OpenMP
Thread Affinity

TACC OpenMP Team

milfeld/lars/agomez@tacc.utexas.edu

THE UNIVERSITY OF TEXAS AT AUSTIN
TEXAS ADVANCED COMPUTING CENTER

Learning Objective

-- Affinity --

- Understand why you should care about affinity
- Basic concepts
  - OMP_PLACES
  - OMP_PROC_BIND
- Standard affinity solutions for
  - Simple OpenMP
  - Nested OpenMP
  - Hybrid OpenMP (with MPI)
- Thread and Memory affinity

OpenMP Pre 3.0

- Affinity not controlled by OpenMP
- System default applied
- Other tools used

RaW = Read after Write, etc.

Why do we care about Thread Affinity?

Why do we want to control where threads are executed?

- Gain some (a little bit) of performance
- Prevent loosing potentially a lot of performance

RaW = Read after Write, etc.

Why do we care about Thread Affinity?

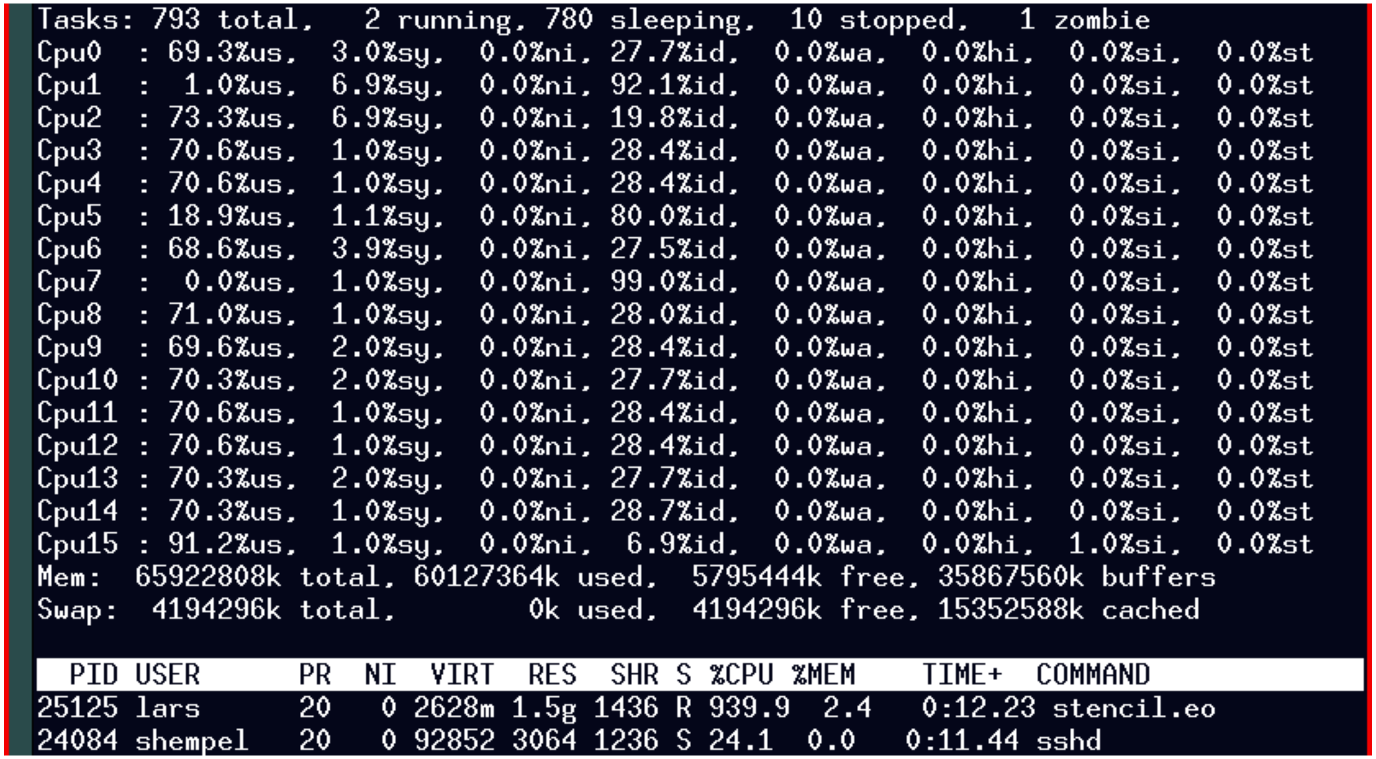
Why do we want to control where threads are executed?

- Gain some (a little bit) of performance
- Prevent loosing potentially a lot of performance

Good News

- It is very easy to do
- Basic setups will cover 95% of use cases

RaW = Read after Write, etc.

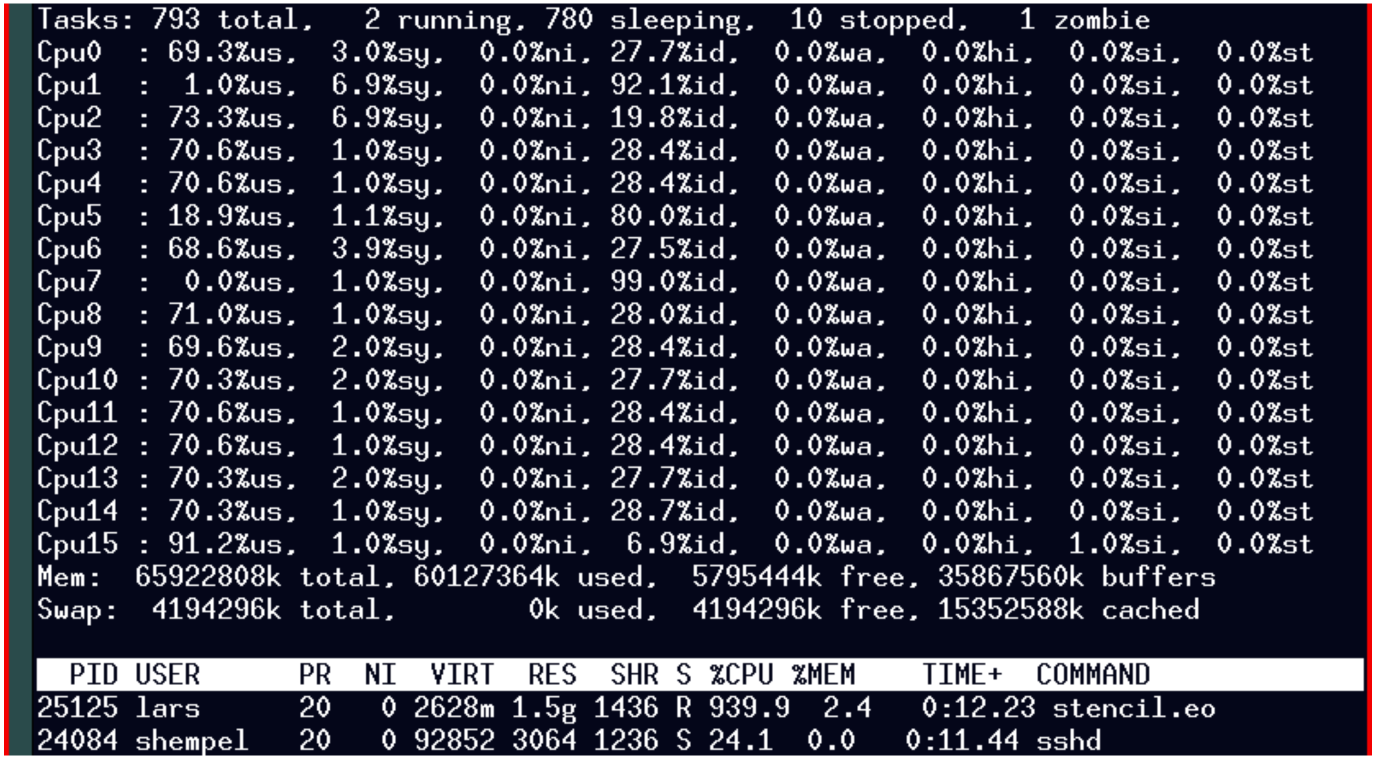


```
Tasks: 793 total,   2 running, 780 sleeping,  10 stopped,   1 zombie
Cpu0  : 69.3%us,  3.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  1.0%us,  6.9%sy,  0.0%ni, 92.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 73.3%us,  6.9%sy,  0.0%ni, 19.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 18.9%us,  1.1%sy,  0.0%ni, 80.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 68.6%us,  3.9%sy,  0.0%ni, 27.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 71.0%us,  1.0%sy,  0.0%ni, 28.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 69.6%us,  2.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 70.3%us,  1.0%sy,  0.0%ni, 28.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 91.2%us,  1.0%sy,  0.0%ni,  6.9%id,  0.0%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:  65922808k total, 60127364k used,  5795444k free, 35867560k buffers
Swap:  4194296k total,        0k used,  4194296k free, 15352588k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25125 lars      20   0 2628m 1.5g 1436 R 939.9  2.4   0:12.23 stencil.eo
24084 shempel   20   0 92852 3064 1236 S 24.1  0.0   0:11.44 sshd
```

