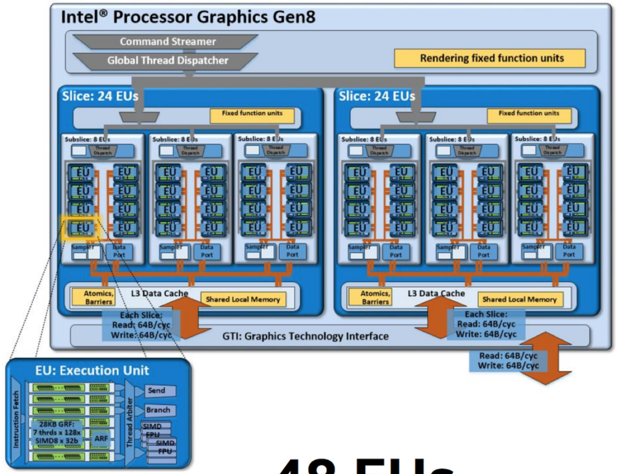
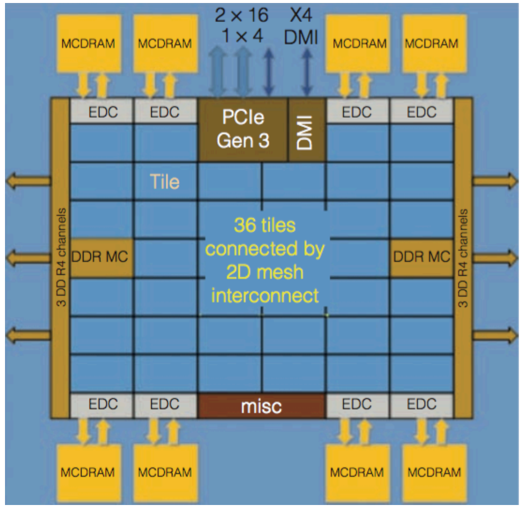
LAB
OpenMP Affinity



48 EUs



PRESENTED BY:

Lars Koesterke

Kent Milfeld

Logon to the Stampede KNL system

Users with Stampede2 login (web/on-site)	On-site "Walk-in"

- Please fill out login sheet – return lower half
- <username> = <trainXXX>

Login to Stampede2

- `$ ssh <username>@stampede2.tacc.utexas.edu`

- password: portal password
- TACC Token Code: TACC token App

- password: login sheet
- TACC Token Code: blackboard

Prep

- Untar labs:

```
$ tar -xvf ~train00/adv_omp_labs.tar
$ cd adv_affinity
```

- You are now on the login node

  This node is NOT a KNL, but a 24 core Broadwell system

- Access a KNL node:

```
$ idev -m 60          # 60 min.   Use -h for options
...
cxxx-yyy[knl] $ module load amask
```

Login again in another window

1.) In another terminal window on your laptop login to stampede2 again.

2.) ssh to the idev knl compute node (note the c###-### prompt in your other window)

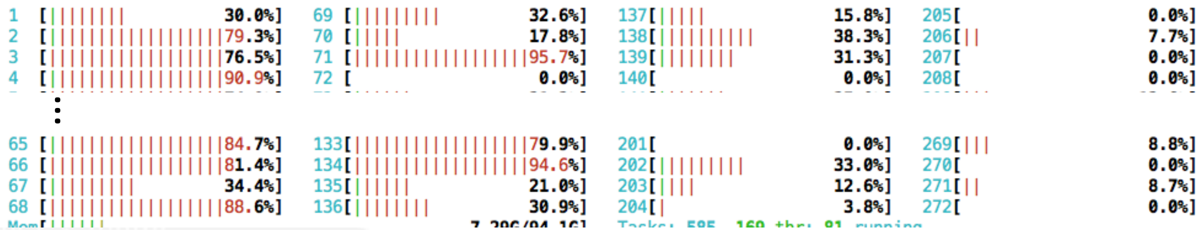
3.) run top so that you can watch the cpu loads.

```
$ ssh <username>@stampede2.tacc.utexas.edu

$ ssh c###-###     #we call this the ssh/top window

$ top
```

Hit the “1” key, and
Adjust the screen size/font so you can see 137 cpus, or at least 69.

