

MC-SD01-I

Introdução ao ambiente SDUMONT/SLURM

Escola Santos Dumont
11 de Janeiro 2021
LNCC
evento online

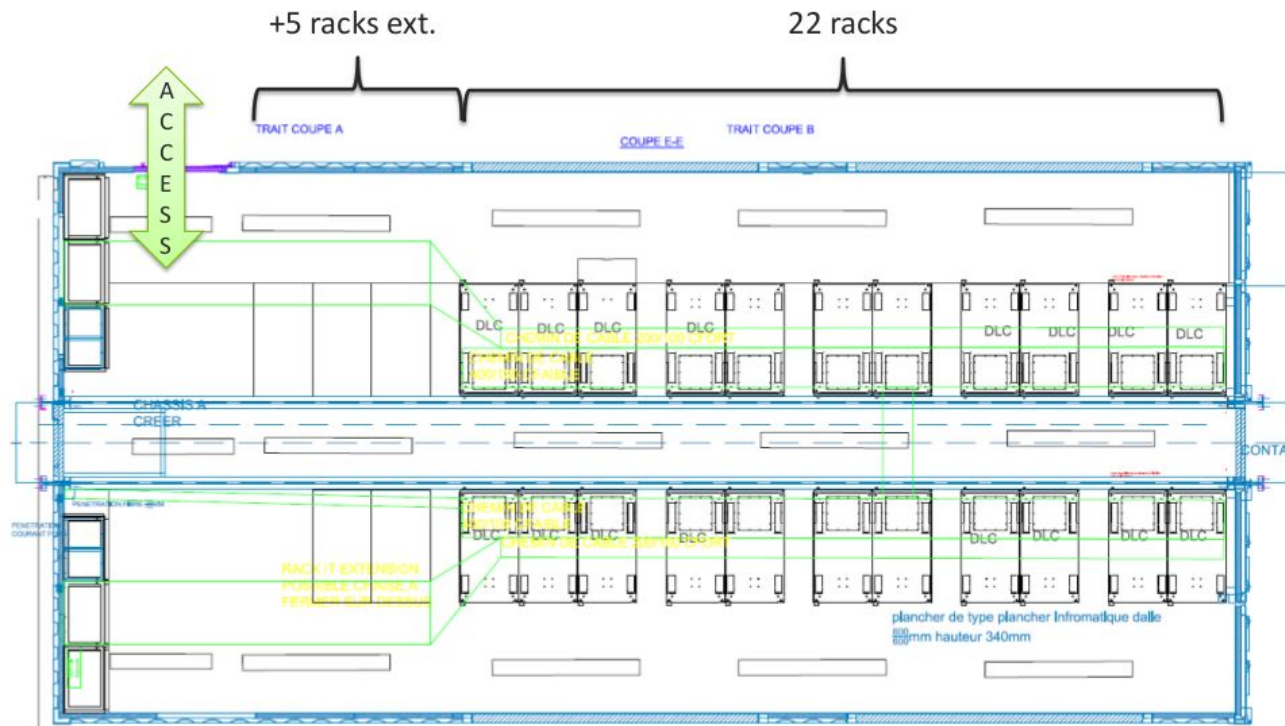
A máquina

Mobull - solução para datacenter baseada em containers.

Plug & Boot

2 containers com 22 racks 42U

A máquina



Arquitetura

Cluster de propósito geral

3 tipos de nodes:

- thin nodes
- hybrid nodes
- fat-node

Arquitetura

Thin nodes (B710)

- 7 racks completos (504 nós computacionais)
- 2 nós computacionais por blade
- Configuração
 - 2x Intel Xeon E5-2695v2 (12c, 2.4Ghz)
 - 64 GB DDR3 RAM (8x 8GB DIMM)
 - 1x 120GB SSD disk
 - 1x Infiniband FDR ConnectX3
 - 1x GbE

Arquitetura

Thin nodes (Bull Sequana) - Machine Learning/Deep Learning

- 1 nó computacional
- Configuração
 - 2x Intel Skylake GOLD 6148, 2,4Ghz (20c)
 - 384 GB DDR4
 - 4x Infiniband EDR 100Gbps
 - 8x NVidia V100 com NVLink

Arquitetura

Hybrid nodes (B715)

- 7 racks completos (252 nós computacionais e 504 aceleradores)
- 198 nodes e 396 nVidia K40
- 54 nodes e 108 Intel Phi 7120P

Arquitetura

Fat-node (S6130)

- 16x Intel Ivy Bridge E7 2870v2 15c 2.3Ghz
- 6TB DDR3 RAM
- 1x 120GB SSD disk
- 1x Infiniband FDR ConnectX3
- 1x GbE

Arquitetura

Login nodes

- 4x bullx R423-E3
- Linux Virtual Server (LVS)
- Cada login node possui:
 - 2x E5-2695v2 12c, 2.4GHz
 - 128 GB DDR3@1866RAM
 - 2x 500GB 7.2krpm SATA2 RAID1
 - 1x GbE network port
 - 1x IB FDR network port
 - 1x Ethernet BMC network port
 - 2x 10GbE network ports

Arquitetura

Armazenamento

- Lustre - Seagate ClusterStor 9000
 - Total 1,7 Petabytes
- DellEMC Isilon
 - Total 650 Terabytes

Estrutura de diretórios

Diretório home (\$HOME):

- NFS
- Acessível apenas nos login nodes
- /prj/NOME_PROJETO/login.name

Diretório de scratch (\$SCRATCH):

- Lustre
- Acessível a todos os nodes do cluster
- /scratch/NOME_PROJETO/login.name

Desempenho

- GPU - 456,8 TFlop/s
- PHI - 363,2 TFlop/s
- CPU - 321,2 TFlop/s
- Total - 1.141,2 TFlop/s

