

Optimizing O2 Jobs



HARVARD
MEDICAL SCHOOL

Information Technology

Optimize YOUR Code

CPU Time

- Run a performance test, be aware of your code bottlenecks.
- Avoid wasting CPU cycles, remove any unnecessary operations.
- Whenever possible avoid computationally expensive operations such as IF blocks.
- Use the most efficient option to solve your problems.

MEMORY

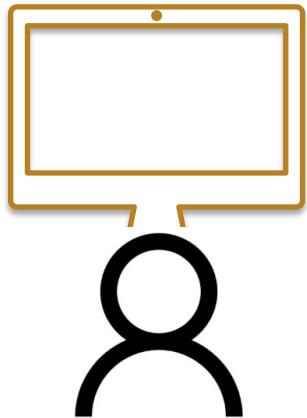
- Clean up any variable that is no longer needed.
- Avoid creating large middle-step variables when possible.
- Do not overestimate when pre-allocating variables.
- Load from large files only what is needed.



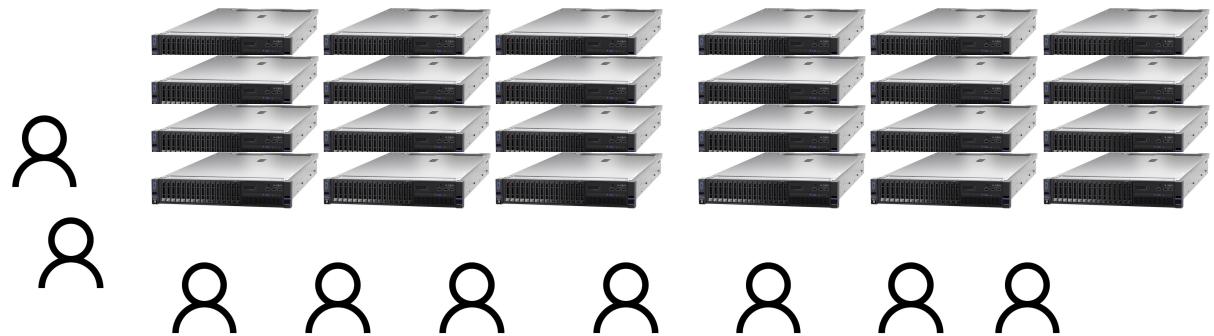
HARVARD
MEDICAL SCHOOL

Information Technology²

Optimize how YOU run on a shared environment



VS



Desktop Scenario

Limited amount of computational resources available only to a single user.
Use all you can, even if not at peak efficiency.

HPC Cluster scenario

Large amount of computational resources shared between many users.
Using resources efficiently will benefit you and everyone in the system.



