TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

PRACE

XSEDE
Extreme Science and Engineering Discovery Environment

OpenMP Affinity

IHPCSS
July 7-12, 2019

PRESENTED BY:
Kent Milfeld

RIKEN R-CCS

SciNet
ADVANCED RESEARCH COMPUTING at the UNIVERSITY OF TORONTO

Lecture and Lab slides available at: tinyurl.com/tacc-2019-ihpcss

Outline

- Motivation
- Affinity →
  - affinity fundamentals, the affinity mask, and how it is used
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility

Motivation

Affinity: Determines where application processes can/will execute.

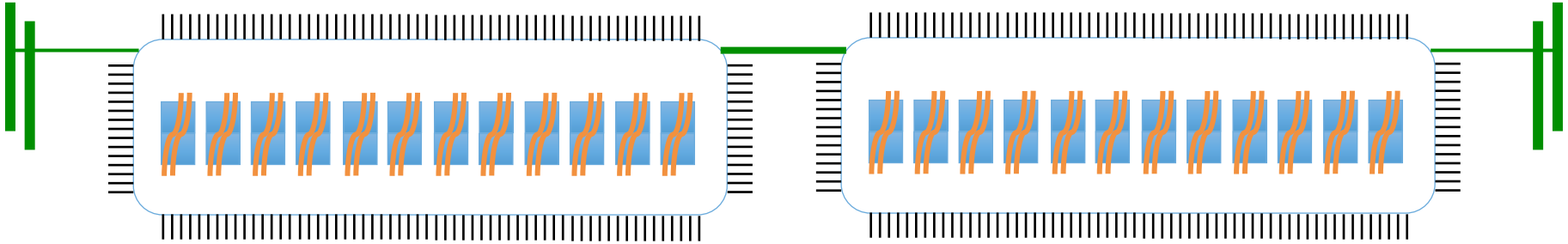
Affinity: Needed for these cases:

- processor count > process count
- processors have local and remote shared memory
- subset of processors have shared cache
- multi-messaging/IO clients may need separation/positioning

Example Architectures

2-socket → Remote and Local Memory
--Non-Uniform Memory Access (NUMA)

Hyper-Threading (a bit more complicated)



Examples:

Lonestar5 Haswell Nodes: 2 sockets (2 Memory NUMA nodes), Hyper-threaded

Stampede2 Skylake* Nodes: 2 sockets (2 Memory NUMA nodes), Hyper-threaded

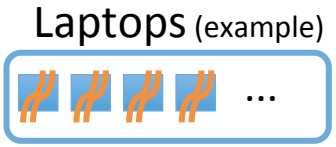
* Has 24 cores/socket.

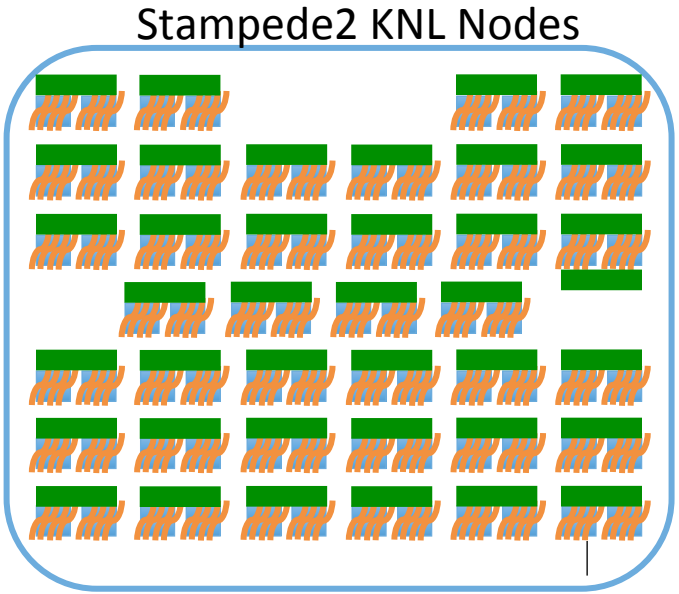
Example Architecture

Single Socket (not always simple)

Hyper-Threading (laptop, 2) (KNL, 4)

2 Cores (a Tile) share a 1MB L2 Cache.





Motivation

- "Often" applications run fine when using the "normal" number of processes (one process|thread per core)
- 1st Approach to Affinity: Do Nothing, Generally Defaults are "good"
  - 2nd order: match process needs to architecture features/resources
  - 2nd order: "pin" processes for limited mobility

Stampede2: KNL -- 68 cores (272 "logical" processors), TILES,...