TOP 500	Total	GPU	PHI	CPU
Jun/15	55	145	177	207
Nov/15	63	200	265	310
Jun/16	75	265	364	433
Nov/16	91	364	476	
Jun/17	107	472		
Nov/17	128			
Jun/18	192			
Nov/18	316			

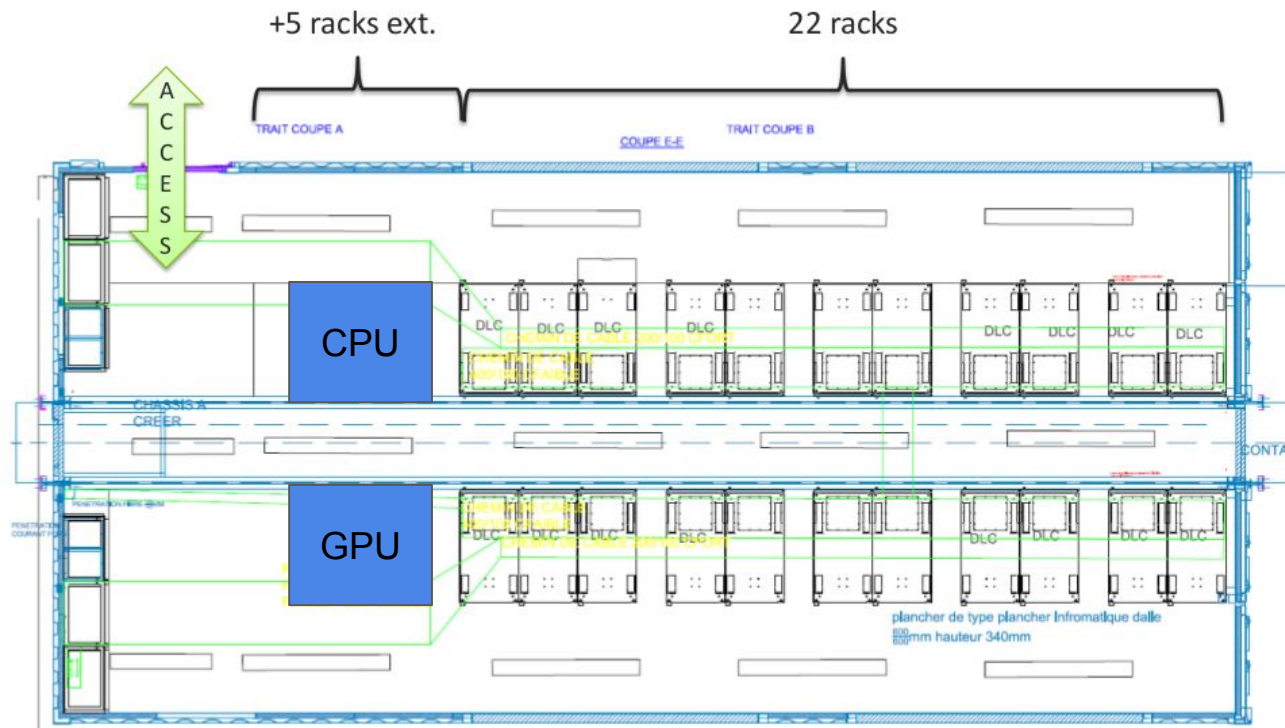
<https://www.top500.org>

Expansão SDumont

- 1 Célula Sequana X1000 CPU
 - 82 Blades X1120 - 384 GB
 - 12 Blades X1120 - 768 GB
 - 282 nós computacionais
 - 2x Intel CascadeLake Gold 6252 (24c)
- 1 Célula Sequana X1000 GPU
 - 94 Blades X1125 - 384 GB
 - 1 nó computacional e 4 aceleradores NVIDIA Volta V100 GPU por blade
 - 2x Intel CascadeLake Gold 6252 (24c)



Expansão SDumont



Desempenho

Novembro 2019



193	Laboratório Nacional de Computação Científica Brazil	Santos Dumont (SDumont) - Bull Sequana X1000, Xeon Gold 6252 24C 2.1GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100 SXM2 Atos	33,856	1,849.0	2,727.0
-----	---	--	--------	---------	---------

Junho 2020

240	Santos Dumont (SDumont) - Bull Sequana X1000, Xeon Gold 6252 24C 2.1GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100 SXM2, Atos Laboratório Nacional de Computação Científica Brazil	33,856	1,849.0	2,727.0
-----	--	--------	---------	---------

Novembro 2020

276	Santos Dumont (SDumont) - Bull Sequana X1000, Xeon Gold 6252 24C 2.1GHz, Mellanox InfiniBand EDR, NVIDIA Tesla V100 SXM2, Atos Laboratório Nacional de Computação Científica Brazil	33,856	1,849.0	2,727.0
-----	--	--------	---------	---------

Filas

Filas	Wall-clock	Nodes	Núcleos	Execução	Na fila
treinamento_gpu	30min	2	48	1	2
treinamento	30min	1 - 4	96	1	2

Módulos de ambiente

module avail

module whatis/help

module load

module unload

module list

Intel Parallel Studio

```
source /scratch/app/modulos/intel-psxe-20[16|17|18|19|20].sh
```

Compiladores

GNU

Versões: 4.8.5, 6.5, 7.4 e 8.3

INTEL

Versões: 2016, 2017, 2018, 2019 e 2020

PGI

Versão 2016.5 e 2019.10 (Community)

Implementações MPI

OpenMPI (Bull)

Versão 2.0.4 - Implementação MPI compilada pela Bull

OpenMPI

Versões: 2.0.4, 3.1.4 e 4.0.1

Intel MPI

Versões 5.1, 2016, 2017, 2018, 2019 e 2020

Slurm

Versão 17.02

Onde encontrar referências?

