Slurm for users

genotoul

06-11-2017
Summary

1. Introduction
   - Slurm architecture
   - Slurm terms
2. User and admin commands
   - Basics
   - User commands (srun, salloc, sbatch)
   - User and admin commands (sinfo, squeue, scancel, scontrol, sstat, sprio, sacctmgr)
3. Additional informations
   - Commands format
   - Slurm environment
   - Job arrays
   - Job dependencies
4. Slurm and MPI
   - Slurm MPI integration
   - Running MPI jobs
   - Running hybrid MPI OpenMP jobs
Introduction
Introduction
SLURM architecture

- One central controller daemon slurmctld
- A daemon upon each computing node slurmd
- One central daemon for the database controls slurmdbd
Introduction

SLURM terms

- Computing node: computer used for the execution of programs
- Partition: group of nodes with specific characteristics (job limit, access controls, etc)
- Job: allocation of resources assigned to a user for some time
- Step: sets of (possible parallel) tasks within a job
User and admin commands
Where can I launch? **sinfo** command displays resource usage and availability information for parallel jobs

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>workq*</td>
<td>up</td>
<td>4:00:00</td>
<td>46</td>
<td>idle</td>
<td>node[101-110,112-147]</td>
</tr>
<tr>
<td>workq*</td>
<td>up</td>
<td>4:00:00</td>
<td>2</td>
<td>down</td>
<td>node[111,148]</td>
</tr>
<tr>
<td>interq</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>mix</td>
<td>genoview</td>
</tr>
<tr>
<td>smpq</td>
<td>up</td>
<td>infinite</td>
<td>1</td>
<td>idle</td>
<td>genosmp02</td>
</tr>
</tbody>
</table>

How can I launch a job script? **sbatch** command

> sbatch HELLO_WORLD.sub
 Submitted batch job 6595

How to cancel a job? **scancel** command

> scancel 6595
## User commands

srund, salloc, sbatch

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>srund</strong></td>
<td>used to submit a job for execution or initiate job steps in real time (option -n number of core, -N number of node, --time for time limit, etc). If necessary, srun will first create resource allocation in which to run the parallel job</td>
</tr>
<tr>
<td><strong>salloc</strong></td>
<td>allocate resources (nodes, tasks, partition, etc), either run a command or start a shell. Request launch srun from shell (interactive commands within one allocation)</td>
</tr>
<tr>
<td><strong>sbatch</strong></td>
<td>allocate resources (nodes, tasks, partition, etc.) Launch a script containing srunds for series of steps</td>
</tr>
</tbody>
</table>
User commands
srn,salloc,sbatch

- All request an allocation of resources
- Similar set of command line options
- Request number of nodes, tasks, cpus, constraints, user info, dependencies, and lots more
- srn launches tasks (command) in parallel on the requested nodes
- salloc obtain a slurm job allocation. It can launch a task such as mpirun on the client, or open a shell on the client. srn is then used to launch tasks within the allocation
- sbatch is a shell script that contains multiple srns within the allocation (multistep job)
- Command line options are preponderant
- The default is one task per node (unless -n or --cpus-per-task is used)
User commands

Sample srun

- srun launches a job that allocates resources (number of nodes, tasks, etc.) and is executed on each allocated cpu. Some basic parameters for srun command:
  - `> srun -l -J hostname --time=00:10:00 -N 2 --mem=40G -p workq --exclusive hostname`:
    - `-l` prepand task number to output
    - `-J` job name
    - `--time=` allocation time limit (format is days-hours:minutes:seconds)
    - `-p workq` specify the partition to use
    - `-N 2` number of nodes required
    - `--mem` memory required per node
    - `--exclusive` exclusive acces to nodes (default is shared ressources)
    - `hostname` command to run

1: node102
0: node101
User commands

sbatch

▶ submit a batch script to slurm
▶ the batch script may contain options proceed with « #SBATCH » before any executable call
▶ assigned a jobId when script is successfully transferred to the slurm controller
▶ when job allocation is finally granted, Slurm runs a single copy of the batch script on the first node in the set of allocated nodes
▶ default stdout and stderr are directed to a file slurm-%j.out (can be modified with options -e and -o). %j is replaced by the jobID
▶ the script can be written in multiples languages like bash, ksh, python, etc
User commands

