High Throughput Computing

Broderick Gardner
SchedMD

SLUG 2019
Focus

through·put
ˈTHrôˌpoot/
noun
noun: throughput; plural noun: throughputs
the amount of material or items passing through a system or process.

How many small, short jobs can we push through the cluster per minute?
Topics

General tuning and recommendations

Test Configuration

Submission rate

Tunable parameters

Feature impact
General Recommendations

- Prioritize higher clock speed over core count for `slurmctld` host
  - `slurmctld` is highly threaded but not highly concurrent
  - So Core i7/i9 over Xeon
General Recommendations

- **Prioritize higher clock speed over core count for `slurmctld` host**
  - `slurmctld` is highly threaded but not highly concurrent
  - So Core i7/i9 over Xeon

- **StateSaveLocation should be on a dedicated fast filesystem**
  - Particularly if it is shared with a backup controller in a High Availability configuration
  - IOPS to this filesystem is one of the main bottlenecks to job throughput
    - At least 1 directory and 2 files created per job
General Recommendations

- Prioritize higher clock speed over core count for slurmctld host
  - slurmctld is highly threaded but not highly concurrent
  - So Core i7/i9 over Xeon
- StateSaveLocation should be on a dedicated fast filesystem
  - Particularly if it is shared with a backup controller in a High Availability configuration
  - IOPS to this filesystem is one of the main bottlenecks to job throughput
    - At least 1 directory and 2 files created per job
- SlurmdSpoolDir should be local to the node, eg. a tmpfs
- Reduce the debug level of slurmctld and slurmd, particularly if they log to a slow or nonlocal filesystem
Accounting and Throughput

Database and slurmdbd

- Reasonably fast filesystem
- InnoDB parameters
  - Recommended minimums to the right
- CommitDelay
  - Seconds between database commits
  - From man slurmdbd.conf: “In testing, 1 second improves the slurmdbd performance dramatically and reduces overhead.”

- Following these guidelines, the slurmdbd should not bottleneck job throughput

# /etc/my.cnf
[mysqld]
innodb_buffer_pool_size=1G
innodb_log_file_size=64M
innodb_lock_wait_timeout=900

# slurmdbd.conf
CommitDelay=1
Accounting Throughput Impact

- Disables step accounting
- My testing shows minimal difference
- You could try this to see if accounting is your bottleneck

# slurm.conf
AccountingStorageEnforce=nosteps
Submission Rate

Array Jobs FTW

```bash
# slurm.conf
MaxArraySize=100000

$ sbatch --array=1-10000 job.sh

for i in {1..10000000}; do
    sbatch --partition=smd ntpbatch.sh &
done
```
Tunable Parameters - Main Scheduler

max_rpc_cnt
- Keep high to allow scheduler to run under high slurmctld RPC load

sched_min_interval
- microseconds
- Rate limit starting the quick scheduler due to many jobs ending

# slurm.conf
SchedulerParameters=max_rpc_cnt=400,\ sched_min_interval=50000
Tunable Parameters - Main Scheduler

sched_max_job_start
- Set to a reasonable number of jobs started at once

batch_sched_delay
- in seconds
- Allow delayed starting of batch jobs during high submission rate

```bash
# slurm.conf
SchedulerParameters=max_rpc_cnt=400,\n  sched_min_interval=50000,\n  sched_max_job_start=300,\n  batch_sched_delay=20
```
Tunable Parameters - Backfill Scheduler

Do you need it? Short, tiny jobs will not benefit. If you do, here are some parameters to consider.

bf_resolution
- in seconds
- Set high to speed up scheduler

```bash
# slurm.conf
SchedulerParameters=max_rpc_cnt=400,
sched_min_interval=50000,
sched_max_job_start=300,
batch_sched_delay=20,
bf_resolution=600,
```
Tunable Parameters - Backfill Scheduler

bf_min_prio_reserve

- Prefer system utilization over priority below a threshold priority

bf_min_age_reserve

- in seconds
- Prefer system utilization over priority for jobs pending less than a threshold time

```
# slurm.conf
SchedulerParameters=max_rpc_cnt=400,\
sched_min_interval=50000,\
sched_max_job_start=300,\
batch_sched_delay=20,\
bf_resolution=600,\
bf_min_prio_reserve=2000,\
bf_min_age_reserve=600
```
# slurm.conf
ClusterName=caesar
TopologyPlugin=topology/tree
FastSchedule=1
SchedulerType=sched/backfill
JobCompType=jobcomp/none