Example from Stampede

- 16 cores → OMP_NUM_THREADS set to 16
- No pinning; uses default from the run-time
- Some cores are idling (Cpu1 and Cpu7)
- Some cores apparently oversubscribed

How much performance would you loose?



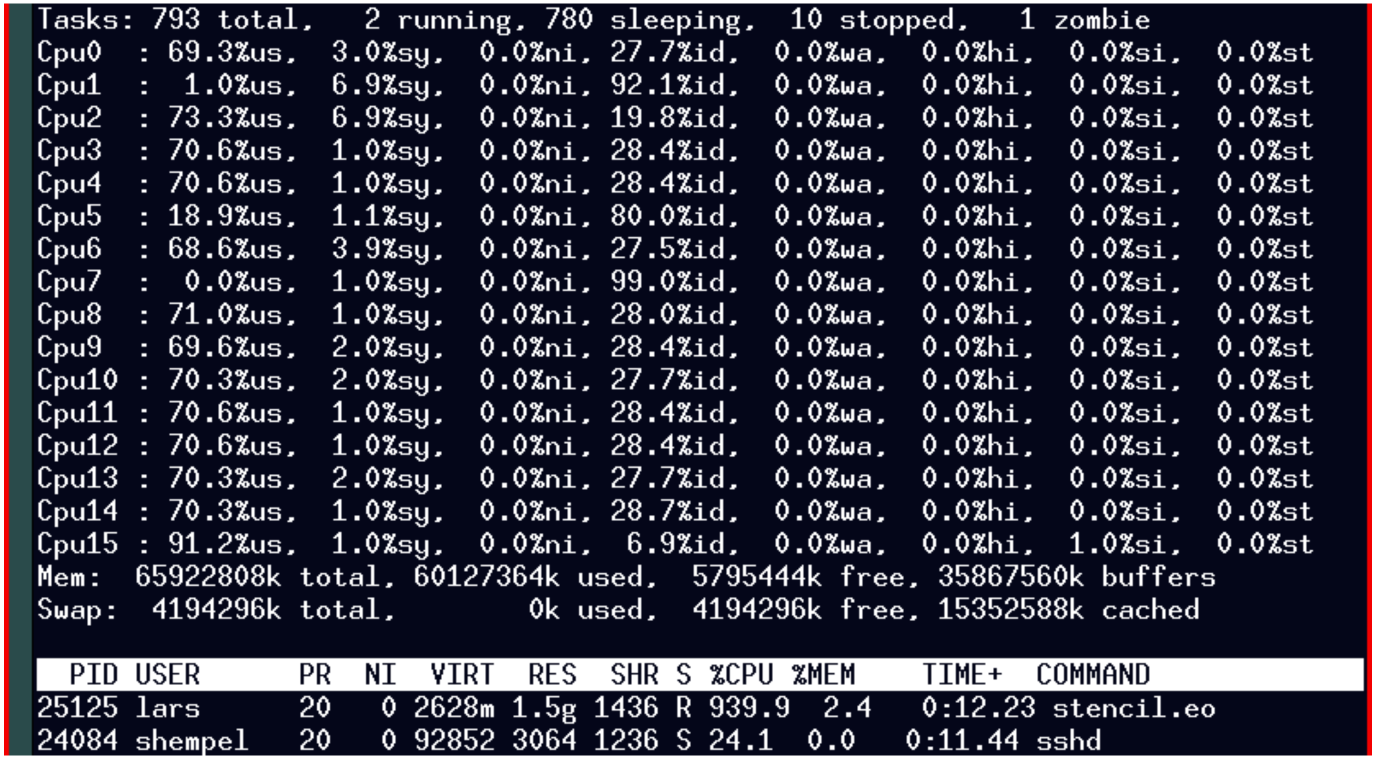
```
Tasks: 793 total,   2 running, 780 sleeping,  10 stopped,   1 zombie
Cpu0  : 69.3%us,  3.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  1.0%us,  6.9%sy,  0.0%ni, 92.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 73.3%us,  6.9%sy,  0.0%ni, 19.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 18.9%us,  1.1%sy,  0.0%ni, 80.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 68.6%us,  3.9%sy,  0.0%ni, 27.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 71.0%us,  1.0%sy,  0.0%ni, 28.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 69.6%us,  2.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 70.3%us,  1.0%sy,  0.0%ni, 28.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 91.2%us,  1.0%sy,  0.0%ni,  6.9%id,  0.0%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:  65922808k total, 60127364k used,  5795444k free, 35867560k buffers
Swap:  4194296k total,        0k used,  4194296k free, 15352588k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25125 lars      20   0 2628m 1.5g 1436 R 939.9  2.4   0:12.23 stencil.eo
24084 shempel   20   0 92852 3064 1236 S 24.1  0.0   0:11.44 sshd
```

Example from Stampede

- 16 cores → OMP_NUM_THREADS set to 16
- No pinning; uses default from the run-time
- Some cores are idling (Cpu1 and Cpu7)
- Some cores apparently oversubscribed

How much performance would you loose?