Sample sbatch

Same as previous srun in sbatch script HOSTNAME.sub

```bash
#!/bin/bash
#SBATCH --time=00:10:00  # job time limit
#SBATCH -J hostname # job name
#SBATCH -N 2        # number of nodes
#SBATCH -p workq    # partition to use
#SBATCH --exclusive # exclusive acces to nodes
srun -l hostname   # submit parallel command
```

Options submitted on command line are preponderant

```bash
> sbatch -J toto HOSTNAME.sub
> squeue -a
Submitted batch job 6604
```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6604</td>
<td>workq</td>
<td>toto</td>
<td>root</td>
<td>R</td>
<td>0:00</td>
<td>2</td>
<td>node[101-102]</td>
</tr>
</tbody>
</table>
salloc is used to allocate resources for a job in real time.

Typically this is used to allocate resources and spawn a shell

The shell be used to execute srun commands

Basic parameters similar with srun and sbatch

```bash
> salloc -N 2
salloc: Granted job allocation 6612
salloc: Waiting for resource configuration
salloc: Nodes node[101-102] are ready for job

> echo $SLURM_JOB_NODELIST
node[101-102]

> salloc -n 1 --constraint=K40 -p interq
salloc: Granted job allocation 6999
salloc: Waiting for resource configuration
salloc: Nodes genoview are ready for job
```

alloc 1 task no GPU node
## User and admin commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo</td>
<td>display characteristics of nodes, partitions, reservations...</td>
</tr>
<tr>
<td>squeue</td>
<td>display jobs and their state</td>
</tr>
<tr>
<td>scancel</td>
<td>cancel a job or set of jobs</td>
</tr>
<tr>
<td>scontrol</td>
<td>administrative tool used to view and/or modify SLURM state. But can also be used to get information on jobs, partitions, reservations ...</td>
</tr>
<tr>
<td>sstat</td>
<td>show status of running jobs</td>
</tr>
<tr>
<td>sprio</td>
<td>view the factors that comprise a job’s scheduling priority</td>
</tr>
<tr>
<td>sacctmgr</td>
<td>setup accounts, specify limitations on users and groups</td>
</tr>
</tbody>
</table>
User and admin commands

sinfo

» sinfo display information about Slurm nodes and partitions

> sinfo -n node[108-110]

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>workq*</td>
<td>up 4-00:00:00</td>
<td>3</td>
<td>idle</td>
<td>node[108-110]</td>
<td></td>
</tr>
<tr>
<td>unlimitq</td>
<td>up infinite</td>
<td>3</td>
<td>idle</td>
<td>node[108-110]</td>
<td></td>
</tr>
<tr>
<td>wflowq</td>
<td>up infinite</td>
<td>3</td>
<td>idle</td>
<td>node[108-110]</td>
<td></td>
</tr>
<tr>
<td>interq</td>
<td>up 1-00:00:00</td>
<td>0</td>
<td>n/a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>smpq</td>
<td>up infinite</td>
<td>0</td>
<td>n/a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

» list reasons nodes are in the down, drained or fail state

> sinfo -Rl

Mon Nov  6 14:42:53 2017

<table>
<thead>
<tr>
<th>REASON</th>
<th>USER</th>
<th>TIMESTAMP</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node unexpectedly re root(0)</td>
<td>2017-10-16T16:30:58</td>
<td>down</td>
<td>node111</td>
<td></td>
</tr>
</tbody>
</table>

» list by state :

> sinfo -t ALLOC

<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>workq*</td>
<td>up 4-00:00:00</td>
<td>3</td>
<td>alloc</td>
<td>node[114,126,135]</td>
<td></td>
</tr>
<tr>
<td>unlimitq</td>
<td>up infinite</td>
<td>3</td>
<td>alloc</td>
<td>node[114,126,135]</td>
<td></td>
</tr>
<tr>
<td>wflowq</td>
<td>up infinite</td>
<td>3</td>
<td>alloc</td>
<td>node[114,126,135]</td>
<td></td>
</tr>
</tbody>
</table>
User and admin commands
squeue

squeue display jobs and their state. Basic parameters for squeue command:

- `-a`: display info about all jobs and partitions
- `-j <job_list>`: report more info about a particular job or jobs
- `-u <user>`: report job information for a specific user
- `-i <seconds>`: repeatedly gather and report the requested information
- `--start`: expected start time of pending job (if backfill scheduling plugin is used)

```plaintext
> squeue -a

<table>
<thead>
<tr>
<th>JOBD</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODELIST(REASON)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6612</td>
<td>workq</td>
<td>bash</td>
<td>root</td>
<td>R</td>
<td>7:56</td>
<td>2 node[101-102]</td>
</tr>
<tr>
<td>6542</td>
<td>interq</td>
<td>TurboVNC</td>
<td>dgorecki</td>
<td>R</td>
<td>1-06:19:31</td>
<td>1 genoview</td>
</tr>
</tbody>
</table>
```
scancel is used to signal jobs or job steps. Usage:
> scancel <jobID> <stepID>

Usefull options
- `-n` : restrict cancel to jobs with this job name
- `-p` : restrict cancel to jobs in this partition
- `-t` : restrict cancel to jobs in this state
- `-s` : send signal to job or step
- `-u` : restrict cancel to jobs owned by this user
- `-w` : cancel any job using any of the given hosts

> scancel -p workq -u myuser -t PD
User and admin commands

scontrol

- scontrol is a tool used to view and modify SLURM configuration state
- can be used to get information on configuration, jobs, nodes, partitions, reservations ...
  - scontrol show config
  - scontrol show job <jobId>
  - scontrol show node <node>
  - scontrol show partition <partition>
  - scontrol show reservation <res>

```
> scontrol show job 6924
JobId=6924 JobName=xhpl
  UserId=dgorecki(13549) GroupId=BULL(3000) MCS_label=N/A
  Priority=1 Nice=0 Account=bull QOS=others
  JobState=RUNNING Reason=none Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:24:54 TimeLimit=03:00:00 TimeMin=N/A
  SubmitTime=2017-11-02T17:30:23 EligibleTime=2017-11-02T17:30:23
  StartTime=2017-11-06T14:35:23 EndTime=2017-11-06T17:35:23 Deadline=N/A
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=workq AllocNode:Sid=genologin1:56975
  ReqNodeList=node135 ExcNodeList=(null)
  NodeList=node135
  BatchHost=node135
  NumNodes=1 NumCPUs=64 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:1
  TRES=cpu=64,mem=250G,node=1
  Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=* 
  MinCPUsNode=1 MinMemoryNode=250G MinTmpDiskNode=0
  Features=(null) DelayBoot=00:00:00
  Gres=(null) Reservation=(null)
  OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
  Command=/home/dgorecki/HPL.1N.node135.job
  WorkDir=/home/dgorecki
  StdErr=/home/dgorecki/slurm-6924.out
  StdIn=/dev/null
  StdOut=/home/dgorecki/slurm-6924.out
  Power=
```
User and admin commands

sprio

▶ use to view the components of a job’s scheduling priority when multi-factor priority plugin is installed
▶ return
▶ the PRIORITY column report the global priority of job (highest is prior)
▶ by default, returns information for all pending jobs

```
sprion -u dgorecki
```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>6930</td>
<td>dgorecki</td>
<td>1</td>
</tr>
<tr>
<td>6931</td>
<td>dgorecki</td>
<td>1</td>
</tr>
<tr>
<td>6933</td>
<td>dgorecki</td>
<td>1</td>
</tr>
</tbody>
</table>

PriorityParameters = (null)
PriorityDecayHalfLife = 7-00:00:00
PriorityCalcPeriod = 00:05:00
PriorityFavorSmall = No
PriorityFlags =
PriorityMaxAge = 7-00:00:00
PriorityUsageResetPeriod = NONE
PriorityType = priority/multifactor
PriorityWeightAge = 0
PriorityWeightFairShare = 0
PriorityWeightJobSize = 0
PriorityWeightPartition = 0
PriorityWeightQOS = 0
PriorityWeightTRES = (null)
User and admin commands

QOS, sacctmgr

- QOS (Quality of Services) are used in SLURM for grouping limitations and priorities
- Show QOS list
  > sacctmgr show qos

