## Table of Contents

# How to implement workflows

## Workflows

The version of Slurm installed on the system enables workflows (chains of jobs) with the possibility of having some overlap between the dependent jobs. This allows them to exchange data over the network rather than writing and reading it using a common file system.

Workflows can be submitted in two ways:

- using the new `--delay` option provided in `sbatch` command, which allows to start a job with a fixed delay from the start of the previous job;
- submitting separate jobs using an `afterok` dependency and later requesting a change in dependency type from `afterok` to `after` (using our provided shared library), which allows the second job to start if resources are available.

An example project that uses all the features discussed is provided ?here.

The following simple example script helps to understand the mechanism of new `delay` switch for workflows.

```
[huda1@deepv scripts]$ cat test.sh
#!/bin/sh

NAME=$(hostname)
echo "$NAME: Going to sleep for $1 seconds"
sleep $1
echo "$NAME: Awake"

[huda1@deepv scripts]$ cat batch_workflow.sh
#!/bin/bash
#SBATCH -p sdv -N2 -t3

#SBATCH packjob

#SBATCH -p sdv -N1 -t3 --delay 2

srun test.sh 175

[huda1@deepv scripts]$
```

In the above `sbatch` script, the usage of `--delay` can be seen. The option takes values in minutes and allows us to delay the subsequent job of by a user-defined number of minutes from the start of the first job in the job pack. After submission of this job pack (which uses the same syntax as a heterogeneous job), Slurm divides it into separate jobs. Also, Slurm ensures that the delay is respected by using reservations, rather than the usual scheduler.

Here is the example execution of this script.

```
[huda1@deepv scripts]$ sbatch batch_workflow.sh
Submitted batch job 81458
[huda1@deepv scripts]$ squeue -u huda1
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             81458       sdv batch_wo    huda1 CF       0:01      2 deeper-sdv[02-03]
             81459       sdv batch_wo    huda1 PD       0:00      1 (Reservation)

[huda1@deepv scripts]$
```

Here the second job (81459) will start 2 minutes after the start of the first job (81458), and it is listed as `PD` (`Pending`) with reason `Reservation` because it will start as soon as its reservation will begin.

Similarly, the output files will be different for each separated job in the job pack. the final outputs are:

```
[huda1@deepv scripts]$ cat slurm-81458.out
deeper-sdv02: Going to sleep for 175 seconds
deeper-sdv03: Going to sleep for 175 seconds
```

```
deeper-sdv02: Awake
deeper-sdv03: Awake

[huda1@deepv scripts]$ cat slurm-81459.out
deeper-sdv01: Going to sleep for 175 seconds
deeper-sdv01: Awake

[huda1@deepv scripts]$
```

Another feature to note is that if there are multiple jobs in a job pack and any number of consecutive jobs have the same `delay` values, they are combined into a new heterogeneous job. This allows to have heterogeneous jobs within workflows. Here is an example of such a script:

```
[huda1@deepv scripts]$ cat batch_workflow_complex.sh
#!/bin/bash

#SBATCH -p sdv -N 2 -t 3
#SBATCH -J first

#SBATCH packjob

#SBATCH -p sdv -N 1 -t 3 --delay 2
#SBATCH -J second

#SBATCH packjob

#SBATCH -p sdv -N 1 -t 2 --delay 2
#SBATCH -J second

#SBATCH packjob

#SBATCH -p sdv -N 2 -t 3 --delay 4
#SBATCH -J third

if [ "$SLURM_JOB_NAME" == "first" ]
then
       srun ./test.sh 150

elif [ "$SLURM_JOB_NAME" == "second" ]
then
       srun ./test.sh 150 : ./test.sh 115

elif [ "$SLURM_JOB_NAME" == "third" ]
then
       srun ./test.sh 155

fi

[huda1@deepv scripts]$
```

Note the `delay` values for the second and third job in the script are equal.

**Attention** The `delay` value for the 4th job (`-J third`) is relative to the start of the first job and not from the start of middle 2 jobs. So it will start after 2 minutes of the start time of the middle jobs. Also, note the usage of the environment variable `SLURM_JOB_NAME` in the script to decide which command to run in which job. The example execution leads to the following:

The example execution leads to the following:

```
[huda1@deepv scripts]$ sbatch batch_workflow_complex.sh
Submitted batch job 81460

[huda1@deepv scripts]$ squeue -u huda1
```

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          81461+0       sdv   second    huda1 PD       0:00      1 (Resources)
          81461+1       sdv   second    huda1 PD       0:00      1 (Resources)
            81463       sdv    third    huda1 PD       0:00      2 (Resources)
            81460       sdv    first    huda1 PD       0:00      2 (Resources)

[huda1@deepv scripts]$
```

Note that the submitted heterogeneous job has been divided into a single job (81460), a job pack (81461) and again a single job (81643). Similarly, three different output files will be generated, one for each new job.

