Slurm Simulator. Mini Tutorial/Installation/User Guide

Nikolay Simakov, Center for Computational Reseach, SUNY, University at Buffalo

09-13-2017

Slurm simulator repositary:

https://github.com/nsimakov/slurm_simulator

Toolkit and documentation for slurm simulator repositary:

https://github.com/nsimakov/slurm_sim_tools

Installation

This installation guide is tested on fresh installation of CenOS 7 (CentOS-7-x86_64-DVD-1611.iso KDE Plasma Workspaces with Development Tools)

Since there are many absolute paths in slurm.conf, it can be helpful to create a separate user for slurm named *slurm* and use it for Slurm Simulator.

The idea of simulator is to run simulator with various configuration to compare them and choose one which is more suitable for particular situation. Because of this there are going to be multiple configurations with multiple outcomes. Therefore it is convenient to keep slurm binaries separately from the configuration and logs from the simulated run. The following directory structure is recommended used here (the respective directories will be created on appropriate steps during the tutorial)

```
/home/slurm                      - Slurm user home derectory
      └── slurm_sim_ws           - Slurm simulator work space
          ├── bld_opt            - Slurm simulato building directory
          ├── sim                - Directory where simulation will be performed
          │   └── <system name>  - Directory where simulation of particular system will be performed
          │       └── <conf name> - Directory where simulation of particular configuration will be performed
          │           ├── etc     - Directory with configuration
          │           ├── log     - Directory with logs
          │           └── var     - Directory varius slurm output
          ├── slurm_opt           - Slurm simulator binary installation directory
          ├── slurm_sim_tools     - Slurm simulator toolkit
          └── slurm_simulator     - Slurm simulator source code
```

Installing Dependencies

Slurm Simulator Dependencies

Install MySQL (MariaDB in this case)

Install maria-db server and developer packages:

```
sudo yum install mariadb-server
sudo yum install mariadb-devel
```

Enable and start mariadb server:

```
sudo systemctl enable mariadb
sudo systemctl start mariadb
```

Run mysql_secure_installation for more secure installation if needed. If SQL server is not accessible from the outside it is ok not to run it

```
sudo mysql_secure_installation
```

Add slurm user to SQL, run in MySQL:

```
create user 'slurm'@'localhost' identified by 'slurm';
```

Slurm Simulator Toolkit Dependencies

Python

Install python3 with pymysql and pandas packages:

```
sudo yum -y install install epel-release
sudo yum -y install python34 python34-libs python34-devel python34-numpy python34-scipy python34-pip
sudo pip3 install pymysql
sudo pip3 install pandas
```

R

Install R:

```
sudo yum -y install R R-Rcpp R-Rcpp-devel
sudo yum -y install python-devel
sudo yum install texlive-*
```

Install R-Studio:

```
wget https://download1.rstudio.org/rstudio-1.0.136-x86_64.rpm
sudo yum -y install rstudio-1.0.136-x86_64.rpm
```

In R-Studio or plain R install depending packages:

```
install.packages("ggplot2")
install.packages("gridExtra")
install.packages("cowplot")
install.packages("lubridate")
install.packages("rPython")
install.packages("stringr")
install.packages("dplyr")
install.packages("rstudioapi")
# install R Slurm Simulator Toolkit
install.packages("~/slurm_sim_ws/slurm_sim_tools/src/RSlurmSimTools", repos = NULL, type="source")
```

Prepering Slurm Simulator Workspace

Create work space for Slurm simulation activities:

```
cd
mkdir slurm_sim_ws
cd slurm_sim_ws
```

```
#create directory for simulations
mkdir sim
```

Installing Slurm Simulator

Obtain Slurm Simulator source code with git:

```
git clone https://github.com/nsimakov/slurm_simulator.git
cd slurm_simulator
```

Ensure what slurm-17.02_Sim branch is used:

```
git branch
```

```
Output:
* slurm-17.02_Sim
```

If it is not the case checkout proper branch:

```
git fetch
git checkout slurm-17.02_Sim
```

Prepare building directory

```
cd ..
mkdir bld_opt
cd bld_opt
```

Run configure:

```
../slurm_simulator/configure --prefix=/home/slurm/slurm_sim_ws/slurm_opt --enable-simulator \
    --enable-pam --without-munge --enable-front-end --with-mysql-config=/usr/bin/ --disable-debug \
    CFLAGS="-g -O3 -D NDEBUG=1"
```

Check config.log and ensure that MySQL is found:

```
configure:4672: checking for mysql_config
configure:4690: found /usr/bin//mysql_config
```

Check that openssl is found:

```
configure:24145: checking for OpenSSL directory
configure:24213: gcc -o conftest -g -O3 -D NDEBUG=1 -pthread -I/usr/include  -L/usr/lib   \
                 conftest.c   -lcrypto >&5
configure:24213: $? = 0
configure:24213: ./conftest
configure:24213: $? = 0
configure:24234: result: /usr
```

Slurm can work without MySQL or OpenSSL so if they are not found slurm still can be configured and built. However in most cases these libraries would be needed for simulation.

Compile and install binaries:

```
make -j install
```

Installing Slurm Simulator Toolkit

Obtain Slurm Simulator Toolkit with git:

```
cd ~/slurm_sim_ws
git clone https://github.com/nsimakov/slurm_sim_tools.git
```

Gerenal Overview

The files needed for this tutorial are located at slurm__sim__tools/tutorials/micro__cluster

In order to perform Slurm simulation, one need to:

- Install Slurm in **simulation mode** (see installation guide)
- Create Slurm Configuration
- Create job trace file with jobs description for simulation
- Run Simulation
- Analyse the Results

Number of R and python scripts were created to assist in different stages.

Micro-Cluster

In this tutorial small model cluster, named Micro-Cluster, will be simulated. Micro-Cluster is created to test various aspects of the slurm simulator usage. It consists of 10 compute nodes (see Table 1). There are two types of general compute nodes with different CPU types to test features request (constrains), one GPU node to test GRes and one big memory node to test proper allocation of jobs with large memory requests.