- Dynamic scheduling: 2 out of 16 = 12.5%



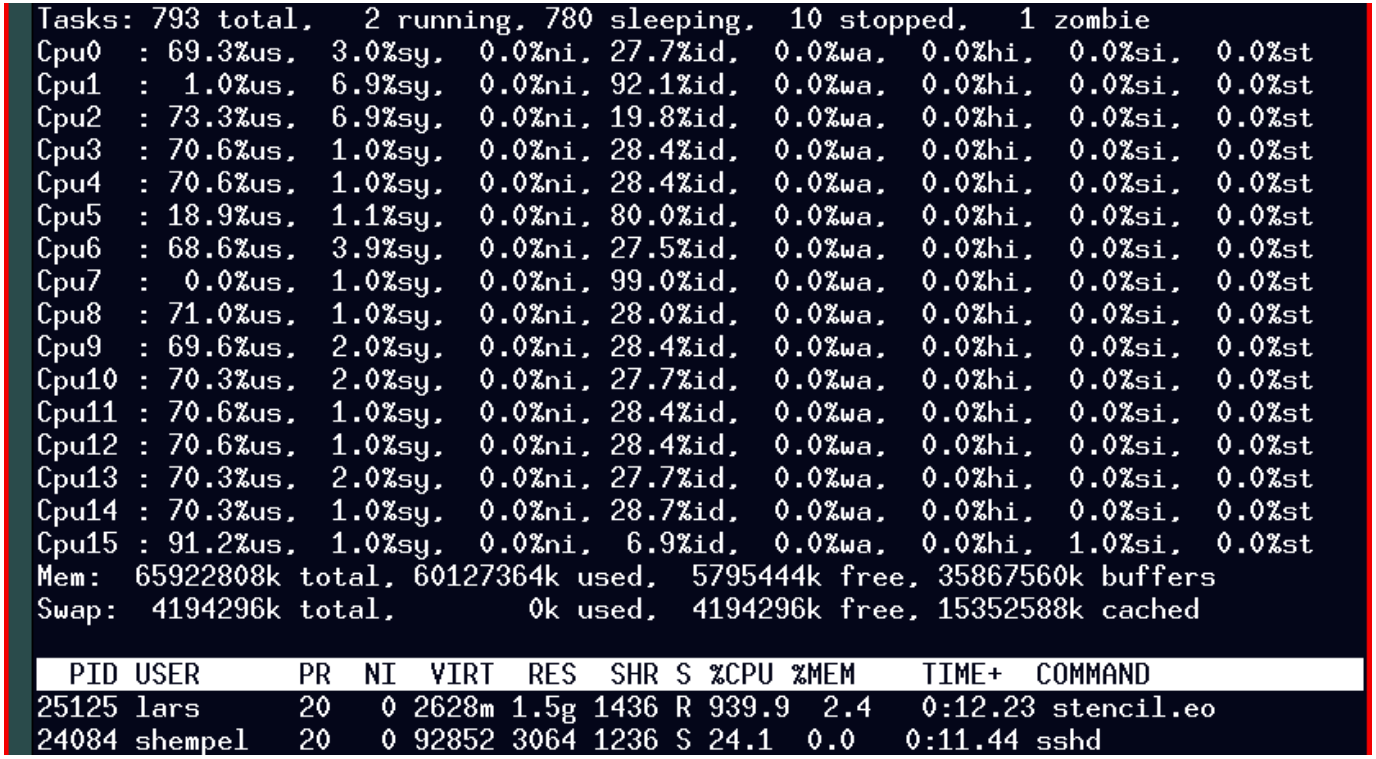
```
Tasks: 793 total,   2 running, 780 sleeping,  10 stopped,   1 zombie
Cpu0  : 69.3%us,  3.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  1.0%us,  6.9%sy,  0.0%ni, 92.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 73.3%us,  6.9%sy,  0.0%ni, 19.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 18.9%us,  1.1%sy,  0.0%ni, 80.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 68.6%us,  3.9%sy,  0.0%ni, 27.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 71.0%us,  1.0%sy,  0.0%ni, 28.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 69.6%us,  2.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 70.3%us,  1.0%sy,  0.0%ni, 28.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 91.2%us,  1.0%sy,  0.0%ni,  6.9%id,  0.0%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:   65922808k total, 60127364k used,  5795444k free, 35867560k buffers
Swap:   4194296k total,        0k used,  4194296k free, 15352588k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25125 lars      20   0 2628m 1.5g 1436 R 939.9  2.4   0:12.23 stencil.eo
24084 shempel   20   0 92852 3064 1236 S 24.1  0.0   0:11.44 sshd
```

Example from Stampede

- 16 cores → OMP_NUM_THREADS set to 16
- No pinning; uses default from the run-time
- Some cores are idling (Cpu1 and Cpu7)
- Some cores apparently oversubscribed

How much performance would you loose?

- Dynamic scheduling: 2 out of 16 = 12.5%
- Static scheduling: 50% (waiting for the oversubscribed cores to finish)



```
Tasks: 793 total,   2 running, 780 sleeping,  10 stopped,   1 zombie
Cpu0  : 69.3%us,  3.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  1.0%us,  6.9%sy,  0.0%ni, 92.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 73.3%us,  6.9%sy,  0.0%ni, 19.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 18.9%us,  1.1%sy,  0.0%ni, 80.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 68.6%us,  3.9%sy,  0.0%ni, 27.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 71.0%us,  1.0%sy,  0.0%ni, 28.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 69.6%us,  2.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 70.6%us,  1.0%sy,  0.0%ni, 28.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 70.3%us,  2.0%sy,  0.0%ni, 27.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 70.3%us,  1.0%sy,  0.0%ni, 28.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 91.2%us,  1.0%sy,  0.0%ni,  6.9%id,  0.0%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:  65922808k total, 60127364k used,  5795444k free, 35867560k buffers
Swap:  4194296k total,        0k used,  4194296k free, 15352588k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25125 lars      20   0 2628m 1.5g 1436 R 939.9  2.4   0:12.23 stencil.eo
24084 shempel   20   0 92852 3064 1236 S 24.1  0.0  0:11.44 sshd
```

Example from Stampede

- 16 cores → OMP_NUM_THREADS set to 16
- No pinning; uses default from the run-time
- Some cores are idling (Cpu1 and Cpu7)
- Some cores apparently oversubscribed

How much performance would you loose?

- Dynamic scheduling: 2 out of 16 = 12.5%
- Static scheduling: 50% (waiting for the oversubscribed cores to finish)
- Implications for MPI codes even more severe

How to use ‘**top**’ to look at core utilization

- Open a terminal
- Type **top**
- Press the 1 key (#1 key top left corner)
- **top** may complain about the window size
- You may have to increase the window so that there are more lines in the window than cores
- You may increase the update frequency. Type ‘s’ and 1, which will change the update interval to 1 second

We will use 2 windows in the lab

One to run the experiment

The other to inspect core utilization

Terminology

- Compute node: basic building block of a cluster
  - A single desktop is also a node
- Socket: each node may have 1, 2, or 4 sockets
  - Sockets are often called CPUs (confusion!)
- Cores: each socket may have any (small) number of cores; 2, 4, 8, 12, 16, 20
  - Cores are sometimes called CPUs (more confusiion)
- Each core may have 1 or 2 (Hyperthreading) hardware threads (hw threads)
  - Some architectures have 4 hw threads
  - Lonestar has Hyperthreading enabled
- OpenMP threads use the hardware threads

Typical Thread Utilization

All cores are being used

- Without Hyperthreading
  - 1 OpenMP thread per hardware thread
  - 1 OpenMP thread per core
- With Hyperthreading
  - 1 OpenMP thread per hardware thread
  - 2 OpenMP threads per core

Or

  - 1 OpenMP thread per core
  - Every other hardware thread is not used = Hyperthreading enabled but not ustilized

Typical Thread Utilization

Exceptions

- Oversubscribing one core
  - 16 cores and 17 threads
  - Extra thread doing different work
    - Scheduling, MPI communication, or I/O, etc.
- Leaving cores idle
  - Memory bandwidth limited code
  - Increasing shared L3 cache per thread
- Experiment to find best performance
  - One thread per core
  - One thread per hardware thread
  - 'Odd' configurations

How to pin OpenMP threads to a resource

`OMP_PLACES` and **`OMP_PROC_BIND`**

- Step 1: Defining OpenMP Places on a node
  - Environment variable: **`OMP_PLACES`**
  - Default: 1 node is one big place
  - Divide node in multiple places
  - Example: 2 sockets, each a place (2 places total)
  - or: each core a separate place (16 places on Stampede)
  - Environment variable: **`OMP_PLACES`**
- Step 2: Pin OpenMP threads (procs) to the Places
  - Environment variable: **`OMP_PROC_BIND`**
  - Pinning can be 'spread out' among places
  - Pinning can be 'close together'
- Places and pinning are controlled by environment variables

Defining OpenMP Places: **OMP_PLACES**

Example without Hyperthreading

- Each core is 1 hardware thread
- The cores are numbered
- Stampede: 16 cores, numbered from 0 to 15
- Example for 2 places on a node
  - `export OMP_PLACES="{0,1,2,3,4,5,6,7}{8,9,10,11,12,13,14,15}"`
  - First place: Cores 0-7 (socket 0)
  - Second place: Cores 8-15 (socket 1)
- Example for 16 places
  - `Export OMP_PLACES="{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}"`
- Syntax shortcuts available (not shown, buggy at times)

