Resource Management for Multi-Core/Multi-Threaded Usage

Martin Perry
email: martin.perry@bull.com

Architect of an Open World

Outline

- Background, Basic Concepts, Terminology
- CPU Management Steps, Configuration Options & Command Line Options
- CPU Allocation & Distribution Methods
- CPU Resource Sharing

- Task/cgroup plugin
- Accounting Limits on CPU usage

- Getting Information about CPU Usage

- CPU Management Examples

CPU Management Documentation

- Included in SLURM distribution
- Also available on SchedMD website (http://www.schedmd.com/slurmdocs/documentation.html)

HTML documents

- CPU Management User and Administrator Guide (http://www.schedmd.com/slurmdocs/cpu_management.html)

man pages

- slurm.conf
- srun
- salloc
- sbatch
- sacctmgr (for accounting limits)

Background

- HPC cluster resource managers originally managed CPU resources in units of whole nodes.
- With the growth of multi-socket, multi-core, multi-thread hardware architectures, it became necessary to manage individual CPU resources within nodes to improve resource utilization efficiency, job performance and system throughput.
- To that end, two new resource management plugins were developed for SLURM:

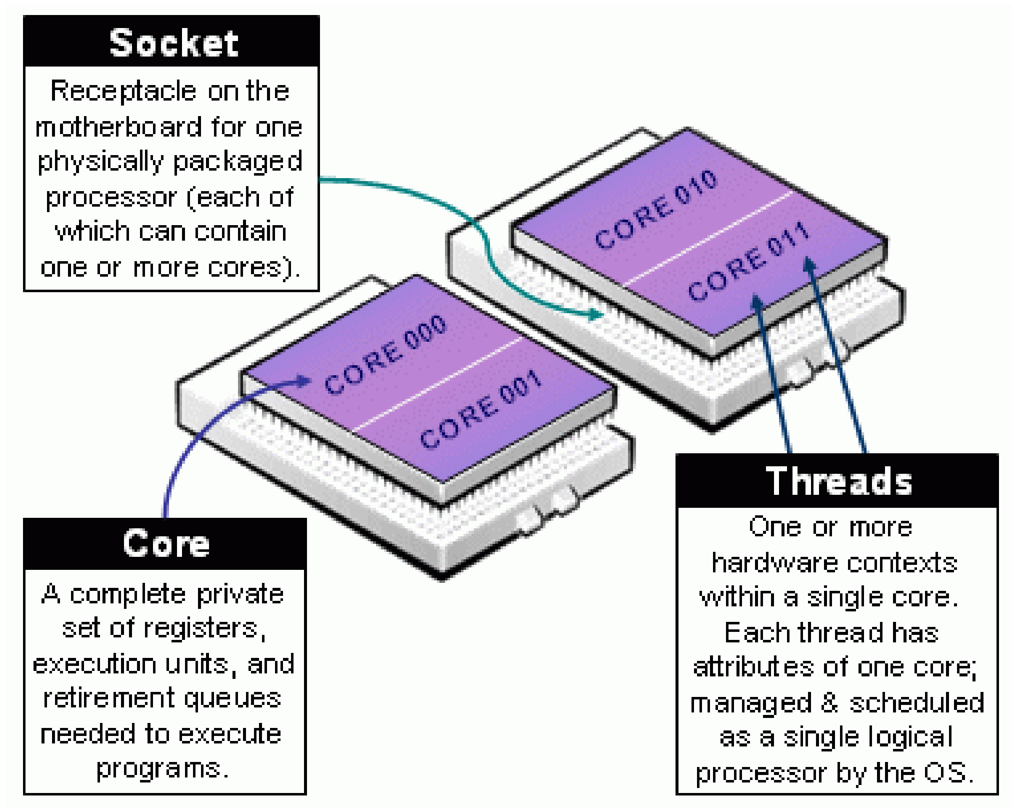
  The **select/cons_res plugin** allows sockets, cores and threads to be allocated individually as consumable resources within nodes. This can enhance throughput by allowing jobs to share nodes.