Table 1: Micro-Cluster compute nodes description

Node Type	Node Name	Number of Cores	CPU Type	Memory Size	Number of GPUs
General Compute	n[1-4]	12	CPU-N	48GiB	-
General Compute	m[1-4]	12	CPU-M	48GiB	-
GPU node	g1	12	CPU-G	48GiB	2
Big memory	b1	12	CPU-G	512GiB	-

Installation Note

Slurm Simulator is a fork of original Slurm. Follow installation guide to install it.

The following directory layout is assumed from installation.

```
/home/slurm                   - Slurm user home derectory
      └── slurm_sim_ws        - Slurm simulator work space
          ├── bld_opt         - Slurm simulato building directory
          ├── sim             - Directory where simulation will be performed
          │   └── micro       - Directory where simulation of Micro-Cluster will be performed
          ├── slurm_opt       - Slurm simulator binary installation directory
          ├── slurm_sim_tools - Slurm simulator toolkit
          └── slurm_simulator - Slurm simulator source code
```

Initiate Workspace for Micro-Cluster Simulation

Create top directory for Micro-Cluster simulations

```
cd ~/slurm_sim_ws
mkdir -p ~/slurm_sim_ws/sim/micro
```

Create Slurm Configuration

In this tutorial we will use already existent *etc/slurm.conf* and *etc/slurmdbd.conf*. Slurm configuration files contains a number of parameters with absolute file-system paths which needed to be updated upon the changing of the slurm directories. This happens if we want to tests multiple configurations and it is helpful to have separate directory for each configuration.

To aid the paths updates, the toolkit has *cp_slurm_conf_dir.py* utility which copies files from specified *etc* directory to new location.

In this tutorial we will perform simulation in *~/slurm_sim_ws/sim/micro/baseline*. To initiate new directory from supplied Slurm configuration run:

```
cd ~/slurm_sim_ws
~/slurm_sim_ws/slurm_sim_tools/src/cp_slurm_conf_dir.py -o -s ~/slurm_sim_ws/slurm_opt \
    ~/slurm_sim_ws/slurm_sim_tools/tutorials/micro_cluster/etc ~/slurm_sim_ws/sim/micro/baseline
```

Output:

```
[2017-04-27 15:16:22,800]-[INFO]: Creating directory: /home/slurm/slurm_sim_ws/sim/micro/baseline
[2017-04-27 15:16:22,800]-[INFO]: Copying files from \
    /home/slurm/slurm_sim_ws/slurm_sim_tools/tutorials/micro_cluster/etc \
    to sim/micro/baseline/etc
[2017-04-27 15:16:22,803]-[INFO]: Updating sim/micro/baseline/etc/slurm.conf
[2017-04-27 15:16:22,804]-[INFO]: Updating sim/micro/baseline/etc/slurmdbd.conf
[2017-04-27 15:16:22,805]-[INFO]: Updating sim/micro/baseline/etc/sim.conf
```

sim/micro/baseline now contains etc directory with Slurm configuration. The paths in *etc/slurm.conf* and *etc/slurmdbd.conf* are updated to place outputs within *~/slurm_sim_ws/sim/micro/baseline*.

Generate Users Look-up File

The HPC cluster can have more than several hundreds of users. In simulation performed on desktop we don't want to flood the system with all these users. So Slurm simulator read a simulated user names and UID from *etc/users.sim*. It has following format:

```
username:UID
```

For Micro-Cluster simulation we will simulate five users. Below is the content of *etc/users.sim*:

```
user1:1001
user2:1002
user3:1003
user4:1004
user5:1005
```

Populate SlurmDB

Start slurmdbd in foreground mode:

```
SLURM_CONF=/home/slurm/slurm_sim_ws/sim/micro/baseline/etc/slurm.conf /home/slurm/slurm_sim_ws/slurm_opt/sbin/slurm
```

In separate terminal populate SlurmDB using Slurm *sacctmgr* utility:

```
export SLURM_CONF=/home/slurm/slurm_sim_ws/sim/micro/baseline/etc/slurm.conf
SACCTMGR=/home/slurm/slurm_sim_ws/slurm_opt/bin/sacctmgr

# add QOS
$SACCTMGR  -i modify QOS set normal Priority=0
$SACCTMGR  -i add QOS Name=supporters Priority=100
# add cluster
$SACCTMGR -i add cluster Name=micro Fairshare=1 QOS=normal,supporters
# add accounts
#$SACCTMGR -i add account name=account0 Fairshare=100
$SACCTMGR -i add account name=account1 Fairshare=100
$SACCTMGR -i add account name=account2 Fairshare=100
#$SACCTMGR -i add user name=mikola DefaultAccount=account0 MaxSubmitJobs=1000 AdminLevel=Administrator
# add users
$SACCTMGR -i add user name=user1 DefaultAccount=account1 MaxSubmitJobs=1000
$SACCTMGR -i add user name=user2 DefaultAccount=account1 MaxSubmitJobs=1000
$SACCTMGR -i add user name=user3 DefaultAccount=account1 MaxSubmitJobs=1000
$SACCTMGR -i add user name=user4 DefaultAccount=account2 MaxSubmitJobs=1000
$SACCTMGR -i add user name=user5 DefaultAccount=account2 MaxSubmitJobs=1000

$SACCTMGR -i modify user set qoslevel="normal,supporters"

unset SLURM_CONF
```

Terminate slurmddbd in first terminal.

In this simulation there are 5 users which uses 2 accounts. *user1*, *user2* and *user3* use *account1*. *user4* and *user5* use *account2*

Generate Job Trace File

Job trace file contains information about jobs to be submitted to the simulator it defines time to be submitted and resource requests similarly to normal Slurm batch job submission.

Here we use R to generate job traces. In this tutorial there are two R script which do that job: *11_prep_jobs_for_testrun.R* and *12_prep_jobs_for_testrun.R*. The first one generate several small trace files where all jobs are manually entered and it is good to test Slurm Simulator. The second script generates job trace file with large number of jobs to simulate load which would involve competition between jobs for resource allocation.

