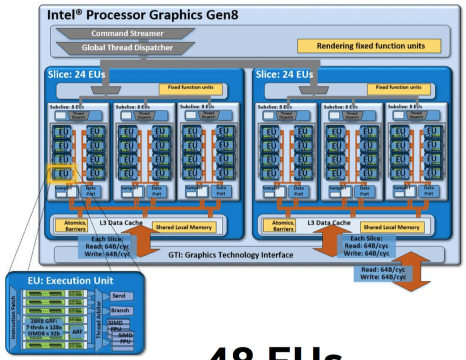
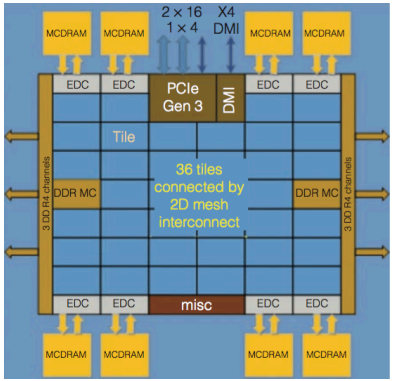
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

OpenMP Affinity



48 EUs



PRESENTED BY:

Kent Milfeld

Lars Koesterke

Outline

- Motivation
- Affinity →
  - what is a mask, where is it set, and what does it mean
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility
- How OpenMP Affinity works with Hybrid Computing

Motivation

Affinity: Determines where application processes can/will execute.

Affinity: Needed for these cases:

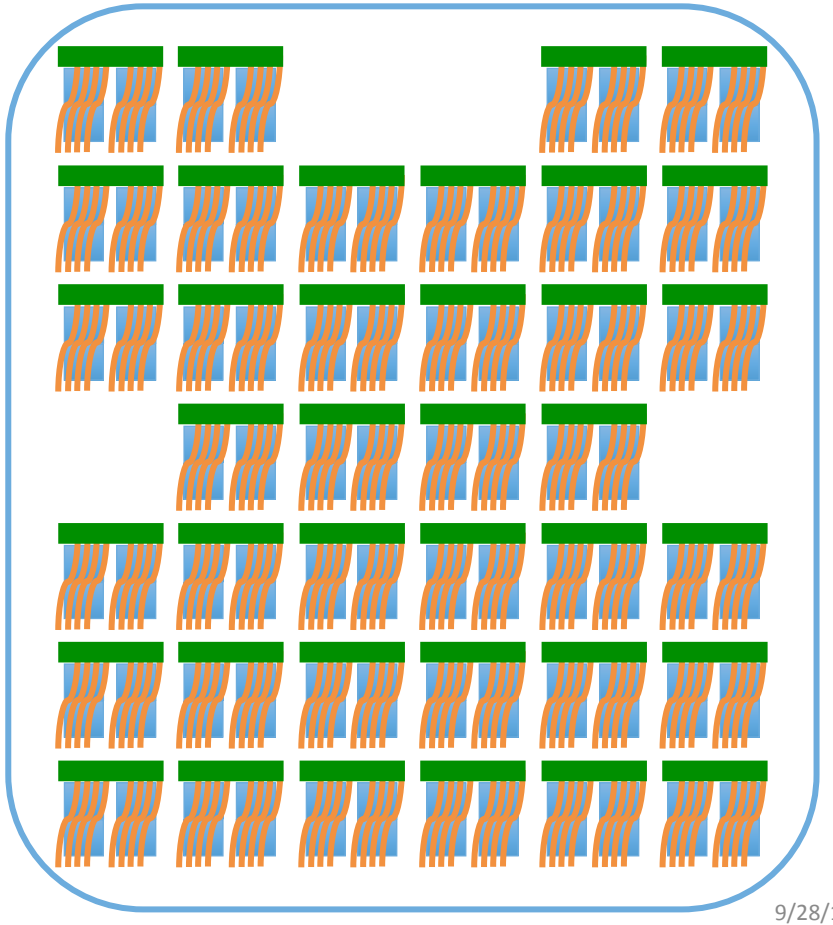
- "logical" processor count > process count
- processors have local and remote shared memory
- subset of processors have shared cache
- multi-messaging/IO clients may need separation/positioning

KNL

Cores share a Cache.

Example:
Stampede2 KNL Nodes
A Tile -- 2 cores + 1MB L2 Shared Cache

KNL: Physical Core Layout
Chip design accommodates 76 Cores



Motivation

- Often applications run fine when using the “normal” number of processes:
- First order Affinity:

  use defaults, prescribed settings or tacc_affinity (See User Guide).

Stampede2: KNL	--	68 cores (272 “logical” processors)
Stampede2: Skylake (soon)	--	48 cores (96 “logical” processors)
Lonestar5: Haswell	--	24 cores (48 “logical” processors)

Motivation

Customized Affinity may help for:
Fewer OpenMP threads/ranks than cores.
When Memory Affinity (NUMA) is important.