Pinning Policy: **OMP_PROC_BIND**

Example without Hyperthreading

- Chose between ‘**spread**’ and ‘**close**’
  - **Spread**: Choose places far away from each other
  - **Close**: Choose places close together
  - A third possibility is ‘master’

Let’s look at some examples

- Different combinations of
  - **OMP_NUM_THREADS**
  - **OMP_PLACES**
  - **OMP_PROC_BIND**

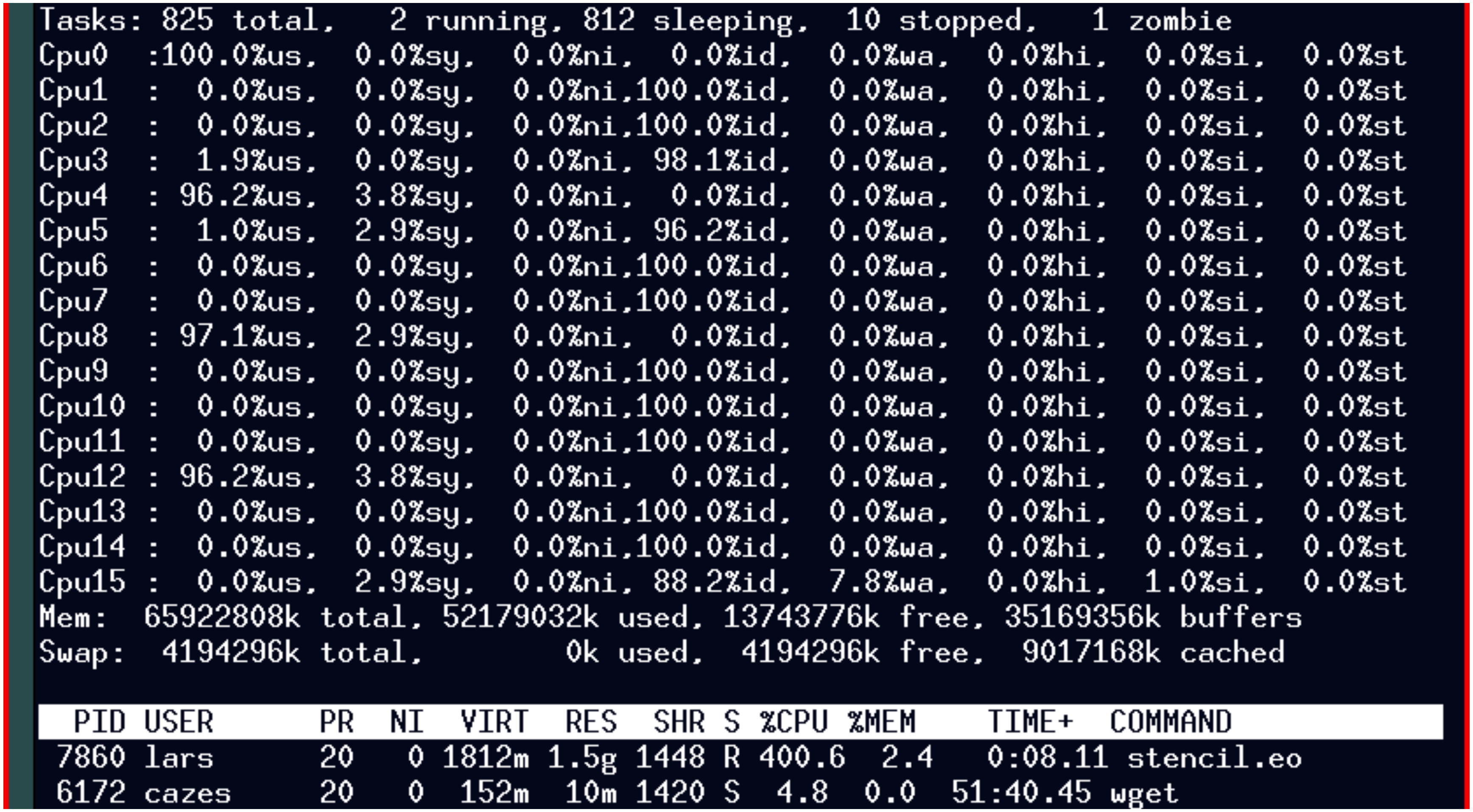
Example 1 (No Hyperthreading)

Stampede: 16 cores, 16 Places

- export OMP_NUM_THREADS=4
- export OMP_PLACES=“{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}”
- export OMP_PROC_BIND=spread

- Threads may be bound to cores 0, 4, 8, and 12 or 1, 5, 9, and 13, etc.
  - ‘spread’ = maximum distance is 4 (16 places, 4 threads)

Example 1



```
Tasks: 825 total,   2 running, 812 sleeping,  10 stopped,   1 zombie
Cpu0  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  1.9%us,  0.0%sy,  0.0%ni, 98.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 96.2%us,  3.8%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  1.0%us,  2.9%sy,  0.0%ni, 96.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 97.1%us,  2.9%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 96.2%us,  3.8%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.0%us,  2.9%sy,  0.0%ni, 88.2%id,  7.8%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:  65922808k total, 52179032k used, 13743776k free, 35169356k buffers
Swap:  4194296k total,        0k used,  4194296k free,  9017168k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7860 lars      20   0 1812m 1.5g 1448 R 400.6  2.4   0:08.11 stencil.eo
 6172 cazes     20   0  152m  10m 1420 S  4.8  0.0  51:40.45 wget
```

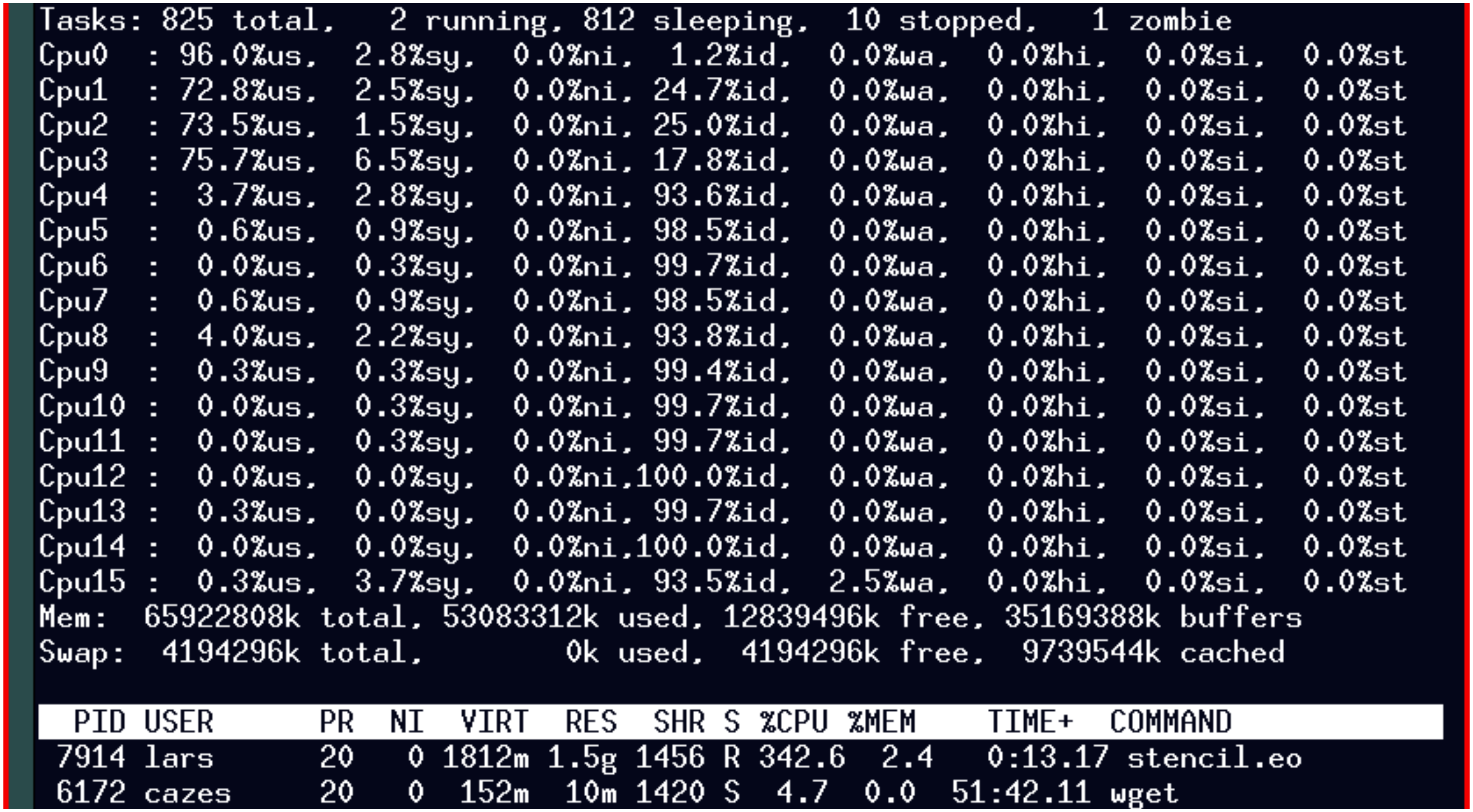
Example 2 (No Hyperthreading)