Stampede2: Skylake -- 48 cores (96 "logical" processors), 2 SOCKETS,...

Lonestar5: Haswell -- 24 cores (48 "logical" processors), 2 SOCKETS,...

Outline

- Motivation
- Affinity →
  - affinity fundamentals, the affinity mask, and how it is used
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility

CPU Affinity -- the mask

processes map onto processors

MPI rank-num or thread-num

proc-id (core-id)

Each process has a bit mask (array of bits)

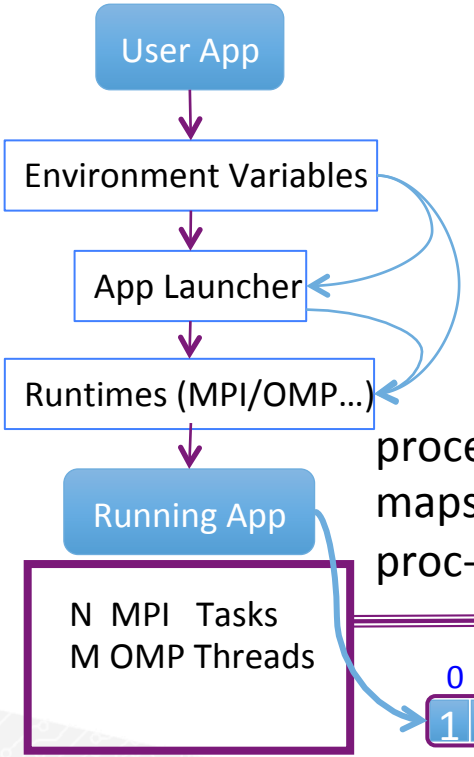
# of bits = # of processors (proc-ids) set bit→ process can run on proc-id (core-id).

Setting Mask

The mask can be set at various places:

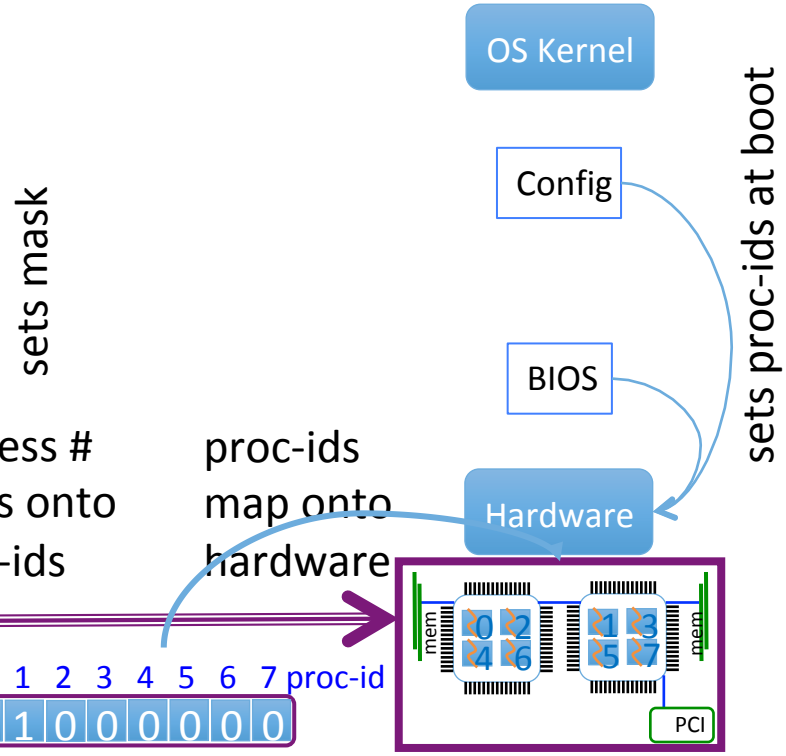
OpenMP Affinity
KMP Affinity
Intel/MV2 MPI

