Expanse:SLURM Runtime Configurations and Tuning
April 21, 2021

Mahidhar Tatineni
Manu Shantharam
SDSC

EXPANSE
COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER

NSF Award 1928224

Outline

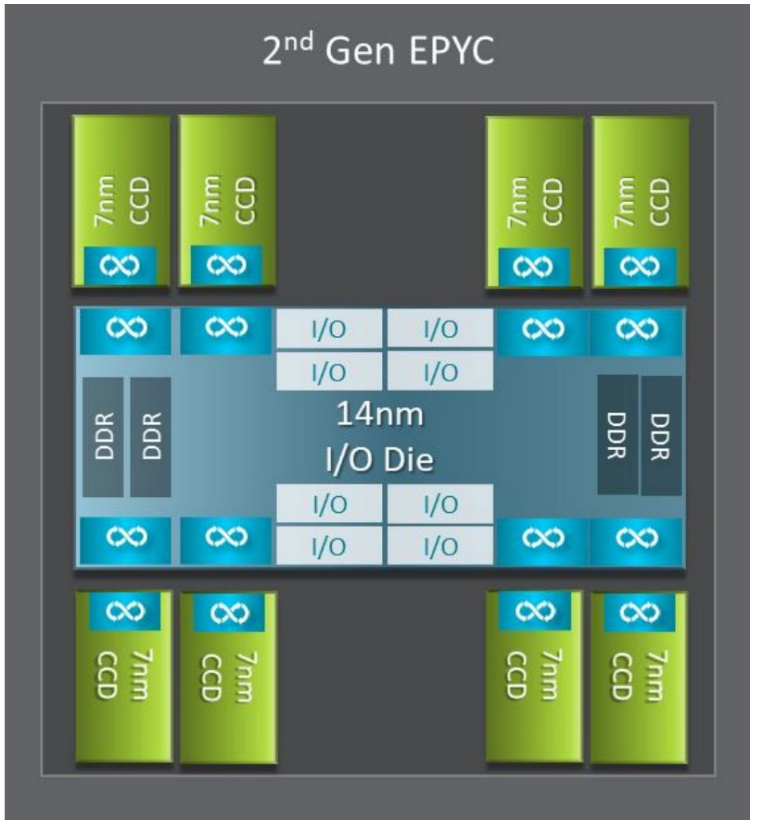
- AMD EPYC Processor Architecture
  - Hardware details
  - NUMA options
  - Applications
- Expanse task layout and affinity scripts
  - ibrun
  - affinity
  - slurm-aff-prod (for MPI/pthreads)
- Summary

Outline

- AMD EPYC Processor Architecture
  - Hardware details
  - NUMA options
  - Applications
- Expanse task layout and affinity scripts
  - ibrun
  - affinity
  - slurm-aff-prod (for MPI/pthreads)
- Summary

AMD EPYC 7742 Processor Architecture

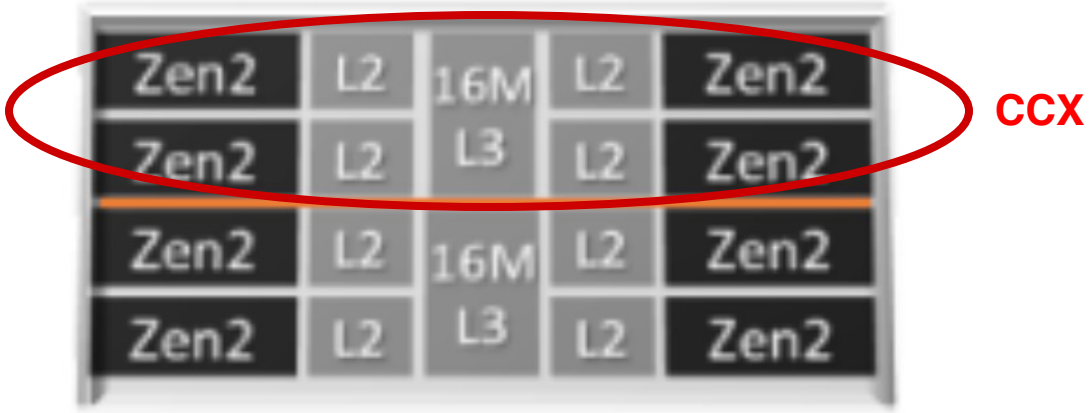
- 8 Core Complex Dies (CCDs).
- CCDs connect to memory, I/O, and each other through the I/O Die.
- 8 memory channels per socket.
- DDR4 memory at 3200MHz.
- PCI Gen4, up to 128 lanes of high speed I/O.
- Memory and I/O can be abstracted into separate quadrants each with 2 DIMM channels and 32 I/O lanes.