- Pure MPI or Pure OpenMP:
  - 1 thread/core (Pure OpenMPI) -- possibly 2 threads/core
  - max 1 rank/core (Pure MPI) – probably best to run hybrid
- Hybrid:
  - 1, 2, 4, 17 or 16 MPI ranks/node with 68/1, 68/2, 68/4, 68/17 or 64/16 threads per rank. (Fill “gaps” with threads. Use “power of 2” ranks for MPI-collective intensive code.)
  - Pitfalls

    Too few threads

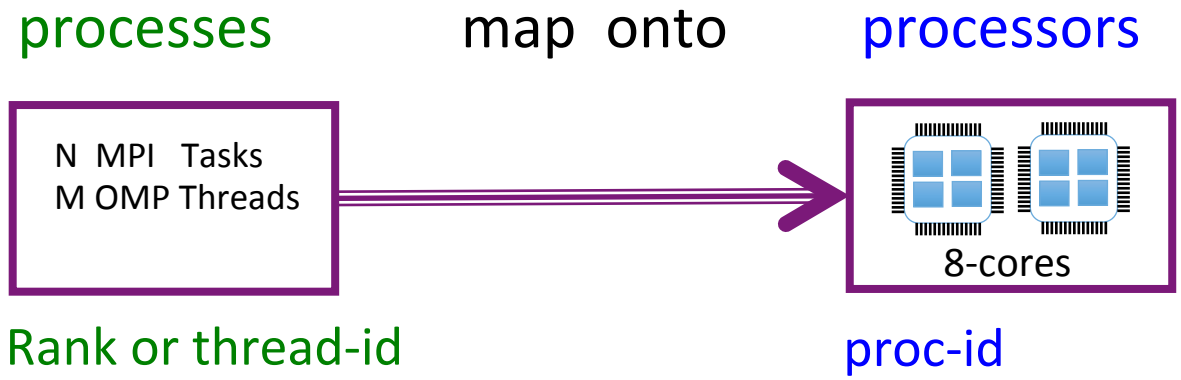
    Too many ranks on KNL (Don’t try 272!)

    Threads not pinned; threads may bunch up on some cores

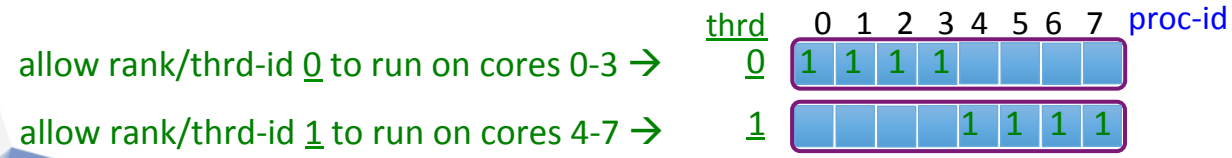
Outline

- Motivation
- Affinity →
  - what is a mask, where is it set, and what does it mean
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility
- Hybrid Computing – things to consider

CPU Affinity -- the mask



Each process has a bit mask: a set bit→ process can run on proc-id.



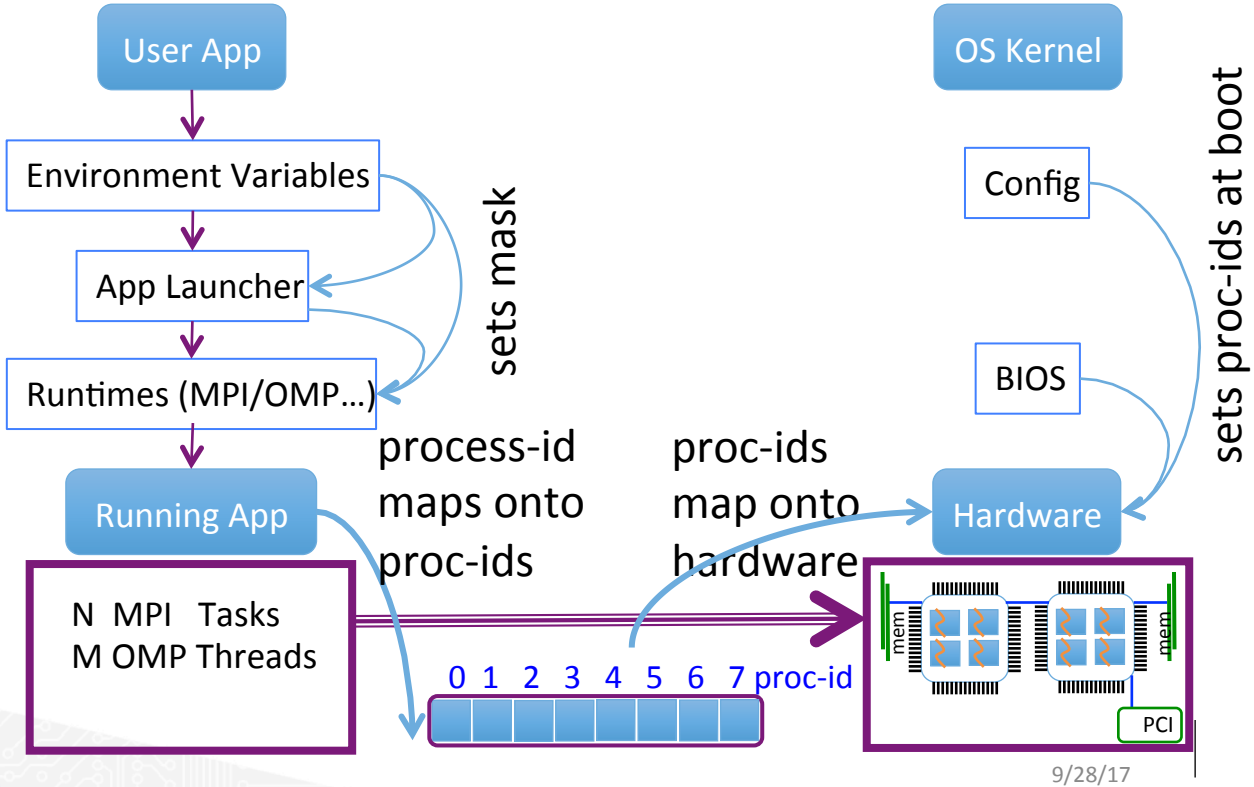
Setting Mask

Hardware/OS Setup

The mask can be set at various places:

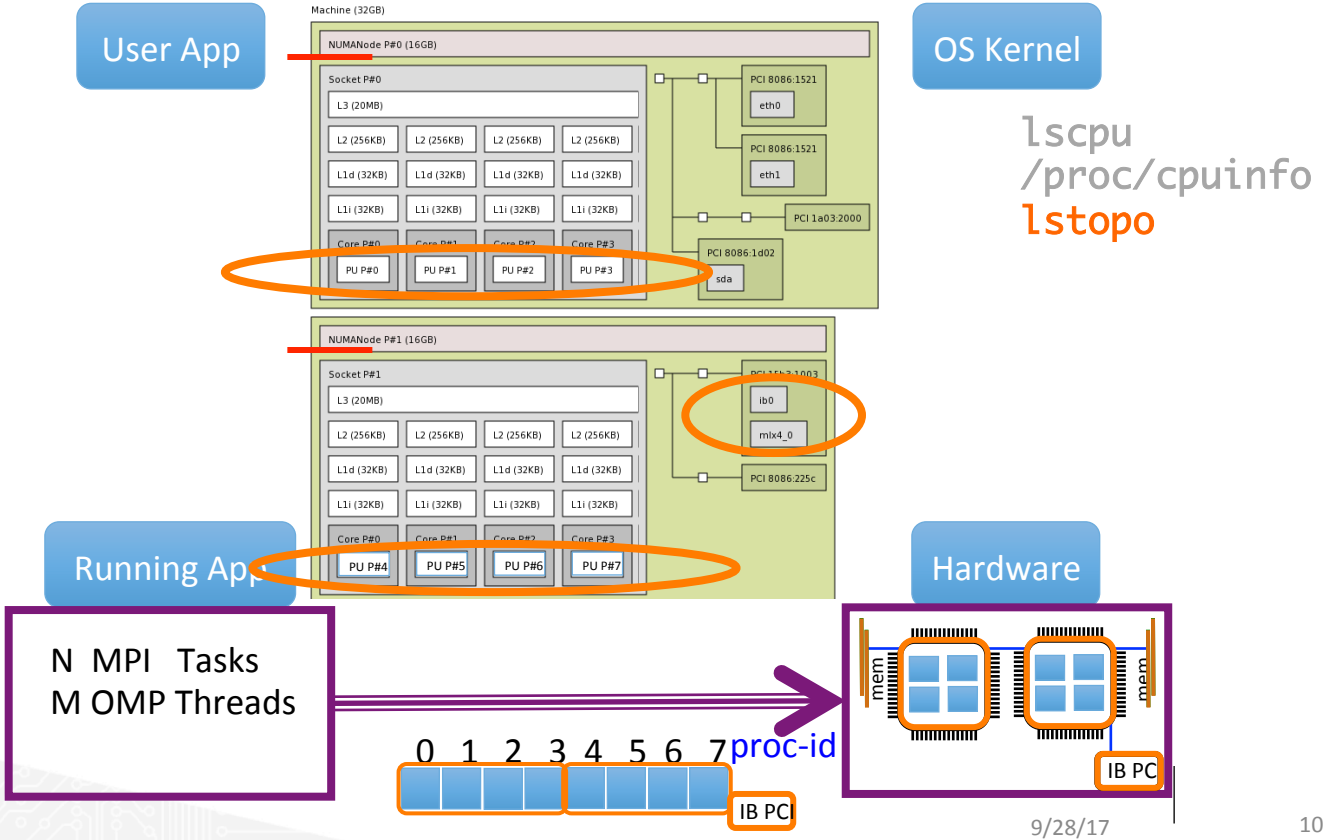
OpenMP Affinity
KMP Affinity
Intel/MV2 MPI

Lib/Compiler/site/
Vendor defaults



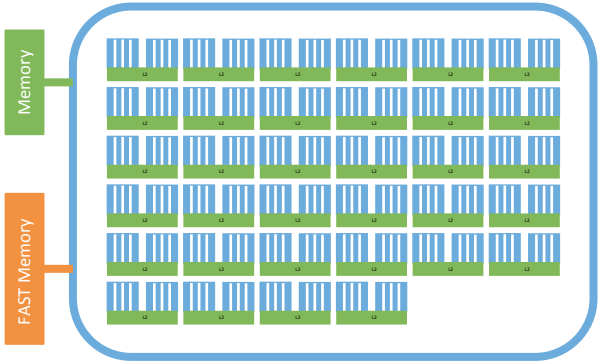
Placing processes– depends on Hardware

Unix proc-id info



lscpu

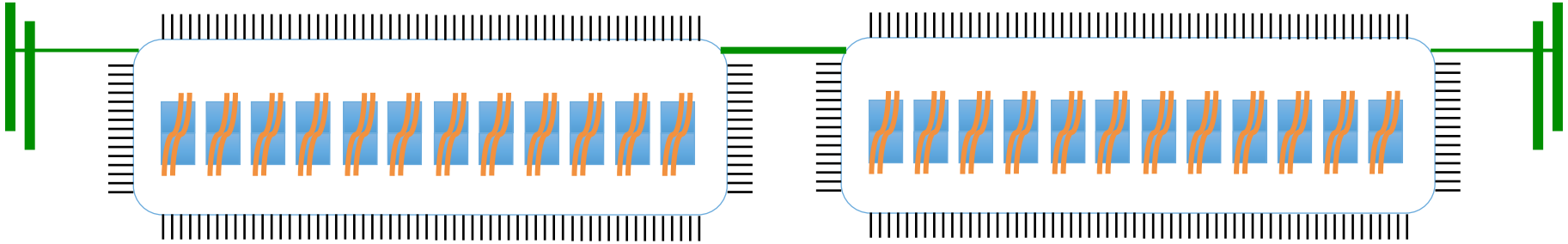
```
knl$ lscpu | grep -i 'core\|thread\|socket'
Thread(s) per core:    4
Core(s) per socket:    68
Socket(s):             1
```



Example:

2-socket →Remote and Local Memory

Hyper-Threading



Examples:

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

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

* Has 24 cores/socket.