Lib/Compiler/site/
Vendor defaults



Hardware/OS Setup

Placing processes– depends on Hardware

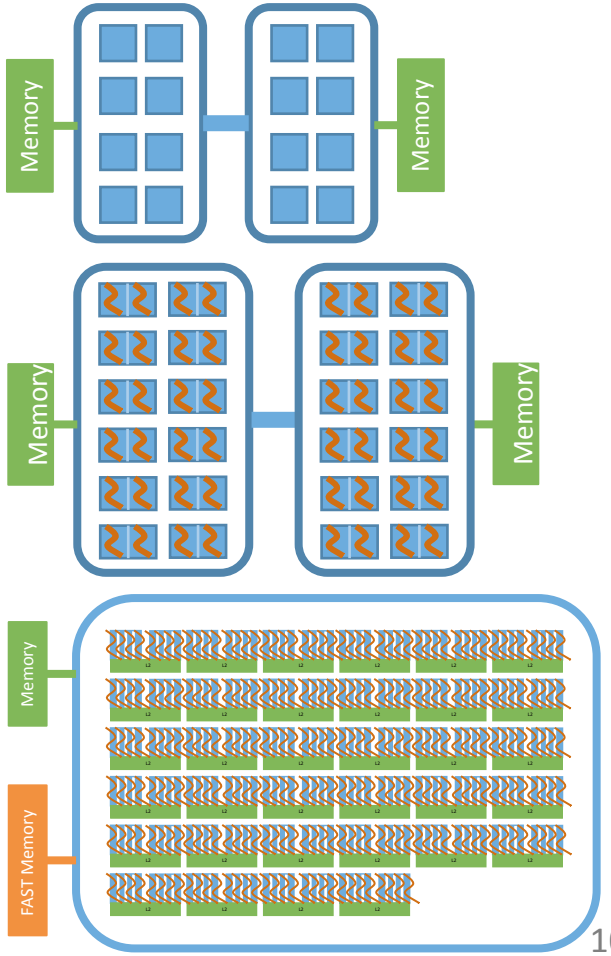


Unix Architecture from *lscpu*

```
sp$ lscpu | grep -i 'core\|thread\|Socket'
Thread(s) per core:    1
Core(s) per socket:    8
Socket(s):             2
```