Reference: https://developer.amd.com/wp-content/resources/56827-1-0.pdf

AMD EPYC 7742 Processor: Core Complex Die (CCD)

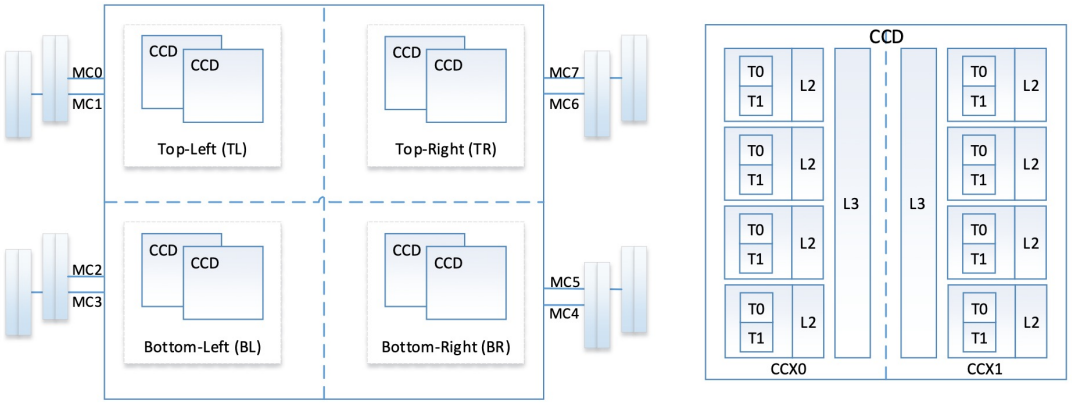
- 2 Core Complexes (CCXs) per CCD
- 4 Zen2 cores in each CCX shared a 16M L3 cache. Total of 16 x 16 = 256MB L3 cache.
- Each core includes a private 512KB L2 cache.



Reference: https://developer.amd.com/wp-content/resources/56827-1-0.pdf

AMD EPYC 7742 Processor : NUMA Nodes Per Socket

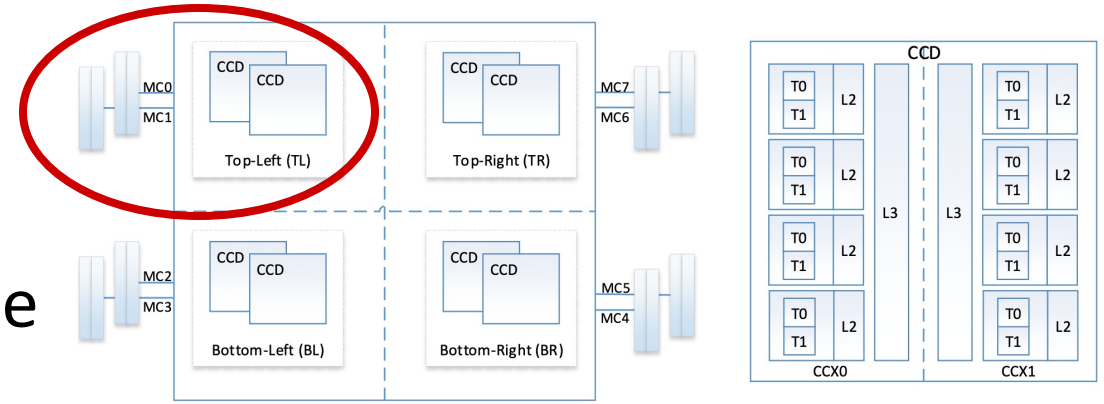
- The four logical quadrants allow the processor to be partitioned into different NUMA domains. Options set in BIOS.
- Domains are designated as NUMA per socket (NPS).
- **NPS4:** Four NUMA domains per socket is the typical HPC configuration.



https://developer.amd.com/wp-content/resources/56338_1.00_pub.pdf

NPS4 Configuration

- The processor is partitioned into four NUMA domains.
- Each logical quadrant is a NUMA domain.
- Memory is interleaved across the two memory channels
- PCIe devices will be local to one of four NUMA domains (the IO die that has the PCIe root for the device)
- ***This is the typical HPC configuration*** as workload is NUMA aware, ranks and memory can be pinned to cores and NUMA nodes.



https://developer.amd.com/wp-content/resources/56338_1.00_pub.pdf

Outline

- AMD EPYC Processor Architecture
  - Hardware details
  - NUMA options
  - Applications
- Expanse task layout and affinity scripts
  - ibrun
  - affinity
  - slurm-aff-prod (for MPI/pthreads)
- Summary

Using MPI options

- All MPI implementations have affinity options.
- Example OpenMPI run command:

  mpirun -np 32 --mca pml ucx --mca osc ucx --map-by l3cache xhpl

- Example Intel MPI setup:

  export OMP_NUM_THREADS=16

  mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./hello_hybrid

- Can also combine with application pinning options. For example for NAMD:

  mpirun -np 8 --map-by ppr:4:node namd2 +setcpuaffinity +ppn 31 +commap 0,32,64,96 +pemap 1-31,33-63,65-95,97-127 stmv.namd

ibrun and affinity options

- Basic usage
  - ibrun ./executable <executable_opions>
- With affinity
  - ibrun affinity <hints> ./executable <executable_opions>
- Affinity options
  - **scatter**: scatters the ranks across all numa domains in a cyclic manner
  - **scatter-ccd**: scatters the ranks across all AMD CCD domains in a cyclic manner
  - **scatter-ccx**: scatters the ranks across all AMD CCX domains in a cyclic manner
  - **scatter blk <blk_size>**: scatters the ranks across all numa domains in a cyclic manner, but with 'blk_size' (1-16) consecutive ranks packed into a single numa domain
  - **scatter-ccd blk <blk_size>**: scatters the ranks across AMD CCD domains in a cyclic manner, but with 'blk_size' (1-8) consecutive ranks packed into a single CCD domain
  - **scatter-ccx blk <blk_size>**: scatters the ranks across AMD CCX domains in a cyclic manner, but with 'blk_size' (1-4) consecutive ranks packed into a single CCX domain