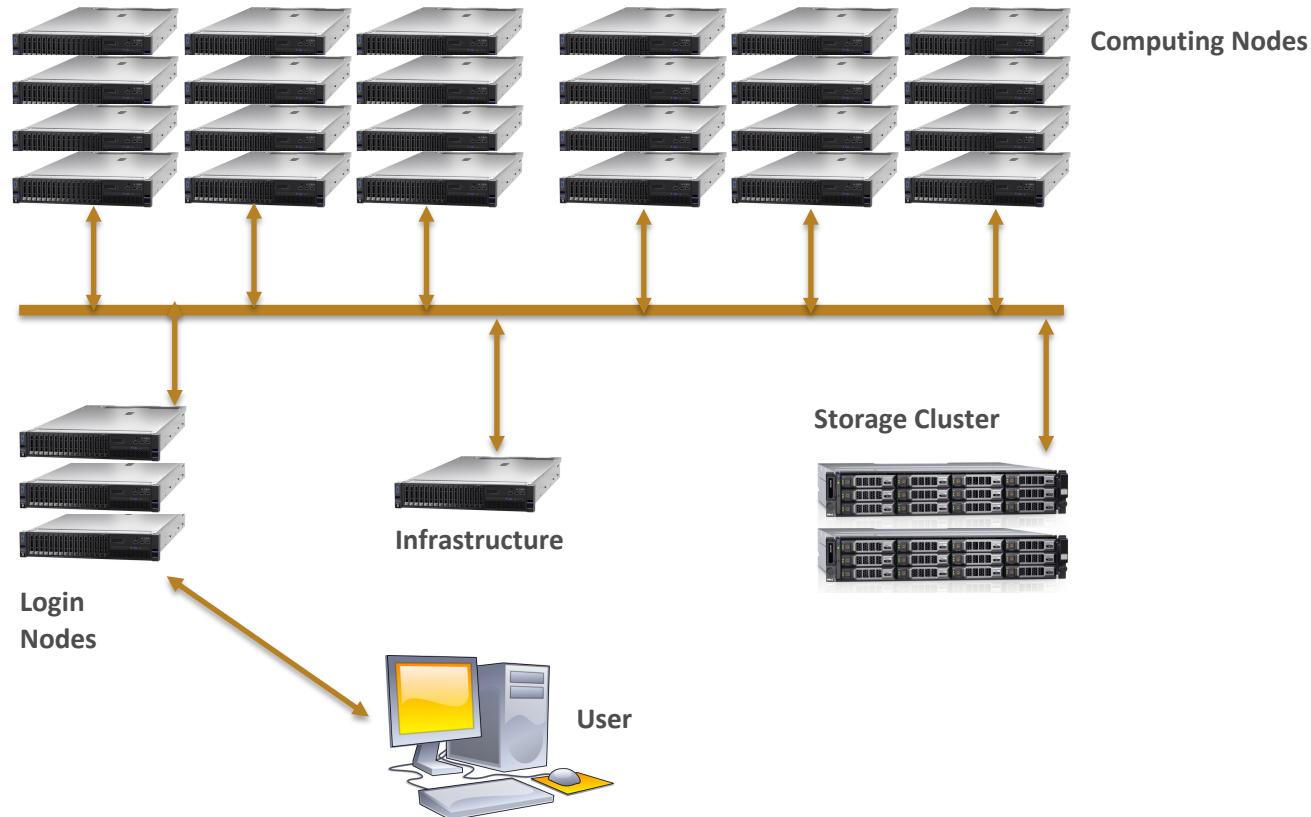
**HARVARD
MEDICAL SCHOOL**

Information Technology 3

The O2 HPC cluster

The O2 cluster is composed of:

- *Login Nodes*, to connect remotely and submit jobs.
- *Compute nodes*, with high memory and multiple cores, to process many jobs simultaneously
- *Storage nodes*, for hosting the users data in O2
- *Scheduler*, handles O2 resources and dispatches jobs on computing nodes



O2 currently includes:

— **390 Compute Nodes — 12260 CPU cores — 106 TiB of RAM — 136 GPU cards**



**HARVARD
MEDICAL SCHOOL**

Information Technology⁴

The Slurm scheduler

The Slurm scheduler handles resources allocation on the O2 cluster and also provides accounting information about past jobs and cluster utilization. Slurm periodically dispatches O2 jobs in two different ways:

- ***Direct Scheduling***

At every cycle Slurm tries to dispatch pending jobs with the highest priority on available idle resources (memory, cpu cores, gpu). If it cannot find available resources to dispatch a given high priority job, Slurm will determine *when* and *where* the required resources will be available, and will schedule a future start time and node allocation for the high priority job.

- ***Backfill Scheduling***

At every cycle Slurm tries to dispatch pending jobs with lower priority on available idle resources, as long as dispatching those lower priority jobs will not impact the expected start time of the higher priority jobs

In either cycle Slurm checks only a limited number of jobs per user, and it uses the “priority value” of each job as sorting criteria.



The Slurm scheduler

A simplified cpu-only scenario to understand jobs scheduling:

Sorted by priority ↓

- JOB_5, 8 cpu cores for 12hrs
- JOB_1, 6 cpu cores for 12hrs
- JOB_4, 4 cpu cores for 12hrs
- JOB_3, 2 cpu cores for 3hrs
- JOB_6, 8 cpu cores for 12hrs
- JOB_2, 5 cpu cores for 12hrs

JOB_3 would have been pending for at least 7 hours without backfilling, many O2 jobs are backfilled not directly dispatched !!!

Scheduled to start in 4 hours

Dispatched

Backfilled (dispatched)