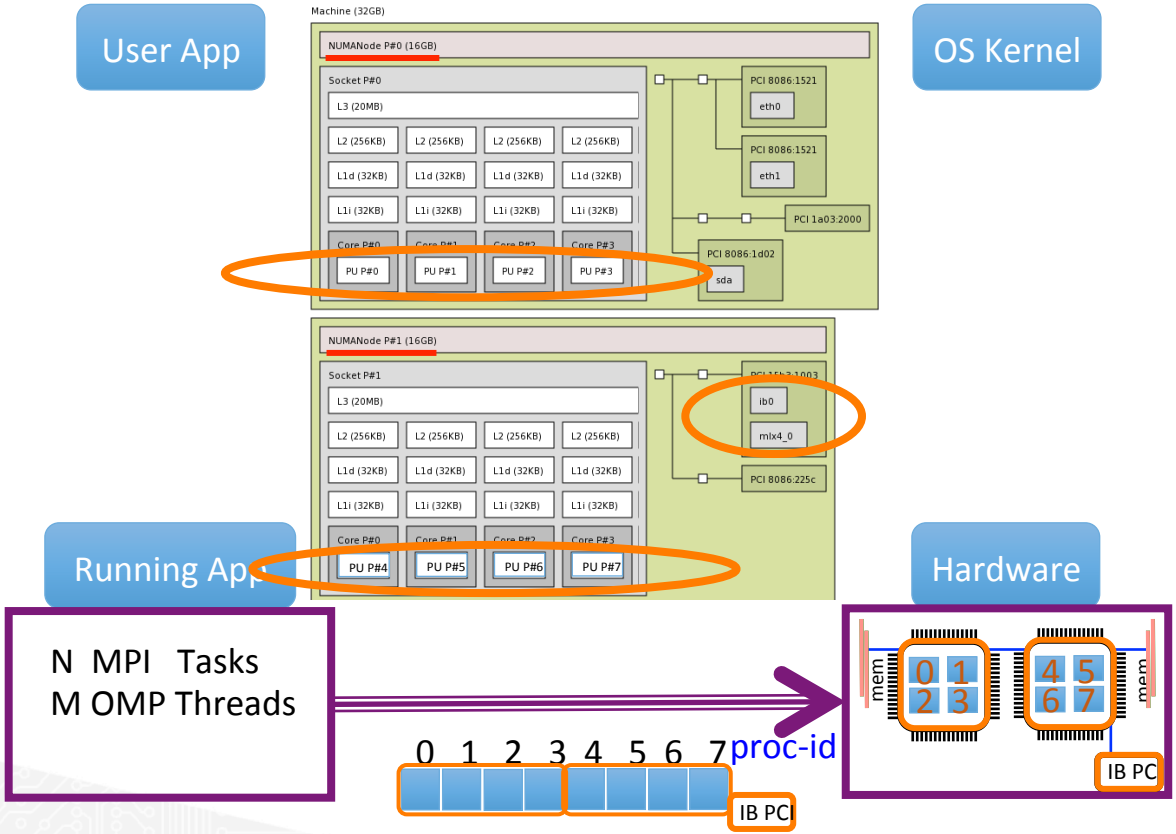
```
ls5%  lscpu | grep -i 'core\|thread\|Socket'
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
```

```
knl$ lscpu | grep -i 'core\|thread\|socket'
Thread(s) per core:    4
Core(s) per socket:    68
Socket(s):             1
```

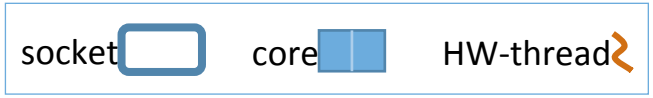


Unix *proc-id* info

```
$ lstopo
```

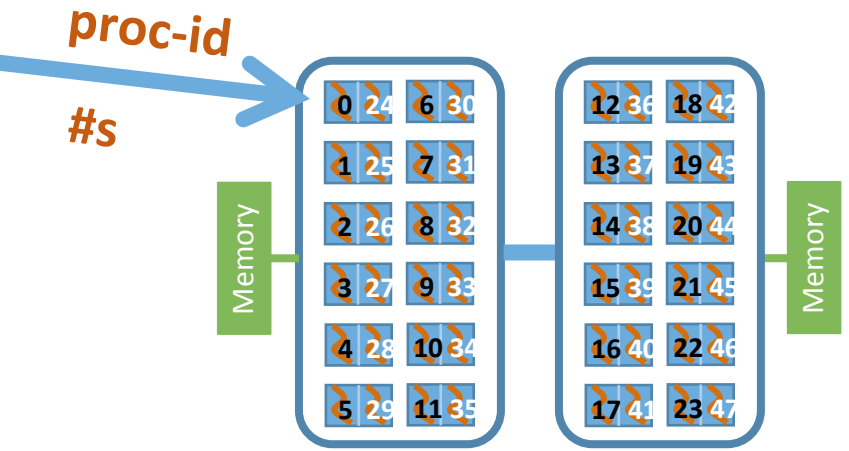


proc-id #'s in /cpu/info




```
ls5% cat /proc/cpuinfo ...awk* ...

   physical id : 0    core id  : 0    processor : 0
   physical id : 0    core id  : 0    processor : 24
   physical id : 0    core id  : 1    processor : 1
   physical id : 0    core id  : 1    processor : 25
   ...
   physical id : 1    core id  : 0    processor : 12
   physical id : 1    core id  : 0    processor : 36
   physical id : 1    core id  : 1    processor : 13
   ...
```




```
ls5% * awk '/processor|core id|physical id/ {arr[j++]=$0};
       END{for(i=0;i<j;i+=3) {printf "%-20s %-20s %-20s\n",
       arr[i+1],arr[i+2],arr[i]} }' /proc/cpuinfo | \
       sed -e 's/[ \t]/ /g' | sort -n -k4,4 -k8,8 -k11,11
```