<http://sdumont.lncc.br/>

<https://slurm.schedmd.com/archive/slurm-17.02.11>

Política de escalonamento

Backfill: prioridade, tempo na fila, quantidade de memória e etc.

Acesso

Somente os alunos que já possuem conta no SDumont poderão acessar o ambiente.

```
$ ssh meu.login@login.sdumont.lncc.br
```

Não serão distribuídas credenciais para os demais usuários.

SLURM: comandos básicos

- **sinfo:** visualiza informação sobre os nós e partições do SLURM
- **squeue:** visualiza informação sobre o status dos jobs e escalonamento
- **srun:** alocação de recursos e distribuição de tarefas
- **scontrol:** ferramenta para visualizar e/ou modificar o estado de um job
- **salloc:** obtém uma alocação para o job
- **sbatch:** submete um script para alocação em uma partição do SLURM
- **scancel:** cancela um job
- **sacct:** mostra informação de jobs já submetidos

Comandos básicos

sinfo

```
$ sinfo -s
```

PARTITION	AVAIL	TIMELIMIT	NODES (A/I/O/T)	NODELIST
treinamento	up	infinite	0/10/0/10	sdumont[5000-5009]
treinamento_gpu	up	infinite	0/2/0/2	sdumont[3000-3001]

Comandos básicos

sinfo

```
$ sinfo -s
```

PARTITION	AVAIL	TIMELIMIT	NODES (A/I/O/T)	NODELIST
cpu*	up	infinite	350/229/7/586	sdumont[1000-1503,3110-3159,5012-5043]
nvidia	up	infinite	119/71/2/192	sdumont[3006-3197]
phi	up	infinite	13/19/0/32	sdumont[5012-5043]
mesca2	up	infinite	0/1/0/1	sdumont57
cpu_small	up	infinite	350/229/7/586	sdumont[1000-1503,3110-3159,5012-5043]
nvidia_small	up	infinite	119/71/2/192	sdumont[3006-3197]
cpu_dev	up	infinite	287/220/7/514	sdumont[1000-1503,5044-5053]
nvidia_dev	up	infinite	119/71/2/192	sdumont[3006-3197]
phi_dev	up	infinite	13/19/0/32	sdumont[5012-5043]
cpu_scal	up	infinite	350/229/7/586	sdumont[1000-1503,3110-3159,5012-5043]
cpu_long	up	infinite	300/229/7/536	sdumont[1000-1503,5012-5043]
nvidia_scal	up	infinite	119/71/2/192	sdumont[3006-3197]
nvidia_long	up	infinite	119/71/2/192	sdumont[3006-3197]
all	inact	2-00:00:00	419/328/9/756	sdumont[1000-1503,3000-3197,5000-5053]

Comandos básicos

sinfo

Outras opções:

- -s
- --long
- --state
- -R

Comandos básicos

squeue

Outras opções

- -s
- -u (user)
- -A (account)
- -p

Comandos básicos

squeue

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
1767	cpu	mpiblast	labinfo	R	9:40:13	1	sdumont1128
1769	cpu	mpiblast	labinfo	R	9:35:10	1	sdumont1130
1770	cpu	mpiblast	labinfo	R	9:33:36	2	sdumont[1000-1001]
1772	cpu	mpiblast	labinfo	R	9:29:48	2	sdumont[1106-1107]
1777	cpu	brams-5.	xrpsouto	R	1:12:10	1	sdumont1126
1776	mesca2	TEST_bla	labinfo	R	8:21:18	1	sdumont57

SLURM: comandos básicos

- **sinfo**: visualiza informação sobre os nós e partições do SLURM
- **squeue**: visualiza informação sobre o status dos jobs e escalonamento
- **srun**: alocação de recursos e distribuição de tarefas
- **scontrol**: ferramenta para visualizar e/ou modificar o estado de um job
- **salloc**: obtém uma alocação para o job
- **sbatch**: submete um script para alocação em uma partição do SLURM
- **scancel**: cancela um job
- **sacct**: mostra informação de jobs já submetidos

Comandos básicos

srun

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

Alocação de recursos computacionais (“CPUs”:*sockets*/núcleos) a partir de um conjunto de nós. Neste exemplo, 3 nós com 8 recursos cada.

Node	n0	n1	n2
Allocate CPUS:	8	6	1

Alocação padrão dos nós é por blocos (***block***): o recurso é alocado em um nó, de modo que os recursos consecutivos estejam neste mesmo nó.

Outro método de alocação é o cíclico (***cyclic***): o recurso é alocado em um nós, de modo que os recursos consecutivos sejam alocados em nós consecutivos (escalonamento *round-robin*)

Comandos básicos

srun

```
$ srun --ntasks=6
```

Alocação de recursos computacionais a partir de um conjunto de *sockets*, dentro de um nó. Neste exemplo, um nó com 2 *sockets*, com 4 núcleos cada.

Socket	0	1
Allocate CPUS:	3	3

O método de alocação padrão aqui é o cíclico (**cyclic**): o recurso é atribuído para um *socket*, de modo que os recursos consecutivos sejam alocados em *sockets* consecutivos.

Outro método de alocação é por blocos (**block**): o recurso é atribuído para um *socket*, de modo que os tarefas consecutivos sejam alocados neste mesmo *socket*.

Comandos básicos

srun

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

Distribuição das **tarefas** entre os recursos computacionais (“CPUs”:*sockets*/núcleos) alocados. Neste exemplo, entre 3 nós com 8 recursos cada.

