



TACC

TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU



Understanding and Controlling Processor Affinity

PRESENTED BY:

Kent Milfeld

Slides at: tinyurl.com/chpc-2017-affinity



CHPC
CENTRE FOR HIGH
PERFORMANCE COMPUTING

National Conference

Outline

- Motivation
- Affinity -- what is it
- OpenMP Affinity
- Ways to show process masks
- Hybrid Computing
 - How it works
 - Advancing Standards



Motivation

We may need to know|control the location of application processes.

- Xeon-Phi (**KNL**) Node: 72 Cores **288 hardware threads**
- Xeon (**SKL-X**) Node: 2x28=56 Cores **112 hardware threads**
- process count < hardware thread count
- L2/L3 cache sharing (tiles/sockets)
- Minimal Memory Distance
- Messaging/IO process near PCI-e



Motivation

Practical Consulting Question:

- Where does my master thread run?
- How are my threads assigned in a hybrid run?
- How can I pin my threads.
- ...

Control
Affinity

View
Affinity

Outline

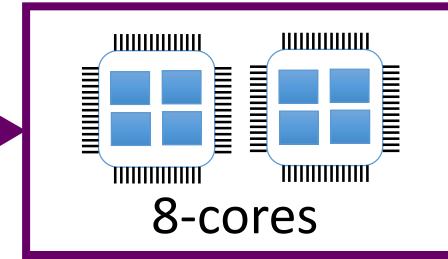
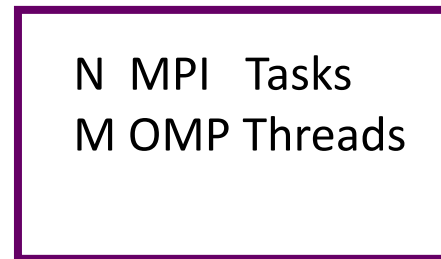
- Motivation
- Affinity -- what is it
- OpenMP Affinity
- Ways to show process masks
- Hybrid Computing
 - How it works
 - Advancing Standards

CPU Affinity -- the mask

processes

map onto

processors



Rank or thread-id

proc-id

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

	thrd	0	1	2	3	4	5	6	7	proc-id
allow rank/thrd-id <u>0</u> to run on cores 0-3 →	<u>0</u>	1	1	1	1					
allow rank/thrd-id <u>1</u> to run on cores 4-7 →	<u>1</u>					1	1	1	1	