Outline

- Motivation
- Affinity →
  - affinity fundamentals, the affinity mask, and how it is used
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility

OpenMP Affinity

Portable: works with GNU, Intel, IBM, Oracle, …

There are two components to setting affinity:

(after setting the number of threads)

Distribution Policy: PROC_BIND policy

Set Locations: PLACES explicit list of proc-ids or abstract set

OpenMP Affinity (distribution policy)

- PROC_BIND Policy (set in ENV. VAR.)
  - export OMP_PROC_BIND=close | spread | master

Default

Applies to all parallel regions, except where proc_bind clause is used.

- (within code as clause of a parallel directive)
- #pragma omp parallel proc_bind(close | spread | master)

Clause: Policy applies only to the parallel region with clause.

OpenMP Affinity (Place List)

- Places List (place = “smallest unit of execution”= proc-id # in Linux systems)
  - export OMP_PLACES=<place_list>

A place:	{0}	
Place List:	{0},{1},{2},{3}	List of proc-ids
Interval notation:	<place>:<len>:<stride>	e.g. {0}:4:1

Default Place list: with HyperThreading =HW-threads #’s, without HT =core #’s.

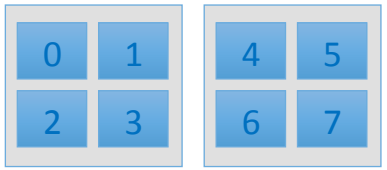
  - export OMP_PLACES=<abstract_name>

abstract name: sockets, cores, threads

Defines a set of places

The precise definitions of the abstract names are implementation defined. An implementation may also add abstract names as appropriate for the target platform.

OpenMP Affinity
PROC_BIND Distribution



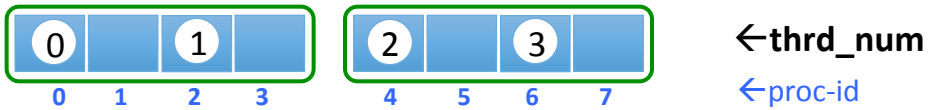

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=close

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=spread

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

COMPACT PACKING



SCATTER PACKING

OpenMP Affinity Places

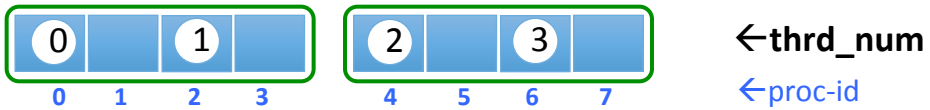
```
export OMP_NUM_THREADS=4
export OMP_PLACES='{0},{1},{2},{3}'

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

```
export OMP_NUM_THREADS=4
export OMP_PLACES='{0},{2},{4},{6}'

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

COMPACT PACKING



SCATTER PACKING

OpenMP Affinity
Places → Interval Expression

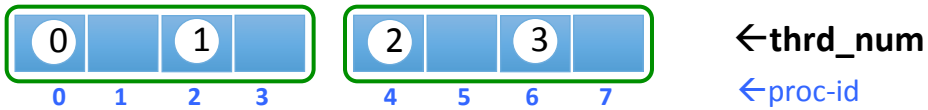
```
export OMP_NUM_THREADS=4
export OMP_PLACES='{0}:4'

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

```
export OMP_NUM_THREADS=4
export OMP_PLACES='{0}:4:2'