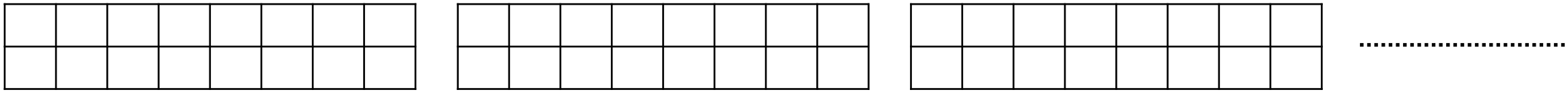
NOTE: valid blk_sizes depend on the **cpus-per-task a**nd the **domain type** (numa, CCD, CCX). 'blk' is optional and is set to '1' by default

Guide for Layout Diagrams (for upcoming slides)

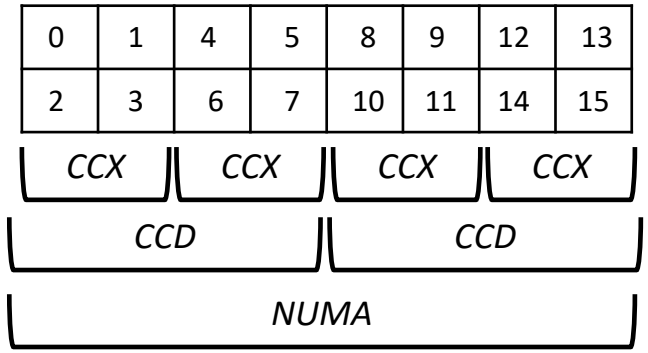
NUMA Domain 1

NUMA Domain 2

NUMA Domain 3

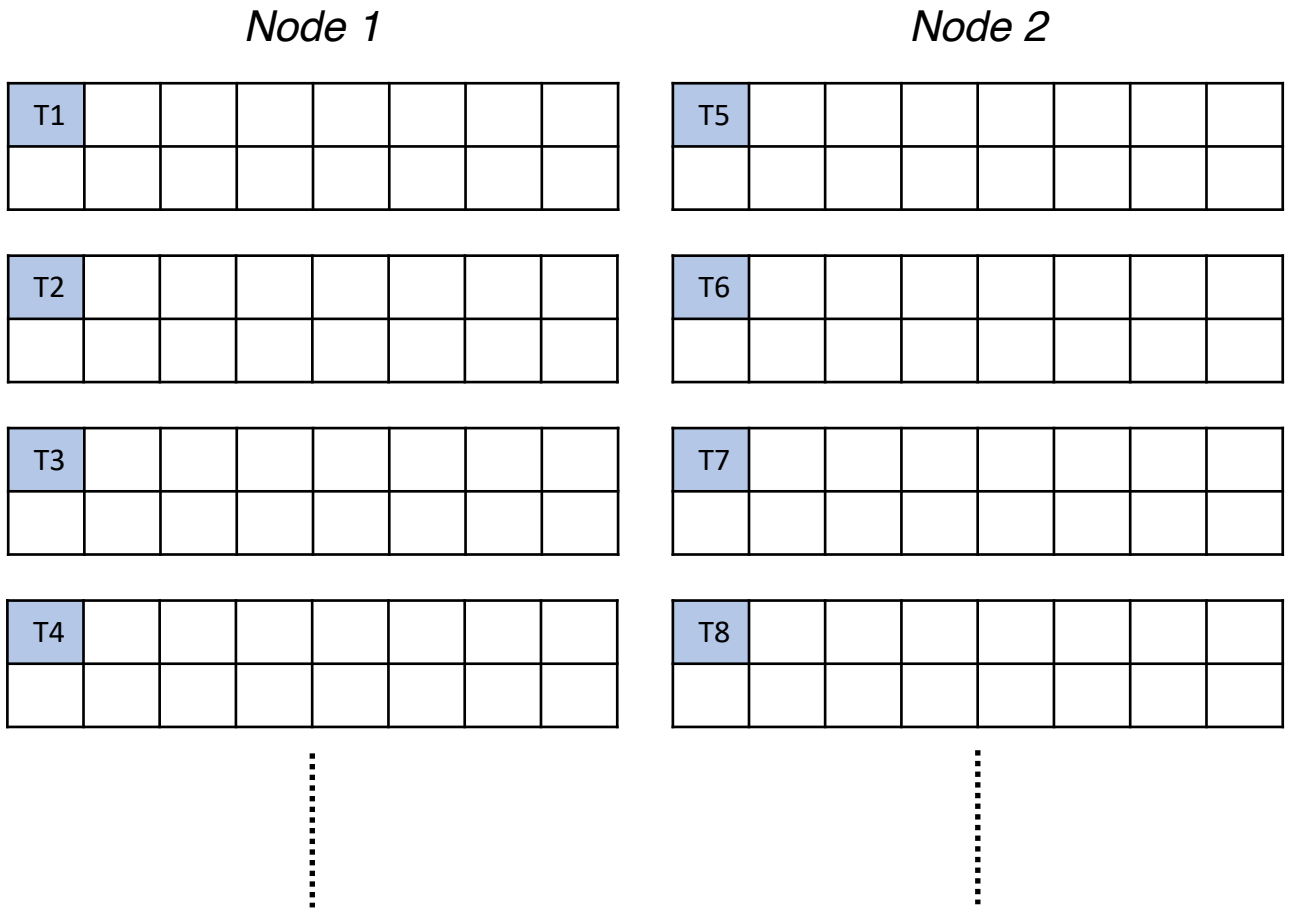


.....