- Show usable QOS by user
  > sacctmgr show assoc user=dgorecki

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Account</th>
<th>User</th>
<th>Partition</th>
<th>Share GrpJobs</th>
<th>GrpTRES GrpSubmit</th>
<th>GrpWall</th>
<th>GrpTRESMins</th>
<th>MaxJobs</th>
<th>MaxTRES</th>
<th>MaxTRESPerNode M</th>
<th>maxSubmit</th>
<th>MaxWall</th>
<th>MaxTRESMins</th>
<th>QOS</th>
<th>Def QOS</th>
<th>GrpTRESRunMin</th>
</tr>
</thead>
<tbody>
<tr>
<td>genobull</td>
<td>bull</td>
<td>dgorecki</td>
<td>unlimitq</td>
<td>1</td>
<td>2500</td>
<td>cpu=6000000</td>
<td>others_unlimit others_u+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>genobull</td>
<td>bull</td>
<td>dgorecki</td>
<td></td>
<td>1</td>
<td>2500</td>
<td>cpu=6000000</td>
<td>contributors,others others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3 Additional informations
some commands like sacct and squeue give the possibility to tune output format

```bash
sacct -D --format=jobid%-13,user%-15,uid,jobname%-15,state %20,exitcode,Derivedexitcode,nodelist% -X --job 6969
```

<table>
<thead>
<tr>
<th>JobID</th>
<th>User</th>
<th>UID</th>
<th>JobName</th>
<th>State</th>
<th>ExitCode</th>
<th>DerivedExitCode</th>
<th>NodeList</th>
</tr>
</thead>
<tbody>
<tr>
<td>6969</td>
<td>root</td>
<td>0</td>
<td>toto</td>
<td>COMPLETED</td>
<td>0:0</td>
<td>0:0</td>
<td>node[101-102]</td>
</tr>
</tbody>
</table>

```bash
squeue --format="%10i %12u %12j %.8M %.8l %.10Q %10P %10q %10r %11v %12T %D %R" -S "T"
```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>NAME</th>
<th>TIME</th>
<th>TIME_LIM</th>
<th>PRIORITY</th>
<th>PARTITION</th>
<th>QOS</th>
<th>REASON</th>
<th>RESERVATION</th>
<th>STATE</th>
<th>NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>6612</td>
<td>root</td>
<td>bash</td>
<td>16:09</td>
<td>4-00:00:00:00</td>
<td>1</td>
<td>workq</td>
<td>normal</td>
<td>None</td>
<td>(null)</td>
<td>RUNNING</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node[101-102]</td>
</tr>
<tr>
<td>6542</td>
<td>dgorecki</td>
<td>TurboVNC</td>
<td>1-06:27:44</td>
<td>UNLIMITE</td>
<td>1</td>
<td>interq</td>
<td>normal</td>
<td>None</td>
<td>(null)</td>
<td>RUNNING</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>genoview</td>
</tr>
</tbody>
</table>
## Slurm job environment

**Environment variables**

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLURM_JOBID</td>
<td>Job ID</td>
</tr>
<tr>
<td>SLURM_NNODES</td>
<td>#SBATCH -N</td>
</tr>
<tr>
<td>SLURM_NODELIST</td>
<td>Nodelist which is allocated to the job</td>
</tr>
<tr>
<td>SLURM_NTASKS</td>
<td>#SBATCH -n</td>
</tr>
<tr>
<td>SLURM_NTASKS_PER_NODE</td>
<td>#SBATCH --tasks-per-node</td>
</tr>
<tr>
<td>SLURM_CPUS_PER_TASK</td>
<td>#SBATCH -c</td>
</tr>
<tr>
<td>SLURM_SUBMIT_DIR</td>
<td>Job submission directory</td>
</tr>
</tbody>
</table>

[https://slurm.schedmd.com/sbatch.html](https://slurm.schedmd.com/sbatch.html)
Slurm job environment

Environment variables

--export option identify which environment variables are propagated to the batch job

- --export=ALL: all current shell variables are propagated
- --export=NONE: no variable propagated
- --export=VARIABLE=value: propagate the current variable

```bash
srun --export=LOGTYPE=debug,LOGFILE=log.out ./program
```
Job arrays

sbatch

- sbatch -a | --array=<indexes>
  - submit a job array, multiple jobs to be executed with identical parameters
  - multiple values may be specified using a comma separated list and/or a range of values with a "-" separator
    - --array=1-10
    - --array=0,6,16-32
    - --array=0-15:4 : a step of 4
    - --array=1-10%2 : a maximum of 2 simultaneously running tasks