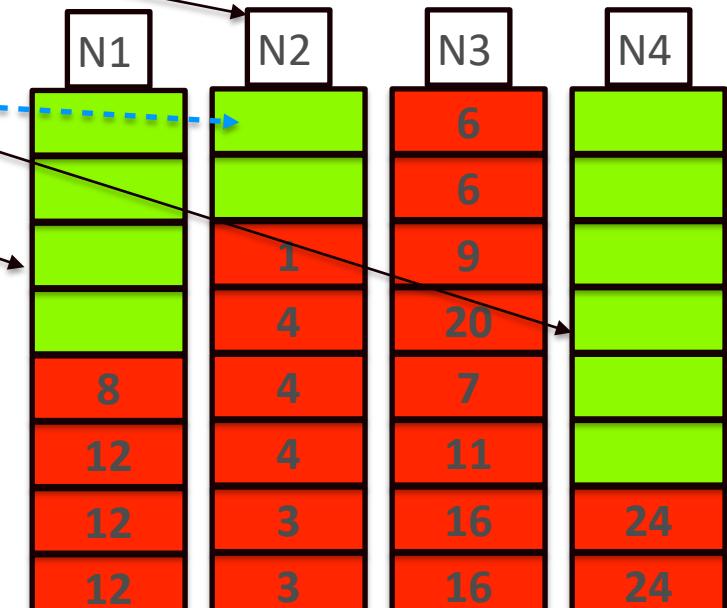
Idle CPU core



Allocated CPU core

WallTime allocated hours remaining

```
*****
sdiag output at Fri Sep 10 10:54:14 2021 (1631285654)
Data since    Thu Sep 09 20:00:00 2021 (1631232000)
*****
...
Jobs submitted: 39147
Jobs started:  37846
Jobs completed: 38946
Jobs canceled: 149
Jobs failed:   0
...
Backfilling stats
    Total backfilled jobs (since last slurm start): 317877
    Total backfilled jobs (since last stats cycle start): 18049
...
```



Cluster Computing Nodes



HARVARD
MEDICAL SCHOOL

Information Technology6

The Slurm scheduler

The job ***priority value*** is based on several factors:

- **Age:** proportional to the job's pending time (since eligible), it has a significant impact on overall priority.
- **JobSize:** relates to the resources requested, has negligible contribution.
- **Partition***: based on the partition used, this factor has a minor impact on jobs overall priority
- **QOS:** “Quality Of Service”, is a custom factor priority, by default set to zero. It can have a significant impact on job's priority.
- **TRES:** not currently active, should always be zero
- **FairShare:** It is the most important factor in determining the overall job priority. It represents the fraction of resources consumed by the user in the last 48 hours respect to all the other users in the cluster.

* The scheduler tries to dispatch first jobs submitted to partitions ***interactive*** and ***priority***, independently by the jobs priority values.



HARVARD
MEDICAL SCHOOL

Information Technology⁷

Why should you optimize your jobs ?

There are three good reasons why you should invest some time and optimize how your jobs are being executed on a cluster environment.