  The **task/affinity plugin** allows tasks to be bound to specific sets of CPUs within a node. This can enhance performance by increasing the utilization of CPU cache and local memory.

Basic Concepts & Techniques

- **Physical CPU Resource Types**
  - Nodes
  - Sockets
  - Cores
  - Threads
- **Physical CPU vs. Logical CPU**
  - Node definitions, FastSchedule option in slurm.conf
- **Allocation Units**
  - Whole Nodes
  - Consumable Resources
- **Exclusive Allocation vs. Shared Allocation**
  - CPU Sharing Between Jobs
  - CPU Sharing Between Tasks of the Same Job
- **Allocation and Distribution Methods**
  - Block
  - Cyclic
- **Optional Task-to-CPU Binding**
  - Task/affinity plugin
  - Task/cgroup plugin

Illustration of Socket, Core, Thread


Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 010
CORE 011
CORE 000
CORE 001
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

CPU Management Terminology

Allocation

Assignment of a specific set of CPU resources (nodes, sockets, cores and/or threads) to a specific job or step

Distribution

1. Assignment of a specific task to a specific node, or
2. Assignment of a specific task to a specific set of CPUs within a node (used for optional Task-to-CPU binding)

Binding

Confinement/locking of a specific set of tasks to a specific set of CPUs within a node

Basic CPU Management Steps

SLURM uses four basic steps to manage CPU resources for a job/step:

- **Step 1: Selection of Nodes**
- **Step 2: Allocation of CPUs from Selected Nodes**
- **Step 3: Distribution of Tasks to Selected Nodes**
- **Step 4: Optional Distribution and Binding of Tasks to Allocated CPUs within a Node**

- SLURM provides a rich set of configuration and command line options to control each step
- Many options influence more than one step
- Interactions between options can be complex and difficult to predict
- Users may be constrained by Administrator's configuration choices

Notable Options for Step 1: Selection of Nodes

Configuration options in slurm.conf

Nodename: Defines a node and its characteristics. This includes the layout of sockets, cores, threads and the number of logical CPUs on the node.

FastSchedule: Allows administrators to define “virtual” nodes with different layout of sockets, cores and threads and logical CPUs than the physical nodes in the cluster.

PartitionName: Defines a partition and its characteristics. This includes the set of nodes in the partition.

Command line options on srun/salloc/sbatch commands

--partition, --nodelist: Specifies the set of nodes from which the selection is made

-N, --nodes: Specifies the minimum/maximum number of nodes to be selected

-B, --sockets-per-node, --cores-per-socket, --threads-per-core: Limits node selection to nodes with the specified characteristics

Notable Options for Step 2: Allocation of CPUs from Selected Nodes

Configuration options in slurm.conf:

SelectType:

- **SelectType=select/linear**: Restricts allocation to whole nodes
- **SelectType=select/cons_res**: Allows allocation of individual sockets, cores or threads as consumable resources

SelectTypeParameters: For select/cons_res, specifies the consumable resource type and default allocation method within nodes

Command line options on srun/salloc/sbatch:

-n, --ntasks: Specifies the number of tasks. This may affect the number of CPUs allocated to the job/step

-c, --cpus-per-task: Specifies the number of CPUs per task. This may affect the number of CPUs allocated to the job/step

Notable Options for Step 3: Distribution of Tasks to Nodes

Configuration options in slurm.conf:

MaxTasksPerNode: Specifies maximum number of tasks per node

Command Line options on srun/salloc/sbatch:

-m, --distribution: Controls the order in which tasks are distributed to nodes.

Notable Options for Step 4: Optional Distribution & Binding

Configuration options in slurm.conf:

TaskPlugin:

TaskPlugin=task/none: Disables this step.

TaskPlugin=task/affinity: Enables task binding using the task affinity plugin.

TaskPlugin=task/cgroup: Enables task binding using the new task cgroup plugin.