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

COMPACT PACKING



SCATTER PACKING

OpenMP Affinity
Places → abstract name

Hyper-Threading Enabled

```
export OMP_NUM_THREADS=8
export OMP_PLACES=threads

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

```
export OMP_NUM_THREADS=8
export OMP_PLACES=cores

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

Binding to Single HW-thread

Binding to Core

←thrd_num

←proc-id

HW-thread

OpenMP Affinity
Places expression

Hyper-Threading Enabled

```
export OMP_NUM_THREADS=8
export OMP_PLACES='{0}:8'   CASE 1
export OMP_PLACES='{8}:8'   CASE 2

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

```
export OMP_NUM_THREADS=8
export OMP_PLACES='{0,8}:8'

!$omp parallel private(thrd_num);
thrd_num =omp_get_thread_num();
```

Binding to Single HW-thread

Binding to Core

←thrd_num

←proc-id

HW-thread

OpenMP Affinity
Places abstract name

Hyper-Threading Enabled

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

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

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

!$omp parallel private(thrd_num);
thrd_num=omp_get_thread_num();
```

Binding to Single HW-thread

Binding to Core

←thrd_num

←proc-id

HW-thread

Outline

- Motivation
- Affinity →
  - affinity fundamentals, the affinity mask, and how it is used
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility

Viewing Affinity mask with amask

Old Stampede: 16-core

export OMP_NUM_THREADS=8 OMP_PROC_BIND=spread

amask_omp

```
        0   proc-id    15
thrd    v              v
 | 0   1000000000000000
 v 1   0010000000000000
   2   0000100000000000
   3   0000001000000000
   4   0000000010000000
   5   0000000000100000
   6   0000000000001000
   7   0000000000000010
```

Bit Mask

```
                proc-id
thrd    |  0       |    10    |
   0    1---------------
   1    --1-------------
   2    ----1-----------
   3    ------1---------
   4    -------1--------
   5    ---------1------
   6    ------------1---
   7    --------------1-
rank    |  0       |    10    |
```

More Readable

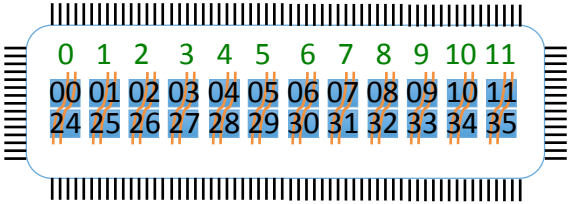
```
                proc-id
thrd    |  0       |    10    |
   0    0---------------
   1    --2-------------
   2    ----4-----------
   3    ------6---------
   4    --------8-------
   5    ----------0-----
   6    ------------2---
   7    --------------4-
rank    |  0       |    10    |
```

Even More Readable

Masks for hyper-threaded platform

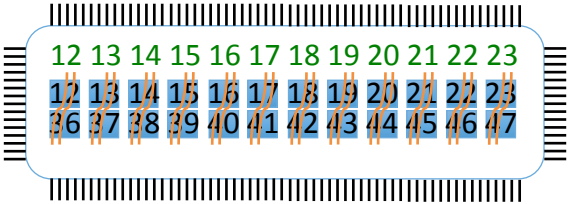
Hyper-Threaded systems 2 sockets x 12 cores → 48 hardware threads (lonestar)