RSlurmSimTools library provides various routines for job trace generation and results analysis. *write_trace* function takes *data.frame trace* with resource requests as input and write it to file for future use in Slurm Simulator. *data.frame trace* has single job per row format and columns specify the requested resources by that job. There are large number of parameters supported by Simulator. Entering all of them manually can be overwhelming. To help with that function *sim_job* generate list with requested resources for single job, all of its arguments has default values and thus by specifying only what is different can shorten the time for job trace generation. Multiple of such lists can be combine to form a single job trace file.

Open 11__prep__jobs__for__testrun.R R-script from this tutorial in RStudio. Below is shown a portion of that script:

```r
# load RSlurmSimTools
library(RSlurmSimTools)


# change working directory to this script directory
top_dir <- NULL
top_dir <- dirname(rstudioapi::getActiveDocumentContext()$path)
print(top_dir)
setwd(top_dir)


#dependency check
trace <- list(
    sim_job(
        job_id=1001,
        submit="2016-10-01 00:01:00",
        wclimit=300L,
        duration=600L,
        tasks=12L,
        tasks_per_node=12L
    ),sim_job(
        job_id=1002,
        submit="2016-10-01 00:01:00",
        wclimit=300L,
        duration=600L,
        tasks=12L,
        tasks_per_node=12L,
        dependency="afterok:1001"
    ),sim_job(
        job_id=1003,
        submit="2016-10-01 00:01:00",
        wclimit=300L,
        duration=600L,
        tasks=12L,
        tasks_per_node=12L
    ),sim_job(
        job_id=1004,
        submit="2016-10-01 00:01:00",
        wclimit=300L,
        duration=600L,
        tasks=12L,
        tasks_per_node=12L,
        dependency="afterok:1002:1003"
    ),sim_job(
        job_id=1005,
        submit="2016-10-01 00:01:00",
        wclimit=300L,
        duration=600L,
        tasks=24L,
        tasks_per_node=12L,
        dependency="afterok:1004"
    )
)
#convert list of lists to data.frame
trace <- do.call(rbind, lapply(trace,data.frame))


write_trace(file.path(top_dir,"dependency_test.trace"),trace)
```

First RSlurmSimTools library is loaded, followed by change of working directory. Next the future job trace is specified as list of lists and *sim_job* function is used to set the job request. Execute `r ?sim_job` to see help page for *sim_job* function. Next list of lists if converted to data.frame which is supplied to *write_trace* function.

Execute the script, record the location of generated trace files. The resulting job traces are design to test does Slurm Simulator process dependencies, features and GRes properly.

Open and execute 12_prep_jobs_for_testrun.R R-script. This script would generate job trace file with large number of jobs.

Create Slurm Simulator Configuration

Parameters specific for simulator are set in etc/sim.conf. Open and examine ~/slurm_sim_ws/sim/micro/baseline/etc/sim.co

```
#initial start time, the start time will be reset to the submit time of first job minus X seconds
StartSecondsBeforeFirstJob=45
#stop time, if it is 0 then spin forever if 1 finish after all jobs are done
TimeStop=1
#step in time between cycles in simulator loop
TimeStep=1.0
#time to add after each job launch, kinda mimic time to spin off extra thread and bumping with it
AfterJobLaunchTimeIncreament=0

BFBetweenJobsChecksTimeIncreament=0

#bf_model_real_prefactor=0.0003755
bf_model_real_prefactor=1.0
bf_model_real_power=1.0
#bf_model_sim_prefactor=6.208e-05
bf_model_sim_power=1.0
bf_model_sim_power=1.0


sdiagPeriod=60
sdiagFileOut = /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sdiag.out
sprioPeriod=60
sprioFileOut = /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sprio.out
sinfoPeriod=60
sinfoFileOut = /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sinfo.out
squeuePeriod=60
squeueFileOut = /home/slurm/slurm_sim_ws/sim/micro/baseline/log/squeue.out

SimStats = /home/slurm/slurm_sim_ws/sim/micro/baseline/log/simstat.out

#file with jobs to simulate
```

```
JobsTraceFile=/home/slurm/slurm_sim_ws/slurm_sim_tools/tutorials/micro_cluster/test.trace
```

Update JobsTraceFile parameter to one of the generated job trace file

Perform Simulation

run_sim.py utility simplify the execution of Slurm Simulator. It does following:

1 With -d option it removes results from previous simulation (clean DB, remove logs, shared memory, etc)

2 Check the configuration files, some of the option is not supported in Simulator, some other should be set to particular value. The configuration checking helps to run simulation.