Example ibrun options and layouts

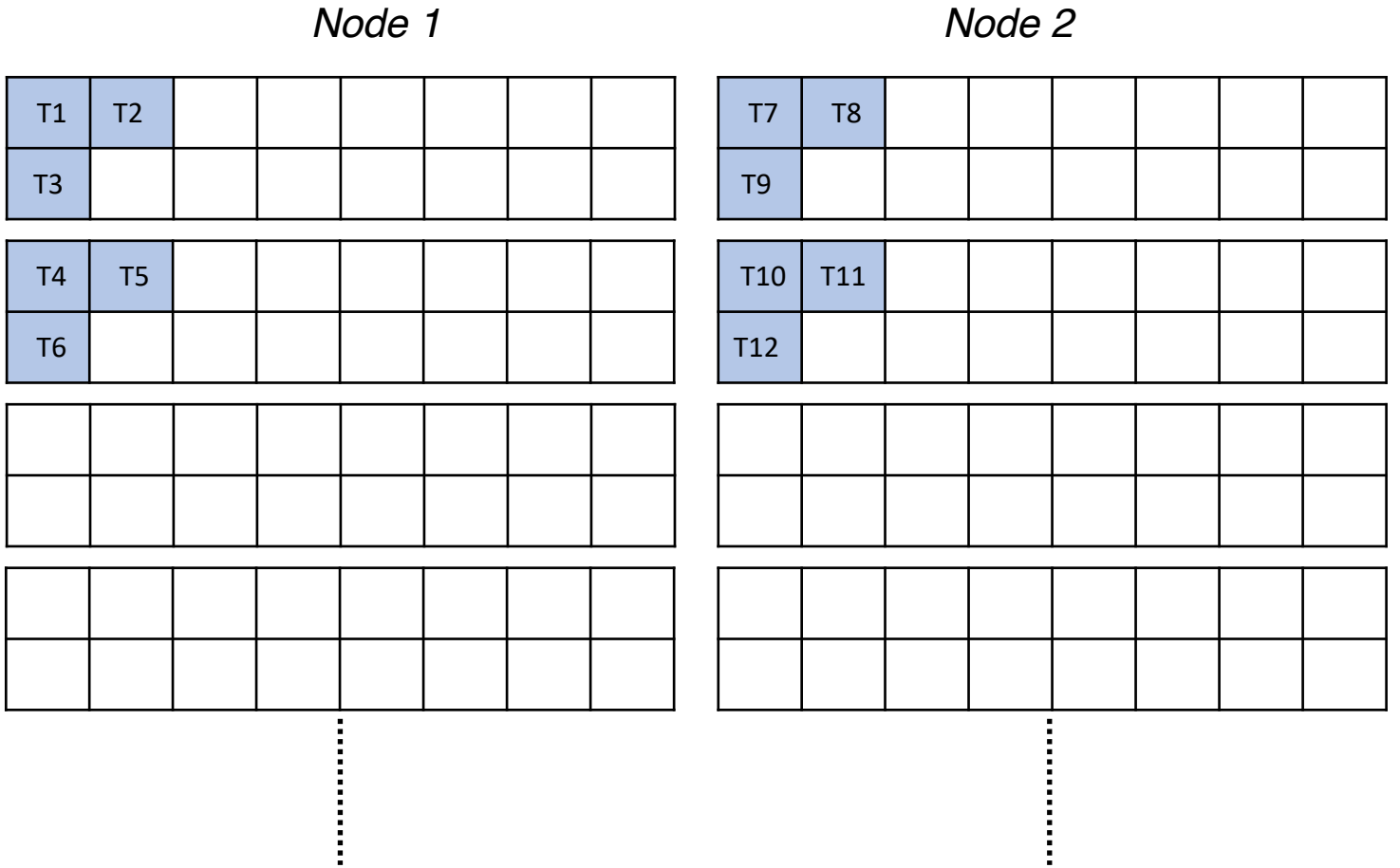
```
#!/bin/bash
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -A <ACCT>
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=4
#SBATCH -t 00:20:00

### Expanse modules
module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4

ibrun ./hy-gcc-openmpi.exe

(same as srun –n 8 ./hy-gcc-openmpi.exe)
```



Example ibrun options and layouts

```
#!/bin/bash
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -A <ACCT>
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=6
#SBATCH -t 00:20:00

### Expanse modules
module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4
```

