

# Using GNU Parallel to Package Multiple Jobs in a Single PBS Job

## Category: Effective Use of PBS

GNU is a complete, free software system, upward-compatible with Unix. GNU parallel is a shell tool for executing jobs in parallel. It uses the lines of its standard input to modify shell commands, which are then run in parallel. Detailed information about this tool can be found on the [GNU Operating System](#) website and its related [parallel man page](#).

On Pleiades, a copy of GNU parallel is available under `/usr/bin`.

The three examples below demonstrate how you can use GNU parallel to run multiple tasks in a single PBS batch job.

### Example 1

This example script runs 64 copies of a serial executable file, and assumes that 4 copies will fit in the available memory of one node and that 16 nodes are used.

#### gnu\_serial1.pbs

```
#PBS -lselect=16:ncpus=4
#PBS -lwalltime=4:00:00

cd $PBS_O_WORKDIR

seq 64 | parallel -j 4 -u --sshloginfile $PBS_NODEFILE \
  "cd $PWD;./myscript.csh {}"
```

In the above PBS script, the last command uses the parallel command to simultaneously run 64 copies of `myscript.csh` located under `$PBS_O_WORKDIR`. Here is the specific breakdown:

- seq 64

Generates a set of integers *1, 2, 3, ..., 64* that will be passed to the parallel command.

- -j 4

GNU parallel will determine the number of processor cores on the remote computers and run the number of tasks as specified by `-j`. In this case, `-j 4` tells the the

parallel command to run 4 tasks in parallel on one compute node.

- `-u`

Tells the parallel command to print output as soon as possible. This may cause output from different commands to be mixed. GNU parallel runs faster with `-u`. This can be reversed with `--group`.

- `--sshloginfile $PBS_NODEFILE`

Distributes tasks to the compute nodes listed in `$PBS_NODEFILE`.

- `"cd $PWD; ./myscript.csh {}"`

Changes directory to the current working directory and runs `myscript.csh` located under `$PWD`. At this point, `$PWD` is the same as `$PBS_O_WORKDIR`. The `{}` is an input to `myscript.csh` (see below) and will be replaced by the sequence number generated from `seq 64`.

## myscript.csh

```
#!/bin/csh -fe
date
mkdir -p run_$(seq 1 64)
cd run_$(seq 1 64)

echo "Executing run $(seq 1 64) on" `hostname` "in $PWD"

$HOME/bin/a.out < ../input_$(seq 1 64) > output_$(seq 1 64)
```

In this above sample script executed by the parallel command:

- `$(seq 1 64)` refers to the sequence numbers (1, 2, 3, ..., 64) from the `seq` command that was piped into the parallel command
- For each serial run, a subdirectory named `run_$(seq 1 64)` (`run_1`, `run_2`, ...) is created
- The echo line prints information back to the PBS `stdout` file
- The serial `a.out` is located under `$HOME/bin`
- The input for each run, `input_$(seq 1 64)` (`input_1`, `input_2`, ...) is located under `$PBS_O_WORKDIR`, which is the directory above `run_$(seq 1 64)`
- The output for each run (`output_1`, `output_2`, ...) is created under `run_$(seq 1 64)`

## Potential Modifications to Example 1

There are multiple ways to pass arguments to parallel. For example, instead of using the `seq` command to pass a sequence of integers, you can also pass in a list of directory names or filenames using `ls -1` or `cat mylist`, where the file `mylist` contains a list of entries.

## Example 2

This script is similar to Example 1, except that 6 nodes are used instead of 16 nodes. This means that 24 serial `a.out`s can be run simultaneously, since a total of 24 cores (6 nodes x 4 cores) are requested. As each `a.out` completes its work on a core, another `a.out` is launched by the parallel command to run on the same core.

`myscript.csh` is the same as that shown in the previous example.

### gnu\_serial2.pbs

```
#PBS -lselect=6:ncpus=4
#PBS -lwalltime=4:00:00

cd $PBS_O_WORKDIR

seq 64 | parallel -j 4 -u --sshloginfile $PBS_NODEFILE \
  "cd $PWD; ./myscript.csh {}"
```

## Example 3

In this example, an OpenMP executable is run with 12 OpenMP threads on one Westmere node. To run 64 copies of this executable with 10 copies running simultaneously on 10 nodes:

### gnu\_openmp.pbs

```
#PBS -lselect=10:ncpus=12:mpiprocs=1:ompthreads=12:model=wes
#PBS -lwalltime=4:00:00

cd $PBS_O_WORKDIR

seq 64 | parallel -j 1 -u --sshloginfile $PBS_NODEFILE \
  "cd $PWD; ./myopenmpscript.csh {}"
```

### myopenmpscript.csh

```
#!/bin/csh -fe
date
mkdir -p run_$1
cd run_$1

setenv OMP_NUM_THREADS 12
```

```
echo "Executing run $1 on" `hostname` "in $PWD"  
$HOME/bin/a.out < ../input_$1 > output_$1
```

---

Article ID: 303

Last updated: 31 Jul, 2012

Computing at NAS -> Running Jobs with PBS -> Effective Use of PBS -> Using GNU  
Parallel to Package Multiple Jobs in a Single PBS Job

<http://www.nas.nasa.gov/hecc/support/kb/entry/303/?ajax=1>