Setting Mask

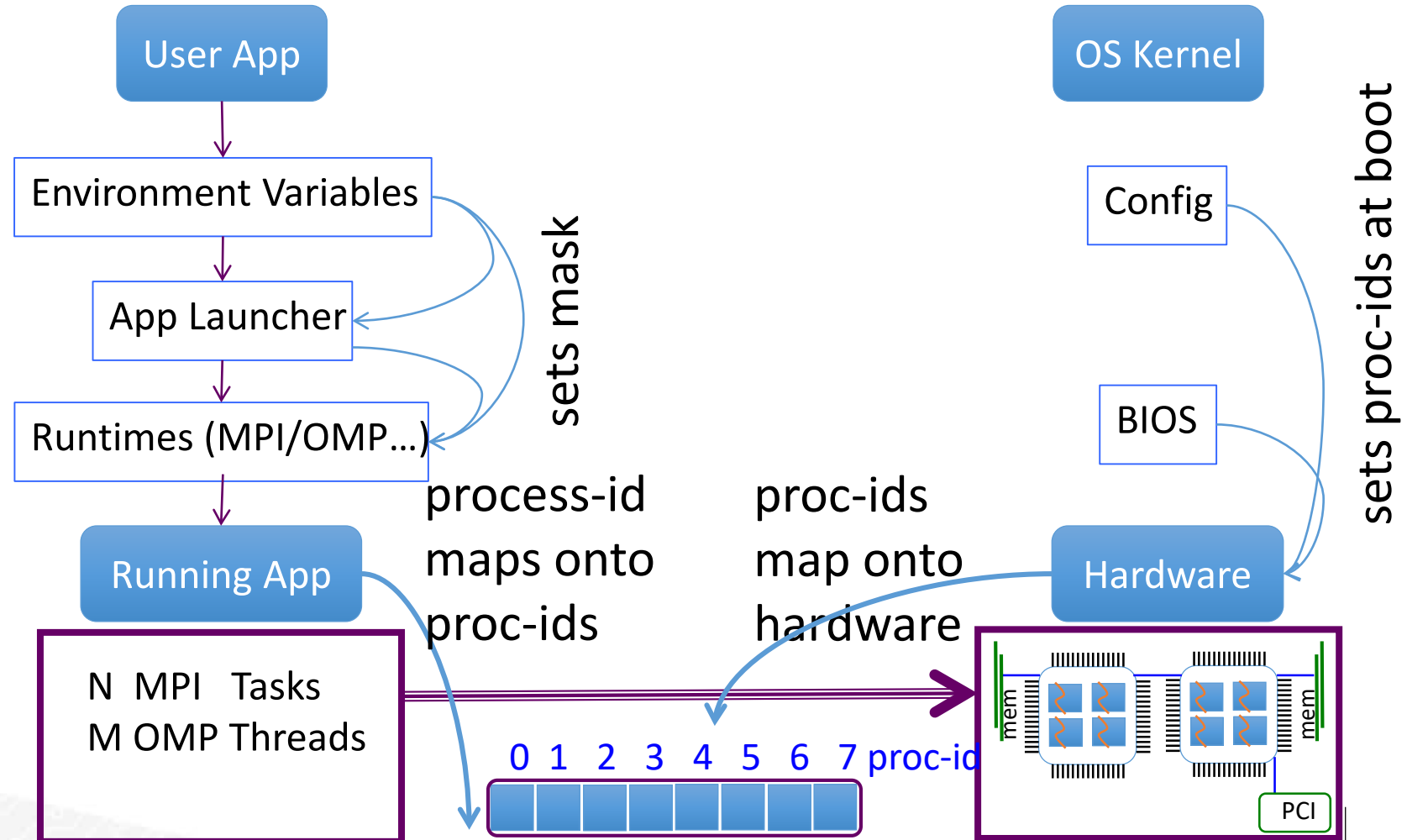
Hardware/OS Setup

The mask can be set at various places:

Placing processes— depends on Hardware

OpenMP Affinity
KMP Affinity
Intel/MV2 MPI

Lib/Compiler/site/
Vendor defaults



Unix proc-id info

User App

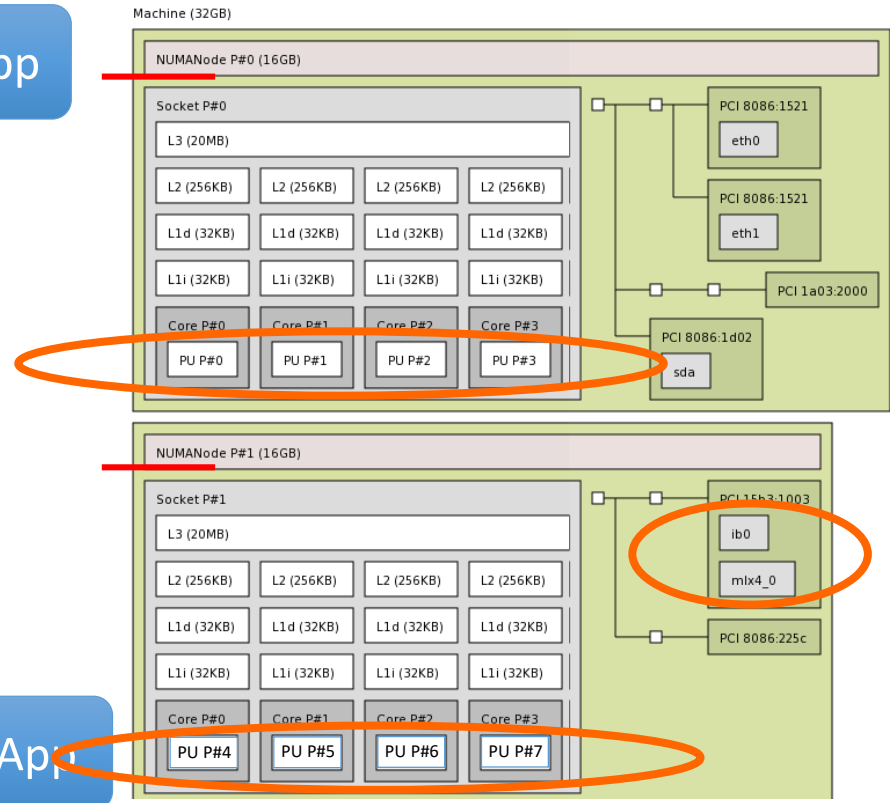
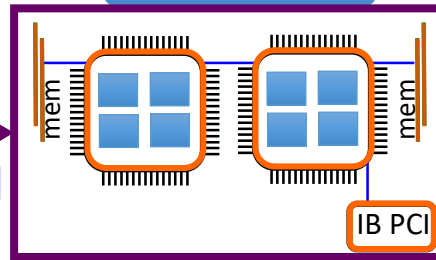
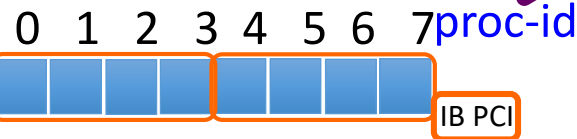
OS Kernel