TaskPluginParam: For task/affinity, specifies the binding unit (sockets, cores or threads) and binding method (sched_setaffinity or cpusets)

Command Line options on srun/salloc/sbatch:

--cpu_bind: Controls many aspects of task affinity

-m, --distribution: Controls the order in which tasks are distributed to allocated CPUs on a node for binding

Allocation & Distribution Methods

SLURM uses two default methods for allocating and distributing individual CPUs from a set of resources

- **block** method: Consume all eligible CPUs consecutively from a single resource before using the next resource in the set
- **cyclic** method: Consume eligible CPUs from each resource in the set consecutively in a round-robin fashion

The following slides illustrate the default method used by SLURM for each step.

Allocation & Distribution Methods

Step 2a: Allocation of CPUs from a set of Nodes

The default allocation method for this case is **block**
Example: A partition contains 3 nodes. Each node has 8 available CPUs.

```
srun --nodes=3 --ntasks=15 ...
```

Node	n0	n1	n2
Number of allocated CPUs	8	6	1

Users can override this default using the appropriate command line options, e.g.
`--nodes, --ntasks-per-node`

Allocation & Distribution Methods

Step 2b: Allocation of CPUs from a set of Sockets within a Node

The default allocation method for this case is **cyclic**
Example: A node contains 2 sockets. Each socket has 4 available CPUs.

```
srun --ntasks=6 ...
```

Socket#	**0**	**1**
Number of allocated CPUs	3	3

Administrators can change the default allocation method for this case to **block** with
`SelectTypeParameters=CR_CORE_DEFAULT_DIST_BLOCK`

Users can override the default using the command line option `--distribution`

Allocation & Distribution Methods

Step 3: Distribution of Tasks to Nodes

The default distribution method for this case is **block**
Example: A partition has 3 nodes. Each node has 8 available cpus.

```
srun --nodes=3 --ntasks=8 --cpus-per-task=2 ...
```

Node	n0	n1	n2
Number of allocated CPUs	8	6	2
Distribution of Tasks, by Task#	0 - 3	4 - 6	7

Users can override this default using the command line option `--distribution`. The option supports three alternate methods for distributing tasks to nodes: **cyclic**, **plane** and **arbitrary**.

Allocation & Distribution Methods

Step 4: Distribution of Tasks to Allocated CPUs within a Node for Task-to-CPU binding

The default distribution method for this case is **cyclic**
Example: A node has 2 sockets. Each socket has 4 available CPUs (cores).

```
srun --ntasks=8 --cpu_bind=cores
```

Node	n0							
Socket#	0				1			
CPU#	0	1	2	3	4	5	6	7
Bound Task#	0	2	4	6	1	3	5	7

Users can override this default and specify **block** distribution of tasks to CPUs using the command line option `--distribution`

CPU Resource Sharing

- **CPU Resource Sharing between Jobs**

  - **Shared** keyword on partition definition in slurm.conf
  - **--share**, **--exclusive** options on srun/salloc/sbatch command line

Example:

```
Nodename=n0 CPUs=8
PartitionName=p1 Nodes=n0 Shared=YES

srun --partition=p1 --ntasks=8 --share …
srun --partition=p1 --ntasks=8 --share …
```

CPU Resource Sharing

- **<u>CPU Resource Sharing between Tasks of the same Job</u>**

- **--overcommit** option on srun/salloc/sbatch command line

Example:

```
NodeName=n0 CPUs=8
PartitionName=p1 Nodes=n0

srun --partition=p1 --ntasks=12 --overcommit …
```

<u>Note</u>:
It is important to understand that the Linux scheduler is not aware of CPU allocations by SLURM. Unless Task-to-CPU binding is used, Linux may run any task on any CPU on the node to which the task was distributed. In this way, CPUs may be shared between tasks and jobs even in the absence of shared CPU allocation by SLURM.