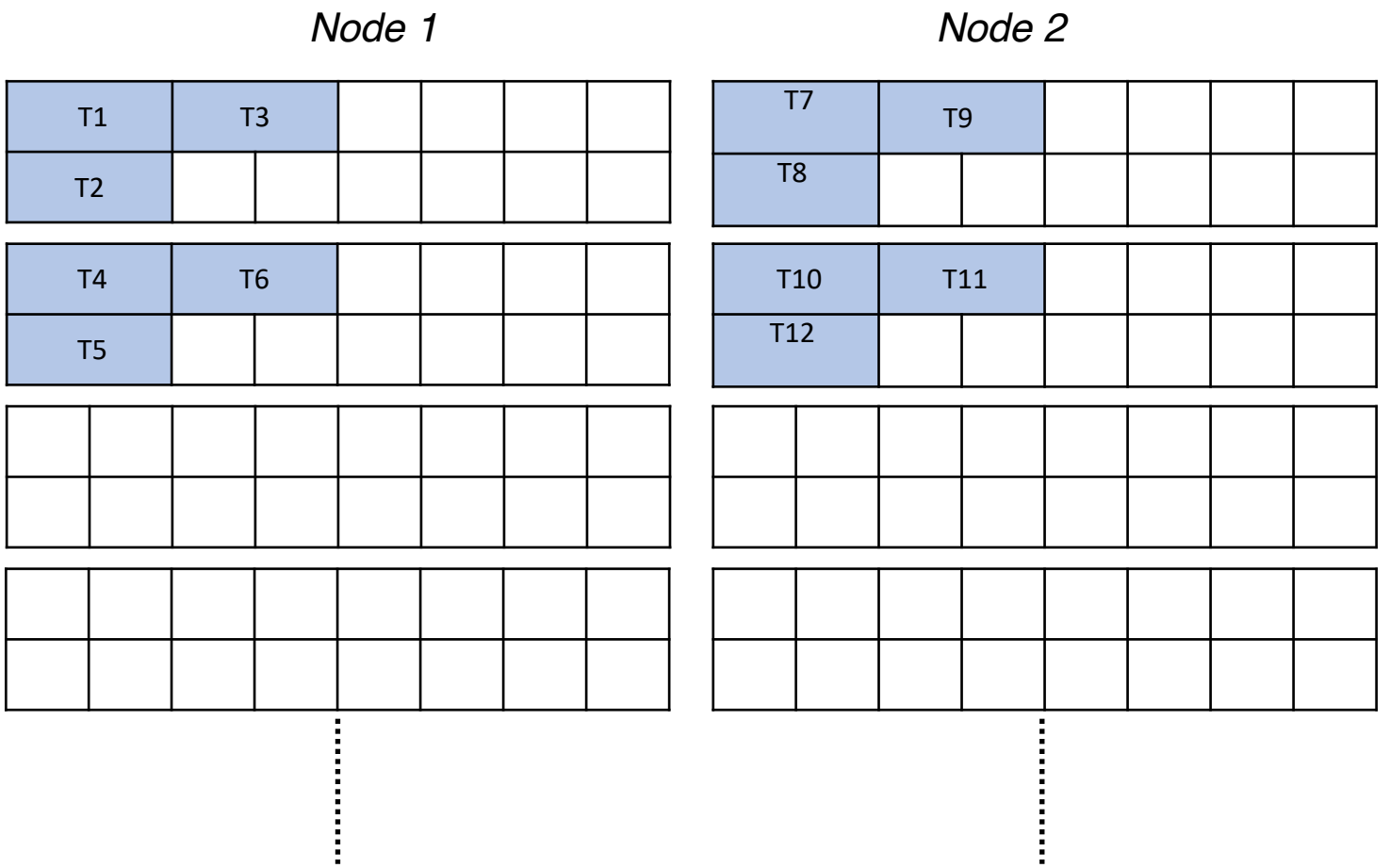
ibrun affinity scatter blk 3 ./hy-gcc-openmpi.exe