Stampede: 16 cores, 16 Places

- export OMP_NUM_THREADS=4
- export OMP_PLACES="{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}"
- export OMP_PROC_BIND=close

- Threads may be bound to cores 0, 1, 2, and 3 or 1, 2, 3, 4, etc.
  - 'close' = minimum distance is 1

Example 2



```
Tasks: 825 total,   2 running, 812 sleeping,  10 stopped,   1 zombie
Cpu0  : 96.0%us,  2.8%sy,  0.0%ni,  1.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 72.8%us,  2.5%sy,  0.0%ni, 24.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 73.5%us,  1.5%sy,  0.0%ni, 25.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 75.7%us,  6.5%sy,  0.0%ni, 17.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  3.7%us,  2.8%sy,  0.0%ni, 93.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.6%us,  0.9%sy,  0.0%ni, 98.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.6%us,  0.9%sy,  0.0%ni, 98.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  :  4.0%us,  2.2%sy,  0.0%ni, 93.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  :  0.3%us,  0.3%sy,  0.0%ni, 99.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.3%us,  0.0%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.3%us,  3.7%sy,  0.0%ni, 93.5%id,  2.5%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65922808k total, 53083312k used, 12839496k free, 35169388k buffers
Swap:  4194296k total,        0k used,  4194296k free,  9739544k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7914 lars      20   0 1812m 1.5g 1456 R 342.6  2.4   0:13.17 stencil.eo
 6172 cazes     20   0  152m  10m 1420 S  4.7  0.0  51:42.11 wget
```

Example 3 (No Hyperthreading)

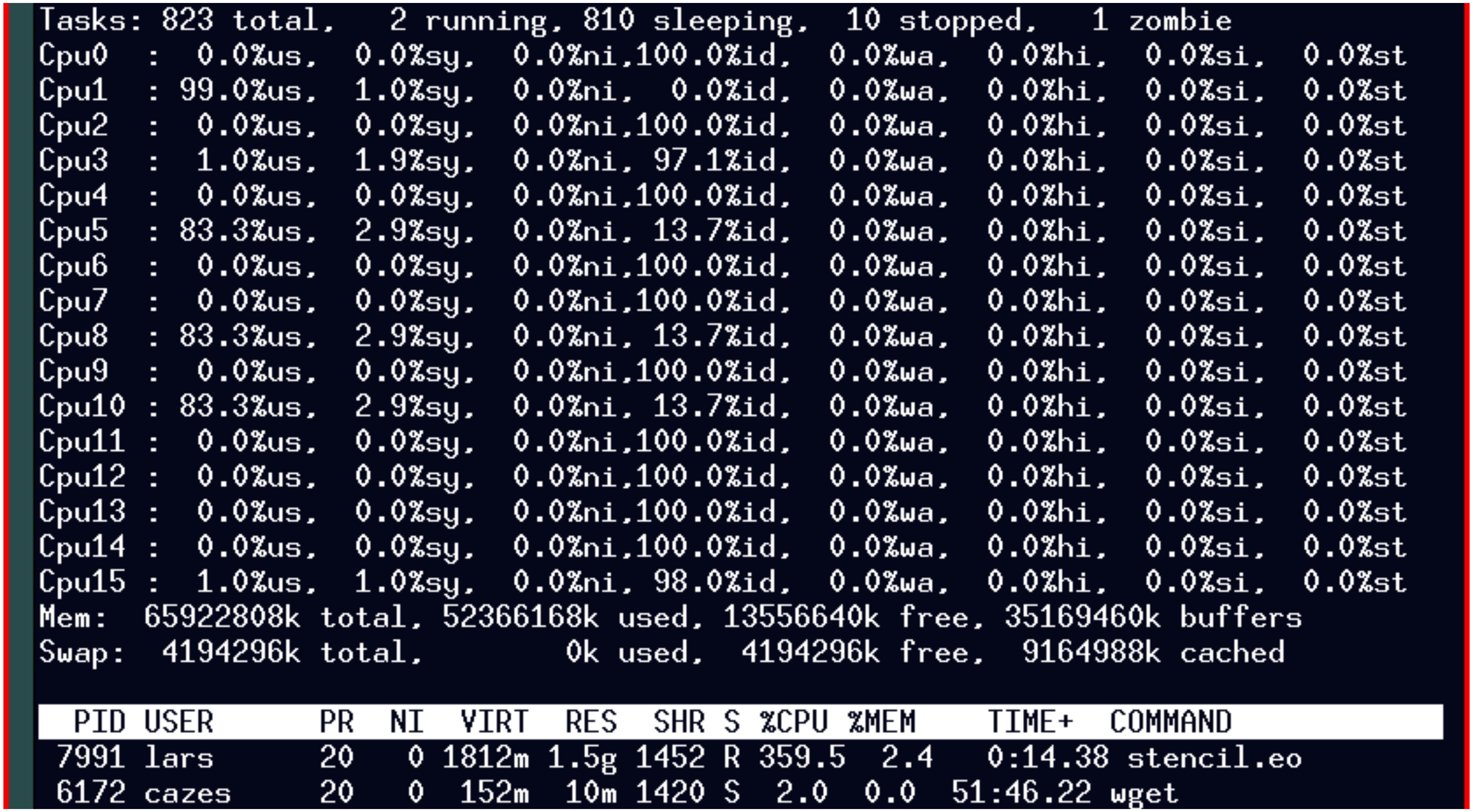
Stampede: 16 cores, 2 Places

- export OMP_NUM_THREADS=4
- export OMP_PLACES="{0,1,2,3,4,5,6,7}{8,9,10,11,12,13,14,15}"
- export OMP_PROC_BIND=spread

- 2 threads will be bound to the first place (threads 0 and 2)
- 2 threads will be bound to the second place (threads 1 and 3)

- Within a place threads can move around, so
  - Threads 0 and 2 may be executed by any core 0-7
  - Threads 1 and 3 may be executed on any core 8-15