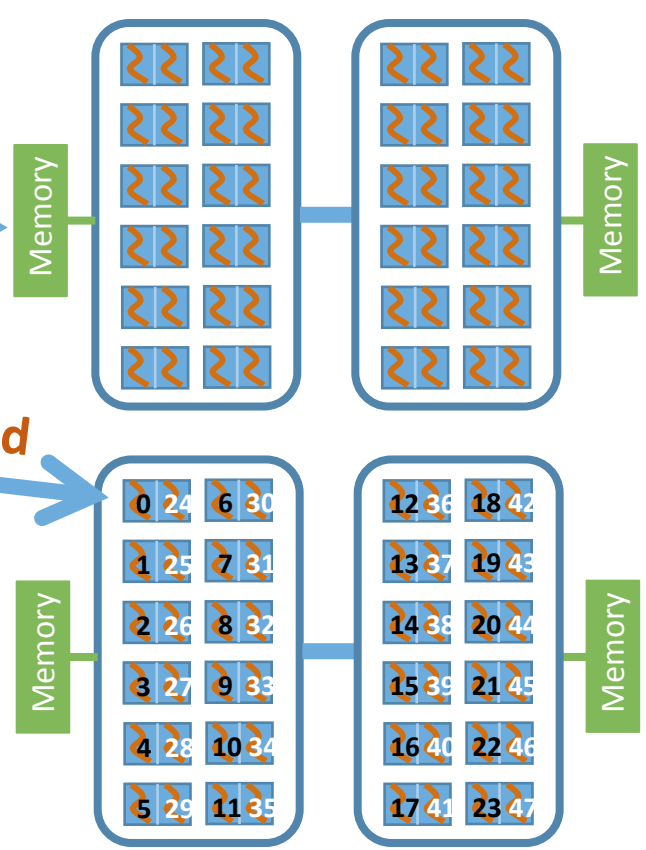
proc-id #'s in /cpu/info

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

Hierarchy summary

```
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
  ...
```

proc-id #s




```
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 →
  - what is a mask, where is it set, and what does it mean
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility
- Hybrid Computing – things to consider

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	list

OpenMP Affinity (distribution policy)

- PROC_BIND Policy:
  - #pragma omp parallel proc_bind(close | spread | master) *
  - export OMP_PROC_BIND=close | spread | master **

Default

* Clause: Policy applies only to the parallel region with clause.
**Env. Var: Applies to all parallel regions, except where proc_bind clause is used.

OpenMP Affinity (set of places)

- OMP_PLACES
  - export OMP_PLACES=<abstract_name>

Defines a set of places

abstract name: sockets, cores, threads

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 (List)

- OMP_PLACES, as a 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}	
Interval notation:	<place>:<len>:<stride>	e.g. {0}:4:1

List of proc-ids

Default Place list: with HyperThreading =list of HW-threads #’s
without HyperThreading =list of cores

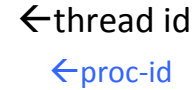
OpenMP Affinity
PROC_BIND Distribution

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=close

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

COMPACT PACKING

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=spread

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

SCATTER PACKING

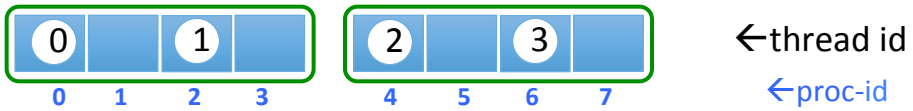


OpenMP Affinity Places

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

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

COMPACT PACKING

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

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



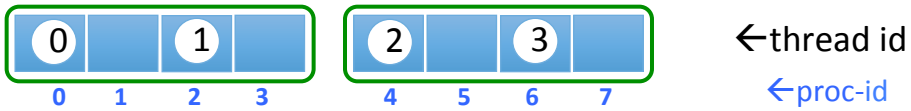
SCATTER PACKING

OpenMP Affinity
Places → Interval Expression

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

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

COMPACT PACKING

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


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



SCATTER PACKING

OpenMP Affinity
Places abstract name

Hyper-Threading Enabled

```
export OMP_NUM_THREADS=8
export OMP_PLACES=threads

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

Binding to Single HW-thread

```
export OMP_NUM_THREADS=8
export OMP_PLACES=cores

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

Binding to Core

←thread id

←proc-id

HW-thread

OpenMP Affinity
Places expression

Hyper-Threading Enabled

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

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

Binding to Single HW-thread

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

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

Binding to Core

←thread id

←proc-id

HW-thread

OpenMP Affinity
Places abstract name

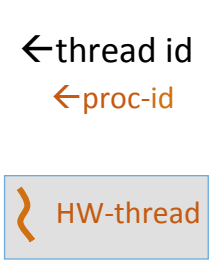
Hyper-Threading Enabled

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


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

Binding to Single HW-thread

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


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

Binding to Core



Outline

- Motivation
- Affinity →
  - what is a mask, where is it set, and what does it mean
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility
- Hybrid Computing – things to consider

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   |
```