lscpu
/proc/cpuinfo
lstopo

Running App

Hardware

N MPI Tasks
M OMP Threads



lscpu

```
sp$ lscpu | grep -i 'core\|thread\|Socket'
```

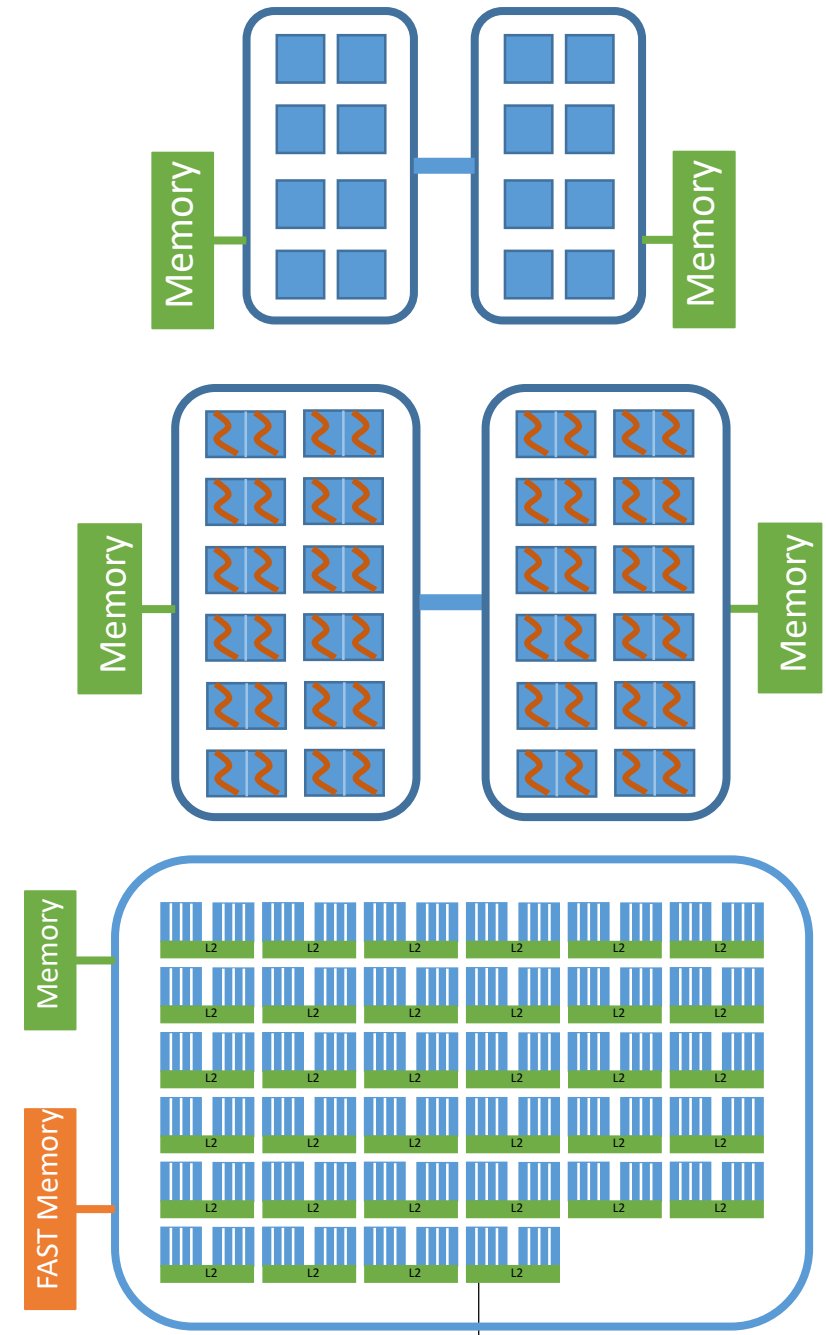
```
Thread(s) per core: 1  
Core(s) per socket: 8  
Socket(s): 2
```

```
ls5% lscpu | grep -i 'core\|thread\|Socket'
```

```
Thread(s) per core: 2  
Core(s) per socket: 12  
Socket(s): 2
```

```
kn1$ lscpu | grep -i 'core\|thread\|socket'
```

```
Thread(s) per core: 4  
Core(s) per socket: 68  
Socket(s): 1
```