NodeName=DEFAULT State=UNKNOWN CoresPerSocket=4 ThreadsPerCore=2 RealMemory=7940
NodeName=smd1_[0-15] NodeHostname=smd1 Port=19100-19115
NodeName=smd2_[0-15] NodeHostname=smd2 Port=19100-19115
NodeName=smd3_[0-15] NodeHostname=smd3 Port=19100-19115
NodeName=smd4_[0-15] NodeHostname=smd4 Port=19100-19115
PartitionName=smd Nodes=smd[1-4]_[0-15] PriorityJobFactor=1000
Test System

slurmctld host

$ lscpu | egrep 'Model name|CPU MHz'
Model name: Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz
CPU MHz: 4700.000

$ grep "MemTotal" /proc/meminfo
MemTotal: 16255456 kB

$ udevadm info --query=all --name=/dev/nvme0n1 | grep ID_MODEL
E: ID_MODEL=Samsung SSD 970 EVO Plus 500GB

$ uname -sr
Linux 5.2.14-zen2-1-zen

$ srun --partition=smd /usr/bin/lscpu | grep 'Model name'
Model name: Intel(R) Xeon(R) CPU E31230 @ 3.20GHz

$ srun --partition=smd /bin/uname -sr
Linux 5.0.0-15-generic
```bash
$ cat s_jobspermin.sh
#!/bin/bash
# Returns number of jobs completed in each minute since $1 or an hour ago

timeback=$(date +%FT%R -d "-1 hour");
if [[ -n $1 ]]; then
date -d "$1" > /dev/null 2>&1;
if [[ $? -eq 0 ]]; then
timeback=$(date +%FT%R -d "$1");
else
exit
fi;
fi;

echo "Since $timeback";
SLURM_TIME_FORMAT="%FT%H:%M" sacct -Xa --noheader -S$timeback -Enow -oEnd --state=CD |\  
  awk '{sums[$1]++}END{for (s in sums) print s, sums[s]}' |\  
sort
```
Test Jobs

- Test array job

```
$ cat tpbatch.sh
#!/bin/sh
#SBATCH --array=0-1000000
#SBATCH --ntasks=1
#SBATCH --output=/dev/null
#SBATCH --time=00:01
srun /bin/true

$ sbatch --partition=smd tpbatch
```
Usefulness and Limitations

- Test configuration is highly idealized
- The goal is to expose the relative impact of Slurm-specific features, parameters, and configurations
- Hardware is held constant and so is not considered here
- Running massive numbers of tiny jobs is not efficient or recommended due to job launch overhead
  - If the workflow can be adapted to it, job steps would be better
Feature Impact

- What impact do each of Slurm’s major features have on throughput performance?
  - Cgroups, accounting, cons_tres, etc.
- We will start with a baseline configuration with features tuned and turned off for the highest possible throughput
- Then we will measure the independent impact of each feature
Test Configuration

# slurm.conf
ProctrackType=
   proctrack/cgroup
   proctrack/linuxproc
JobAcctGatherType=
   jobacct_gather/none
   jobacct_gather/linux
   jobacct_gather/cgroup
PriorityType=
   priority/basic
   priority/multifactor
SelectType=
   select/cons_res
   select/cons_tres

PrologFlags=
   None
   Alloc
   Contain
TaskPlugin=
   task/affinity
   task/affinity,task/cgroup
SlurmctldDebug=
   error
   debug2
SlurmdDebug=
   error
   debug2

*Underline* indicates baseline configuration
ProctrackType=proctrack/linuxproc

Throughput Impact of Slurm Features

Impact: Small
JobAcctGatherType=jobacct_gather/linux

Impact: Negligible
JobAcctGatherType=jobacct_gather/cgroup

Throughput Impact of Slurm Features

Impact: Large
PriorityType=priority/multifactor

Impact: Negligible
SelectType=select/cons_tres

Impact: Negligible
PrologFlags=Alloc

Impact: Small
PrologFlags=Contain

Impact: Large

Throughput Impact of Slurm Features

Throughput (jobs/min)
TaskPlugin=task/affinity,task/cgroup

# cgroup.conf
ConstrainCores=yes
ConstrainRamSpace=yes
ConstrainSwapSpace=yes
ConstrainDevices=yes

Impact: Large
TaskPlugin=task/affinity,task/cgroup
Arbitrary Combination

JobAcctGatherType = jobacct_gather/linux
ProctrackType = proctrack/cgroup
PrologFlags = Alloc, Contain
TaskPlugin = task/cgroup
ConstrainCores = yes
ConstrainRAMSpace = yes
ConstrainSwapSpace = yes
ConstrainDevices = yes
PriorityType = priority/multifactor

Impact: Large
Slurmctld Debug Level

Impact of Slurmctld Debug Level

Small
Slurmd Debug Level

Impact of Slurmd Debug Level

Small
Questions?