Example ibrun options and layouts

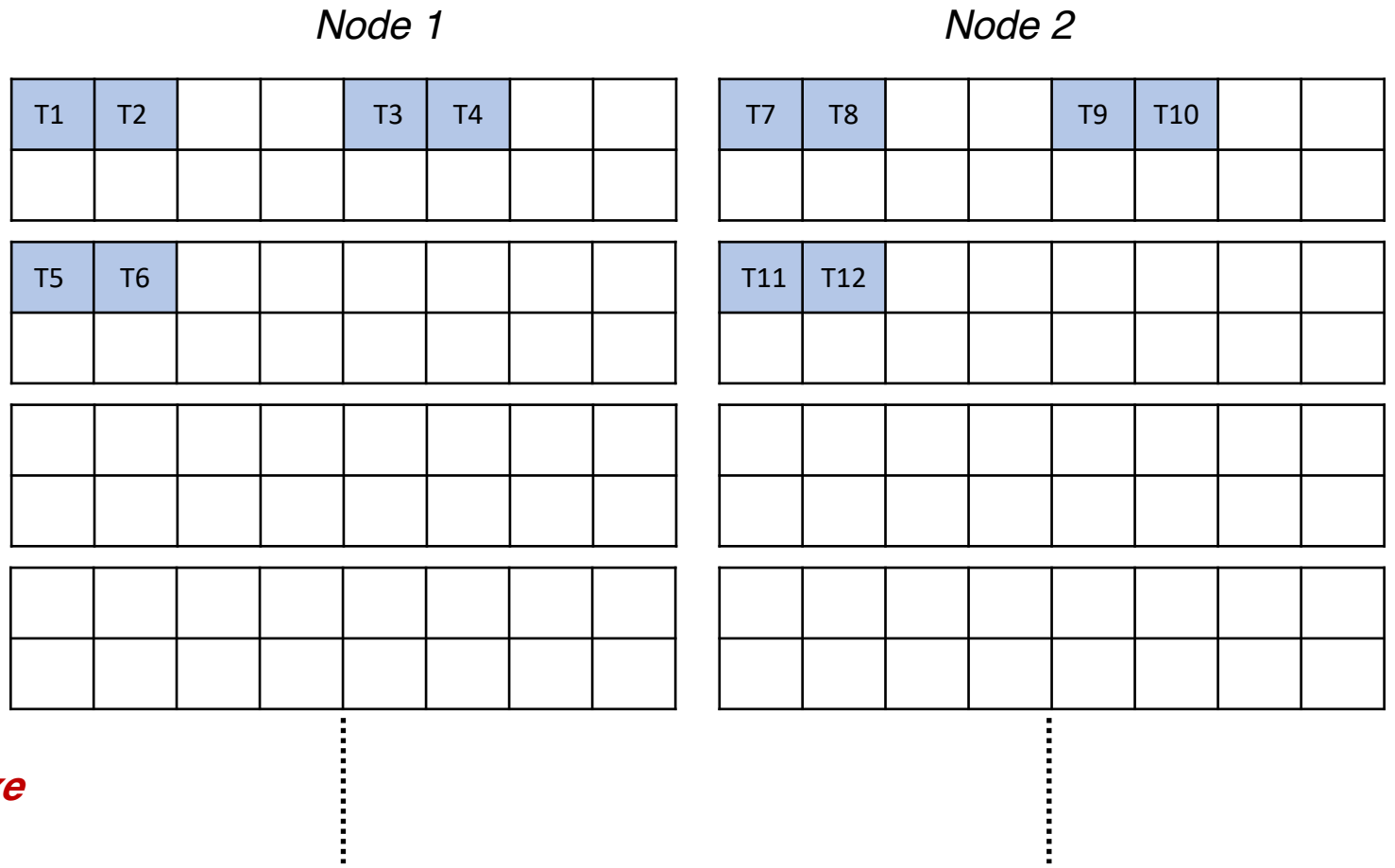
```
#!/bin/bash
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -A <ACCT>
#SBATCH --cpus-per-task=2
#SBATCH --ntasks-per-node=6
#SBATCH -t 00:20:00

### Expanse modules
module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4
```

ibrun affinity scatter blk 3 ./hy-gcc-openmpi.exe



Example ibrun options and layouts

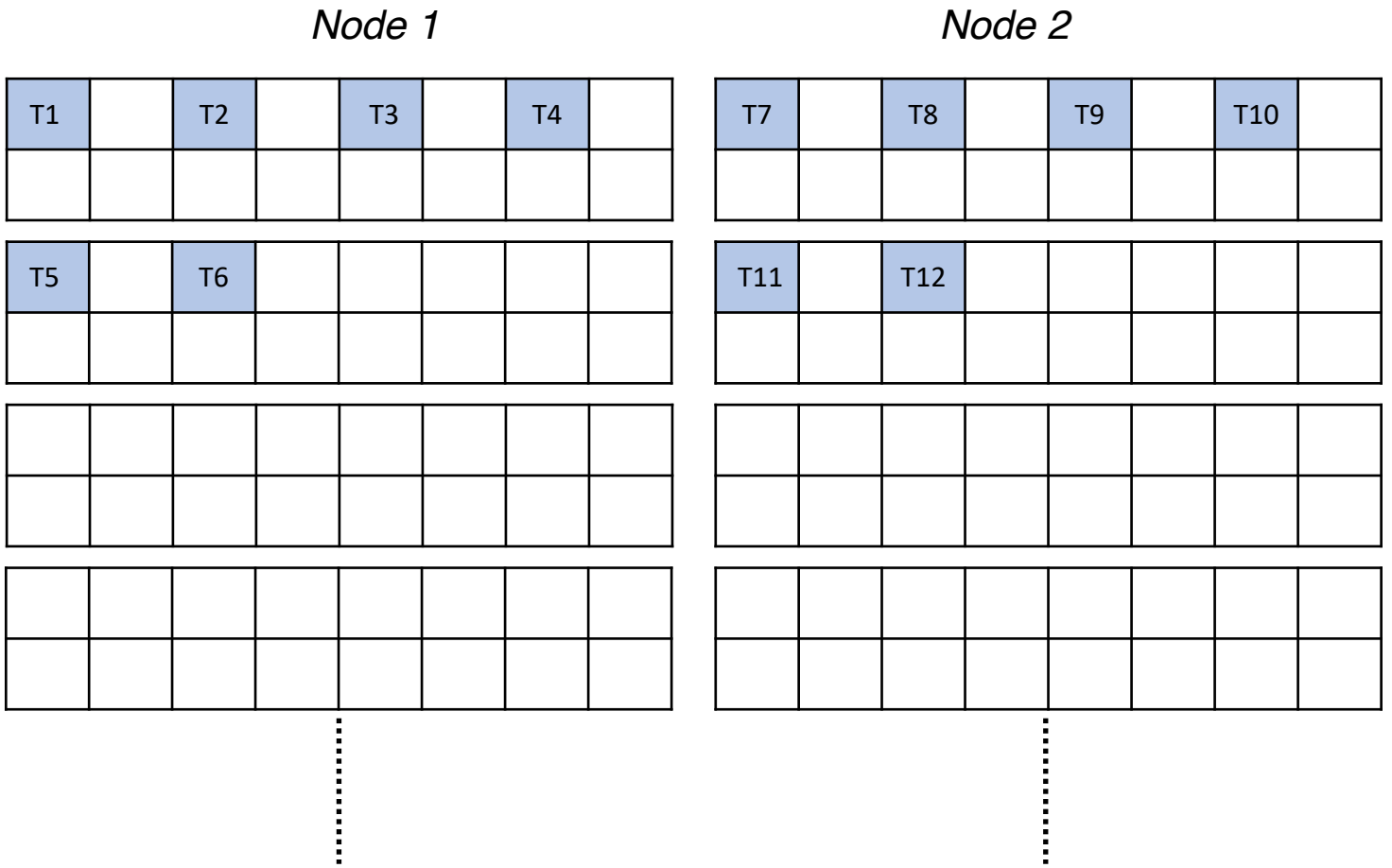
```
#!/bin/bash
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -A <ACCT>
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=6
#SBATCH -t 00:20:00

### Expanse modules
module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4
```



