISSEMINATION & TRAINING
Tirana, 17 November 2011

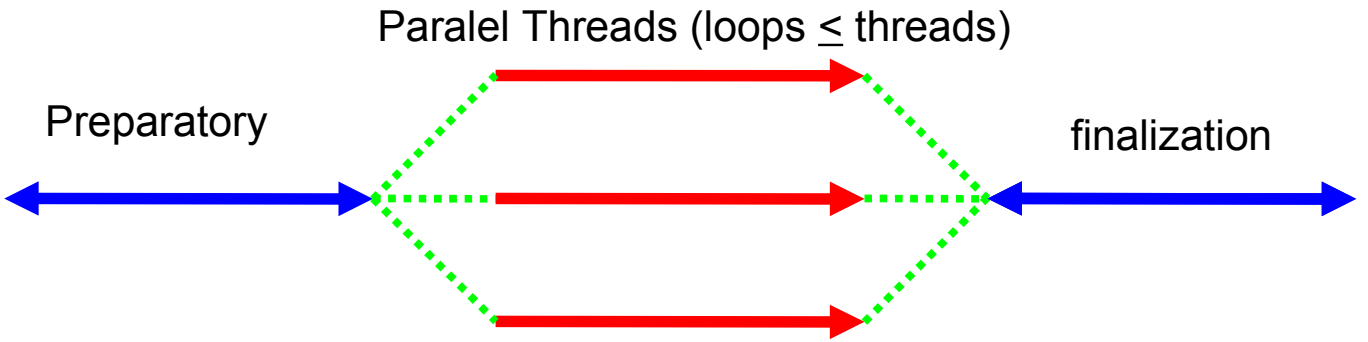
OpenMP

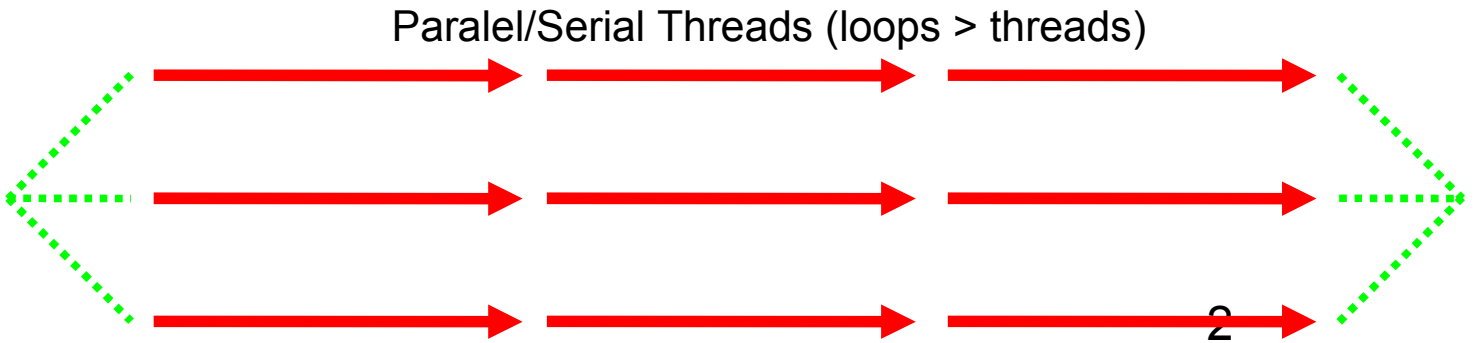
EC FP7 Project HP-SEE
http://www.hp-see.eu/

N. Frasheri
Polytechnic University of Tirana

Parallelization ~ OpenMP







Processes ~ Threads

- Process – execution of a piece of code using its own memory segments
- Threads – processes that share a part of memory segments
- Process can be split
  - In parallel processes each in its own memory segment
  - In parallel threads sharing a part of memory
    - Case of OpenMP

Download & Install

- http://openmp.org/
- Usage
  - Compiler directives
  - OpenMP library (specific to compiler)
- Downloads
  - Linux
    - Gcc => `/usr/lib/libm.a`
  - `MS Windows`
    - `http://msdn.microsoft.com/en-us/library/tt15eb9t%28v=vs.80%29.aspx`

Split a Loop in Threads

- Directive "**`pragma`**"
- Traditional loop:

```
for (i=0; i<n; i++)
        printf("hello world %d \n",i);
```

- Parallelized loop:

```
#pragma omp parallel
for (i=0; i<n; i++)
        printf("hello world %d \n",i);
```

```
// EXAMPLE
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{ int i,threid,thread,Nthread,Niterat;
   Nthread=1; Niterat=16;
   if (argc>1) sscanf (argv[1],"%d",&Nthread);
   if (argc>2) sscanf (argv[1],"%d",&Niterat);

   #pragma omp parallel for num_threads(Nthread)
     private(threid,thread)
   for (i=0; i<Niterat; i++)
   {     threid = omp_get_thread_num();
      thread = omp_get_num_threads();
      printf("hello iter %d thread %d  of %d \n", i,
        threid,thread);
   } return 0; }
```