proc-id #'s in /cpu/info

socket  core  HW-thread 

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

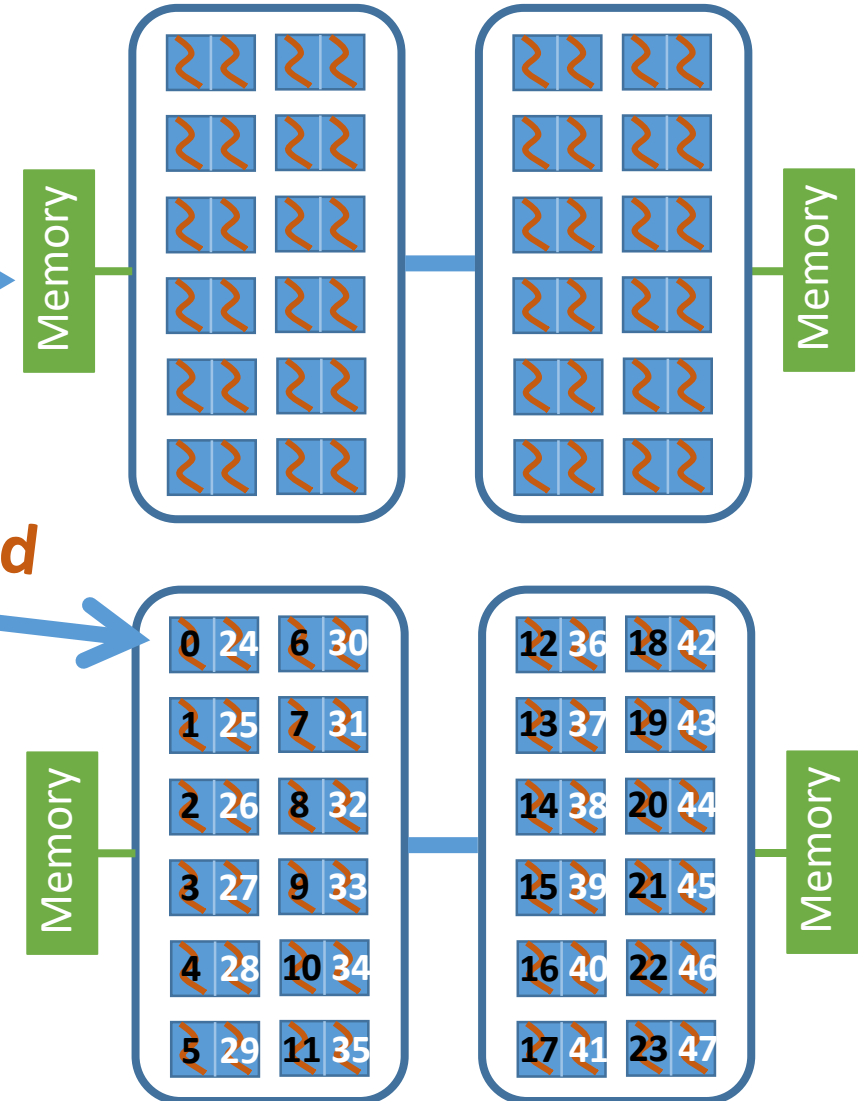
Hierarchy summary

```
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 it
- OpenMP Affinity
- Ways to show process masks
- Hybrid Computing
 - How it works
 - Advancing Standards



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

Locations: PLACES abstract name or list

OpenMP Affinity (distribution policy)

- PROC_BIND Policy:

export

OMP_PROC_BIND=close | spread | master **

Default



**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>**

abstract name: **sockets, cores, threads**

} Defines a set of places

OpenMP Affinity (List)

- **OMP_PLACES**, as a list

(place = “smallest unit of execution”= proc-id # in Linux)

- export **OMP_PLACES=<place_list>**

A place: {0}

Place List: {0},{1},{2},{3}

Interval notation: <place>:<len>:<stride> e.g. {0}:4:1

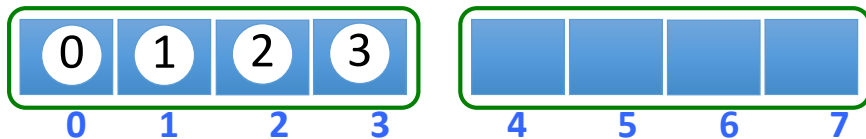
Default Place list: with HyperThreading =list of HW-threads #'s
without HyperThreading = list of cores.

OpenMP Affinity

PROC_BIND Distribution

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=close

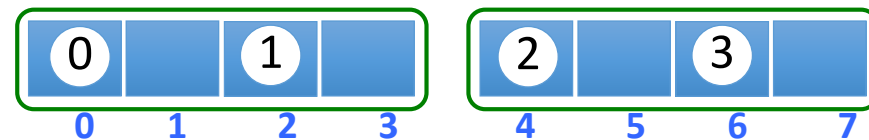
!$omp parallel private(tid);
  tid=omp_get_thread_num();
```