ibrun affinity scatter-ccd blk 2 ./hy-gcc-openmpi.exe

Example ibrun options and layouts

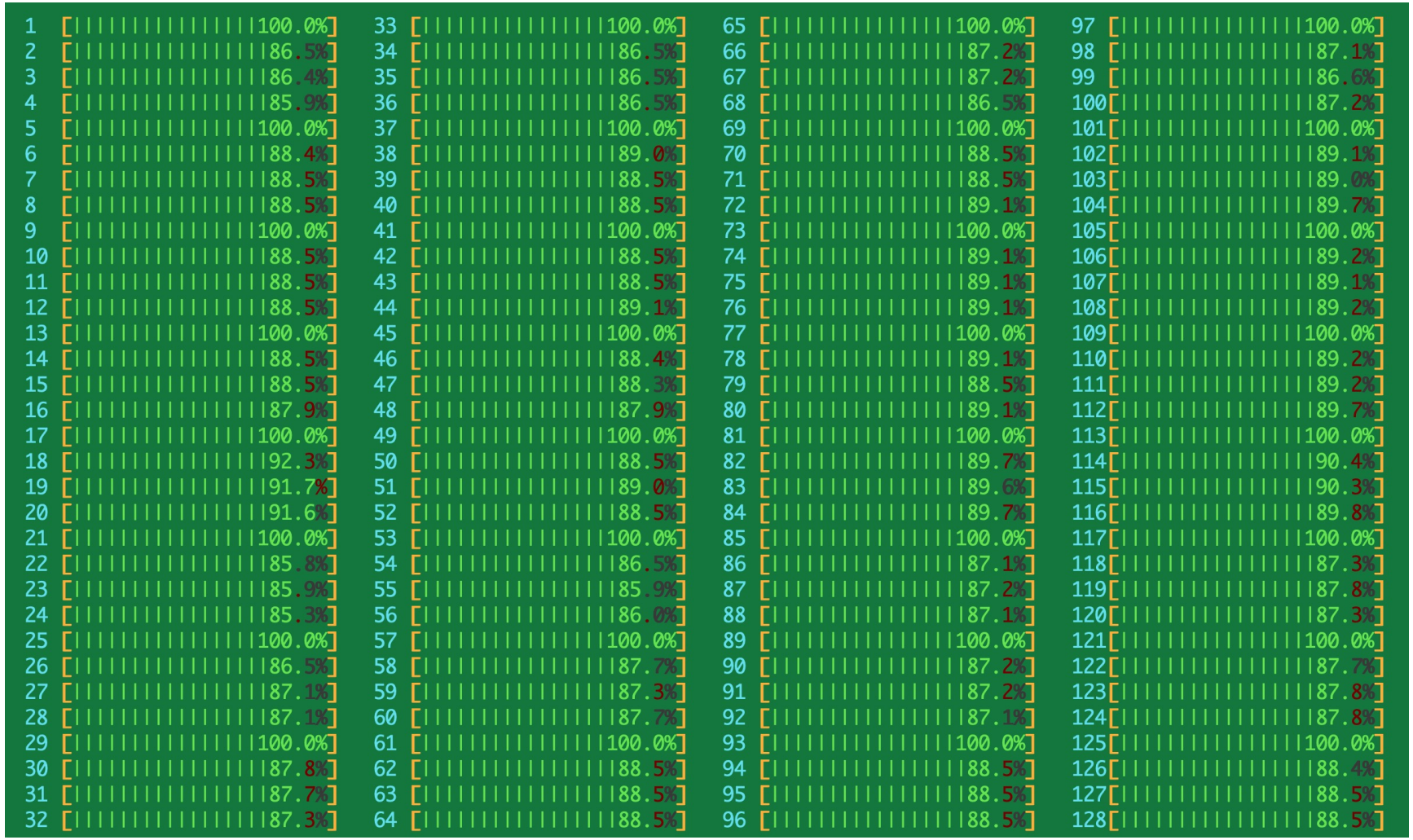
```
#!/bin/bash
#SBATCH -p compute
#SBATCH -N 2
#SBATCH -A <ACCT>
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=6
#SBATCH -t 00:20:00

### Expanse modules
module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4

ibrun affinity scatter-ccx ./hy-gcc-openmpi.exe
```