1. Reduce pending time, increase your throughput and overall processing speed



2. Reduce the cost, if applicable



3. Avoid locking idle resources that could be allocated to other users

INEFFICIENT

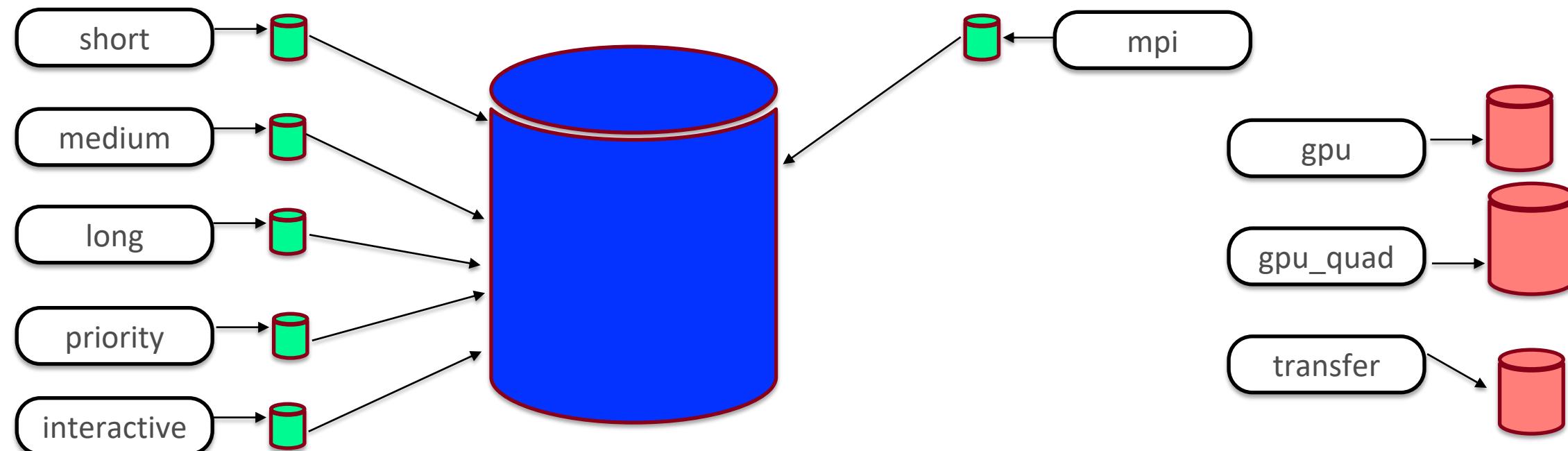


**HARVARD
MEDICAL SCHOOL**

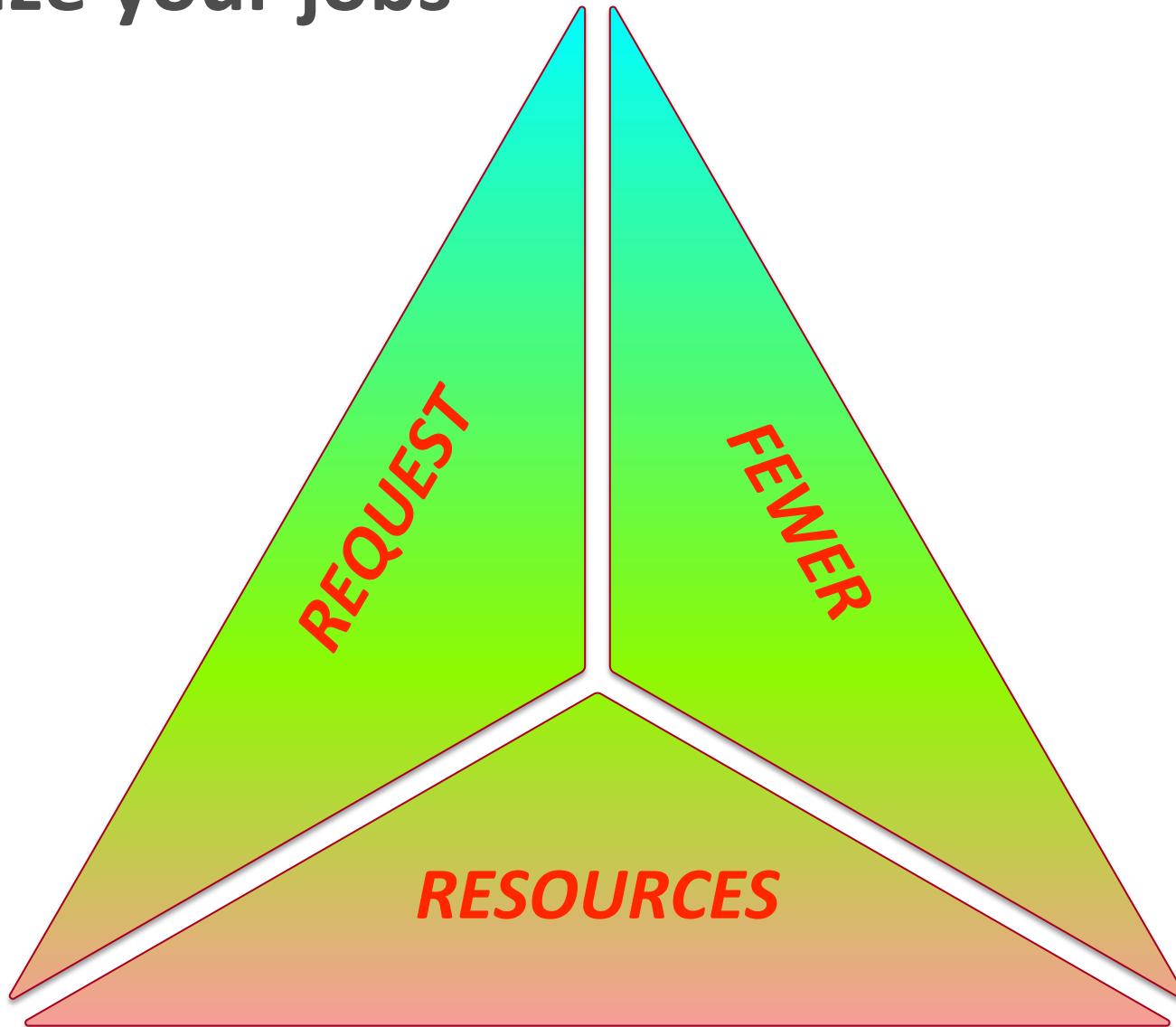
Information Technology 8

How NOT to optimize your jobs

The O2 cluster currently includes 12 different shared partitions (i.e. queues), one might think to optimize jobs by requesting arbitrary wall time and submitting the jobs across different partitions. This is not a good idea because those partitions share the vast majority of the computing resources.