3 Create missing directories

4 Launch slurmdbd

5 Launch slurmctld

6 On finishing slurmctld process it terminates slurmdbd.

7 Preprocess some outputs (squeue, sinfo and sstat like) so that they can be loaded to R.

8 Copy all resulting files to results directory (set with -r option)

```
cd ~/slurm_sim_ws/sim/micro/baseline
~/slurm_sim_ws/slurm_sim_tools/src/run_sim.py -e ~/slurm_sim_ws/sim/micro/baseline/etc -s ~/slurm_sim_ws/slurm_opt
```

```
[2017-05-04 11:30:20,280]-[INFO]: slurm.conf: /home/slurm/slurm_sim_ws/sim/micro/baseline/etc/slurm.conf
[2017-05-04 11:30:20,281]-[INFO]: slurmdbd: /home/slurm/slurm_sim_ws/slurm_opt/sbin/slurmdbd
[2017-05-04 11:30:20,281]-[INFO]: slurmctld: /home/slurm/slurm_sim_ws/slurm_opt/sbin/slurmctld
[2017-05-04 11:30:20,283]-[INFO]: trancating db from previous runs
TRUNCATE TABLE `micro_assoc_usage_day_table`
TRUNCATE TABLE `micro_assoc_usage_hour_table`
TRUNCATE TABLE `micro_assoc_usage_month_table`
TRUNCATE TABLE `micro_event_table`
TRUNCATE TABLE `micro_job_table`
TRUNCATE TABLE `micro_last_ran_table`
TRUNCATE TABLE `micro_resv_table`
TRUNCATE TABLE `micro_step_table`
TRUNCATE TABLE `micro_suspend_table`
TRUNCATE TABLE `micro_usage_day_table`
TRUNCATE TABLE `micro_usage_hour_table`
[2017-05-04 11:30:20,311]-[INFO]: deleting previous JobCompLoc file: /home/slurm/slurm_sim_ws/sim/micro/baseline/log/
[2017-05-04 11:30:20,312]-[INFO]: deleting previous SlurmctldLogFile file: /home/slurm/slurm_sim_ws/sim/micro/baselin
[2017-05-04 11:30:20,330]-[INFO]: deleting previous SlurmSchedLogFile file: /home/slurm/slurm_sim_ws/sim/micro/basel
[2017-05-04 11:30:20,330]-[INFO]: deleting previous SlurmdbdPidFile file: /home/slurm/slurm_sim_ws/sim/micro/baselin
[2017-05-04 11:30:20,331]-[INFO]: deleting previous sdiagFileOut file: /home/slurm/slurm_sim_ws/sim/micro/baseline/l
[2017-05-04 11:30:20,331]-[INFO]: deleting previous sprioFileOut file: /home/slurm/slurm_sim_ws/sim/micro/baseline/l
[2017-05-04 11:30:20,333]-[INFO]: deleting previous SimStats file: /home/slurm/slurm_sim_ws/sim/micro/baseline/log/s
[2017-05-04 11:30:20,333]-[INFO]: deleting previous sinfoFileOut file: /home/slurm/slurm_sim_ws/sim/micro/baseline/l
[2017-05-04 11:30:20,334]-[INFO]: deleting previous squeueFileOut file: /home/slurm/slurm_sim_ws/sim/micro/baseline/
[2017-05-04 11:30:20,335]-[INFO]: deleting previous StateSaveLocation files from /home/slurm/slurm_sim_ws/sim/micro/b
[2017-05-04 11:30:35,362]-[INFO]: slurmctld took 14.01592206954956 seconds to run.
[2017-05-04 11:30:35,363]-[INFO]: Done with simulation
[2017-05-04 11:30:35,363]-[INFO]: Copying results to :/home/slurm/slurm_sim_ws/slurm_simulator/results
[2017-05-04 11:30:35,363]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/jobcomp.log
[2017-05-04 11:30:35,364]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/slurmctld.l
```

```
[2017-05-04 11:30:35,509]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sdiag.out to
[2017-05-04 11:30:35,511]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sprio.out to
[2017-05-04 11:30:35,520]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/simstat.out
[2017-05-04 11:30:35,521]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/sinfo.out to
[2017-05-04 11:30:35,521]-[INFO]: copying resulting file /home/slurm/slurm_sim_ws/sim/micro/baseline/log/squeue.out t
[2017-05-04 11:30:35,527]-[INFO]: processing /home/slurm/slurm_sim_ws/slurm_simulator/results/sprio.out to /home/slu
writing output to: /home/slurm/slurm_sim_ws/slurm_simulator/results/sprio.csv
[2017-05-04 11:30:35,781]-[INFO]: processing /home/slurm/slurm_sim_ws/slurm_simulator/results/simstat.out to /home/s
[2017-05-04 11:30:36,134]-[INFO]: Done
```

In case if it exit without much useful information it can be helpful to redirect standard output from slurmctld and slurmdbd (-octld and -odbd option). slurmctld and slurmdbd can exit prior to output redirection to log files, reading standard output can provide hint on what is wrong:

```
cd ~/slurm_sim_ws/sim/micro/baseline
~/slurm_sim_ws/slurm_sim_tools/src/run_sim.py \
    -e ~/slurm_sim_ws/sim/micro/baseline/etc \
    -s ~/slurm_sim_ws/slurm_opt \
    -octld slurmctld.out -odbd slurmdbd.out \
    -d
```

Analyse Results

Results from simulation are located in their location as specified in Slurm Configuration. Some of the most interesting files are copied to results directory (specified by -r argument, default is "results"). Examine, jobcomp.log file it has sacct like format and can be loaded to R right away.

Recorded Output

jobcomp.log sdiag.out simstat__backfill.csv simstat.out sinfo.out slurmctld.log sprio.csv sprio.out squeue.out squeue.out ##
t: Sun Jan 1 00:03:01 2017 JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON) 1001 normal 1001 user5 R 0:46 1 m1 ##
t: Sun Jan 1 00:04:01 2017 JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON) 1001 normal 1001 user5 R 1:46 1 m1

sinfo.out sinfo like output ##
t: Sun Jan 1 00:01:30 2017 PARTITION AVAIL TIMELIMIT NODES STATE NODELIST normal up infinite 10 idle b1,g1,m[1-4],n[1-4] debug up infinite 2 idle n[1-2] ######################################
t: Sun Jan 1 00:02:01 2017 PARTITION AVAIL TIMELIMIT NODES STATE NODELIST normal up infinite 10 idle b1,g1,m[1-4],n[1-4] debug up infinite 2 idle n[1-2]

sprio.out sprio like output ##
t: Sun Jan 1 00:24:01 2017 JOBID USER PRIORITY AGE FAIRSHARE JOBSIZE PARTITION QOS NICE TRES 1021 user4 1070697 0 40697 30000 1000000 0 0 1025 user1 1085394 0 45394 40000 1000000 0 0 sprio.cvs processed sprio.out file jobid,t,user,priority,age,fairshare,jobsize,partition,qos,nice,tres 1021,2017-01-01 00:23:01,user4,1070697,0,40697,30000,1000000,0,0,NA 1021,2017-01-01 00:24:01,user4,1070697,0,40697,30000,1000000,0,0,NA 1025,2017-01-01 00:24:01,user1,1085394,0,45394,40000,1000000,0,0,NA

Load Libraries

```
library(dplyr)
library(ggplot2)
```

```r
library(gridExtra)
library(RSlurmSimTools)
```

