Hybrid Parallel Programming
Part 1

Bhupender Thakur
HPC @ LSU
Overview

• Basic Hybrid model
• NUMA Challenges
  – Understanding Hardware
• Communication support for threads in MPI-2
• Hello hybrid
Why do we need a hybrid approach

• Threading reduces memory footprint of MPI processes. MPI processes on a node replicate memory.

• Threading reduces MPI communication between processes

• Reduces programming effort for certain OpenMP parallel sections of the code
Hybrid Programming: Basic Model

Serial or MPI Parallel code

OpenMP/OpenACC Parallel code

Fork threads

Join threads

Thread 0
Thread 1
Thread 2
Thread 3

MPI 0

MPI 0

6/4/14

LONI Programming Workshop 2014
Understanding NUMA architecture

Compute Node 001
2 NUMA nodes/ CPU- Sockets
8 Cores/Socket
2 HyperThds /Core

Compute Node 002
2 NUMA nodes/ CPU- Sockets
8 Cores/Socket
2 HyperThds /Core

Infiniband interconnect
NUMA: SuperMike-II

===== Processor =====

Processor name: Intel(R) Xeon(R) E5-2670 0
Packages (sockets): 2
Cores: 16
Processors (CPUs): 16
Cores per package: 8
Threads per core: 1

$ numactl --hardware | grep cpus
--------------
--------------
node 0 cpus: 0 1 2 3 4 5 6 7
node 1 cpus: 8 9 10 11 12 13 14 15
NUMA: SuperMike-II

<table>
<thead>
<tr>
<th>Processor identification</th>
<th>Placement on packages</th>
<th>Cache sharing</th>
<th>Processor composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proc</td>
<td>Thread Id.</td>
<td>Core Id.</td>
<td>Package Id.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

Processor name: Intel(R) Xeon(R) E5-2670
Packages(sockets): 2
Cores: 16
Processors(CPUs): 16
Cores per package: 8
Threads per core: 1
NUMA : Quiz 1

Quiz: Can you identify the node structure here?

$ numactl --hardware|grep cpus

node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 1 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
### Processor Composition

<table>
<thead>
<tr>
<th>Proc</th>
<th>Thread Id.</th>
<th>Core Id.</th>
<th>Package Id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

### Processor Name
- **Intel(R) Xeon(R) X5670**

### Packages (Sockets)
- **2**

### Cores
- **12**

### Processors (CPUs)
- **24**

### Cores per Package
- **6**

### Threads per Core
- **2**

### Command

```
$ numactl --hardware|grep cpus
```

node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 1 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
### Placement on packages

<table>
<thead>
<tr>
<th>Package Id.</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0,12)(1,13)(2,14)(3,15)(4,16)(5,17)</td>
</tr>
<tr>
<td>1</td>
<td>(6,18)(7,19)(8,20)(9,21)(10,22)(11,23)</td>
</tr>
</tbody>
</table>

### Cache sharing

<table>
<thead>
<tr>
<th>Cache</th>
<th>Size</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3</td>
<td>12 MB</td>
<td>(0,1,2,3,4,5,12,13,14,15,16,17)(6,7,8,9,10,11,18,19,20,21,22,23)</td>
</tr>
</tbody>
</table>
NUMA : Quiz 2

Quiz: Can you identify the node structure here?

$ numactl --hardware | grep cpus

node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23
### NUMA : Quiz 2

#### Processor

<table>
<thead>
<tr>
<th>Processor Thread Id.</th>
<th>Core Id. Package Id.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>5</td>
</tr>
</tbody>
</table>

**Processor name**: Intel(R) Xeon(R) E5-2640 0

**Packages (sockets)**: 2

**Cores**: 12

**Processors (CPUs)**: 24

**Cores per package**: 6

**Threads per core**: 2

```bash
$ numactl --hardware | grep cpus
node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23
```
NUMA : Quiz 3

Quiz: Can you identify the node structure here?

```bash
$numactl --hardware | grep cpus
```

```
mike441
```

```
node 0 cpus:  0  4  8 12 16 20 24 28 32 36
node 1 cpus: 1  5  9 13 17 21 25 29 33 37
node 2 cpus: 2  6 10 14 18 22 26 30 34 38
node 3 cpus:  3  7 11 15 19 23 27 31 35 39
```
NUMA : Quiz 3

===== Processor ======

Processor name : Intel(R) Xeon(R) E7-4870
Packages(sockets) : 4
Cores : 40
Processors(CPUs) : 40
Cores per package : 10
Threads per core : 1