How to optimize your jobs



HARVARD
MEDICAL SCHOOL

Information Technology 10

How to optimize your jobs

There are four factors that strongly impact jobs' efficiency in O2

- Wasted resources on FAILED jobs



There were 6,44M of CPU hours from FAILED or CANCELLED jobs in 2021

- Allocated but unused RAM



- Allocated but unused CPU cores



- Overestimated or arbitrary wall-time



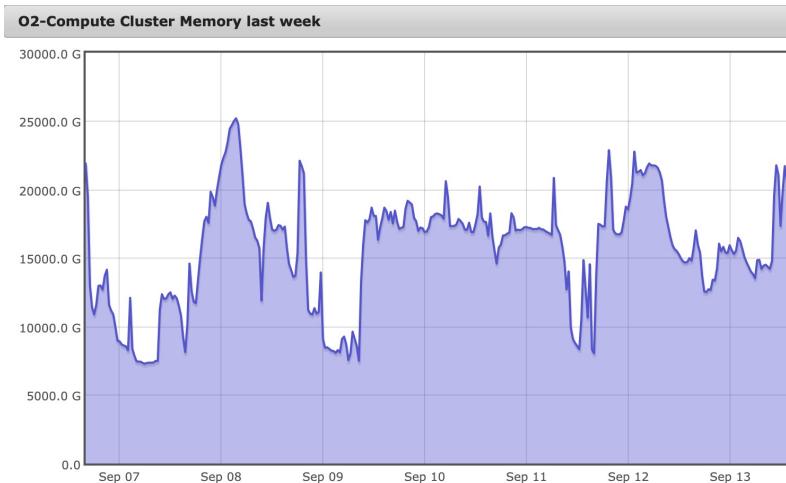
HARVARD
MEDICAL SCHOOL

Information Technology 11

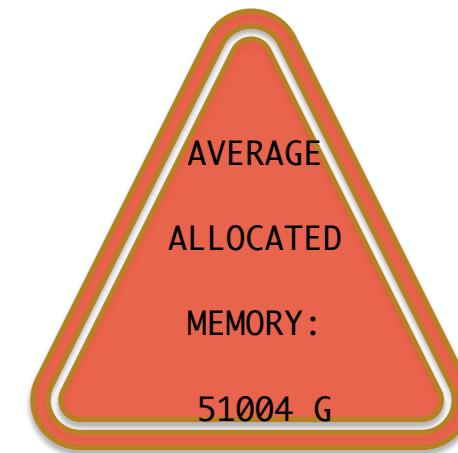
Allocating memory in O2 jobs

The scheduler allocates by default 1000 MiB of RAM for each requested CPU core.

Users can request a custom amount of memory per job using the flag **--mem=** and very often request much more memory than what is really needed.



VS



Allocating more memory than needed will:

- Unnecessarily increase the pending time of each job
- Consume more of your fair share, which will lower your jobs' priority
- Increase the utilization cost (when applicable)
- Stop other people's jobs from starting



HARVARD
MEDICAL SCHOOL

Information Technology 12

Allocating memory in O2 jobs

There are two typical behaviors we see on the cluster:

(A) users over-allocated memory for all jobs

(B) users over-allocated memory for 99% of the jobs. In this case it is always better to plan for the 99% and resubmit the failed 1%

In either case the users below would have increased their throughput and reduced the cost by requesting no more than 1/2 of the memory originally requested.

User	Njobs	AvgReqMem(GB)	AvgUsedMem(GB)	AbsMaxUsed(GB)	Njob>1/2Req	AvgCoreEff(%)	AvgWallTimeUsed(%)
A	13	199.89	24.96	31.41	0	98.86	27.42
	61	48.24	0.17	1.00	0	98.61	3.92
	14	100.50	4.10	6.28	0	97.74	7.02
	430	50.00	1.39	3.99	0	99.61	3.93
	49	174.99	5.20	5.90	0	57.27	2.26
	7576	50.66	0.86	19.82	0	82.23	11.52
User	Njobs	AvgReqMem(GB)	AvgUsedMem(GB)	AbsMaxUsed(GB)	Njob>1/2Req	AvgCoreEff(%)	AvgWallTimeUsed(%)
B	9307	24.68	3.93	81.19	17	95.78	3.09
	252	62.20	11.02	32.53	1	85.54	6.26
	1110	113.92	27.36	181.75	13	27.30	3.67

week_end_2021-09-13 (END)



Allocating CPU cores in O2 jobs

Users can request multiple CPU cores for their jobs. Requesting more cores will not automatically increase the job's performance!

The user requests several CPU assuming the job will run faster but...