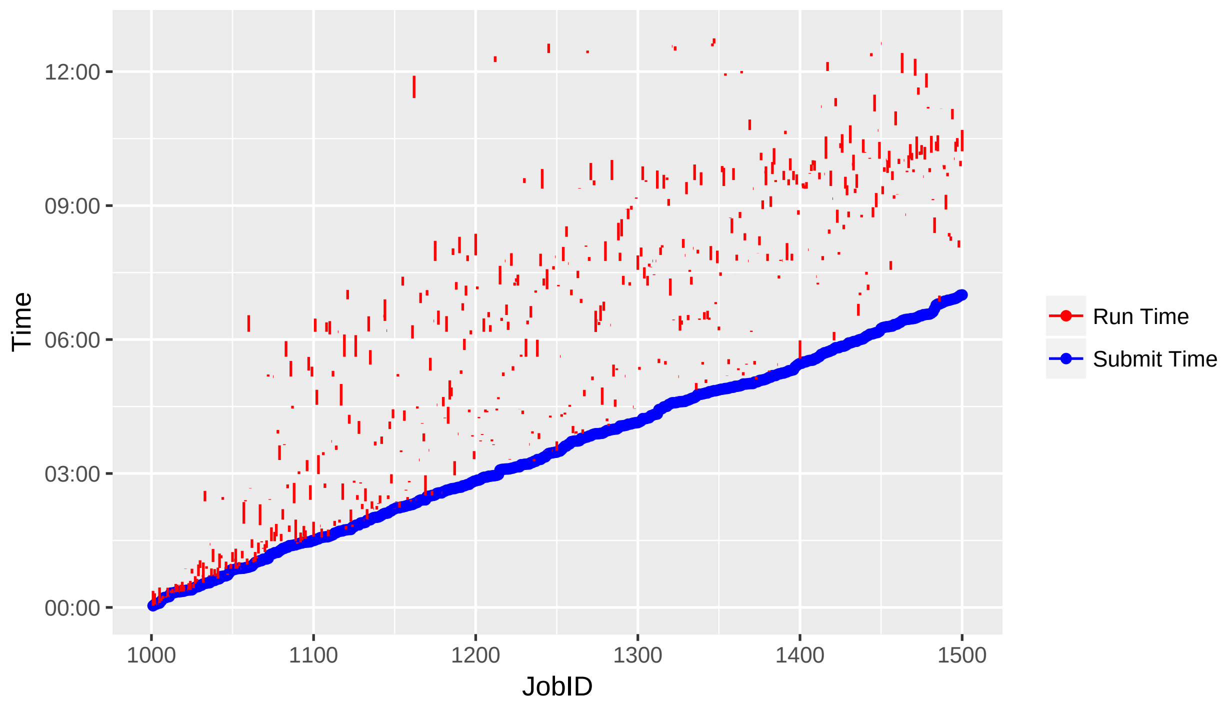
Load Data

```r
#Read jobs execution log
sacct_base <- read_sacct_out("~/slurm_sim_ws/sim/micro/baseline/log/jobcomp.log")
#Read backfill stats
bf_base <- read_simstat_backfill("~/slurm_sim_ws/sim/micro/baseline/results/simstat_backfill.csv")
```

Plotting Jobs Submittion and Execution Times

Job's events plot. On y-axis is time on x-axis is JobID. Submittion time is shown by blue point and the job's execution is shown by line, line starts when the jobs starts and end at job end time.

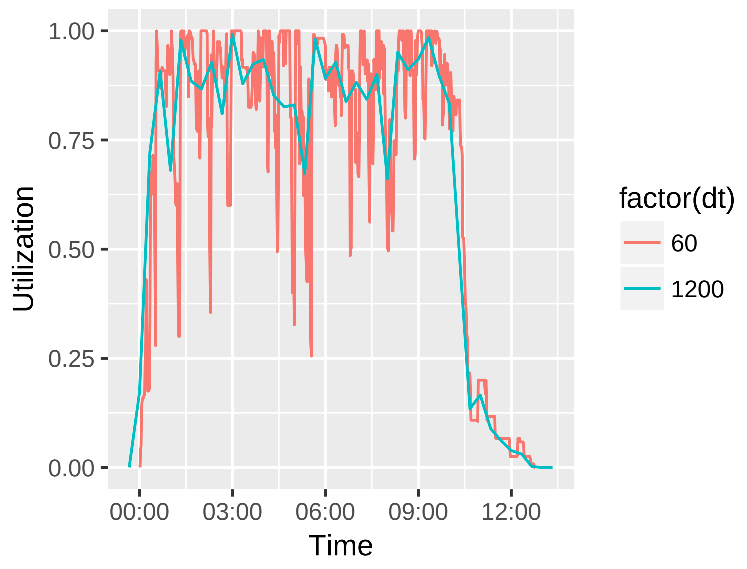
```r
ggplot(sacct_base)+
    geom_point(aes(x=local_job_id,y=Submit,colour="Submit Time"))+
    geom_segment(aes(x=local_job_id,y=Start,xend=local_job_id,yend=End,colour="Run Time"))+
    scale_colour_manual("",values = c("red","blue", "green"))+
    xlab("JobID")+ylab("Time")
```



Utilization

Utilization of resource over time, lets also play with aggrigation duration

```r
util<-data.frame()
for(dt in c(60L,1200L)){
    util_tmp <- get_utilization(sacct_base,total_proc=120L,dt=dt)
    util_tmp$dt<-dt
    util <- rbind(util,util_tmp)
    rm(util_tmp)
}


ggplot(data=util)+xlab("Time")+ylab("Utilization")+
    geom_line(aes(x=t,y=total_norm,color=factor(dt)))
```