OR $ export PATH=~train00/bin:$PATH

```
$ htop
```



What you will do

Play with different settings for OMP_PLACES and OMP_PROC_BIND

- On the Stampede KNL system
- To understand how to use all cores and/or hardware-threads
  - In pure OpenMP mode
  - In hybrid mode (OpenMP + MPI)
- We will use special tools developed by TACC (https://github.com/tacc/amask)
  - ‘Dummy’ executable
  - Creating a condensed output to visualize binding mask of each thread
  - Name of the tools:
    - **`amask_omp`**
    - **`amask_hybrid`**

amask_omp

Dummy executable **`amask_omp`**: usage and example output

- Execute with: **`amask_omp -vk -w <time in seconds>`**
- Example: **`amask_omp -vk –w 1`**
  - Execute for 1 second (watch top) with number of threads and thread placement taken from
    - **`OMP_NUM_THREADS`**, **`OMP_PLACES`**, and **`OMP_PROC_BIND`**
  - Prints a list of potential places for all threads
- Example output (shortened): amask_omp -vk | cut -c 1-61

```
 Each row of matrix is an Affinity mask. A set mask bit
thrd |         |   10    |   20    |   30    |   40    |   50
0000 0-------------------------------------------------------
0001 -1------------------------------------------------------
0002 ---3----------------------------------------------------
0003 -------7--------------2---------------------------------
0004 ------6-8---2-------------------------------------------
```

Thread 0 on core 0
Thread 1 on core 1
Thread 2 on core 3
Thread 3 on cores 7, 22
Thread 4 on cores 6, 8, 12

Exercise 1

Binding policies **`close`** and **`spread`**

- **`cd ex1`**
- **`readme.txt`** contains commands and instructions
  - You may cut-and-paste from the readme file or execute: `source sourceme`
- **`export OMP_NUM_THREADS=4`**
- **`export OMP_PLACES='{0},{4},{8},{12},{16},{20},{24},{28}'`**
- **`export OMP_PROC_BIND=close`**
- Run: **`amask_omp -vk`** #see readme.txt --> -vk views the true kernel mask

- Change the bind-policy to spread
- Rerun experiment

TACC

Exercise 1: Results

Dummy executable **`amask_omp -vk`**

First experiment: Binding = 'close'

```
thrd |         |   10    |   20    |   30    |   40    |   50
0000 0-------------------------------------------------------
0001 ----4---------------------------------------------------
0002 --------8-----------------------------------------------
0003 ------------2-------------------------------------------
```

Second experiment: Binding = 'spread'

```
thrd |         |   10    |   20    |   30    |   40    |   50
0000 0-------------------------------------------------------
0001 --------8-----------------------------------------------
0002 ----------------6---------------------------------------
0003 ------------------------4-------------------------------
```

Exercise 1: Confirmation with top | htop

Confirm thread location with top command

- Find idev host name: prompt has the compute node (cxxx-xxx), or use ***hostname*** cmd
- Open a second window
- ssh into the idev compute node: **`ssh c???-???`**
- Make the window really long (many lines)
- Type **`top`**, and press the **`1`** button (this will show the load on each HW thread)
- Rerun the 2 experiments in the original idev session for a longer time
- Example: **`amask_omp -kv -w 60 | cut -c 1-61`**
- Observe the load in the 'top' window
- Even better: use ~train00/bin/htop (htop labels proc-id 0-271 as 1-272 ☹)

Exercise 2

Places with 2 hardware threads -- in the “core view”

- **`cd ex2`**
- **`readme.txt`** contains commands and instructions
  - You may cut-and-paste from the Readme file
- **`export OMP_NUM_THREADS=8`**
- **`export OMP_PLACES=`**`'{0,68},{1,69},{2,70},{3,71},{4,72},{5,73},{6,74},{7,75}'`
- Run: **`amask_omp –vk | cut -c 1-120`**
- Run: **`amask_omp`** #core view, no –ls, no need for cut !!!!

- Change the number of threads to **`OMP_NUM_THREADS=16`**
- Rerun experiment. Are the places now oversubscribed.

amask_omp -vk   Exercise 2: Results

8 threads
8 places
Each place (core) with
2 SMTs (HW thrds)

```
thrd |    0    |   10    |   20    |   30    |   40    |   50    |   60    |   70
0000 0-------------------------------------------------------------------8-----------
0001 -1-------------------------------------------------------------------9----------
0002 --2-------------------------------------------------------------------0---------
0003 ---3-------------------------------------------------------------------1--------
0004 ----4-------------------------------------------------------------------2-------
0005 -----5-------------------------------------------------------------------3------
0006 ------6-------------------------------------------------------------------4-----
0007 -------7-------------------------------------------------------------------5----
```