Compilation & Execution

- Compilation of source (script `build.sh`)

```
gcc -fopenmp $1.c /usr/lib/libm.a -o $1
```

where: $1 ~ source name

Ex: ./build.sh test1

- Execution (script `run.sh`)

```
./$1 $2 $3
```

where: $1 ~ code; $2 ~ number of threads; $3 ~ iterations

Ex: ./run.sh test1 4 8

Example: Output

- test-openmp$ `./run.sh test1 4 8`

hello iter 6 thread 3 of 4

hello iter 7 thread 3 of 4

hello iter 2 thread 1 of 4

hello iter 3 thread 1 of 4

hello iter 4 thread 2 of 4

hello iter 5 thread 2 of 4

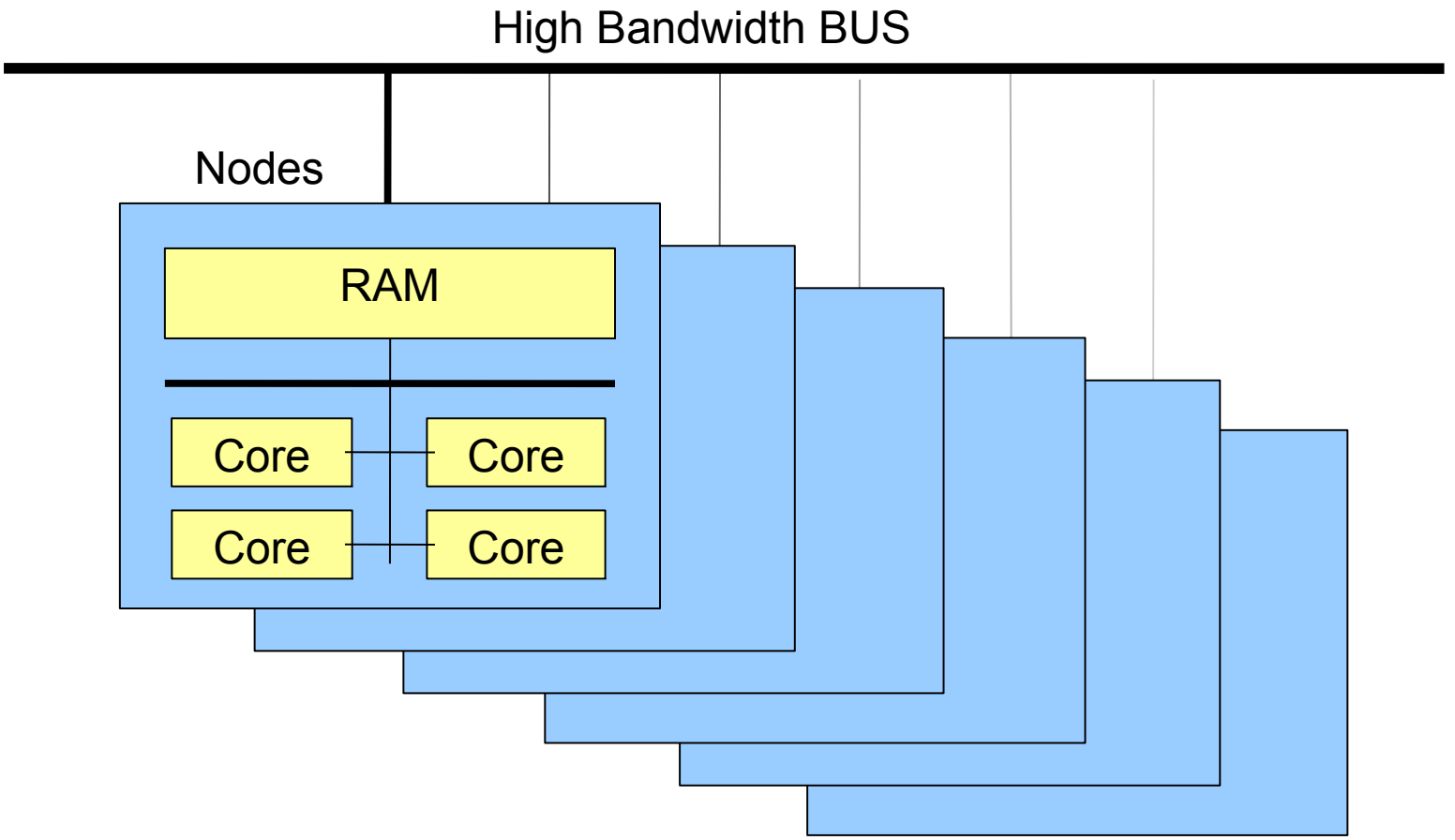
hello iter 0 thread 0 of 4

hello iter 1 thread 0 of 4

- test-openmp$

Hardware Issues

- Typical parallel hardware



Problem with OpenMP

- OpenMP uses *threads* which share the memory
- Cores within a node share the node's memory
- Cores in different nodes may not share memory
- OpenMP parallelization may depend on number of cores per node :-(
- In HP-SEE for OpenMP
  - only one SUN SGE system in Pecs has 1052 cores
  - other systems have 4–8 cores usable with OpenMP
- The alternative is MPI ...

Measuring the Time

- OpenMP routines

```
double time_start, time_stop, seconds;
time_start=omp_get_wtime();
...
time_stop=omp_get_wtime();
seconds = time_stop – time_start
```

- Resulting “walltime” depends on CPU% usage

Measuring Time and CPU%

- Shell script:

```
(/usr/bin/time ./run.sh) 1>out.txt 2>time.txt
```

  - example of `time.txt`:

```
111871.09 user 870.97 system
28:31.88 elapsed 6585% [CPU]
```

walltime: 1711.78 [omp_get_wtime]

__cores: 256

real user time: 111871/256 = 437 sec

mean CPU%: 7585/256 = 24.85%

real walltime: 1711*24.85% = 425 sec

Need for Planing

- Parallel systems are used in shared mode by many users (grouped in virtual organizations)
- Small problems may result in low performance
- Request for many cores may led to low CPU%
- Walltime may result higher than expected …
- Coordination necessary for good exploitation
- Need for sharing of resources nationally

Thank You

- Q & A

HP-SEE

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