- 1. The additional cores remain idle and no speedup is achieved. If your executable supports parallelization, make sure to pass any specific flag required.*
- 2. The parallelization does not scale very well and the overall benefit - cost balance is negative.*

User	Tot.Jobs	AverageCPU	Efficiency
<hr/>			
	176	2.98	31.88 %
	126	9.49	14.88 %
	57	3.75	17.55 %
	301	7.86	68.09 %
	146	5.50	31.11 %
	31	6.00	53.57 %
	5	20.00	30.06 %

Allocating more CPU than needed will again:

- Unnecessarily increase the pending time
- Consume fair share, and lower jobs' priority
- Increase the utilization cost (when applicable)

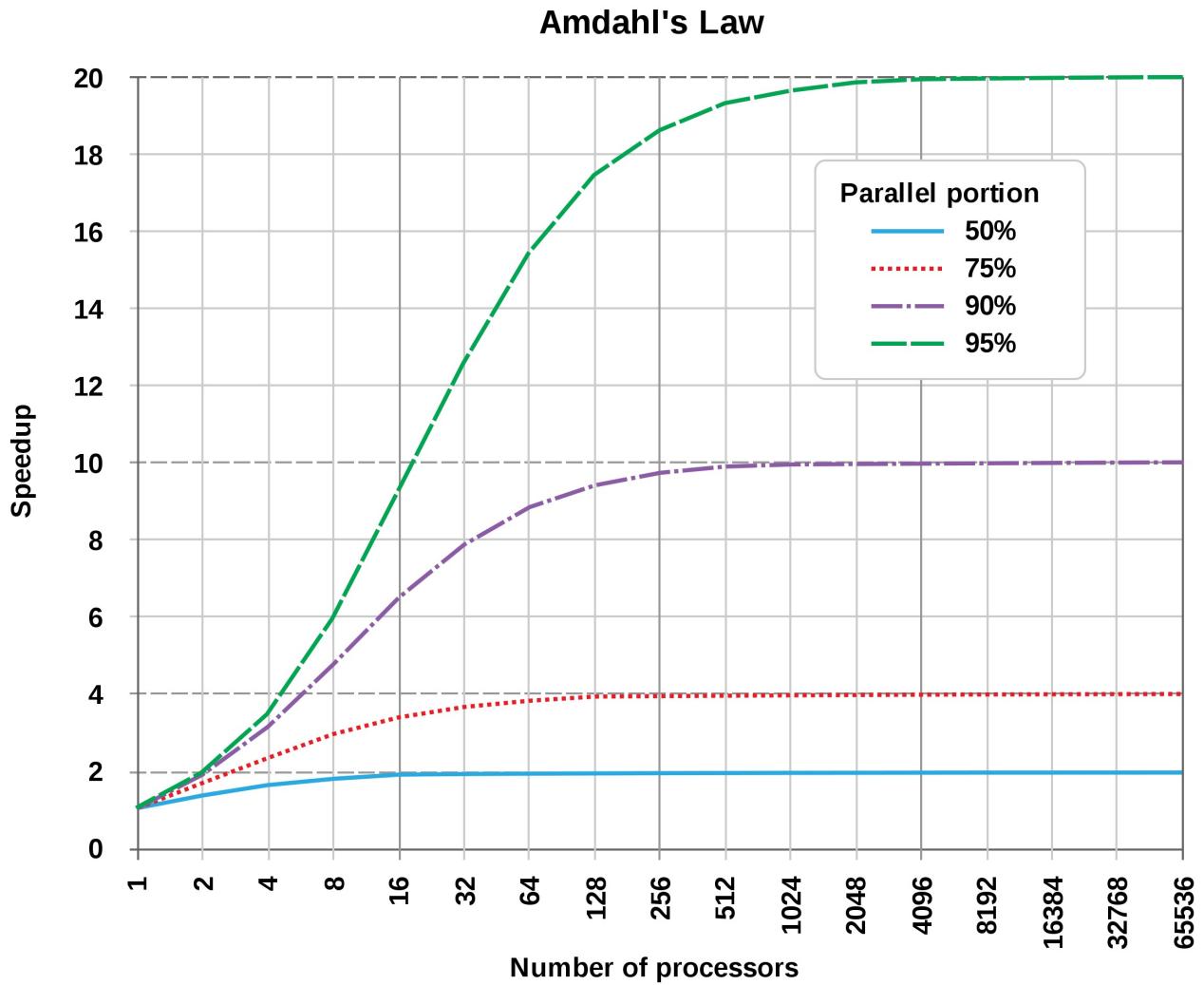


Allocating CPU cores in O2 jobs

$$SpeedUp = \frac{1}{\frac{P}{N} + S}$$

if $N \rightarrow \infty$

$$SpeedUp \rightarrow \frac{1}{1 - P}$$



Allocating wall-time in O2 jobs

The majority of users tend to overestimate the wall time required, often setting it to the maximum value allowed.