Example 3



```
Tasks: 823 total,   2 running, 810 sleeping,  10 stopped,   1 zombie
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 99.0%us,  1.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  1.0%us,  1.9%sy,  0.0%ni, 97.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 83.3%us,  2.9%sy,  0.0%ni, 13.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 83.3%us,  2.9%sy,  0.0%ni, 13.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 83.3%us,  2.9%sy,  0.0%ni, 13.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  1.0%us,  1.0%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65922808k total, 52366168k used, 13556640k free, 35169460k buffers
Swap:  4194296k total,        0k used,  4194296k free,  9164988k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7991 lars      20   0 1812m 1.5g 1452 R 359.5  2.4   0:14.38 stencil.eo
 6172 cazes     20   0  152m  10m 1420 S  2.0  0.0  51:46.22 wget
```

Example 4 (No Hyperthreading)

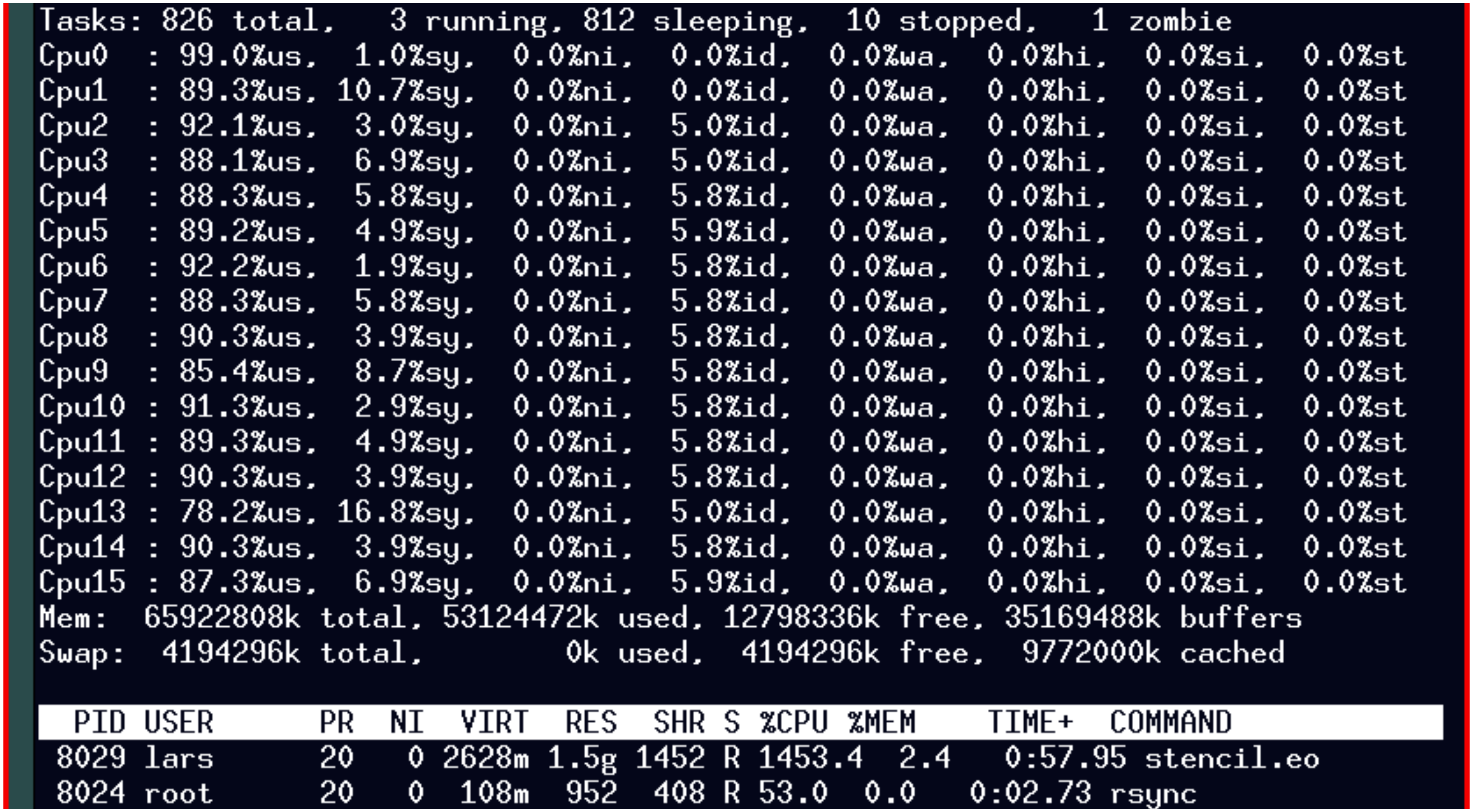
Stampede: 16 cores, 16 Places

- export OMP_NUM_THREADS=16
- export OMP_PLACES="{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}"
- export OMP_PROC_BIND=spread

- Threads may be bound to cores 0, 4, 8, and 12 or 1, 5, 9, and 13, etc.
  - 'spread' = maximum distance is 1 (16 places, 16 threads)
  - So each thread is bound to a different place

- This is a standard situation: covering 95% of the applications
  - 1 thread per core, no 'overlap'
  - Threads cannot move around

Example 4



```
Tasks: 826 total,   3 running, 812 sleeping,  10 stopped,   1 zombie
Cpu0  : 99.0%us,  1.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 89.3%us, 10.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 92.1%us,  3.0%sy,  0.0%ni,  5.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 88.1%us,  6.9%sy,  0.0%ni,  5.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 88.3%us,  5.8%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  : 89.2%us,  4.9%sy,  0.0%ni,  5.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 92.2%us,  1.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  : 88.3%us,  5.8%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 90.3%us,  3.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 85.4%us,  8.7%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 91.3%us,  2.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 89.3%us,  4.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 90.3%us,  3.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 : 78.2%us, 16.8%sy,  0.0%ni,  5.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 90.3%us,  3.9%sy,  0.0%ni,  5.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 : 87.3%us,  6.9%sy,  0.0%ni,  5.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  65922808k total, 53124472k used, 12798336k free, 35169488k buffers
Swap:  4194296k total,        0k used,  4194296k free,  9772000k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8029 lars      20   0 2628m 1.5g 1452 R 1453.4  2.4   0:57.95 stencil.eo
 8024 root      20   0  108m  952  408 R 53.0  0.0   0:02.73 rsync
```

Standard Setup (95% of cases)

Stampede: 16 cores, 16 places, 16 threads

- export OMP_NUM_THREADS=16
- export OMP_PLACES=cores
- export OMP_PROC_BIND=spread
- Every OpenMP thread is pinned to a core/hardware thread

Lonestar: 24 cores, 24 places, 24 or 48 threads

- export OMP_NUM_THREADS=24
- export OMP_NUM_THREADS=48
- export OMP_PLACES=cores (or threads)
- export OMP_PROC_BIND=spread
- One or two OpenMP thread(s) pinned to a core with 2 hardware threads

Summary OpenMP code

No Hyperthreading

- export OMP_NUM_THREADS=<number of cores>
- export OMP_PLACES=cores (shortcut for a list of cores)
- export OMP_PROC_BIND=spread

With Hyperthreading

- export OMP_NUM_THREADS=<1 or 2 x number of cores>
- export OMP_PLACES=cores
- export OMP_PROC_BIND=spread

- Each thread has its own core
- With Hyperthreading 2 threads are sharing a core

- Increase/reduce number of OpenMP threads if you want to oversubscribe cores or want to leave cores idle

Nested OpenMP (1)

Example: 2-level nested OpenMP

- Upper level with 2 threads (Outer parallel region)
- Lower level with 4 threads (Inner parallel region)

You spawn 2 OpenMP threads

Each thread spawns 4 sub-threads

Good configuration would be to start the 2 upper-level threads as far away as possible. This would be one thread per socket.