16 threads
8 places
Each place (core) with
2 SMTs (HW thrds)

```
thrd |    0    |   10    |   20    |   30    |   40    |   50    |   60    |   70
0000 0-------------------------------------------------------------------8-----------
0001 0-------------------------------------------------------------------8-----------
0002 -1-------------------------------------------------------------------9----------
0003 -1-------------------------------------------------------------------9----------
0004 --2-------------------------------------------------------------------0---------
0005 --2-------------------------------------------------------------------0---------
0006 ---3-------------------------------------------------------------------1--------
0007 ---3-------------------------------------------------------------------1--------
0008 ----4-------------------------------------------------------------------2-------
0009 ----4-------------------------------------------------------------------2-------
0010 -----5-------------------------------------------------------------------3------
0011 -----5-------------------------------------------------------------------3------
0012 ------6-------------------------------------------------------------------4-----
0013 ------6-------------------------------------------------------------------4-----
0014 -------7-------------------------------------------------------------------5----
0015 -------7-------------------------------------------------------------------5----
```

Exercise 2:

`amask_omp`

8 threads
8 places
Each place (core) with
2 SMTs (HW thrds)

```
CORE ID  = matrix digit + column group # in |...|

thrd |    0    |   10    |   20    |   30    |   40    |   50    |   60    |
0000 0===============================================================
     0---------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
0001 =1==============================================================
     -1--------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
0002 ==2=============================================================
     --2-------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
...
```

```
CORE ID = matrix digit + column group # in |...

thrd |    0    |   10    |   20    |   30    |
0000 0===============================================================
     0---------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
0001 0===============================================================
     0---------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
0002 =1==============================================================
     -1--------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
0003 =1==============================================================
     -1--------------------------------------------------------------
     ----------------------------------------------------------------
     ----------------------------------------------------------------
...
```

16 threads
8 places
Each place (core) with
2 SMTs (HW thrds)

Exercise 3

Using all cores: affinity with one and multiple SMTs on each core.

- **`cd ex3`**
- **`readme.txt`** contains commands and instructions
  - You may cut-and-paste from the readme file
- **`export OMP_NUM_THREADS=68`**
- **`export OMP_PLACES=cores`**
- Run: **`amask_omp`**

- Change the abstract name to **threads**.
- Rerun experiment – which SMT is occupied?

Exercise 4

Hybrid computing

- **`cd ex4`**
- Exit your current idev session and start a new one with:
- **`idev -n 4 -N 1 -m 10`** (single node, 4 MPI tasks)
- **`readme.txt`** contains commands and instructions
  - You may cut-and-paste from the Readme file
- **`ml amask`** `#ml == module load`
- **`export OMP_NUM_THREADS=17 OMP_PLACES=cores`**
- Run: **`ibrun amask_hybrid`**
- Check it out with the kernel view: **ibrun `amask_hybrid -vk`**

- Change the number of threads to 34, then 8 and rerun
- What is the distribution of threads within the MPI mask for 8 threads?
- Set OMP_PROC_BIND=close and rerun for 8 threads. What is the distribution now?

Exercise 4: Results

```
ibrun amask_hybrid -vk | cut -c 1-80
```

4 MPI tasks; 17 threads

OMP_NUM_THREADS=16

OMP_PROC_BIND=cores

(16 places per rank)

```
  Each row of matrix is an Affinity mask. A set mask bit
rank |    0    |    10    |    20    |    30    |    40    |    5
0000 01234567890123456--------------------------------------
0001 -----------------78901234567890123---------------------
0002 ----------------------------------45678901234567890----
0003 ---------------------------------------------------1234

      Each row of matrix is an Affinity mask. A set mask bit
rank thrd |    0    |    10    |    20    |    30    |    40
0000 0000 0-------------------------------------------------
0000 0001 -1------------------------------------------------
0000 0002 --2-----------------------------------------------
0000 0003 ---3----------------------------------------------
```

34 threads and 16 places

```
rank thrd |    0    |    10    |    20    |    30    |    40
0000 0000 0-------------------------------------------------
0000 0001 0-------------------------------------------------
0000 0002 -1------------------------------------------------
0000 0003 -1------------------------------------------------
0000 0004 --2-----------------------------------------------
0000 0005 --2-----------------------------------------------
0000 0006 ---3----------------------------------------------
0000 0007 ---3----------------------------------------------
```