<table>
<thead>
<tr>
<th>Variable</th>
<th>Correspondance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLURM_ARRAY_TASK_ID</td>
<td>Job array ID (index) number</td>
</tr>
<tr>
<td>SLURM_ARRAY_JOB_ID</td>
<td>Job array's master job ID number</td>
</tr>
<tr>
<td>SLURM_ARRAY_TASK_MAX</td>
<td>Job array's maximum ID (index) number</td>
</tr>
<tr>
<td>SLURM_ARRAY_TASK_MIN</td>
<td>Job array's minimum ID (index) number</td>
</tr>
<tr>
<td>SLURM_ARRAY_TASK_COUNT</td>
<td>total number of tasks in a job array</td>
</tr>
</tbody>
</table>
Job dependencies

sbatch

- `sbatch -d | --dependency=<dependency_list>`
  - defer the start of this job until the specified dependencies have been satisfied completed
  - `<dependency_list>` is of the form `<type:jobId[:jobID][,type:jobId[:jobID]]>`, example:
    ```bash
    sbatch --dependency=afterok:6265 HELLO.job
    ```

<table>
<thead>
<tr>
<th>Type</th>
<th>Correspondance</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>this job can begin execution after the specified jobs have begun execution</td>
</tr>
<tr>
<td>afterany</td>
<td>this job can begin execution after the specified jobs have terminated</td>
</tr>
<tr>
<td>afterok</td>
<td>This job can begin execution after the specified jobs have successfully executed (ran to completion with an exit code of zero)</td>
</tr>
<tr>
<td>afternotok</td>
<td>This job can begin execution after the specified jobs have terminated in some failed state (non-zero exit code, node failure, timed out, etc)</td>
</tr>
<tr>
<td>singleton</td>
<td>This job can begin execution after any previously launched jobs sharing the same job name and user have terminated</td>
</tr>
</tbody>
</table>
SLURM and MPI
SLURM MPI integration

▶ MPI use with slurm depends upon the type of MPI being used. There are three fundamentally different modes of operation used by these various MPI implementations:
  – Slurm directly launches the tasks and performs initialization of communications (use of PMI library)
  – Slurm creates a resource allocation for the job and then mpirun launches tasks using Slurm's infrastructure
  – Slurm creates a resource allocation for the job and then mpirun launches tasks using some mechanism other than Slurm, such as SSH or RSH (outside slurm's monitoring or control)

▶ PMI library:
  – The use of a PMI library offer tight integration with slurm and the simplest way to launch an MPI application.
  – The PMI library has been used for quite some time as a means of exchanging information needed for interprocess communication.
OpenMPI is configured with pmi2 support (compiled with --with-pmi=). The OMPI jobs can be launched directly using the srun command.

```
> module load compiler/intel-2018.0.128 mpi/openmpi-1.8.8-intel2018.0.128
> mpicc -o hello_world hello_world.c
> srun -n 12 -N 2 --ntasks-per-node=6 hello_world
Hello world from process 4 of 12 - node101
Hello world from process 5 of 12 - node101
Hello world from process 3 of 12 - node101
Hello world from process 0 of 12 - node101
Hello world from process 2 of 12 - node101
Hello world from process 9 of 12 - node102
Hello world from process 6 of 12 - node102
Hello world from process 7 of 12 - node102
Hello world from process 10 of 12 - node102
Hello world from process 8 of 12 - node102
Hello world from process 1 of 12 - node101
Hello world from process 11 of 12 - node102
```
But it is also possible to launch an MPI application through the common mpirun command. Example of a full bash script:

```
#!/bin/bash
#SBATCH -J mpi_job
#SBATCH --nodes=2
#SBATCH --tasks-per-node=6
#SBATCH --time=00:10:00
cd $SLURM_SUBMIT_DIR
module purge
module load compiler/intel-2018.0.128 mpi/openmpi-1.8.8-intel2018.0.128
mpirun -n $SLURM_NTASKS -npernode $SLURM_NTASKS_PER_NODE ./hello_world
```
You can use --cpus-per-task in order to set the number of OpenMP threads

#!/bin/bash
#SBATCH -J hybrid_mpi_openmp_job
#SBATCH --nodes=2
#SBATCH --tasks-per-node=4
#SBATCH --cpus-per-task=8
cd $SLURM_SUBMIT_DIR
module purge
module load compiler/intel-2018.0.128 mpi/openmpi-1.8.8-intel2018.0.128
[ -n "$SLURM_CPUS_PER_TASK" ] && export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun ./hybrid_program
Thanks

For more information please contact:
M+ 33 6 14655608
cedric.trivino@atos.net