COMPACT PACKING

```
export OMP_NUM_THREADS=4
export OMP_PROC_BIND=spread

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



SCATTER PACKING

←thread id
←proc-id



2 sockets x 4 cores

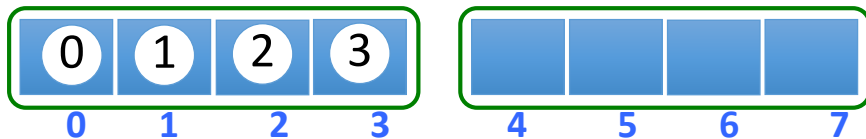
OpenMP Affinity Places

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

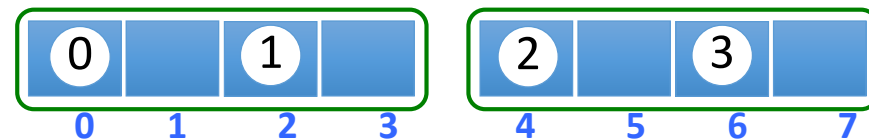
!$omp parallel private(tid);
  tid=omp_get_thread_num();
```

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

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



COMPACT PACKING

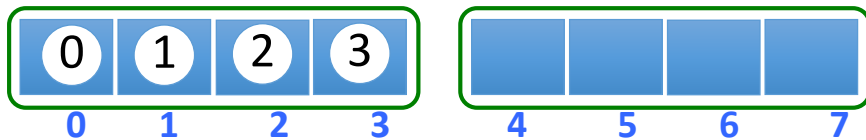


SCATTER PACKING

←thread id
←proc-id

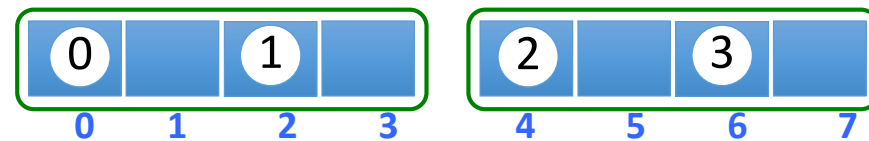
OpenMP Affinity Places → Interval Expression

```
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

←thread id
←proc-id

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

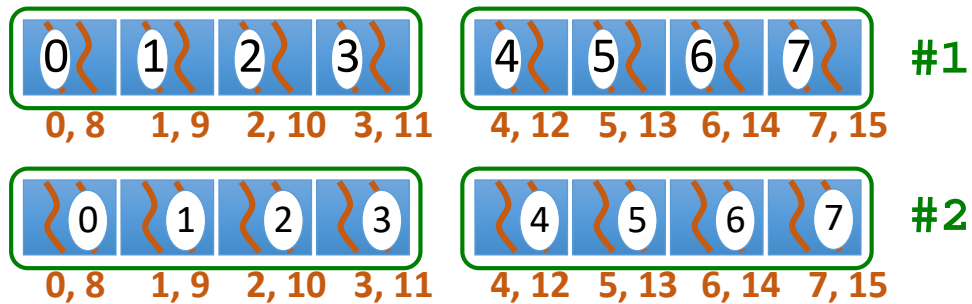
OpenMP Affinity

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

←thread id
←proc-id

⌋ HW-thread

Outline

- Motivation
- Affinity -- what is it
- OpenMP Affinity
- Showing Masks with *amask* Tool
- Hybrid Computing
 - How it works
 - Advancing Standards

Unix Process View

User App

OS Kernel