Node	n0	n1	n2
Allocate CPUS:	8	6	2
Distribution	0-3	4-6	7

Distribuição padrão aqui é por blocos (***block***): a tarefa é atribuída para um recurso, de modo que as tarefas consecutivas sejam atribuídas para este mesmo recurso.

Outro método de distribuição é a cíclica (***cyclic***): a tarefa é atribuída para um recurso, de modo que as tarefas consecutivas sejam distribuídas no recursos consecutivos (escalonamento *round-robin*)

Comandos básicos

srun

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

Distribuição de tarefas para os recursos alocados dentro de um nó.
Neste exemplo, em um nó contendo 2 sockets com 4 núcleos cada.

Node	n0							
Socket#	0				1			
CPI#	0	1	2	3	4	5	6	7
Bounds Task#	0	2	4	6	1	3	5	7

Distribuição padrão aqui é a cíclica (**cyclic**): a tarefa é atribuída para um *socket*, de modo que as tarefas consecutivas sejam distribuídas em núcleos de *sockets* consecutivos.

Outro método de distribuição é por blocos (**block**): a tarefa é atribuída para um *socket*, de modo que as tarefas consecutivas utilizem os núcleos deste mesmo *socket*.

Mapeamento (*mapping*) e vinculação (*binding*)

Mapping define como as tarefas são distribuídas:

- no nível de núcleos
- no nível de sockets
- no nível de nós

Binding define a afinidade das tarefas:

- por núcleo
- por socket
- por nó (sem *binding*)

NUMANode P#0 (32GB)

Package P#0

L3 (30MB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

Core P#0

Core P#1

Core P#2

Core P#3

Core P#4

Core P#5

Core P#8

Core P#9

Core P#10

Core P#11

Core P#12

Core P#13

PU P#0

PU P#1

PU P#2

PU P#3

PU P#4

PU P#5

PU P#6

PU P#7

PU P#8

PU P#9

PU P#10

PU P#11

NUMANode P#1 (32GB)

Package P#1

L3 (30MB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L2 (256KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1d (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

L1i (32KB)

Core P#0

Core P#1

Core P#2

Core P#3

Core P#4

Core P#5

Core P#8

Core P#9

Core P#10

Core P#11

Core P#12

Core P#13

PU P#12

PU P#13

PU P#14

PU P#15

PU P#16

PU P#17

PU P#18

PU P#19

PU P#20

PU P#21

PU P#22

PU P#23

srun: distribuição das tarefas

```
$ srun -p treinamento --nodes=3 --ntasks=6 --cpus-per-task=8 sleep 60
```

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
159893	treinamen	sleep	professo	R	0:10	3	sdumont[5000-5002]

```
$ scontrol --details show job 159893
```

```
NumNodes=3 NumCPUs=48 CPUs/Task=8 ReqB:S:C:T=0:0:*:*
```

```
Nodes=sdumont5000 CPU_IDs=0-23 Mem=64000
```

```
Nodes=sdumont5001 CPU_IDs=0-15 Mem=64000
```

```
Nodes=sdumont5002 CPU_IDs=0-7 Mem=64000
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 sleep 60 (forma compacta)
```

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
159893	treinamen	sleep	professo	R	0:10	3	sdumont[5000-5002]

```
$ scontrol --details show job 159893
```

```
NumNodes=3 NumCPUs=48 CPUs/Task=8 ReqB:S:C:T=0:0:*:*
```

```
Nodes=sdumont5000 CPU_IDs=0-23 Mem=64000
```

```
Nodes=sdumont5001 CPU_IDs=0-15 Mem=64000
```

```
Nodes=sdumont5002 CPU_IDs=0-7 Mem=64000
```

srun: distribuição das tarefas

```
$ scontrol --details show job 159893
JobId=159893 JobName=sleep
  UserId=professor(63001) GroupId=treinamento(61052)
  Priority=1791 Nice=0 Account=treinamento QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
  DerivedExitCode=0:0
  RunTime=00:00:27 TimeLimit=00:10:00 TimeMin=N/A
  SubmitTime=2018-02-25T07:54:18 EligibleTime=2018-02-25T07:54:18
  StartTime=2018-02-25T07:54:18 EndTime=2018-02-25T08:04:18
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=treinamento AllocNode:Sid=sdumont13:28904
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=sdumont[5000-5002]
  BatchHost=sdumont5000
  NumNodes=3 NumCPUs=48 CPUs/Task=8 ReqB:S:C:T=0:0:*:*
  Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
    Nodes=sdumont5000 CPU_IDS=0-23 Mem=64000
    Nodes=sdumont5001 CPU_IDS=0-15 Mem=64000
    Nodes=sdumont5002 CPU_IDS=0-7 Mem=64000
  MinCPUsNode=8 MinMemoryNode=62.50G MinTmpDiskNode=0
  Features=(null) Gres=(null) Reservation=(null)
  Shared=OK Contiguous=0 Licenses=(null) Network=(null)
  Command=sleep
  WorkDir=/prj/treinamento/professor
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block:cyclic --cpu_bind=cores,verbose sleep 1
```

```
cpu_bind=MASK - sdumont5000, task 0 0 [18472]: mask 0xff set  
cpu_bind=MASK - sdumont5000, task 1 1 [18473]: mask 0xff000 set  
cpu_bind=MASK - sdumont5000, task 2 2 [18474]: mask 0xf00f00 set  
cpu_bind=MASK - sdumont5001, task 3 0 [41701]: mask 0xff set  
cpu_bind=MASK - sdumont5001, task 4 1 [41702]: mask 0xff000 set  
cpu_bind=MASK - sdumont5002, task 5 0 [88601]: mask 0xff set
```

método de distribuição das tarefas entre os núcleos

-m [node:socket] ou --distribution[node:socket]