Task/cgroup Plugin

- New version of Task plugin
- Alternative to task/affinity plugin for Task-to-CPU binding
- Uses Portable Hardware Locality (hwloc) library
- Uses cgroup cpuset subsystem
- Also used for resource confinement of jobs/steps for other types of resource (memory, generic resources)

Slurm Accounting Limits on CPU Usage

CPU management by SLURM users is subject to limits imposed by SLURM Accounting. Accounting limits may be applied on CPU usage at the level of users, associations and clusters. CPU-related accounting limits include the following:

MaxNodes: Maximum number of nodes that can be selected for each job.

MaxCPUs: Maximum number of CPUs that can be allocated to each job.

GrpNodes: Maximum number of nodes that can be selected for all running jobs combined in this association

GrpCPUs: Maximum number of CPUs that can be allocated to all running jobs combined in this association

For more details, see the sacctmgr man page.

Getting Information About CPU Usage

Information about Node Selection and CPU allocation (Steps 1 & 2)

```
[sulu] (slurm) mnp> srun --nodes=2 --ntasks=3 --cpus-per-task=2 sleep 60
[2] 22841

[sulu] (slurm) mnp> squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
    309  allnodes    sleep    slurm   R       0:02      2  n[6-7]

[sulu] (slurm) mnp> scontrol --details show job 309
    . . .
   NumNodes=2 NumCPUs=6 CPUs/Task=2 ReqS:C:T=*:*:*
     Nodes=n6 CPU_IDs=4-7 Mem=0
     Nodes=n7 CPU_IDs=6-7 Mem=0
    . . .
```

Note:
The CPU_IDs reported by scontrol are SLURM abstract CPU numbers, not physical CPU numbers known to Linux.

Getting Information About CPU Usage (cont'd)

Information about Distribution of Tasks to Nodes (Step 3)

```
[sulu] (slurm) mnp> srun --nodes=3 --ntasks=5 sleep 60 &


[sulu] (slurm) mnp> squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
    311  allnodes    sleep    slurm   R       0:03      3 n[6-7,13]


[sulu] (slurm) mnp> sstat --allsteps --jobs 311 --pidformat
       JobID                 Nodelist                 Pids
------------ -------------------- --------------------
311.0                         n13            7994,7995
311.0                          n6            3838,3839
311.0                          n7                 7199
```

Getting Information About CPU Usage (cont'd)

Information about Task-to-CPU binding (Step 4)

```
[sulu] (slurm) mnp> srun --partition=bones-scotty --nodes=2 --ntasks=4
--cpu_bind=cores,verbose --label cat /proc/self/status | grep
Cpus_allowed_list

0: cpu_bind=MASK - scotty, task  0  0 [4070]: mask 0x8 set
3: cpu_bind=MASK - bones, task  3  0 [23808]: mask 0x80 set
1: cpu_bind=MASK - scotty, task  1  1 [4071]: mask 0x20 set
2: cpu_bind=MASK - scotty, task  2  2 [4072]: mask 0x80 set

3: Cpus_allowed_list:   7
0: Cpus_allowed_list:   3
1: Cpus_allowed_list:   5
2: Cpus_allowed_list:   7
```

CPU Management Examples

Example Node and Partition Configuration in slurm.conf

In the following examples, the default partition contains three nodes: n0, n1 and n2. Each node has 2 sockets, 4 cores per socket and 1 thread per core, for a total of 8 logical CPUs per node. It is assumed that all CPUs on all nodes are available.

```
Nodename=n0 NodeAddr=node0 Sockets=2 CoresPerSocket=4 ThreadsPerCore=1
   CPUs=8 State=IDLE
Nodename=n1 NodeAddr=node1 Sockets=2 CoresPerSocket=4 ThreadsPerCore=1
   CPUs=8 State=IDLE
Nodename=n2 NodeAddr=node2 Sockets=2 CoresPerSocket=4 ThreadsPerCore=1
   CPUs=8 State=IDLE

PartitionName=regnodes Nodes=n0,n1,n2 Shared=YES Default=YES State=UP
```

CPU Management Examples

Example 1: Allocation of Whole Nodes

Allocate a minimum of two whole nodes to a job.