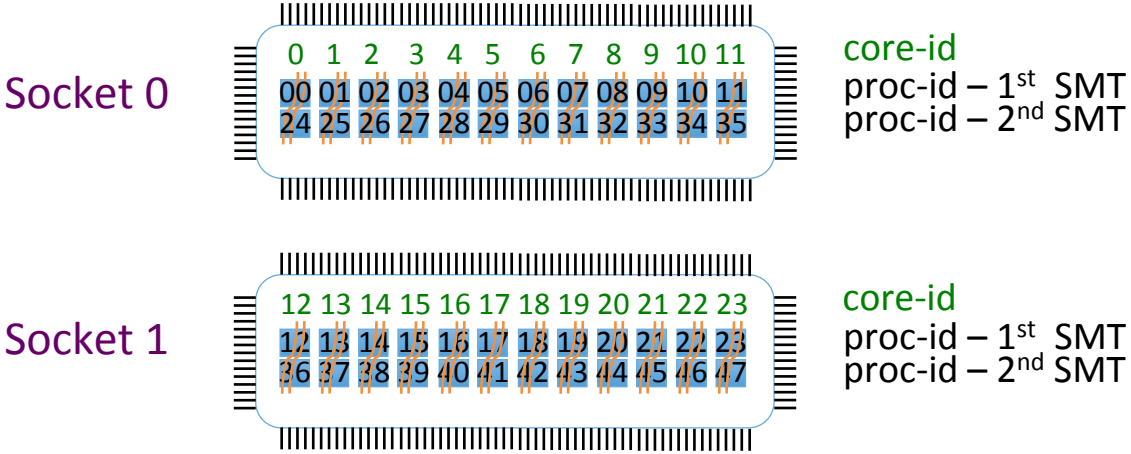
More Readable

```
                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

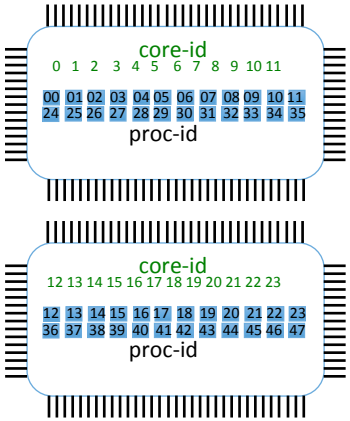
Hyper-Threaded systems 2 sockets x 12 cores → 48 hardware threads (lonestar)



What about hyper-threading…

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

SMT-Centric MAP




```
$ export OMP_NUM_THREADS=4 OMP_PLACES=cores;  amask_omp  -ls

      Each row of matrix is an Affinity mask.
      A set mask bit = matrix digit + column group # in |...|

thrd |    0    |   10    |   20    |   30    |   40    |
0000 0-----------------------4-----------------------
0001 ------6-----------------------0----------------
0002 ------------2-----------------------6-----------
0003 ------------------8-----------------------2-----
```

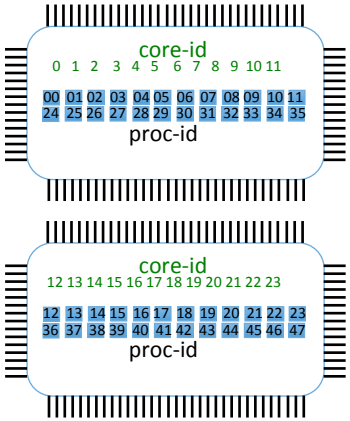
SMT 0 ——— SMT 1

socket 0 socket 1 socket 0 socket 1

What about hyper-threading…

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

Core-Centric MAP




```
$ export OMP_NUM_THREADS=4 OMP_PLACES=cores;  amask_omp
```

Each row of matrix is a CORE mask for a SMT.
core id = matrix digit + column group # in |...|
proc-id = core id + add 24 to each additional row

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

What about hyper-threading… on KNL

KNL: 68 cores
4 SMT

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

```
thrd |    0    |   10    |   20    |   30    |   40    |   50    |   60    |
0000 0=========================================================================
     0-------------------------------------------------------------------------
     0-------------------------------------------------------------------------
     0-------------------------------------------------------------------------
0001 ====4====================================================================
     ----4---------------------------------------------------------------------
     ----4---------------------------------------------------------------------
     ----4---------------------------------------------------------------------
0002 ========8================================================================
     --------8-----------------------------------------------------------------
     --------8-----------------------------------------------------------------
     --------8-----------------------------------------------------------------
...
0016 ============================================================================4===
     ------------------------------------------------------------------------4---
     ------------------------------------------------------------------------4---
     ------------------------------------------------------------------------4---
```

Outline

- Motivation
- Affinity →
  - what is a mask, where is it set, and what does it mean
- OpenMP Affinity: PROC_BIND and PLACES
- Showing Mask with *amask* utility
- How OpenMP Affinity works with Hybrid Computing

Hybrid Affinity-- MPI

- An OpenMP team inherits the MPI process mask
- A thread's mask can only contain the set, or subset of the MPI process mask.
- Easiest Approach:
  - Use Distribution of OMP_PROC_BIND (close, spread)
  - Use granularity of OMP_PLACES (threads, cores, <vendor_defined>)

- Complicated (Advanced) Approach:
  - Launch script for each MPI process that uses proc-id list in OMP_PLACES tailored for each process (depending upon rank).