Average Waiting Time

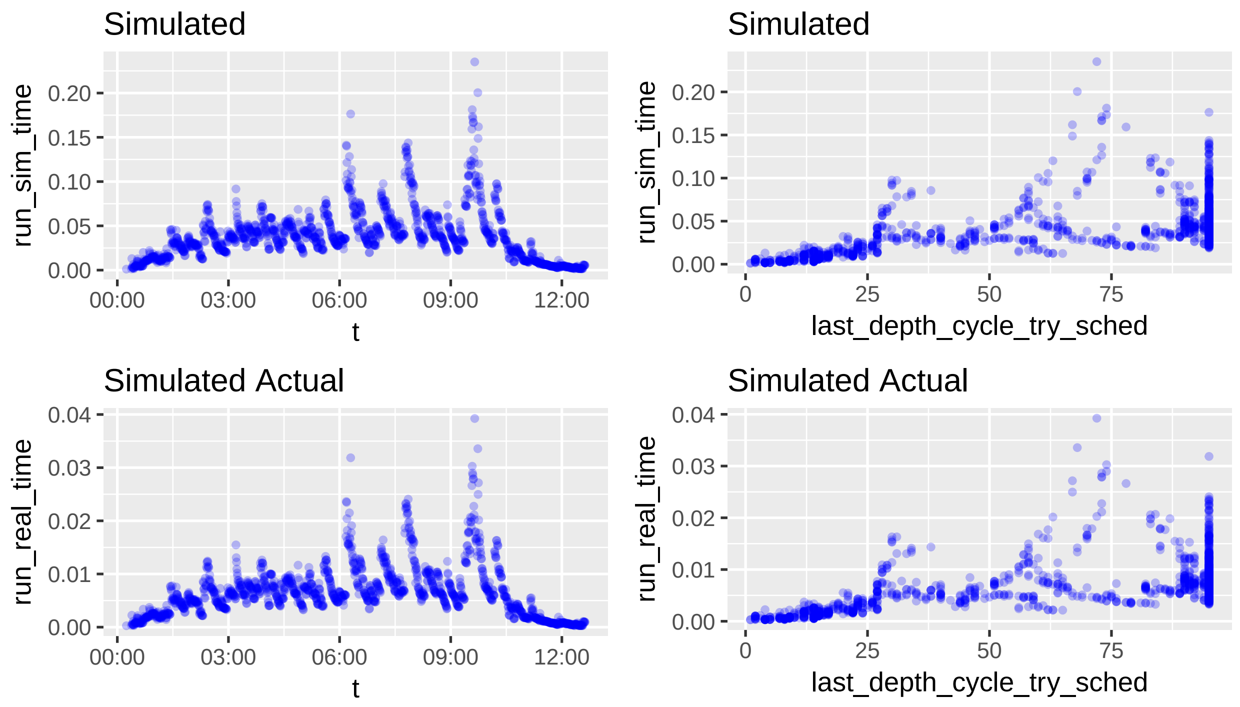
```
print(paste(
    "Average Waiting Time:",
        mean(as.integer(sacct_base$Start-sacct_base$Submit)/3600.0),
    "Hours"
))
```

```
## [1] "Average Waiting Time: 2.40518333333333 Hours"
```

Backfill Performance

Backfill scheduler usually place large number of jobs for execution, It also can take more than several minutes to run on large system. Performance of backfiller both in sence good job placement and time it takes to run can be of interest for resource providers. So here we simply plot some of back fill characteristics.

```
grid.arrange(

    ggplot(bf_base,aes(x=t,y=run_sim_time))+ggtitle("Simulated")+
        geom_point(size=1,colour="blue",alpha=0.25),
    ggplot(bf_base,aes(x=last_depth_cycle_try_sched,y=run_sim_time))+ggtitle("Simulated")+
        geom_point(size=1,colour="blue",alpha=0.25),

    ggplot(bf_base,aes(x=t,y=run_real_time))+ggtitle("Simulated Actual")+
        geom_point(size=1,colour="blue",alpha=0.25),
    ggplot(bf_base,aes(x=last_depth_cycle_try_sched,y=run_real_time))+ggtitle("Simulated Actual")+
        geom_point(size=1,colour="blue",alpha=0.25),
    ncol=2)
```



Modify Slurm Configuration

The characteristics of single set-up is not as much interesting as to compare how chanches in Slurm configuration would be comparable to the base line. To illustrate it we will modify bf_max_job_user from 20 to 3

First initiate now simulation directory from baseline

```
~/slurm_sim_ws/slurm_sim_tools/src/cp_slurm_conf_dir.py -o \
    -s ~/slurm_sim_ws/slurm_opt \
    ~/slurm_sim_ws/sim/micro/baseline/etc ~/slurm_sim_ws/sim/micro/bf3
```

Change bf_max_job_user from 3 to 20 in ~/slurm_sim_ws/sim/micro/baseline/etc/slurm.conf.

Now rerun simulator

```
cd ~/slurm_sim_ws/sim/micro/bf3
~/slurm_sim_ws/slurm_sim_tools/src/run_sim.py -e ~/slurm_sim_ws/sim/micro/bf3/etc -s ~/slurm_sim_ws/slurm_opt -d
```