Socket 0



core-id
proc-id – 1st HW-thread
proc-id – 2nd HW-thread

Socket 1

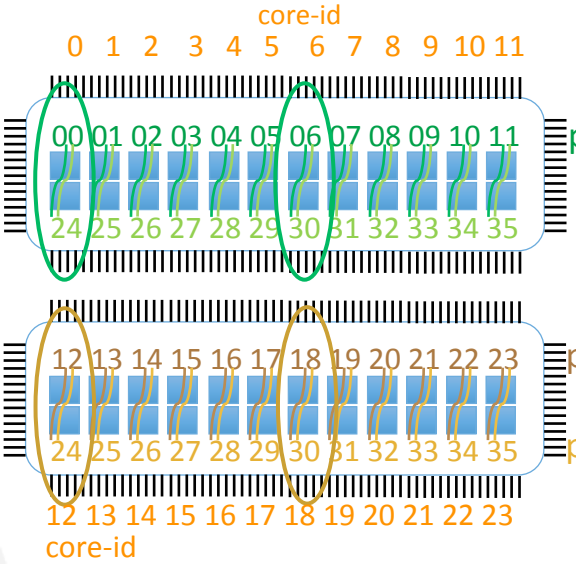


core-id
proc-id – 1st HW-thread
proc-id – 2nd HW-thread

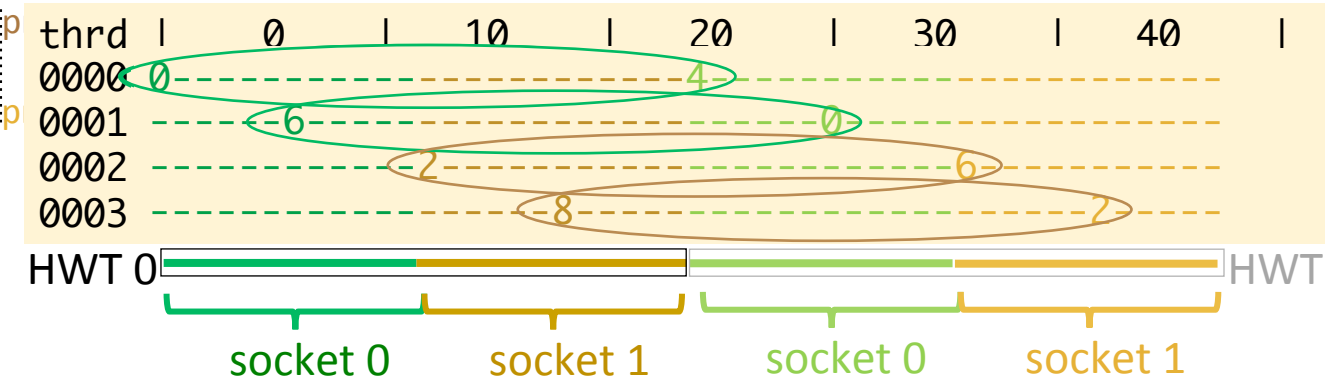
What about hyper-threading…

Hyper-Threaded systems 2 x 12 cores → 48 hardware threads (HWT)

```
$ export OMP_NUM_THREADS=4 OMP_PLACES=cores
$ amask_omp   -vk
```



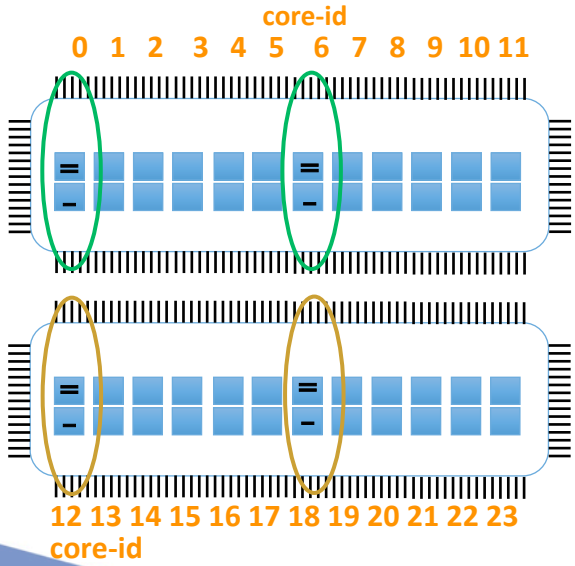
view kernel mask (proc-ids)



What about hyper-threading…

Hyper-Threaded systems 2 x 12 cores → 48 hardware threads

```
$ export OMP_NUM_THREADS=4 OMP_PLACES=cores
$ amask_omp  -vc
```



view core mask (core-ids)