```
top -load average: 0.77, 0.33, 0.29
Tasks: 584 total, 3 running, 579 sleeping, ...
```

mpirun -np 2 ...
top -- hit 1 key

```
Cpu0 : 99.7%us, 0.3%sy, ... 0.0%id, ...
Cpu1 : 0.0%us, 0.0%sy, ... 100.0%id, ...
Cpu2 : 0.0%us, 0.0%sy, ... 100.0%id, ...
Cpu3 : 0.3%us, 0.3%sy, ... 99.3%id, ...
Cpu4 : 100.0%us, 0.0%sy, ... 0.0%id, ...
Cpu5 : 0.0%us, 0.0%sy, ... 100.0%id, ...
Cpu6 : 0.0%us, 0.0%sy, ... 100.0%id, ...
Cpu7 : 0.0%us, 0.0%sy, ... 100.0%id, ...

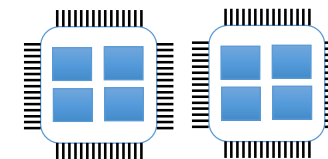
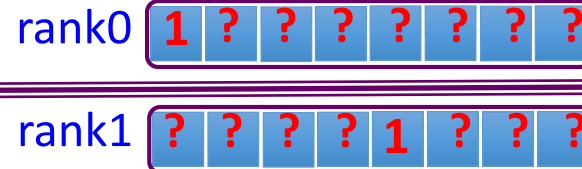
Mem: ... total, ... used, ... free, ...
Swap: ... total, ... used, ... free, ...
```

Running App

```
PID USER ... COMMAND
60673 milfeld ... mpi_affinity
60672 milfeld ... mpi_affinity
58787 milfeld ... top
```

Hardware

N MPI Tasks
M OMP Threads



Viewing Affinity: Intel I_MPI_DEBUG=4

```
KNL IMPI $ export I_MPI_DEBUG=4  
$ mpirun -np 16 my_favorite_app
```

Rank	Node name	Pin cpu
0	c401-001	{0,1,2,3,4,68,69,70,71,136,137,138,139,204,205,206,207}
1	c401-001	{5,6,7,8,72,73,74,75,76,140,141,142,143,208,209,210,211}
2	c401-001	{9,10,11,12,77,78,79,80,144,145,146,147,148,212,213,214,215}
3	c401-001	{13,14,15,16,81,82,83,84,149,150,151,152,216,217,218,219,220}
4	c401-001	{17,18,19,20,21,85,86,87,88,153,154,155,156,221,222,223,224}
5	c401-001	{22,23,24,25,89,90,91,92,93,157,158,159,160,225,226,227,228}
6	c401-001	{26,27,28,29,94,95,96,97,161,162,163,164,165,229,230,231,232}
7	c401-001	{30,31,32,33,98,99,100,101,166,167,168,169,233,234,235,236,237}
8	c401-001	{34,35,36,37,38,102,103,104,105,170,171,172,173,238,239,240,241}
9	c401-001	{39,40,41,42,106,107,108,109,110,174,175,176,177,242,243,244,245}
10	c401-001	{43,44,45,46,111,112,113,114,178,179,180,181,182,246,247,248,249}
11	c401-001	{47,48,49,50,115,116,117,118,183,184,185,186,250,251,252,253,254}
12	c401-001	{51,52,53,54,55,119,120,121,122,187,188,189,190,255,256,257,258}
13	c401-001	{56,57,58,59,123,124,125,126,127,191,192,193,194,259,260,261,262}
14	c401-001	{60,61,62,63,128,129,130,131,195,196,197,198,199,263,264,265,266}
15	c401-001	{64,65,66,67,132,133,134,135,200,201,202,203,267,268,269,270,271}

Viewing Affinity mask with amask

Old Stampede: 16-core

export OMP_NUM_THREADS=8 OMP_PROC_BIND=spread

```
0 proc-id 15
thrd v      v
| 0 10000000000000000
v 1 00100000000000000
  2 00001000000000000
  3 00000010000000000
  4 00000000100000000
  5 00000000001000000
  6 00000000000001000
  7 00000000000000010
```

Bit Mask

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

Even More Readable



How to get masks with amask executables

Pure OpenMP

```
$ export OMP_NUM_THREADS=68  
$ amask_omp # amask executable  
$ my_omp_app
```

Pure MPI

```
$ mpirun -np 4 amask_mpi # amask executable  
$ mpirun -np 4 my_mpi_app
```

Hybrid