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m cyclic:cyclic --cpu_bind=cores,verbose sleep 1
```

```
cpu_bind=MASK - sdumont5000, task 0 0 [18543]: mask 0xff set  
cpu_bind=MASK - sdumont5001, task 1 0 [41762]: mask 0xff set  
cpu_bind=MASK - sdumont5002, task 2 0 [88656]: mask 0xff set  
cpu_bind=MASK - sdumont5000, task 3 1 [18544]: mask 0xff000 set  
cpu_bind=MASK - sdumont5001, task 4 1 [41763]: mask 0xff000 set  
cpu_bind=MASK - sdumont5000, task 5 2 [18545]: mask 0xf00f00 set
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block:cyclic --cpu_bind=cores,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list
```

```
0: cpu_bind=MASK - sdumont5000, task 0 0 [18849]: mask 0xff set  
1: cpu_bind=MASK - sdumont5000, task 1 1 [18850]: mask 0xff000 set  
2: cpu_bind=MASK - sdumont5000, task 2 2 [18851]: mask 0xf00f00 set  
3: cpu_bind=MASK - sdumont5001, task 3 0 [42038]: mask 0xff set  
4: cpu_bind=MASK - sdumont5001, task 4 1 [42039]: mask 0xff000 set  
5: cpu_bind=MASK - sdumont5002, task 5 0 [88908]: mask 0xff set
```

```
0: Cpus_allowed_list: 0-7  
1: Cpus_allowed_list: 12-19  
2: Cpus_allowed_list: 8-11,20-23  
3: Cpus_allowed_list: 0-7  
4: Cpus_allowed_list: 12-19  
5: Cpus_allowed_list: 0-7
```


srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block:cyclic --cpu_bind=cores,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list
```

```
0: cpu_bind=MASK - sdumont5000, task 0 0 [18849]: mask 0xff set  
1: cpu_bind=MASK - sdumont5000, task 1 1 [18850]: mask 0xff000 set  
2: cpu_bind=MASK - sdumont5000, task 2 2 [18851]: mask 0xf00f00 set  
3: cpu_bind=MASK - sdumont5001, task 3 0 [42038]: mask 0xff set  
4: cpu_bind=MASK - sdumont5001, task 4 1 [42039]: mask 0xff000 set  
5: cpu_bind=MASK - sdumont5002, task 5 0 [88908]: mask 0xff set
```

```
0: Cpus_allowed_list:      0-7  
1: Cpus_allowed_list:      12-19  
2: Cpus_allowed_list:      8-11,20-23 (diferentes sockets)  
3: Cpus_allowed_list:      0-7  
4: Cpus_allowed_list:      12-19  
5: Cpus_allowed_list:      0-7
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block: block --cpu_bind=cores,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list
```

```
0: cpu_bind=MASK - sdumont5000, task 0 0 [19142]: mask 0xff set  
1: cpu_bind=MASK - sdumont5000, task 1 1 [19143]: mask 0xff00 set  
2: cpu_bind=MASK - sdumont5000, task 2 2 [19144]: mask 0xff0000 set  
3: cpu_bind=MASK - sdumont5001, task 3 0 [42305]: mask 0xff set  
4: cpu_bind=MASK - sdumont5001, task 4 1 [42306]: mask 0xff00 set  
5: cpu_bind=MASK - sdumont5002, task 5 0 [89156]: mask 0xff set
```

```
0: Cpus_allowed_list: 0-7  
1: Cpus_allowed_list: 8-15  
2: Cpus_allowed_list: 16-23 (mesmo socket)  
3: Cpus_allowed_list: 0-7  
4: Cpus_allowed_list: 8-15  
5: Cpus_allowed_list: 0-7
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block:block --cpu_bind=socket,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list
```

```
0: cpu_bind=MASK - sdumont5000, task 0 0 [19301]: mask 0xffff set  
1: cpu_bind=MASK - sdumont5000, task 1 1 [19302]: mask 0xffffffff set  
2: cpu_bind=MASK - sdumont5000, task 2 2 [19303]: mask 0xffff000 set  
3: cpu_bind=MASK - sdumont5001, task 3 0 [42454]: mask 0xffff set  
4: cpu_bind=MASK - sdumont5001, task 4 1 [42455]: mask 0xffff set  
5: cpu_bind=MASK - sdumont5002, task 5 0 [89293]: mask 0xff set
```

```
0: Cpus_allowed_list:      0-11  
1: Cpus_allowed_list:      0-23 (sem garantia de ser no mesmo socket)  
2: Cpus_allowed_list:      12-23  
3: Cpus_allowed_list:      0-11  
4: Cpus_allowed_list:      12-15  
5: Cpus_allowed_list:      0-7
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N3 -n6 -c8 -m block:cyclic --cpu_bind=socket,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list
```

```
0: cpu_bind=MASK - sdumont5000, task 0 0 [18982]: mask 0xffff set  
1: cpu_bind=MASK - sdumont5000, task 1 1 [18983]: mask 0xffff000 set  
2: cpu_bind=MASK - sdumont5000, task 2 2 [18984]: mask 0xffffffff set  
3: cpu_bind=MASK - sdumont5001, task 3 0 [42155]: mask 0xff set  
4: cpu_bind=MASK - sdumont5001, task 4 1 [42156]: mask 0xff000 set  
5: cpu_bind=MASK - sdumont5002, task 5 0 [89018]: mask 0xff set
```

```
0: Cpus_allowed_list: 0-11  
1: Cpus_allowed_list: 12-23  
2: Cpus_allowed_list: 0-23  
3: Cpus_allowed_list: 0-7  
4: Cpus_allowed_list: 12-19  
5: Cpus_allowed_list: 0-7
```