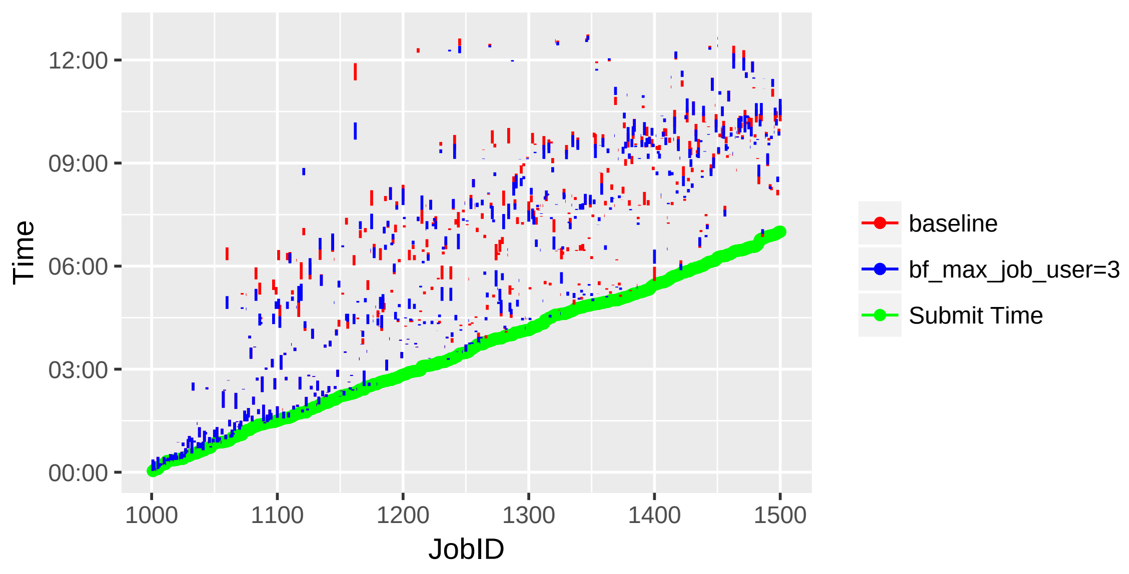
Analyse Results

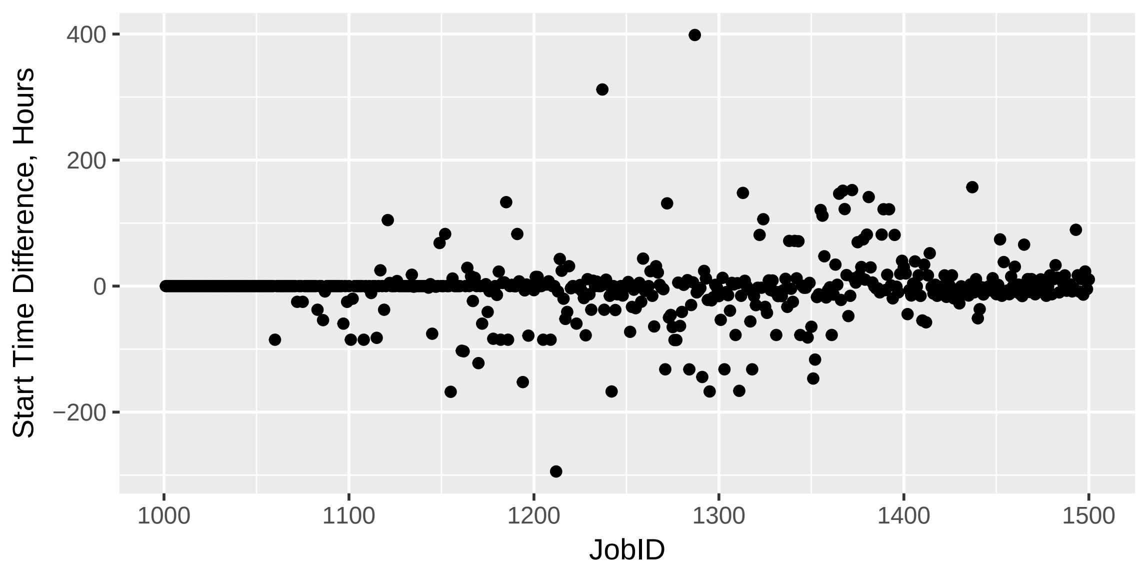
Now we can analyse and compare the changes with baseline

```
#Read jobs execution log
sacct_base <- read_sacct_out("~/slurm_sim_ws/sim/micro/baseline/log/jobcomp.log")
sacct_base$RunID<-"baseline"
```

```r
sacct_bf3 <- read_sacct_out("~/slurm_sim_ws/sim/micro/bf3/log/jobcomp.log")
sacct_bf3$RunID<-"bf_max_job_user=3"
sacct <- rbind(sacct_base,sacct_bf3)

#Read backfill stats
bf_base <- read_simstat_backfill("~/slurm_sim_ws/sim/micro/baseline/results/simstat_backfill.csv")
bf_base$RunID<-"baseline"
bf_bf3 <- read_simstat_backfill("~/slurm_sim_ws/sim/micro/bf3/results/simstat_backfill.csv")
bf_bf3$RunID<-"bf_max_job_user=3"
bf <- rbind(bf_base,bf_bf3)

rm(sacct_base,sacct_bf3,bf_base,bf_bf3)
```

```r
dStart <- merge(subset(sacct,RunID=="baseline",c("local_job_id","Submit","Start")),
      subset(sacct,RunID=="bf_max_job_user=3",c("local_job_id","Start")),
      by = "local_job_id",suffixes = c("_base","_alt"))
dStart$dStart<-(unclass(dStart$Start_alt)-unclass(dStart$Start_base))/60.0
dStart$WaitTime_base<-(unclass(dStart$Start_base)-unclass(dStart$Submit))/60.0
dStart$WaitTime_alt<-(unclass(dStart$Start_alt)-unclass(dStart$Submit))/60.0

grid.arrange(
    ggplot(data=sacct)+ylab("Time")+xlab("JobID")+
         geom_point(aes(x=local_job_id,y=Submit,colour="Submit Time"))+
         geom_segment(aes(x=local_job_id,y=Start,xend=local_job_id,yend=End,colour=RunID))+
         scale_colour_manual("",values = c("red","blue", "green")),
    ggplot(data=dStart)+ylab("Start Time Difference, Hours")+xlab("JobID")+
         geom_point(aes(x=local_job_id,y=dStart)),
    ncol=1
)
```





```
print(paste(
    "Average job start time differences between two simulation is",
    mean(dStart$dStart),
    "minutes"
))
```

```
## [1] "Average job start time differences between two simulation is -3.224 minutes"
```

```
print(quantile(dStart$dStart))
```

```
##          0%         25%         50%         75%        100%
## -294.266667  -10.004167    0.000000    2.516667  398.533333
```

```
print(paste(
    "Average job start time differences between two simulation is",
    100.0*mean(dStart$dStart)/mean(c(dStart$WaitTime_base,dStart$WaitTime_alt)),
    "% of verage wait time"
))
```

```
## [1] "Average job start time differences between two simulation is -2.25930104625821 % of verage wait time"
```