$numactl --hardware | grep cpus

--------
mike441
--------
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 1 cpus: 1 5 9 13 17 21 25 29 33 37
node 2 cpus: 2 6 10 14 18 22 26 30 34 38
node 3 cpus: 3 7 11 15 19 23 27 31 35 39
Design decisions

• How many MPI processes per node, per socket?
  How many OpenMP threads per MPI process
Hybrid 0: Basic Model
Hybrid 1: Basic Model

Process memory shared by threads

Memory not shared by processes

MPI Communication
Hybrid-2: Task Parallelism

MPI 0

Thread 0

Thread 1

MPI 0

MPI 1

Thread 0

Thread 1

Thread 2

Thread 4

Thread 5

Thread 6

MPI 1
Communication support in MPI-2

MPI-2 Init offers thread support query

MPI_INIT_THREAD( request_thd_sup, provide_thd_sup )

request_thd_sup: Requested level of communication from a threaded region
provide_thd_sup: Actual provided level of communication from a threaded region

Four levels of thread support on MPI-2

0 : MPI_THREAD_SINGLE: No threading for MPI processes
1 : MPI_THREAD_FUNNELED: Threading with Master Thread allowed MPI calls
2 : MPI_THREAD_SERIALIZED: Multiple threads make MPI calls, but serially
3 : MPI_THREAD_MULTIPLE: Any thread may make MPI calls at any time
MPI_THREAD_FUNNELED

MPI communication allowed for Master Thread
MPI_THREAD_{SERIALIZED,MULTIPLE}

MPI communication allowed between all threads
Thread support in MPI-2

OpenMPI

ompi_info will also show you the level of support

```fortran
program thread_support
  use mpi
  implicit none

  integer :: mpierr
  integer :: thd_sup_req, thd_sup_pvd

  thd_sup_req = MPI_THREAD_MULTIPLE

  call MPI_INIT_THREAD( thd_sup_req, thd_sup_pvd, mpierr)
  call MPI_FINALIZE(mpierr)

  print *, "Requested", thd_sup_req
  print *, "Provided ", thd_sup_pvd

end program
```

program thread_support

use mpi
implicit none

integer :: mpierr
integer :: thd_sup_req, thd_sup_pvd

thd_sup_req=MPI_THREAD_MULTIPLE

call MPI_INIT_THREAD( thd_sup_req, thd_sup_pvd, mpierr)
call MPI_FINALIZE(mpierr)

print *, "Requested", thd_sup_req
print *, "Provided ", thd_sup_pvd

end program

#include <mpi.h>
#include <stdio.h>

int main(int argc, const char* argv[])
{
  int provided;
  int requested=MPI_THREAD_MULTIPLE;

  MPI_Init_thread(NULL, NULL, requested, &provided);
  printf("MPI_Init_thread Requested = %d\n", requested);
  printf("MPI_Init_thread Provided = %d\n", provided);
  MPI_Finalize();

  return 0;
}
MPI Communication within Threads
Hello: Hybrid parallel program

Compiling Hybrid Hello world:

mpif90 –openmp hello_hybrid.f90
Hello: Hybrid parallel program

Compiling Hybrid Hello world:

```bash
mpif90 –openmp hello_hybrid.f90
```
Analyzing a Hybrid parallel program

Running a hybrid (mpi +openmp) process

Global ID
MPI Rank
OpenMP ID
Analyzing a Hybrid parallel program

$ mpirun -n 32 -hostfile $PBS_NODEFILE --bind-to-core -report-bindings ./a.out

NOTE: The host file $PBS_NODEFILE has 16 entries for each node
Analyzing a Hybrid parallel program

$ mpirun -n 32 -hostfile $PBS_NODEFILE --bind-to-core -report-bindings ./a.out

Node 1
Socket 1
  -
  -
  -
  -
  -
  -
  -

Node 1
Socket 2
  -
  -
  -
  -
  -
  -
  -

Jump to another socket after using all cores
Consecutive ranks on consecutive cores
Jump to another node
Analyzing a Hybrid parallel program

To spawn mpi processes correctly you can either manipulate the hostfile or use other runtime options if available.
Analyzing a Hybrid parallel program

$ mpirun -n 32 -hostfile hosts --bind-to-core -report-bindings .a.out

NOTE: The host file hosts has only 2 unique entries
Analyzing a Hybrid parallel program

2 Nodes, 2 MPI processes per node, 8 OMP Threads per process

$ mpirun -n 4 -hostfile $PBS_NODEFILE --bysocket --bind-to-socket -report-bindings -display-map ./a.out