srun: distribuição das tarefas

```
$ srun -p treinamento -N1 -n24 -cl -m block:cyclic --cpu_bind=cores,verbose  
--label cat /proc/self/status | grep Cpus_allowed_list | grep Cpus_allowed_list  
00: Cpus_allowed_list: 0  
01: Cpus_allowed_list: 12  
02: Cpus_allowed_list: 1  
03: Cpus_allowed_list: 13  
04: Cpus_allowed_list: 2  
05: Cpus_allowed_list: 14  
06: Cpus_allowed_list: 3  
07: Cpus_allowed_list: 15  
08: Cpus_allowed_list: 4  
09: Cpus_allowed_list: 16  
10: Cpus_allowed_list: 5  
11: Cpus_allowed_list: 17  
12: Cpus_allowed_list: 6  
13: Cpus_allowed_list: 18  
14: Cpus_allowed_list: 7  
15: Cpus_allowed_list: 19  
16: Cpus_allowed_list: 8  
17: Cpus_allowed_list: 20  
18: Cpus_allowed_list: 9  
19: Cpus_allowed_list: 21  
20: Cpus_allowed_list: 10  
21: Cpus_allowed_list: 22  
22: Cpus_allowed_list: 11  
23: Cpus_allowed_list: 23
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ tar xvf MC-SD01-I.tar
$ cd MC-SD01-I/openmpi/gnu/NPB3.3.1-MZ/
$ ls -Al
Changes.log
env_openmpi
NPB3.3-MZ-MPI
NPB3.3-MZ-OMP
NPB3.3-MZ-SER
README
$ cat env_openmpi
module load openmpi/gnu/2.0.4.2
$ source env_openmpi
$ mpicc -show
gcc -I/opt/mpi/openmpi-gnu/2.0.4.2/include -pthread -L/opt/mpi/openmpi-gnu/2.0.4.2/lib
-lmpi
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ cd NPB3.3-MZ-MPI/config/  
$ ls -A1  
  make.def  
  make.def.template  
  NAS.samples/  
  suite.def  
  suite.def.template  
$ cat make.def
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ cat suite.def
```

```
# config/suite.def
# This file is used to build several benchmarks with a single command.
# Typing "make suite" in the main directory will build all the benchmarks
# specified in this file.
# Each line of this file contains a benchmark name, class, and number
# of nodes. The name is one of "sp-mz", "bt-mz", and "lu-mz".
# The class is one of "S", "W", and "A" through "F".
# No blank lines.
# The following example builds serial sample sizes of all benchmarks.
#sp-mz      S      1
#lu-mz      S      1
#bt-mz      S      1
bt-mz       W      1
bt-mz       W      2
bt-mz       W      4
bt-mz       A      1
```


Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ cd ../  
$ pwd  
  ../MC-SD01-I/openmpi/gnu/NPB3.3.1-MZ/NPB3.3-MZ-MPI  
$ make suite --> compila o bechmark BT-MZ  
$ cd bin  
$ ls -Al  
bt-mz.W.1  
bt-mz.W.16  
bt-mz.W.2  
bt-mz.W.4  
bt-mz.W.8  
BULL_srun_openmpi.sh
```

SLURM: comandos básicos

- **sinfo**: visualiza informação sobre os nós e partições do SLURM
- **squeue**: visualiza informação sobre o status dos jobs e escalonamento
- **srun**: alocação de recursos e distribuição de tarefas
- **scontrol**: ferramenta para visualizar e/ou modificar o estado de um job
- **salloc**: obtém uma alocação para o job
- **sbatch**: submete um script para alocação em uma partição do SLURM
- **scancel**: cancela um job
- **sacct**: mostra informação de jobs já submetidos

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ srun -p treinamento -N1 -n1 ./bt-mz.W.1 --> submete o job com srun
```

-N1 (--nodes=1): define o número de nós a serem alocados

-n1 (--ntasks=1): define o número total de tarefas (MPI ranks) a serem executadas

```
$ salloc -p treinamento -N1 -n1 mpirun ./bt-mz.W.1 --> submete o job com salloc
```

```
salloc: Granted job allocation 105518
```

Class	=			W
Size	=	64x	64x	8
Iterations	=			200
Time in seconds	=			5.82
Total processes	=			1
Total threads	=			1
Mop/s total	=		2464.61	
Mop/s/thread	=		2464.61	

```
salloc: Relinquishing job allocation 105518
```

```
salloc: Job allocation 105518 has been revoked.
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ salloc -p treinamento -N1 -n2 mpirun ./bt-mz.W.2 --> submete o job com salloc
```

```
salloc: Granted job allocation 105519
```

Class	=		W
Size	=	64x	64x 8
Iterations	=		200
Time in seconds	=		2.96
Total processes	=		2
Total threads	=		2
Mop/s total	=		4851.15
Mop/s/thread	=		2425.57

```
salloc: Relinquishing job allocation 105519
```

```
salloc: Job allocation 105519 has been revoked.
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ srun -p treinamento -N1 -n2      ./bt-mz.W.2      --> submete o job com srun
```

Class	=		W
Size	=	64x	64x 8
Iterations	=		200
Time in seconds	=		2.96
Total processes	=		2
Total threads	=		2
Mop/s total	=		4851.15
Mop/s/thread	=		2425.57

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ srun -p treinamento -N1 -n1 -c2 ./bt-mz.W.1 --> para execução multi-thread
```

-N1 (--nodes=1): define o número de nós a serem alocados

-n1 (--ntasks=1): define o número total de tarefas (MPI ranks) a serem executadas

-c2 (--cpus-per-task=2): define o número de cpus por tarefa (MPI rank)

BT-MZ Benchmark Completed.

Class	=			W
Size	=	64x	64x	8
Iterations	=			200
Time in seconds	=		5.85	--> desempenho aquém do esperado
Total processes	=		1	
Total threads	=		1	

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ srun -p treinamento -N1 -n1 -c2 ./bt-mz.W.1 --> para execução multi-thread
```

-N1 (--nodes=1): define o número de nós a serem alocados

-n1 (--ntasks=1): define o número total de tarefas (MPI ranks) a serem executadas

-c2 (--cpus-per-task=2): define o número de cpus por tarefa (MPI rank)

BT-MZ Benchmark Completed.