And then use the rest of the socket to spawn the lower-level threads

Nested OpenMP (2)

```
export OMP_PLACES=cores
export OMP_PROC_BIND=spread,close
export OMP_NUM_THREADS=4
```

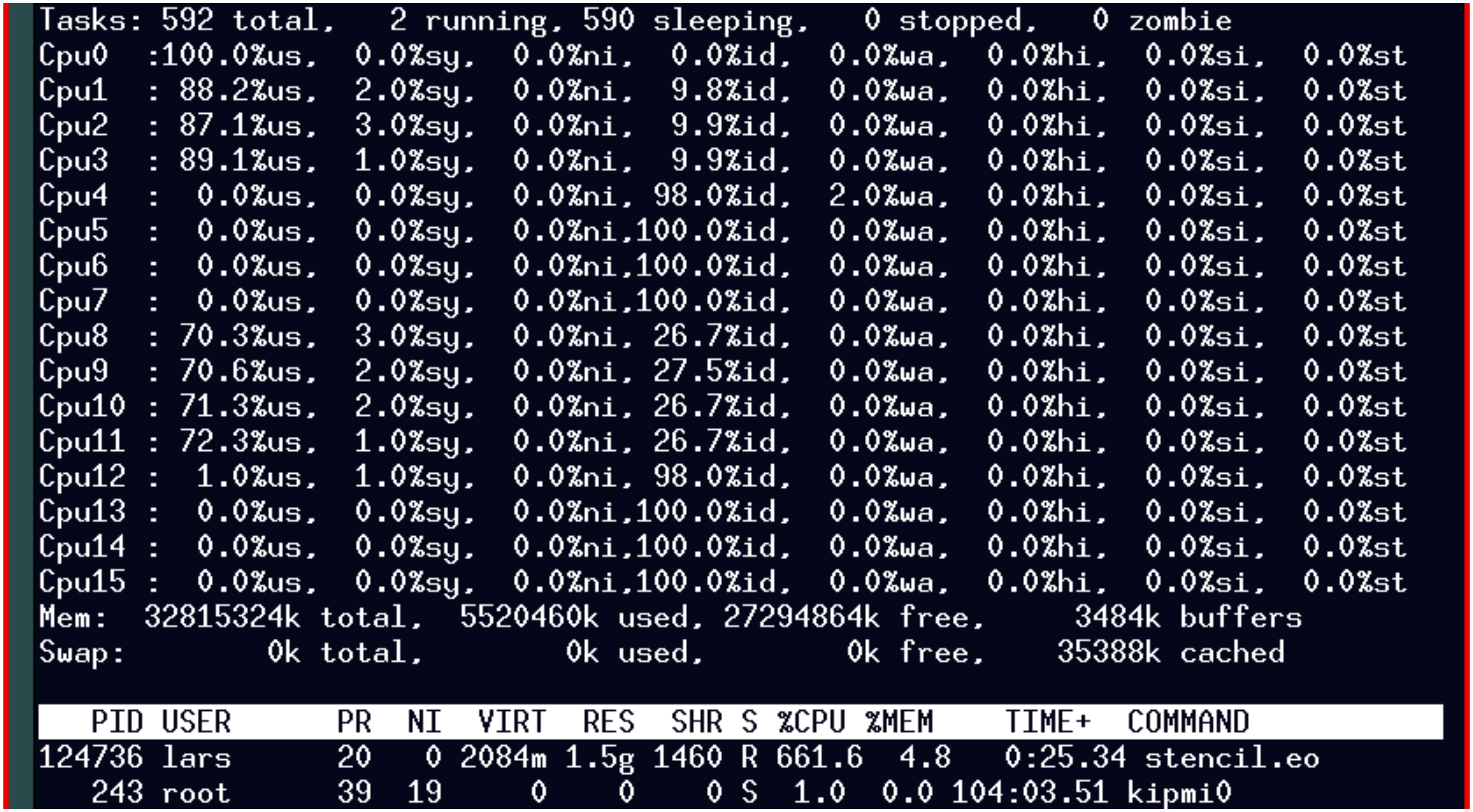
OMP_NUM_THREADS applies to both levels. To change the number of threads in the upper level to 2 I hard-wired this in the code. See example in the lab later.

OMP_PROC_BIND accepts multiple arguments

First argument: Spread is for the outer level

Second argument: Close is for the lower level

Example Nested (close)



```
Tasks: 592 total,   2 running, 590 sleeping,   0 stopped,   0 zombie
Cpu0  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 88.2%us,  2.0%sy,  0.0%ni,  9.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 87.1%us,  3.0%sy,  0.0%ni,  9.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 89.1%us,  1.0%sy,  0.0%ni,  9.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.0%us,  0.0%sy,  0.0%ni, 98.0%id,  2.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 70.3%us,  3.0%sy,  0.0%ni, 26.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 70.6%us,  2.0%sy,  0.0%ni, 27.5%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 71.3%us,  2.0%sy,  0.0%ni, 26.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 72.3%us,  1.0%sy,  0.0%ni, 26.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :  1.0%us,  1.0%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32815324k total,  5520460k used, 27294864k free,     3484k buffers
Swap:        0k total,        0k used,        0k free,    35388k cached

   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
124736 lars      20   0 2084m 1.5g 1460 R 661.6  4.8   0:25.34 stencil.eo
   243 root      39  19     0    0    0 S  1.0  0.0 104:03.51 kipmi0
```

Hybrid: OpenMP and MPI

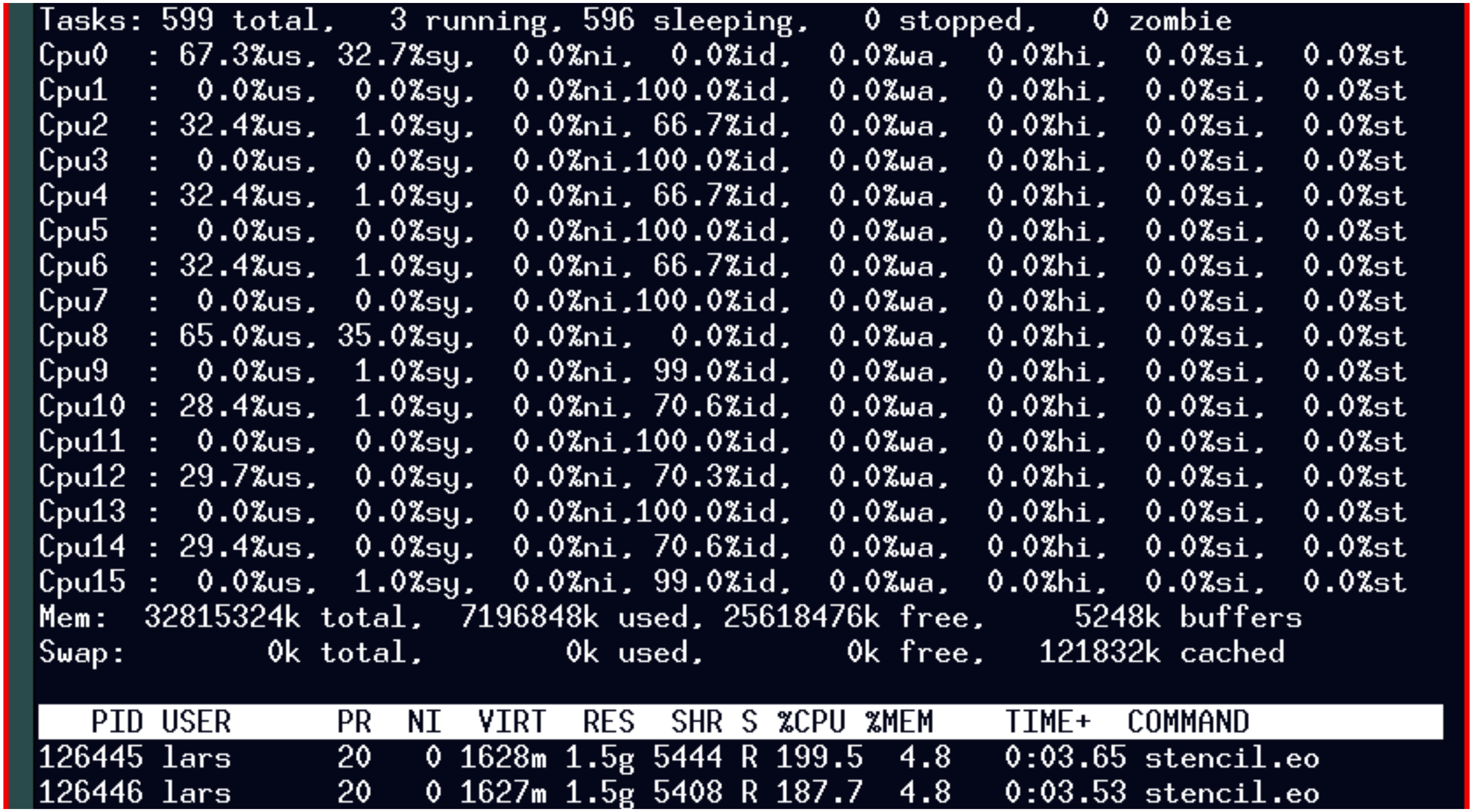
```
export OMP_PLACES=cores
export OMP_PROC_BIND=spread
export OMP_NUM_THREADS=4
```

