Running MPI/MP with Slurm

1. Run MP
2. Run MPI
3. Run ‘parallel’ scripts in bash

Run MP

#SBATCH ntask=1

#SBATCH cpu_per_task=36

srun script.py

Run MPI

#SBATCH ntask=40

Run MPI (with MP)

#SBATCH ntask=40

#SBATCH cpu_per_task=4

#SBATCH --tasks-per-node=10

#careful with the numbers

Run “Parallel” scripts in Bash

1. xargs (simple)
2. Using &

xargs

xargs -I CMD -P 40 bash -c CMD

Example

```
1 echo Hello world
2 echo Goodbye world
3 echo Goodbye cruel world
```

```
ykh22@mahuika01:~> cat commands.txt | xargs -I CMD -P 3 bash -c CMD
Goodbye world
Hello world
Goodbye cruel world
```

Using &

```
#SBATCH ntask=4

srun python script1.py &

srun python script2.py &

srun python script3.py &

srun python script4.py &

wait
```

Using & with MPI scripts

```
#!/bin/bash
#SBATCH nodes=9
#SBATCH time=00:15:00

srun --hint=nomultithread -N 2 --ntasks=72 --ntasks-per-node=36 ./my_exe_1 &
srun --hint=nomultithread –N 3 --ntasks=108 --ntasks-per-node=36 ./my_exe_2 &
srun --hint=nomultithread -N 4 --ntasks=144 --ntasks-per-node=36 ./my_exe_3 &
wait
```

Advanced usage

1. Catch how many task running(pids)
2. Run a job if pids is under max (SLURM_NTASKS)
3. Apped the pid into the list of pids
4. Wait for all pids to finish/change state

See example in: /nesi/project/nesi00213/bash_examples/test_parallel_sleep

```
i=0
echo "slurm_ntask = $SLURM_NTASKS"
for NUM in `cat numbers.txt`;
do
    while [ $i -ge $SLURM_NTASKS ];
    do
        #check all pids
        for pid in ${pids[@]} ;
        do
            kill -0 $pid 2>/dev/null
            res=$?
            if [ $res -ne 0 ];
            then
                #pop the pid out of the array
                pids=(${pids[*]/$pid})
                i=$((i-1))
            fi
        done
        sleep 1
    done
    bash -c "echo start $NUM; sleep $NUM; echo end $NUM" & pids[$i]=$!
    i=$((i+1))
done

#wait for all pids to finish
for pid in ${pids[*]};
do
    wait $pid
done
```

```
i=0
echo "slurm_ntask = $SLURM_NTASKS"
for NUM in `cat numbers.txt`;
do
    while [ $i -ge $SLURM_NTASKS ];
    do
        #check all pids
        for pid in ${pids[@]} ;
        do
            kill -0 $pid 2>/dev/null
            res=$?
            if [ $res -ne 0 ];
            then
                #pop the pid out of the array
                pids=(${pids[*]/$pid})
                i=$((i-1))
            fi
        done
        sleep 1
    done
    bash -c "echo start $NUM; sleep $NUM; echo end $NUM" & pids[$i]=$!
    i=$((i+1))
done

#wait for all pids to finish
for pid in ${pids[*]};
do
    wait $pid
done
```

```
i=0
echo "slurm_ntask = $SLURM_NTASKS"
for NUM in `cat numbers.txt`;
do
    while [ $i -ge $SLURM_NTASKS ];
    do
        #check all pids
        for pid in ${pids[@]} ;
        do
            kill -0 $pid 2>/dev/null
            res=$?
            if [ $res -ne 0 ];
            then
                #pop the pid out of the array
                pids=(${pids[*]/$pid})
                i=$((i-1))
            fi
        done
        sleep 1
    done
    bash -c "echo start $NUM; sleep $NUM; echo end $NUM" & pids[$i]=$!
    i=$((i+1))
done

#wait for all pids to finish
for pid in ${pids[*]};
do
    wait $pid
done
```

```
i=0
echo "slurm_ntask = $SLURM_NTASKS"
for NUM in `cat numbers.txt`;
do
   while [ $i -ge $SLURM_NTASKS ];
   do
      #check all pids
      for pid in ${pids[@]} ;
      do
         kill -0 $pid 2>/dev/null
         res=$?
         if [ $res -ne 0 ];
         then
            #pop the pid out of the array
            pids=(${pids[*]/$pid})
            i=$((i-1))
         fi
      done
      sleep 1
   done
   bash -c "echo start $NUM; sleep $NUM; echo end $NUM" & pids[$i]=$!
   i=$((i+1))
done

#wait for all pids to finish
for pid in ${pids[*]};
do
   wait $pid
done
```

```
i=0
echo "slurm_ntask = $SLURM_NTASKS"
for NUM in `cat numbers.txt`;
do
    while [ $i -ge $SLURM_NTASKS ];
    do
        #check all pids
        for pid in ${pids[@]} ;
        do
            kill -0 $pid 2>/dev/null
            res=$?
            if [ $res -ne 0 ];
            then
                #pop the pid out of the array
                pids=(${pids[*]/$pid})
                i=$((i-1))
            fi
        done
        sleep 1
    done
    bash -c "echo start $NUM; sleep $NUM; echo end $NUM" & pids[$i]=$!
    i=$((i+1))
done

#wait for all pids to finish
for pid in ${pids[*]};
do
    wait $pid
done
```