Snapshot of task layout with scatter-ccx option

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4

ibrun affinity scatter-ccx $XHPL



Snapshot of task layout with scatter-ccx option

#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4

ibrun affinity scatter-ccx $XHPL

```
82667 xhpl            mahidhar    2526976 R 98.4 00:02:10   0
82667 xhpl            mahidhar    2526976 S  0.0 00:00:00   0
82667 xhpl            mahidhar    2526976 S  0.8 00:00:01   0
82667 xhpl            mahidhar    2526976 R 87.9 00:01:49   1
82667 xhpl            mahidhar    2526976 R 87.9 00:01:49   2
82667 xhpl            mahidhar    2526976 R 87.9 00:01:49   3
82668 xhpl            mahidhar    2527544 R 98.4 00:02:09  40
82668 xhpl            mahidhar    2527544 S  0.0 00:00:00  40
82668 xhpl            mahidhar    2527544 S  0.9 00:00:01  40
82668 xhpl            mahidhar    2527544 R 88.3 00:01:49  41
82668 xhpl            mahidhar    2527544 R 88.3 00:01:49  42
82668 xhpl            mahidhar    2527544 R 88.3 00:01:49  43
82669 xhpl            mahidhar    2527532 R 98.3 00:02:09   4
82669 xhpl            mahidhar    2527532 S  0.0 00:00:00   4
82669 xhpl            mahidhar    2527532 S  0.7 00:00:00   4
82669 xhpl            mahidhar    2527532 R 87.7 00:01:48   5
82669 xhpl            mahidhar    2527532 R 87.7 00:01:48   6
82669 xhpl            mahidhar    2527532 R 87.7 00:01:48   7
```

slurm-aff-prod script for MPI/Pthreads codes

- The **slurm-aff-prod** script is used to bind Pthreads after MPI job launch. Can be used by wrapping binary based on name.

```
#!/bin/bash
#SBATCH -p shared
#SBATCH -N 1
#SBATCH --ntasks-per-node=10
#SBATCH --cpus-per-task=4
#SBATCH --mem=77G
#SBATCH -t 00:10:00
#SBATCH -J A76BE.GTRGAMMA.mpi10pt4NautoMRExfa
#SBATCH -o A76BE.GTRGAMMA.mpi10pt4NautoMRExfa.%j.%N.out
#SBATCH -e A76BE.GTRGAMMA.mpi10pt4NautoMRExfa.%j.%N.err
#SBATCH -A use300

export NP=$SLURM_TASKS_PER_NODE
export THREADS=$SLURM_CPUS_PER_TASK

rm RAx*

export AFFINITY_INFO=0
export AFFINITY_DEBUG=0

srun --mpi=pmi2 -n $NP ./raxmlHPC-HYBRID_8.2.12_expanse -s ./A76BE.txt -n A76BE.GTRGAMMA.mpi10pt4NautoMRExfa -m GTRGAMMA -N a
utoMRE -p 12345 -x 12345 -f a
```

```
#!/bin/sh

# This is for running on EXPANSE

#source $HOME/.bashrc

module reset
module load sdsc
module load gcc/10.2.0
module load openmpi/4.0.4
module load raxml/8.2.12
module load slurm

EXE='raxmlHPC-HYBRID-AVX'
slurm-aff-prod-test $EXE &

echo "running:"
echo " $EXE -T ${THREADS} $*"

$EXE -T ${THREADS} $*
```

Summary

- AMD Processor on Expanse has 4 NUMA domains with 16 cores each.
- 8 Core Complex Dies (CCDs) per processor, with 2 Core Complexes (CCXs) per CCD. Four cores in a CCX share L3 cache.
- For hybrid MPI/OpenMP and MPI/Pthreads codes it is important to lay out tasks correctly and binding is important for performance.
- **ibrun, affinity, and slurm-aff-prod** scripts available to make it easier to lay out and bind tasks.
- Follow all things Expanse at https://www.sdsc.edu/support/user_guides/expanse.html
- Tools are being updated so feedback is encouraged!

Thank you to our collaborators, partners, users, and the SDSC team!

Ilkay Altintas
Haisong Cai
Amit Chourasia
Trevor Cooper
Jerry Greenberg
Eva Hocks
Tom Hutton
Christopher Irving
Marty Kandes
Amit Majumdar
Dima Mishin
Sonia Nayak
Mike Norman
Wayne Pfeiffer
Scott Sakai
Fernando Silva
Bob Sinkovits
Subha Sivagnanam
Michele Strong
Shawn Strande
Mahidhar Tatineni
Mary Thomas
Nicole Wolter
Frank Wuerthwein

EXPANSE
COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER

IN PRODUCTION OCTOBER 2020