Job launched with ibrun and tacc_affinity

Example is with 2 MPI tasks on a node; one per socket

```
Ibrun tacc_affinity a.out
```

Example Nested (spread)



```
Tasks: 599 total,   3 running, 596 sleeping,   0 stopped,   0 zombie
Cpu0  : 67.3%us, 32.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 32.4%us,  1.0%sy,  0.0%ni, 66.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 32.4%us,  1.0%sy,  0.0%ni, 66.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  : 32.4%us,  1.0%sy,  0.0%ni, 66.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  : 65.0%us, 35.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 28.4%us,  1.0%sy,  0.0%ni, 70.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 : 29.7%us,  0.0%sy,  0.0%ni, 70.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 : 29.4%us,  0.0%sy,  0.0%ni, 70.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32815324k total,  7196848k used, 25618476k free,     5248k buffers
Swap:        0k total,        0k used,        0k free,   121832k cached

   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
126445 lars      20   0 1628m 1.5g 5444 R 199.5  4.8   0:03.65 stencil.eo
126446 lars      20   0 1627m 1.5g 5408 R 187.7  4.8   0:03.53 stencil.eo
```

Hybrid: OpenMP and MPI

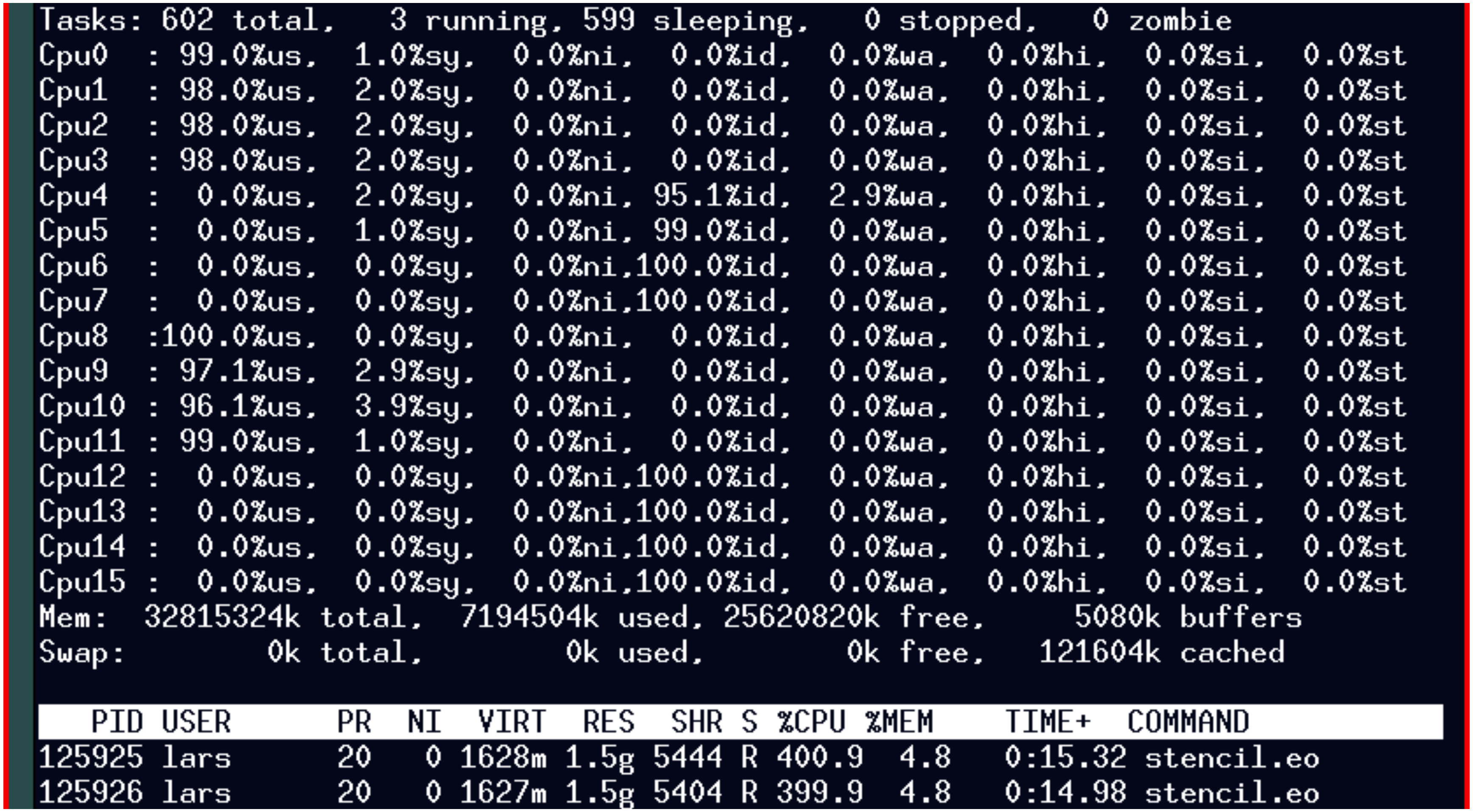
```
export OMP_PLACES=cores
export OMP_PROC_BIND=close
export OMP_NUM_THREADS=4
```

Job launched with ibrun and tacc_affinity

Example is with 2 MPI tasks on a node; one per socket

```
Ibrun tacc_affinity a.out
```

Example Hybrid



```
Tasks: 602 total,   3 running, 599 sleeping,   0 stopped,   0 zombie
Cpu0  : 99.0%us,  1.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 98.0%us,  2.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 98.0%us,  2.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 98.0%us,  2.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  :  0.0%us,  2.0%sy,  0.0%ni, 95.1%id,  2.9%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :  0.0%us,  1.0%sy,  0.0%ni, 99.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu7  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  : 97.1%us,  2.9%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 : 96.1%us,  3.9%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 : 99.0%us,  1.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32815324k total,  7194504k used, 25620820k free,     5080k buffers
Swap:        0k total,        0k used,        0k free,   121604k cached

   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
125925 lars      20   0 1628m 1.5g 5444 R 400.9  4.8   0:15.32 stencil.eo
125926 lars      20   0 1627m 1.5g 5404 R 399.9  4.8   0:14.98 stencil.eo
```

Pinning within the code

- More flexibility
- Different pinning strategies for different parallel regions
- Clause **proc_bind(*arg*)** added to parallel region
- Note: OpenMP places can only be defined once before the execution start

```
                         C/C++
#pragma omp parallel proc_bind(spread)


                         Fortran
!$omp parallel proc_bind(spread)
```

Summary

- Use 3 OpenMP environment variables
  - OMP_PLACES
  - OMP_PROC_BIND
  - OMP_NUM_THREADS
  - Binding and number of threads may also be change in code
- Choose places and binding so that the threads are on dedicated