In slurm.conf:

```
SelectType=select/linear
```

srun command line:

```
srun --nodes=2 --ntasks=2 …
```

The `SelectType=select/linear` configuration option specifies allocation in units of whole nodes. The `--nodes=2` srun option causes SLURM to allocate at least 2 nodes to the job.

CPU Management Examples

<u>Example 2: Consumable Resources with balanced allocation across Nodes</u>

A job requires 9 CPUs (3 tasks and 3 CPUs per task). Allocate 3 CPUs from each of the 3 nodes in the default partition.

<u>In slurm.conf</u>:

```
SelectType=select/cons_res
SelectTypeParameters=CR_Core
```

<u>srun command line</u>:

```
srun --ntasks=3 --cpus-per-task=3 --ntasks-per-node=1 ...
```

To satisfy `--ntasks-per-node=1`, SLURM must allocate 3 CPUs on each of the 3 nodes in the default partition. This overrides the default method for allocating CPUs from a set of nodes (block allocation).

CPU Management Examples

<u>Example 3: Consumable Resources with minimization of resource fragmentation</u>

A job requires 12 CPUs. Allocate CPUs using the minimum number of nodes (2 nodes) and the minimum number of sockets (3 sockets) required for the job in order to minimize fragmentation of allocated/unallocated CPUs in the cluster.

<u>In slurm.conf</u>:

```
SelectType=select/cons_res
SelectTypeParameters=CR_Core,CR_CORE_DEFAULT_DIST_BLOCK
```

<u>srun command line</u>:

```
srun --ntasks=12
```

The default allocation method across a selection of nodes is block. This minimizes the number of nodes used for the job. The configuration option `CR_CORE_DEFAULT_DIST_BLOCK` sets the default allocation method across sockets within a node to block. This minimizes the number of sockets used for the job within a node.

CPU Management Examples

Example 4: Consumable resources with Task-to-CPU binding and non-default allocation and distribution methods

A job requires 18 CPUs. Allocate 6 CPUs on each node using block allocation within nodes. Use cyclic distribution of tasks to nodes and block distribution of
tasks to CPUs for binding.

In slurm.conf:

```
SelectType=select/cons_res
SelectTypeParameters=CR_Core
TaskPlugin=task/affinity
TaskPluginParam=sched
```

srun command line:

```
srun --ntasks=18 --ntasks-per-node=6 --distribution=cyclic:block
--cpu_bind=cores ...
```

(continued)

CPU Management Examples

- `--ntasks-per-node=6` overrides the default allocation method across nodes (block) and causes SLURM to allocate 6 CPUs on each of the 3 nodes in the default partition.

- `--distribution cyclic:xxxxx` overrides the default method for distributing tasks to nodes (block).

- `task/affinity` plus `--cpu_bind=cores` causes SLURM to bind each task to a single allocated core.

- `--distribution xxxxx:block` overrides the default allocation method within nodes (cyclic) and the default method for distributing tasks to CPUs for binding (cyclic).

The following table depicts a possible pattern of allocation, distribution and binding for this job.

(continued)

CPU Management Examples

```
srun --ntasks=18 --ntasks-per-node=6 --distribution=cyclic:block
 --cpu_bind=cores ...
```

Node	n0		n1		n2	
Socket#	0	1	0	1	0	1
Number of Allocated CPUs	4	2	4	2	4	2
Allocated CPU#'s	0 - 5		0 - 5		0 - 5	
Number of Tasks	6		6		6	
Distribution of Tasks to Nodes, by Task#	0, 3, 6, 9, 12, 15		1, 4, 7, 10, 13, 16		2, 5, 8, 11, 14, 17	

Task-to-CPU Binding	n0								n1								n2							
CPU#	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Task#	0	3	6	9	12	15	-	-	1	4	7	10	13	16	-	-	2	5	8	11	14	17	-	-

bullx

instruments for innovation

powered by Bull