```
$ export OMP_NUM_THREADS=16  
$ mpirun -np 17 amask_hybrid # amask executable  
$ mpirun -np 17 my_mpiomp_app
```

What about ... hyper-threading?

- With Hyper-Threading it is more more reasonable to report the SMTs (hardware threads) together.



What about hyper-threading... on KNL

KNL: 68 cores

4 SMT

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

SMT 0
SMT 1
SMT 2
SMT 3



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

Outline

- Motivation
- Affinity -- what is it
- OpenMP Affinity
- Showing Masks with *amask* Tool
- Hybrid Computing
 - How it works
 - Advancing Standards

Hybrid Affinity– MPI + OpenMP

- **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  
  
mpirun ./a.out # 1 thread per tile
```

```
#SLURM -N 2 --tasks-per-node 17  
  
export OMP_NUM_THREADS=2  
  
mpirun ./mpi_script # later
```

MPI mask

This is what IMPI gives on a KNL.

```
$ mpirun -np 2 amask_mpi
```

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



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



```

                                core
rank | 0 | 10 | 20 | 30 | 40 | 50 | 60 |
0000 0123456789012345678901234567890123=====
      0123456789012345678901234567890123-----
      0123456789012345678901234567890123-----
      0123456789012345678901234567890123-----
0001 =====4567890123456789012345678901234567
-----4567890123456789012345678901234567
-----4567890123456789012345678901234567
-----4567890123456789012345678901234567
```

There is NO Standard to set MPI Affinity!

MPI mask

```
$ export OMP_NUM_THREADS=34 OMP_PLACES=cores
```

```
$ mpirun -np 2 amask_hybrid
```

rank	thrd	0	10	20	30	40	50	60	
0000	0000	0	=====						=====
	0	-----							
	0	-----							
	0	-----							
0000	0001	=1	=====						=====
	-1	-----							
	-1	-----							
	-1	-----							
	...								
0000	0033	=====						3	=====
			3	-----					
			3	-----					
			3	-----					
0001	0000	=====						4	=====
			4	-----					
			4	-----					
			4	-----					
	...								
0001	0033	=====						7	=====
						7	-----		
						7	-----		
						7	-----		

rank 0

Rank 1

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

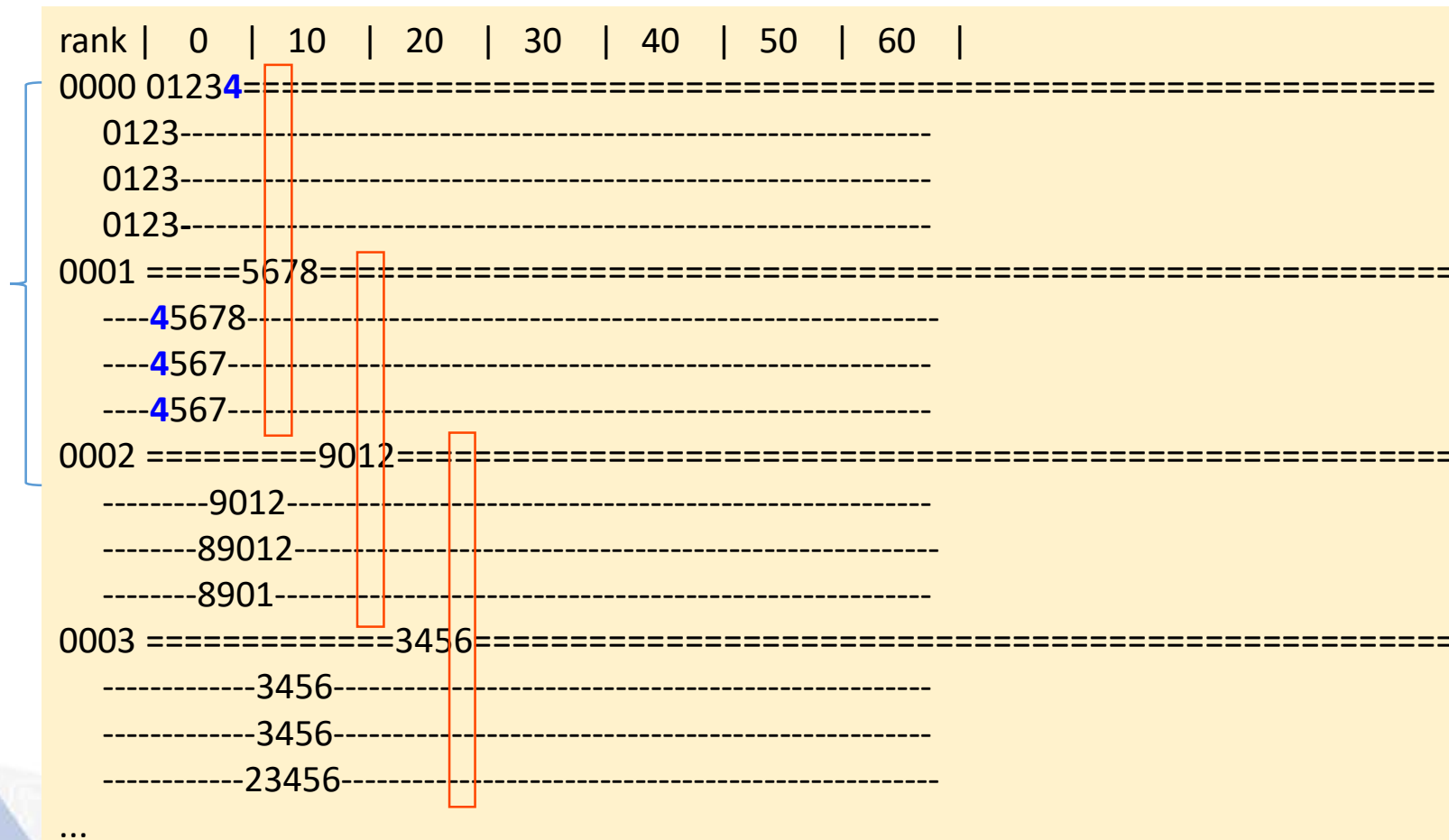
Thread Team inherits mask of MPI process Rank 1-- and adjusts for each thread