```
#SLURM -N 2 --tasks-per-node 17

export OMP_NUM_THREADS=2 \
       OMP_PROC_BIND=spread \
       OMP_PLACES=cores

ibrun ./a.out  # 1 thread per tile
```

```
#SLURM -N 2 --tasks-per-node 17

export OMP_NUM_THREADS=2


ibrun ./mpi_script  # see next page
```

MPI mask

```
$ ibrun -np 2 amask_mpi
```

```
 Each row of matrix is a mask for a SMT-id -- 0, 1, 2, or 3.
     CORE ID  = matrix digit + column group # in |...|
     A set mask bit (proc-id) = core id + add 68 to each additional row.

rank |    0    |   10    |   20    |   30    |   40    |   50    |   60    |
0000 01234567890123456789012345678901234567890123456789012345678901234567
     01234567890123456789012345678901234567890123456789012345678901234567
     01234567890123456789012345678901234567890123456789012345678901234567
     01234567890123456789012345678901234567890123456789012345678901234567
0001 ================================================================
     ----------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
```

MPI process Rank 0 allowed anyplace on 1st 34 cores. →

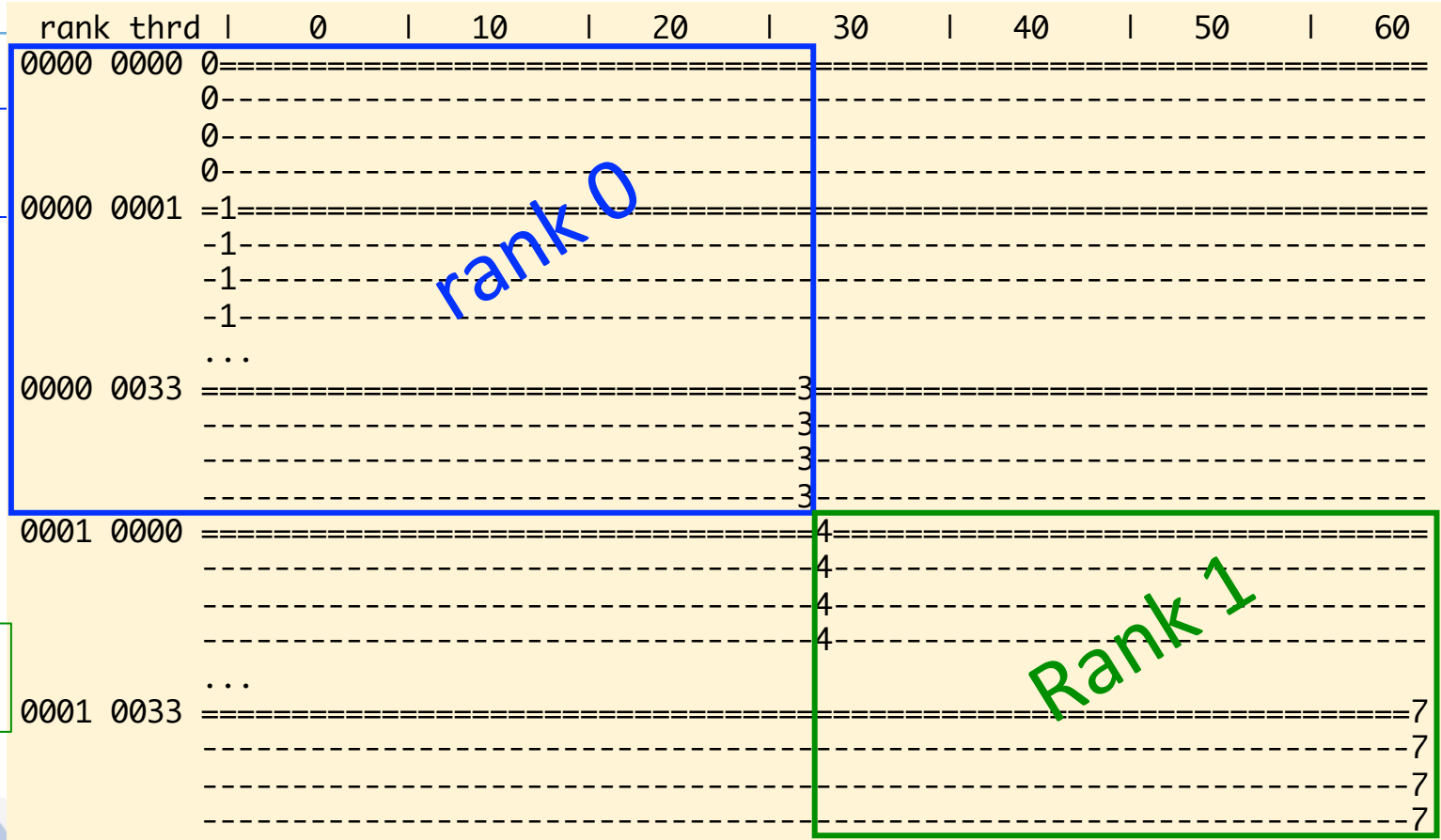
MPI process Rank 1 allowed anyplace on 2nd set of 34 cores. →

MPI mask

```
$ export OMP_NUM_THREADS=34 OMP_PLACES=cores
$ ibrun -np 2 amask_hybrid
```

Thread Team inherits mask of MPI process Rank 0-- and adjusts for each thread

Thread Team inherits mask of MPI process Rank 1-- and adjusts for each thread



Hybrid Affinity-numa script

I_MPI_DOMAIN=numa
On SNC-4 mode compute node

```
#SLURM -N 2 --tasks-per-node 4
# uses numa "aware" script
export I_MPI_DOMAIN=numa
mpirun -np 8 ./numa.sh
```

numa.sh