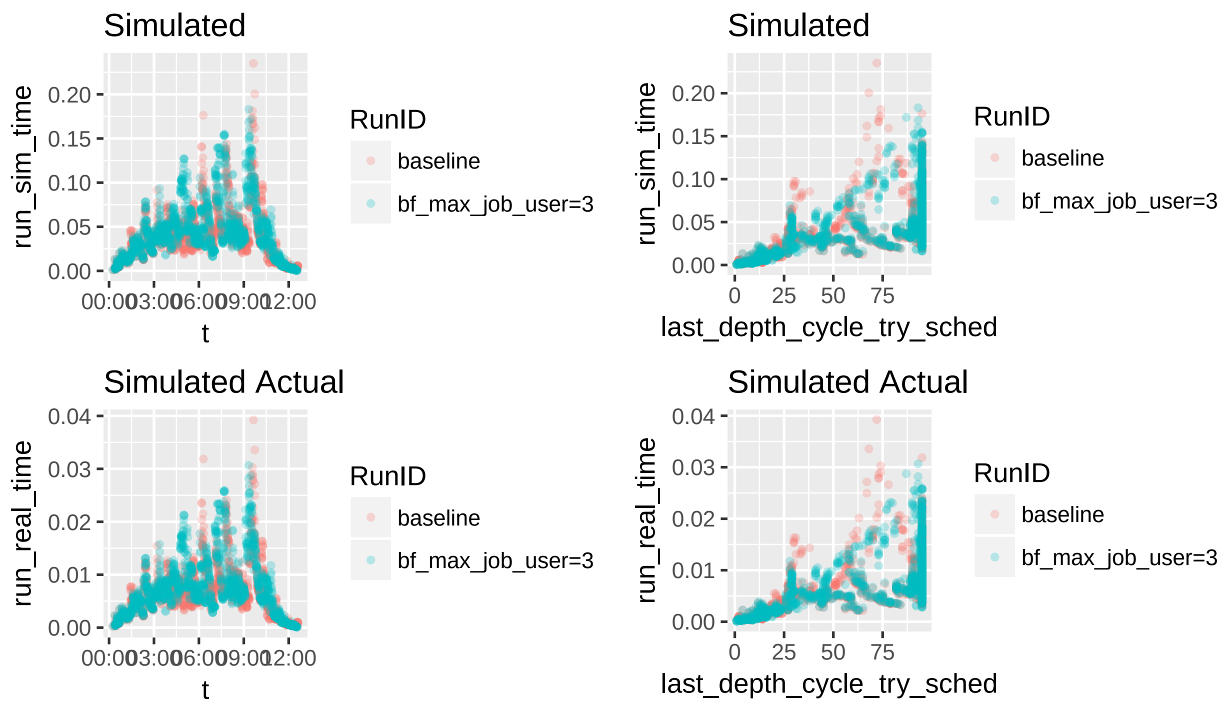
```
print(100*quantile(dStart$dStart)/mean(c(dStart$WaitTime_base,dStart$WaitTime_alt)))
```

```
##          0%         25%         50%         75%        100%
## -206.214947   -7.010677    0.000000    1.763619  279.282499
```

That is for this artificial load decrease of bf_max_job_user from 20 to 3, lead to 2.3% shorter waiting time and maximal difference in start time was 6.6 hours.

Lets examine backfill performance.

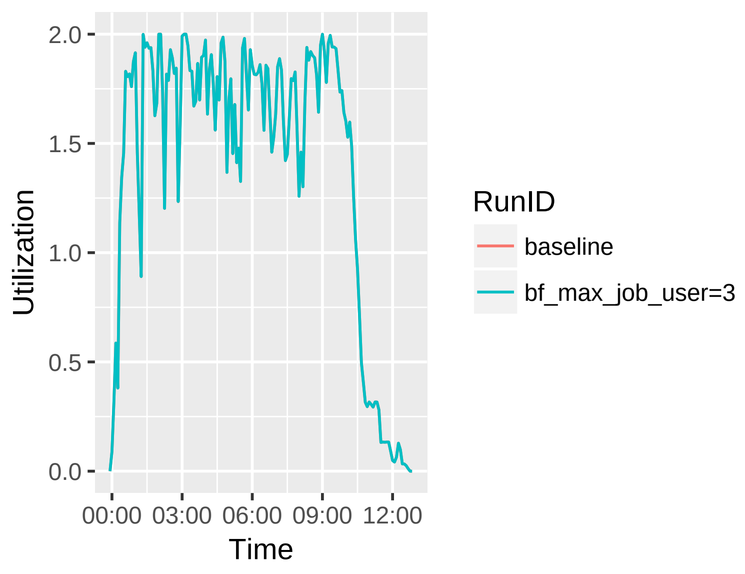
```
grid.arrange(

    ggplot(bf,aes(x=t,y=run_sim_time,colour=RunID))+ggtitle("Simulated")+
        geom_point(size=1,alpha=0.25),
    ggplot(bf,aes(x=last_depth_cycle_try_sched,y=run_sim_time,colour=RunID))+ggtitle("Simulated")+
        geom_point(size=1,alpha=0.25),

    ggplot(bf,aes(x=t,y=run_real_time,colour=RunID))+ggtitle("Simulated Actual")+
        geom_point(size=1,alpha=0.25),
    ggplot(bf,aes(x=last_depth_cycle_try_sched,y=run_real_time,colour=RunID))+ggtitle("Simulated Actual")+
        geom_point(size=1,alpha=0.25),
    ncol=2)
```



Utilization of resource over time, lets also play with aggrigation duration

```
util<-data.frame()
for(RunID in c("baseline","bf_max_job_user=3")){
    util_tmp <- get_utilization(subset(sacct,RunID==RunID),total_proc=120L,dt=300L)
    util_tmp$RunID<-RunID
    util <- rbind(util,util_tmp)
    rm(util_tmp)
}
```

```r
ggplot(data=util)+xlab("Time")+ylab("Utilization")+
    geom_line(aes(x=t,y=total_norm,color=RunID))
```



Utilization is practically the same