Class	=			W
Size	=	64x	64x	8
Iterations	=			200
Time in seconds	=		5.85	--> definir var. de amb. OMP_NUM_THREADS
Total processes	=		1	
Total threads	=		1	

```
$ export OMP_NUM_THREADS=2 --> (igual a --cpus-per-task)
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ srun -p treinamento -N1 -n1 -c2 ./bt-mz.W.1 --> para execução multi-thread
```

-N1 (--nodes=1): define o número de nós a serem alocados

-n1 (--ntasks=1): define o número total de tarefas (MPI ranks) a serem executadas

-c2 (--cpus-per-task=2): define o número de cpus por tarefa (MPI rank)

BT-MZ Benchmark Completed.

Class	=			W
Size	=	64x	64x	8
Iterations	=			200
Time in seconds	=			3.03 --> redução de tempo
Total processes	=			1
Total threads	=			2

SLURM: comandos básicos

- **sinfo**: visualiza informação sobre os nós e partições do SLURM
- **squeue**: visualiza informação sobre o status dos jobs e escalonamento
- **srun**: alocação de recursos e distribuição de tarefas
- **scontrol**: ferramenta para visualizar e/ou modificar o estado de um job
- **salloc**: obtém uma alocação para o job
- **sbatch**: submete um script para alocação em uma partição do SLURM
- **scancel**: cancela um job
- **sacct**: mostra informação de jobs já submetidos

SLURM: comandos básicos

sbatch

- parâmetros na linha de comando
- parâmetros no script

principais opções de configuração:

--time (-t)

--nodes (-N)

--ntasks (-n)

--ntasks-per-node

--cpus-per-task (-c)

--partition (-p)

Exemplo: NAS Parallel Benchmark (BT-MZ)

BULL_srun_openmpi.sh

```
#!/bin/bash
```

```
#SBATCH --nodes=1                # here the number of nodes
#SBATCH --ntasks=1               # here total number of mpi tasks
#SBATCH --cpus-per-task=1        # number of cores per node
#SBATCH -p treinamento          # target partition
#SBATCH -J NPB_BT-MZ             # job name
#SBATCH --time=00:05:00          # time limit
#SBATCH --exclusive               # to have exclusive use of your nodes
```

```
echo "Cluster configuration:"
echo "==="
echo "Partition: " $SLURM_JOB_PARTITION
echo "Number of nodes: " $SLURM_NNODES
echo "Number of MPI processes: " $SLURM_NTASKS " (" $SLURM_NNODES " nodes)"
echo "Number of MPI processes per node: " $SLURM_NTASKS_PER_NODE
echo "Number of threads per MPI process: " $SLURM_CPUS_PER_TASK
echo "NPB Benchmark: " $1
echo "Benchmark class problem: " $2
```

Exemplo: NAS Parallel Benchmark (BT-MZ)

BULL_srun_openmpi.sh (cont.)

```
#####  
#           COMPILER           #  
#####  
module load openmpi/gnu/2.0.4.2  
  
DIR=$PWD  
  
bench=${1}  
class=${2}  
execfile="${bench}.${class}.${SLURM_NTASKS}"  
BIN=$DIR/${execfile}  
  
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK  
  
cd $DIR  
  
srun -n $SLURM_NTASKS $BIN  
  
dirdest="${bench}_${class}_MPI-${SLURM_NTASKS}_OMP-${SLURM_CPUS_PER_TASK}_JOBID-${SLURM_JOBID}"  
mkdir $dirdest  
mv slurm-${SLURM_JOBID}.out $dirdest/
```

Variáveis de ambiente do SLURM

Alguns exemplos:

`SLURM_JOB_PARTITION`

`SLURM_NNODES`

`SLURM_NTASKS`

`SLURM_NTASKS_PER_NODE`

`SLURM_CPUS_PER_TASK`

`SLURM_JOBID`

`SLURM_JOB_NODELIST`

`SLURM_SUBMIT_DIR`

Exemplo: NAS Parallel Benchmark (BT-MZ)

```
$ sbatch BULL_srun_openmpi.sh bt-mz W
```

```
Submitted batch job 105539
```

```
$ squeue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
105539	treinamen	NPB_BT-M	professo	R	0:05	1	sdumont5000

```
$ ls bt-mz_W_MPI-1_OMP-1_JOBID-105539/
```

```
slurm-105539.out    --> arquivo gerado pelo SLURM com a saída da aplicação
```

```
$ sbatch -N2 -n2 -c1 ./BULL_srun_openmpi.sh bt-mz W
```

Os parâmetros por linha de comando têm precedência sobre os definidos no script

```
$ ls bt-mz_W_MPI-2_OMP-1_JOBID-105540
```

```
slurm-105540.out
```

scontrol: alterando parâmetro do job

--dependency (-d): adia o início do job até que a dependência especificada seja satisfeita