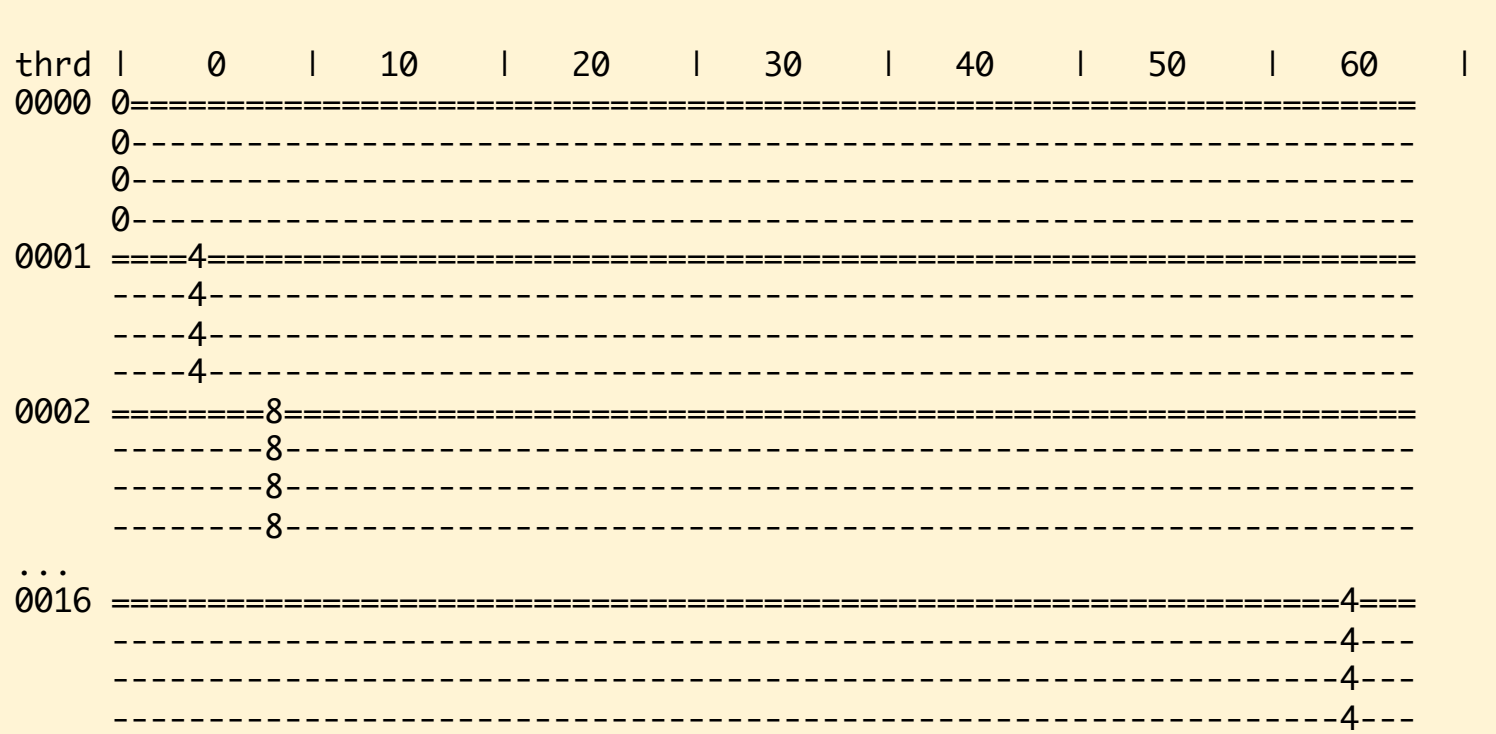
```
thrd |    0    |   10    |   20  ← core-id
0000 0======================      HW-thread 0
     0----------------------      HW-thread 1
0001 ======6=================
     ------6-----------------
0002 ==========2==========
     ----------2----------
0003 ================8=====
     ----------------8-----
```

What about hyper-threading… on KNL

KNL:
68 cores
4 HWT

spread (default) implies distribution with a stride of 4.

```
$ export OMP_NUM_THREADS=17 OMP_PLACES=cores
$ amask_omp
```



Questions!