Requesting too much wall-time does not impact fair-share or cost, however it reduces the possibility of jobs being backfilled.

Combine together very small jobs in larger batches, with each job running at least for 15 minutes, very short jobs can spend more time pending than running.

User	TotJobs	Njobs5X	%_Req_Time_Used	AVGelapsed		%_Partition_used				
6925	0	0	0.31 %	0.04		short	100.00 %			
15965	20	20	1.61 %	0.10		short	99.97 %	medium	0.03 %	
2302	172	172	1.80 %	0.76		short	61.21 %	medium	38.66 %	priority 0.13 %
11064	2	2	1.96 %	0.02		short	99.99 %	medium	0.01 %	



Allocating resources efficiently in O2 jobs

How can you check memory, cpu and wall-time utilization to optimize your future jobs?

1. Check the summary reports that RC sends weekly and adjust your required resources accordingly

Hello Raffaele,

Below you can find a short report about the memory, CPU and wall-time efficiencies for the **non-interactive** jobs you *successfully completed* (had state CD) over the last 7 days (since 2021-09-06).

We encourage you to check and, if needed, adjust the amount of memory, CPU and wall-time requested in order to maximize your job throughput and limit the amount of resources allocated but not used. Asking for fewer resources will usually make your jobs start running sooner, and will help other users' jobs run sooner as well.

User	Njobs	AvgReqMem (GB)	AvgUsedMem (GB)	AbsMaxUsed (GB)	Njob>1/2Req	AvgCoreEff (%)	AvgWallTimeUsed (%)
rp189	171	0.98	0.52	0.57	152	68.33	13.10

LEGEND

User	= your username (rp189)
Njobs	= Number of jobs you ran over the last 7 days marked as completed. Failed, canceled and timed out jobs are not considered.
AvgReqMem	= weighted average memory requested (in GB) by your jobs
AvgUsedMem	= weighted average memory used (in GB) by your jobs
AbsMaxUsed	= Absolute maximum amount of memory (in GB) used by your jobs
Njob>1/2Req	= Number of jobs that used at least 1/2 or more memory than what was requested by the job. Ideally this number should match Njobs
AvgCoreEff(%)	= Average CPU efficiency. Indicates how much CPU was actually used respect to the reserved CPU. Ideally this number should be at least > 70
AvgWallTimeUsed(%)	= Average percent of Wall-Time used. Indicates how much time your jobs actually ran respect to the requested wall-time. Ideally this number should be at least > 50

To get detailed information about resource usage for each job you can run the command O2sacct (O2sacct --help to get more info), and if you have any questions please let us know at rchelp@hms.harvard.edu (or respond to this email).

More information about job accounting and O2sacct can be found here: <https://wiki.rc.hms.harvard.edu/display/O2/Get+information+about+current+and+past+jobs#Getinformationaboutcurrentandpastjobs-O2sacct>

HMS Research Computing



HARVARD
MEDICAL SCHOOL

Information Technology 17

Allocating resources efficiently in O2 jobs

How can you check memory, cpu and wall-time utilization to optimize your future jobs?

2. Use our *O2sacct* wrapper to see detailed information about a custom subset of jobs, see <https://harvardmed.atlassian.net/l/c/XqnwHpgP> for more details.