```
#!/bin/bash

[[ $PMI_RANK == 0 ]] && export OMP_NUM_THREADS=18
[[ $PMI_RANK == 1 ]] && export OMP_NUM_THREADS=18
[[ $PMI_RANK == 2 ]] && export OMP_NUM_THREADS=16
[[ $PMI_RANK == 3 ]] && export OMP_NUM_THREADS=16

./a.out
```

Hybrid Affinity -- omp places script (w. mpi rank offsets)

```
#!/bin/bash

  R=$PMI_RANK
TPN=$SLURM_NTASKS_PER_NODE                  #test with R=3 TPN=2 OMP_NUM_THREADS=2

MPI_OFFSET=$((68/TPN))                      #Divide up cores among ranks (MUST BE EVEN)
OMP_stride=$((MPI_OFFSET/OMP_NUM_THREADS)) #Stride between OMP threads

                 # Assume ranks are numbered sequentially across all nodes
                 # Rank :                          node1      node2    node3   ...
                 # rank number on each node:    {0 1 2 3} {4 5 6 7} {8 9 10 11}...
                 # rank modulo TPN (tasks/node) {0 1 2 3} {0 1 2 3} {0 1  2  3}...

Rm=$((R%TPN))                               # rank modulo tasks-per-node

off0=$((      Rm*MPI_OFFSET))               # proc-id offset for SMT thread 0
off1=$(( 68 + Rm*MPI_OFFSET))               #
off2=$((136 + Rm*MPI_OFFSET))               #
off3=$((204 + Rm*MPI_OFFSET))               # proc-id offset for SMT thread 3

export OMP_PLACES="{$off0,$off1,$off2,$off3}:$OMP_NUM_THREADS:$OMP_stride"

                 #    {Core HW-threads}      no. of PLACES  :  Stride

./a.out          #test this script: Set PMI_RANK & SLURM_NTAKS_PER_NODE
                 # and replace ./a.out with echo `hostname` $R $OMP_PLACES
```

Can go over this at break with anyone.

Questions!

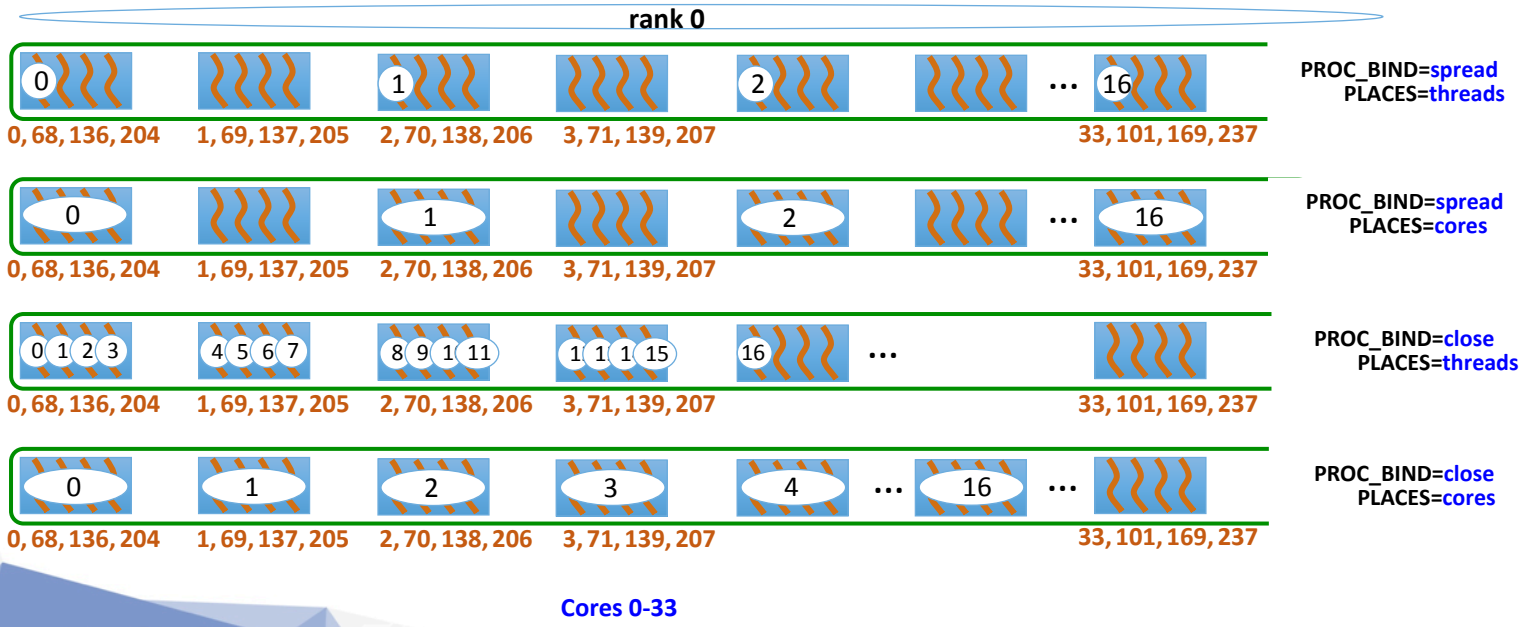
Extras

- How OpenMP Affinity works with Hybrid Computing
  Details will be covered later in Hybrid Computing.

Hybrid Affinity, KNL

Use OMP_PROC_BIND and OMP_PLACES to distribute