```
[huda1@deepv scripts]$ cat slurm-81460.out
deeper-sdv03: Going to sleep for 150 seconds
deeper-sdv04: Going to sleep for 150 seconds
deeper-sdv03: Awake
deeper-sdv04: Awake

[huda1@deepv scripts]$ cat slurm-81461.out
deeper-sdv01: Going to sleep for 150 seconds
deeper-sdv02: Going to sleep for 115 seconds
deeper-sdv02: Awake
deeper-sdv01: Awake

[huda1@deepv scripts]$ cat slurm-81463.out
deeper-sdv01: Going to sleep for 155 seconds
deeper-sdv02: Going to sleep for 155 seconds
deeper-sdv01: Awake
deeper-sdv02: Awake

[huda1@deepv scripts]$
```

If a job exits earlier than the allocated time asked by the user, the corresponding reservation for this job is deleted 5 minutes after the end of the job, automatically and the resources become available for the other jobs. However, users should be careful with the requested time when submitting workflows as the larger time values can delay the scheduling of the workflows depending on the situation of the resources.

The workflows created using `delay` switch ensure overlap between the applications. Instead, using the alternative method (which uses Slurm job dependencies) does not ensure a time overlap between two consecutive jobs of a workflow. Though, in this case users do not need to guess the time a job will take and how much should the delay between jobs starting times should be.

Jobs can be chained in Slurm with the aid of the following script:

```
[huda1@deepv scripts]$ cat chain_jobs.sh
#!/usr/bin/env bash

if [ $# -lt 3 ]
then
   echo "$0: ERROR (MISSING ARGUMENTS)"
   exit 1
fi

LOCKFILE=$1
DEPENDENCY_TYPE=$2
shift 2
SUBMITSCRIPT=$*


if  [ -f $LOCKFILE ]
then
   if [[ "$DEPENDENCY_TYPE" =~ ^(after|afterany|afterok|afternotok)$ ]]; then
       DEPEND_JOBID=`head -1 $LOCKFILE`
       echo "sbatch --dependency=${DEPENDENCY_TYPE}:${DEPEND_JOBID} $SUBMITSCRIPT"
```

```
        JOBID=`sbatch --dependency=${DEPENDENCY_TYPE}:${DEPEND_JOBID} $SUBMITSCRIPT`
    else
        echo "$0: ERROR (WRONG DEPENDENCY TYPE: choose among 'after', 'afterany', 'afterok' or 'afternotok')"
    fi
else
    echo "sbatch $SUBMITSCRIPT"
    JOBID=`sbatch $SUBMITSCRIPT`
fi

echo "RETURN: $JOBID"
# the JOBID is the last field of the output line
echo ${JOBID##* } > $LOCKFILE

exit 0
```

This is a modified version of the of the `chainJobs.sh` included in JUBE, which allows to select the desired dependency type between two consecutive jobs. Here is an example of submission of a workflow with Slurm dependencies using the previous script (here called `chain_jobs.sh`):

```
[huda1@deepv scripts]$ ./chain_jobs.sh lockfile afterok simple_job.sh
sbatch simple_job.sh
RETURN: Submitted batch job 98626
[huda1@deepv scripts]$ ./chain_jobs.sh lockfile afterok simple_job.sh
sbatch --dependency=afterok:98626 simple_job.sh
RETURN: Submitted batch job 98627
[huda1@deepv scripts]$ ./chain_jobs.sh lockfile afterok simple_job.sh
sbatch --dependency=afterok:98627 simple_job.sh
RETURN: Submitted batch job 98628
[huda1@deepv scripts]$ squeue -u huda1
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             98627       sdv simple_j    huda1 PD       0:00      2 (Dependency)
             98628       sdv simple_j    huda1 PD       0:00      2 (Dependency)
             98626       sdv simple_j    huda1  R       0:21      2 deeper-sdv[01-02]
[huda1@deepv scripts]$ scontrol show job 98628 | grep Dependency
  JobState=PENDING Reason=Dependency Dependency=afterok:98627
[huda1@deepv scripts]$ cat lockfile
98628
```

Please note that `lockfile` must not exist previous to the first submission. After the first job submission, that file will contain the id of last submitted job, which is later used by the subsequent call to the `chain_job.sh` script to set the dependency.

### `slurm_workflow` Library

In order to improve the usability of workflows, a library has been developed and deployed on the system to allow users to interact with the scheduler from within applications involved in a workflow. The library is called `slurm_workflow`.

The library has two functions.

The first function is relevant to workflows created using the `--delay` switch and moves all the reservations of the remaining workflow jobs.

```
/*
IN:    number of minutes from now. The start time of the next reservation of the workflow is moved to this time if doable.
OUT:   0 successful, non zero unsuccessful.slurm_wf_error is set.
*/
int slurm_wf_move_all_res(uint32_t t);
```

The minimum value usable for the parameter is currently 2 (minutes).

The second function changes the dependencies type of all jobs dependent on the current job from `afterok:job_id` to `after:job_id`.

```
/*
OUT: 0 successful, error no otherwise.
*/

int slurm_change_dep();
```

This enables the jobs in workflow eligible for allocation by Slurm.

Both functions allow an application to notify the scheduler that it is ready for the start of the subsequent jobs of a workflow. This is particularly relevant in case a network connection must be established between the two applications, but only after a certain time from the start of the first job.

When using the library, the header file can be included using `#include <slurm/slurm_workflow.h>` and the library should be linked against using `-lslurm_workflow -lslurm`.