An Interesting Case (16x17)

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

Core sharing by two MPI processes



Potato (16x17) or Potahito (17x16)

```
$ 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-----
```

SNC-4 mode (4 memory numa nodes)

```
$ mpirun -np 4 amask_mpi # MPI mask
```

NOTE THE
DISTRIBUTION
Should be:
18 x 18 x 16 x 16
Cores according
to lscpu

```
rank | 0 | 10 | 20 | 30 | 40 | 50 | 60 |
0000 01234567890123456=====
1 01234567890123456-----
01234567890123456-----
01234567890123456-----
0001 =====78901234567890123=====
-----78901234567890123-----
-----78901234567890123-----
-----78901234567890123-----
0002 =====45678901234567890=====
-----45678901234567890-----
-----45678901234567890-----
-----45678901234567890-----
0003 =====12345678901234567
-----12345678901234567
-----12345678901234567
-----12345678901234567
```

Hybrid Affinity-script

```
#SLURM -N 2 --tasks-per-node 4  
# uses numa "aware" script  
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-script -- how ugly is this?

```
#!/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
```

```
amask_hybrid #test echo `hostname` $R $OMP_PLACES
```

Perspective, Challenges and Summary:

- User and Developers should be aware of Affinity Control (Capability)
- Affinity Control Needs to be Standardized:
 - Threads: OpenMP PLACES syntax is simple and handles granularity
 - MPI needs to standardize MPI Affinity
 - Needed:
 - Common Affinity Language/Syntax for MPI and OpenMP
 - Interface for OpenMP to acquire basic MPI information
- Affinity Viewers need to include hardware information in reports



TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU



TEXAS

The University of Texas at Austin

**Thanks for your attention!
Questions?**

An Example

TACC KNL: 4 SMTs x 68 cores = 272 hardware threads

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

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

rank	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250	260	270										
0000	01234						8901							6789													4567										
0001		5678						23456							0123													8901									
0002			9012						7890							45678												2345									
0003				3456						1234							9012												67890								
0004					78901						5678							3456											1234								
0005						2345						90123							7890										5678								
0006							6789						4567							12345									9012								
0007								0123						8901							6789									34567							
0008									45678												0123									8901							
0009										9012												67890									4567						
0010											3456																				89012						
0011												1234																				6789					
0012																																	3456				
																																		16-Feb-18			
																																		01234			
																																			40		
																																				5678	

Not a monospace font.

17 masked bits in each row → makes sense, since 16 tasks x 17 threads = 272 → “great” for hybrid.



A closer look:

```
rank | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 ... | 270
0000 01234-----8901----- ...
0001 ----5678-----23456- ...
0002 -----9012-----7 ...
...
.
.
.
0016 -----4567----- ...
```

Maybe a core listing would be insightful?