[mike253:42506] MCW rank 0 bound to socket 0[core 0-7]: [B B B B B B B] [ . . . . . . ]
[mike253:42506] MCW rank 1 bound to socket 1[core 0-7]: [ . . . . . . . ] [ B B B B B B B]
[mike253:42506] MCW rank 2 bound to socket 0[core 0-7]: [B B B B B B B] [ . . . . . . ]
[mike253:42506] MCW rank 3 bound to socket 1[core 0-7]: [ . . . . . . . ] [ B B B B B B B]

NOTE: The host file $PBS_NODEFILE has 16 entries for each node
Analyzing a Hybrid parallel program

2 Nodes, 2 MPI processes per node, 8 OMP Threads per process

$ mpirun -n 4 -hostfile hosts --bysocket --bind-to-socket -report-bindings -display-map ./a.out
[mike015:62126] MCW rank 0 bound to socket 0[core 0-7]: [B B B B B B B][. . . . . . .]
[mike015:62126] MCW rank 2 bound to socket 1[core 0-7]: [. . . . . . .][B B B B B B B]
[mike017:34768] MCW rank 1 bound to socket 0[core 0-7]: [B B B B B B B][. . . . . . .]
[mike017:34768] MCW rank 3 bound to socket 1[core 0-7]: [. . . . . . .][B B B B B B B]

Node 1           Node 1         Node 2           Node 2
Socket 1          Socket 2      Socket 1          Socket 2

NOTE: the hostfile here has unique names
Analyzing a Hybrid parallel program

2 Nodes, 4 MPI processes per node, 4 OMP Threads per process

$ mpirun -n 8 -hostfile hosts --bysocket --bind-to-socket -report-bindings -display-map ./a.out

[mike015:62526] MCW rank 6 bound to socket 1[core 0-7]: [ . . . . . . ] [ B B B B B B B ]
[mike015:62526] MCW rank 0 bound to socket 0[core 0-7]: [ B B B B B B B ] [ . . . . . . ]
[mike015:62526] MCW rank 2 bound to socket 1[core 0-7]: [ . . . . . . ] [ B B B B B B B ]
[mike015:62526] MCW rank 4 bound to socket 0[core 0-7]: [ B B B B B B B ] [ . . . . . . ]
[mike017:35161] MCW rank 1 bound to socket 0[core 0-7]: [ B B B B B B B ] [ . . . . . . ]
[mike017:35161] MCW rank 3 bound to socket 1[core 0-7]: [ . . . . . . ] [ B B B B B B B ]
[mike017:35161] MCW rank 5 bound to socket 0[core 0-7]: [ B B B B B B B ] [ . . . . . . ]
[mike017:35161] MCW rank 7 bound to socket 1[core 0-7]: [ . . . . . . ] [ B B B B B B B ]

NOTE: The hostfile here has unique names
Analyzing a Hybrid parallel program

2 Nodes, 4 MPI processes per node, 4 OMP Threads per process

$ mpirun -n 8 -hostfile hosts --bysocket --bind-to-socket -report-bindings -display-map ./a.out

[Mike015:62526] MCW rank 6 bound to socket 1[core 0-7]: [. . . . . . ] [B B B B B B B]
[Mike015:62526] MCW rank 0 bound to socket 0[core 0-7]: [B B B B B B B][ . . . . . . ]
[Mike015:62526] MCW rank 2 bound to socket 1[core 0-7]: [. . . . . . ] [B B B B B B B]
[Mike015:62526] MCW rank 4 bound to socket 0[core 0-7]: [B B B B B B B][ . . . . . . ]
[Mike017:35161] MCW rank 1 bound to socket 0[core 0-7]: [B B B B B B ] [. . . . . . ]
[Mike017:35161] MCW rank 3 bound to socket 1[core 0-7]: [. . . . . . ] [B B B B B B B]
[Mike017:35161] MCW rank 5 bound to socket 0[core 0-7]: [B B B B B B B][ . . . . . . ]
[Mike017:35161] MCW rank 7 bound to socket 1[core 0-7]: [. . . . . . ] [B B B B B B B]

NOTE: The hostfile here has unique names
Exercise

• Run the hello hybrid program on two nodes using:
  – 1 mpi process per node, 16 openmp threads per process
  – 2 mpi processes per node, 8 openmp threads per process
  – 4 mpi processes per node, 4 openmp threads per process
  – 8 mpi processes per node, 2 openmp threads per process

• Run the executable “shelob.exe.a” as in previous step:
  – What is difference between total time and cpu time?
    Hint: Run with “time mpirun ...“ to see real time
  – Calculate the total cpu utilization as a percentage of cpu time
    vs elapsed time from the output.
  – Find the most efficient combination of mpi and openmp threads