```
#SLURM -n 2      # 34 cores/MPI proc

export OMP_NUM_THREADS=17 \
       OMP_PROC_BIND=<type> \
       OMP_PLACES=<abstract_name>

ibrun ./a.out
```



Defaults

Hybrid Computing on KNL

Suggested MPI process and thread count combinations

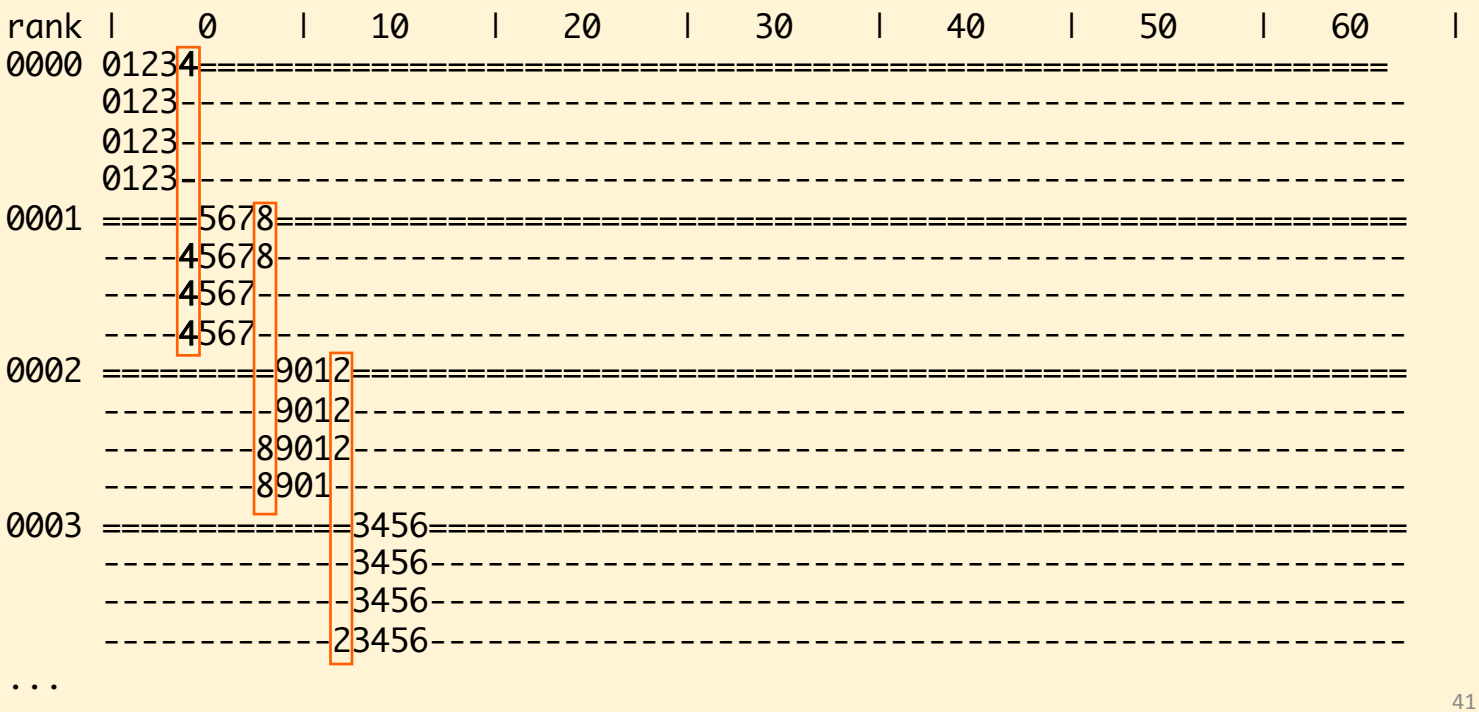
MPI Rank Count	Use Case	Thread Count – OMP_NUM_THREADS (for 1,2,3,4 threads/core)
1	1 MPI communicator	(68, 136, 204,272)
2		(34, 68, 102, 136)
4		(17, 34, 51, 68)
8		(8,16,24,32) (not using all cores, 68/8 != integer)
16		(4,8,12,16) (not using all cores, 68/8 != integer)
17		(4, 8, 12, 16)
34	1 MPI proc/tile	(2, 4, 6, 8)
68	1 MPI proc/core	(1, 2, 3, 4)

Be careful, for 8 and 16 MPI processes– USE OMP_PROC_BIND=spread (not KMP_KMP_AFFINITY=balanced); for MPI ranks/threads = 8/25 and 16/12 some cores have 4 threads and others only 2.

Always consider MPI mask first (hybrid computing)

```
$ mpirun -np 16 amask_mpi     # for hybrid, export OMP_NUM_THREADS=17
```

Core sharing by two MPI processes →



Always consider MPI mask first (hybrid computing)

```
$ mpirun –np 17 amask_mpi  # for hybrid, export OMP_NUM_THREADS=16
```

NO Core sharing by any MPI processes

```
rank |    0    |   10    |   20    |   30    |   40    |   50    |   60
|
0000 0123=============================================================
     0123-------------------------------------------------------------
     0123-------------------------------------------------------------
     0123-------------------------------------------------------------
0001 ====4567=========================================================
     ----4567-------------------------------------------------------
     ----4567-------------------------------------------------------
     ----4567-------------------------------------------------------
0002 ========8901=====================================================
     --------8901-------------------------------------------------
     --------8901-------------------------------------------------
     --------8901-------------------------------------------------
...
0016 ============================================================4567
     ------------------------------------------------------------4567
     ------------------------------------------------------------4567
     ------------------------------------------------------------4567
```