rp189@login03:~ 02sacct	JobID	Partition	State	Nodelist	Start	TimeLimit	Elapsed	CPUefficiency_%	AllocTRES	MaxMemoryUsed
	40740675	priority	COMPLETED	compute-e-16-233	2021-09-16T00:15:14	00:05:00	00:00:32	68.75	billing=1,cpu=1,mem=0.98G,node=1	
	40740675.batch		COMPLETED	compute-e-16-233	2021-09-16T00:15:14		00:00:32	68.75	cpu=1,mem=0.98G,node=1	0.53G
	40740675.extern		COMPLETED	compute-e-16-233	2021-09-16T00:15:14		00:00:32	0.00	billing=1,cpu=1,mem=0.98G,node=1	0
	40741499	priority	COMPLETED	compute-a-16-164	2021-09-16T01:15:34	00:05:00	00:00:41	70.73	billing=1,cpu=1,mem=0.98G,node=1	
	40741499.batch		COMPLETED	compute-a-16-164	2021-09-16T01:15:34		00:00:41	70.73	cpu=1,mem=0.98G,node=1	0.53G
	40741499.extern		COMPLETED	compute-a-16-164	2021-09-16T01:15:34		00:00:42	0.00	billing=1,cpu=1,mem=0.98G,node=1	0
	40743060	priority	COMPLETED	compute-a-16-162	2021-09-16T02:15:04	00:05:00	00:00:41	70.73	billing=1,cpu=1,mem=0.98G,node=1	
	40743060.batch		COMPLETED	compute-a-16-162	2021-09-16T02:15:04		00:00:41	70.73	cpu=1,mem=0.98G,node=1	0.55G
	40743060.extern		COMPLETED	compute-a-16-162	2021-09-16T02:15:04		00:00:41	0.00	billing=1,cpu=1,mem=0.98G,node=1	0
	40743774	priority	COMPLETED	compute-a-16-162	2021-09-16T03:15:50	00:05:00	00:00:39	76.92	billing=1,cpu=1,mem=0.98G,node=1	
	40743774.batch		COMPLETED	compute-a-16-162	2021-09-16T03:15:50		00:00:39	76.92	cpu=1,mem=0.98G,node=1	0.56G
	40743774.extern		COMPLETED	compute-a-16-162	2021-09-16T03:15:50		00:00:39	0.00	billing=1,cpu=1,mem=0.98G,node=1	0



Allocating resources efficiently in O2 jobs

How can you check memory, cpu and wall-time utilization to optimize your future jobs?

3. Build your own custom query using Slurm native **sacct** command.

```
sacct -u $USER --units=G --format=<field1,field2,field3,...>
```

4. Ask for RCC help

rchelp@hms.harvard.edu

<https://it.hms.harvard.edu/our-services/research-computing>

login04:~ sacct -e			
Account	AdminComment	AllocCPUS	AllocGRES
AllocNodes	AllocTRES	AssocID	AveCPU
AveCPUFreq	AveDiskRead	AveDiskWrite	AvePages
AveRSS	AveVMSIZE	BlockID	Cluster
Comment	Constraints	ConsumedEnergy	ConsumedEnergyRaw
CPUTime	CPUTimeRAW	DBIndex	DerivedExitCode
Elapsed	ElapsedRaw	Eligible	End
ExitCode	Flags	GID	Group
JobID	JobIDRaw	JobName	Layout
MaxDiskRead	MaxDiskReadNode	MaxDiskReadTask	MaxDiskWrite
MaxDiskWriteNode	MaxDiskWriteTask	MaxPages	MaxPagesNode
MaxPagesTask	MaxRSS	MaxRSSNode	MaxRSSTask
MaxVMSIZE	MaxVMSIZENode	MaxVMSIZETask	McsLabel
MinCPU	MinCPUNode	MinCPUTask	NCPUS
NNodes	NodeList	NTasks	Priority
Partition	QOS	QOSRAW	Reason
ReqCPUFreq	ReqCPUFreqMin	ReqCPUFreqMax	ReqCPUFreqGov
ReqCPUS	ReqGRES	ReqMem	ReqNodes
ReqTRES	Reservation	ReservationId	Reserved
ResvCPU	ResvCPURAW	Start	State
Submit	Suspended	SystemCPU	SystemComment
Timelimit	TimelimitRaw	TotalCPU	TRESUsageInAve
TRESUsageInMax	TRESUsageInMaxNode	TRESUsageInMaxTask	TRESUsageInMin
TRESUsageInMinNode	TRESUsageInMinTask	TRESUsageInTot	TRESUsageOutAve
TRESUsageOutMax	TRESUsageOutMaxNode	TRESUsageOutMaxTask	TRESUsageOutMin
TRESUsageOutMinNode	TRESUsageOutMinTask	TRESUsageOutTot	UID
User	UserCPU	WCKey	WCKeyID
WorkDir			



HARVARD
MEDICAL SCHOOL

Information Technology 19

Job priority rewards

Users that do not waste too many computational resources are rewarded with an extra priority QoS, equivalent to 30% extra fair-share.

Our goal is for O2 users to check resources actually consumed and adjust job submissions accordingly.

It is often not possible to predict exactly memory and/or run time, reasonable memory and wall-time buffers are OK and advised!

More details are available on our wiki page at <https://harvardmed.atlassian.net/l/c/03G8RYP1>

```
Name|Priority|GraceTime|Preempt|PreemptExemptTime|PreemptMode|Flags|UsageThres|UsageFactor|
normalplus|3|00:00:00|||cluster|||1.000000
~
```



HARVARD
MEDICAL SCHOOL

Information Technology 20

Questions ?



HARVARD
MEDICAL SCHOOL

Information Technology²¹