Issue with slow cluster queues

Thu, 10/04/2018 - 16:54 #1

Daniel Asarnow

Issue with slow cluster queues

Hi Tim et al,

I'm running into an issue with a heavily used SGE queue. Say I have cisTEM launch 100 copies with a hard runtime limit of 1 hour. If the queue is empty, they all run, and 1 hour is more than enough time for the iteration to complete. On the other hand, if the queue is nearly full, my first few jobs will run, but the rest may not start until > 1 hour has passed. In that case, my early jobs will die. AFAICT, the behavior right now is then that cisTEM idles forever - no error output - and all the jobs are removed from the queue.

In other words, the first queued job has to be able to run for at least as long as needed for the last job to queue (and maybe for it to finish). The workaround is to give spuriously high run times for the jobs, but doing so also decreases their queue priority.

It would be great if there was a way for cisTEM to ignore the loss of a worker job (and whatever incomplete work chunk it had) and wait for others to become available so we could "queue and forget" with shorter run time limits. Or maybe I'm missing something?

Thanks!

timgrant

Hi Daniel,

Hi Daniel,

Actually when the worker jobs finish they die, and that is fine. The problem is that the first job to connect back is in made the master that coordinates things, and this needs to remain running the whole time. Another problem is that when the first launched jobs finish their alloted work, if some of the others have not launched yet - they will actually continue and do some of the work of the unlaunched ones - so they will not actually finish until the job is finished.

In the next release of cisTEM I will hopefully add the abilitty to run with MPIRUN, so the queueing system should at least not launch things until they can be launched together.

Cheers,

Tim

rnavaza

This script may help

Hi,

I made this script in order to allocate all the workers at the same time with Torque/Moab but you can adapt it for almost every cluster (I did it for SLURM).

You can find how to configure it in https://cistem.org/unable-get-cistem-job-parallelize-multiple-nodes-cluster (starting from comment #8)

```
#! /bin/bash
# get the current number of instances of this script
that have been# launched by cisTEM and increment it
lockfile="/dev/shm/cisTEM.${@: -4:1}.${@: -1:1}"
lockcount=$(flock -x ${lockfile} /bin/sh -c "read
count <
${lockfile}
                                                let
'count = count +
1'                                                echo
\${count} >
${lockfile}
                                                echo
\${count}")
# only the first instance will launch the qsub
command, for others: Goodbye !if [ "${lockcount}" -ne
1 ]; then exit 0 ; fi
# wait long enough for cisTEM to finish submiting all
the "copies"
sleep 3
# now get the total number of CPUs requested by cisTEM
procs=$(cat ${lockfile})
rm -f ${lockfile}
#
################################################################
#
# physical memory per process (possible suffixes are
b, kb, gb, tb)
mem=2g
# maximun time per qsub job (format is HH:MM:SS)
walltime='00:30:00'
while getopts ":m:w:" OPTION
do
   case "${OPTION}" in
      m) mem="${OPTARG}";;      w)
walltime="${OPTARG}";;
   esac
done
shift $((OPTIND - 1))
```

```
# submit the job with qsub
cat - <<EOF | qsub
#! /bin/sh
#PBS -N cisTEM.${1}
#PBS -j oe
#PBS -l walltime=${walltime}#PBS -l procs=${procs}
#PBS -l pmem=${mem}/usr/bin/pbsdsh
/your_path_to_cistem_binaries/${@}EOF
```

Cheers,

Rafael

Log in or register to post comments