```
$ sbatch BULL_srun_openmpi.sh bt-mz A
Submitted batch job 105558
$ sbatch -d afterany:105558 BULL_srun_openmpi.sh bt-mz A
Submitted batch job 105559
$ sbatch -d afterany:105559 BULL_srun_openmpi.sh bt-mz A
Submitted batch job 105560
$ sbatch -d afterany:105560 BULL_srun_openmpi.sh bt-mz A
Submitted batch job 105561
$ squeue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
105559	treinamen	NPB_BT-M	professo	PD	0:00	1	(Dependency)
105560	treinamen	NPB_BT-M	professo	PD	0:00	1	(Dependency)
105561	treinamen	NPB_BT-M	professo	PD	0:00	1	(Dependency)
105558	treinamen	NPB_BT-M	professo	R	0:26	1	sdumont5000

scontrol: alterando parâmetro do job

```
$ scontrol show jobid 105561
```

```
JobId=105561 JobName=NPB_BT-MZ
  UserId=professor(63001) GroupId=treinamento(61052)
  Priority=1791 Nice=0 Account=treinamento QOS=normal
  JobState=PENDING Reason=Dependency Dependency=afterany:105560
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:00:00 TimeLimit=00:05:00 TimeMin=N/A
  SubmitTime=2017-07-30T17:42:04 EligibleTime=Unknown
  StartTime=Unknown EndTime=Unknown
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=treinamento AllocNode:Sid=sdumont14:19952
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=(null)
  NumNodes=1-1 NumCPUs=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:1
  Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
  MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
  Features=(null) Gres=(null) Reservation=(null)
  Shared=0 Contiguous=0 Licenses=(null) Network=(null)
```


scontrol: alterando parâmetro do job

```
$ scontrol update JobId=105561 Partition=treinamento_gpu
```

```
$ scontrol show jobid 105561
```

```
JobId=105561 JobName=NPB_BT-MZ
  UserId=professor(63001) GroupId=treinamento(61052)
  Priority=1791 Nice=0 Account=treinamento QOS=normal
  JobState=PENDING Reason=Dependency Dependency=afterany:105560
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:00:00 TimeLimit=00:05:00 TimeMin=N/A
  SubmitTime=2017-07-30T17:42:04 EligibleTime=Unknown
  StartTime=Unknown EndTime=Unknown
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=treinamento_gpu AllocNode:Sid=sdumont14:19952
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=(null)
  NumNodes=1-1 NumCPUs=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:1
  Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
  MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
  Features=(null) Gres=(null) Reservation=(null)
  Shared=0 Contiguous=0 Licenses=(null) Network=(null)
```

SLURM: comandos básicos

- **sinfo**: visualiza informação sobre os nós e partições do SLURM
- **squeue**: visualiza informação sobre o status dos jobs e escalonamento
- **srun**: alocação de recursos e distribuição de tarefas
- **scontrol**: ferramenta para visualizar e/ou modificar o estado de um job
- **salloc**: obtém uma alocação para o job
- **sbatch**: submete um script para alocação em uma partição do SLURM
- **scancel**: cancela um job
- **sacct**: mostra informação de jobs já submetidos

sacct: mostra informação de jobs já submetidos

\$ sacct

158796	mpirun	treinamen+	treinamen+	1	FAILED	1:0
158798	mpirun	treinamen+	treinamen+	2	COMPLETED	0:0
158798.0	orted		treinamen+	2	COMPLETED	0:0
158802	bt-mz.A.2	treinamen+	treinamen+	2	CANCELLED+	0:0
158803	bt-mz.A.2	treinamen+	treinamen+	2	COMPLETED	0:0
158804	bt-mz.W.2	treinamen+	treinamen+	2	COMPLETED	0:0
158809	mpirun	treinamen+	treinamen+	2	COMPLETED	0:0
158809.0	orted		treinamen+	2	COMPLETED	0:0
158810	bt-mz.W.2	treinamen+	treinamen+	2	COMPLETED	0:0
158811	bt-mz.W.2	treinamen+	treinamen+	2	COMPLETED	0:0

sacct: mostra informação de jobs já submetidos

```
$ sacct -j 158811
```

JobID	JobName	Partition	Account	AllocCPUS	State	ExitCode
158811	bt-mz.W.2	treinamen+	treinamen+	2	COMPLETED	0:0

sacct: mostra informação de jobs já submetidos

```
$ sacct -e
```

AllocCPUS	AllocGRES	Account	AssocID
AveCPU	AveCPUFreq	AveDiskRead	AveDiskWrite
AvePages	AveRSS	AveVMSize	BlockID
Cluster	Comment	ConsumedEnergy	ConsumedEnergyRaw
CPUTime	CPUTimeRAW	DerivedExitCode	Elapsed
Eligible	End	ExitCode	GID
Group	JobID	JobIDRaw	JobName
Layout	MaxDiskRead	MaxDiskReadNode	MaxDiskReadTask
MaxDiskWrite	MaxDiskWriteNode	MaxDiskWriteTask	MaxPages
MaxPagesNode	MaxPagesTask	MaxRSS	MaxRSSNode
MaxRSSTask	MaxVMSize	MaxVMSizeNode	MaxVMSizeTask
MinCPU	MinCPUNode	MinCPUTask	NCPUS
NNodes	NodeList	NTasks	Priority
Partition	QOS	QOSRAW	ReqCPUFreq
ReqCPUS	ReqGRES	ReqMem	Reservation
ReservationId	Reserved	ResvCPU	ResvCPURAW
Start	State	Submit	Suspended
SystemCPU	Timelimit	TotalCPU	UID
User	UserCPU	WCKey	WCKeyID

sacct: mostra informação de jobs já submetidos

```
$ sacct -j 158811 --format=JobID,JobName,Elapsed,NodeList
```

JobID	JobName	Elapsed	NodeList
158811	bt-mz.W.2	00:00:04	sdumont5000