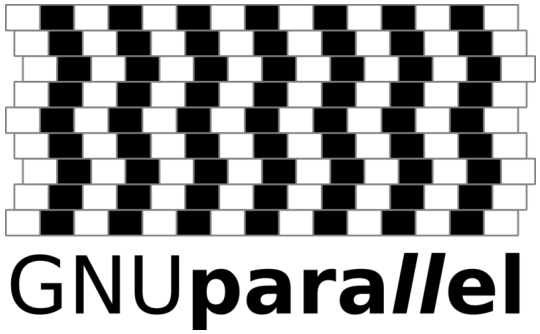
GNU Parallel for Large Batches of Small Jobs



TechTalk, November 2012

Ramses van Zon

SciNet HPC Consortium University of Toronto

SciNet compute • calcul CANADA

Introduction

- SciNet is primarily a parallel computing resource. (Parallel here means OpenMP and MPI, not many serial jobs.)

SciNet compute • calcul CANADA

Introduction

- SciNet is primarily a parallel computing resource. (Parallel here means OpenMP and MPI, not many serial jobs.)
- You should never submit purely serial jobs to the GPC queue. The scheduling queue gives you a full 8-core node. Per-node scheduling of serial jobs would mean wasting 7 cpus.

Introduction

- SciNet is primarily a parallel computing resource. (Parallel here means OpenMP and MPI, not many serial jobs.)
- You should never submit purely serial jobs to the GPC queue. The scheduling queue gives you a full 8-core node. Per-node scheduling of serial jobs would mean wasting 7 cpus.
- Nonetheless, if you can make efficient use of the resources using serial runs and get good science done, that's good too.

Introduction

- SciNet is primarily a parallel computing resource. (Parallel here means OpenMP and MPI, not many serial jobs.)
- You should never submit purely serial jobs to the GPC queue. The scheduling queue gives you a full 8-core node. Per-node scheduling of serial jobs would mean wasting 7 cpus.
- Nonetheless, if you can make efficient use of the resources using serial runs and get good science done, that's good too.
- Users need to utilize whole nodes by running at least 8 serial runs at once.

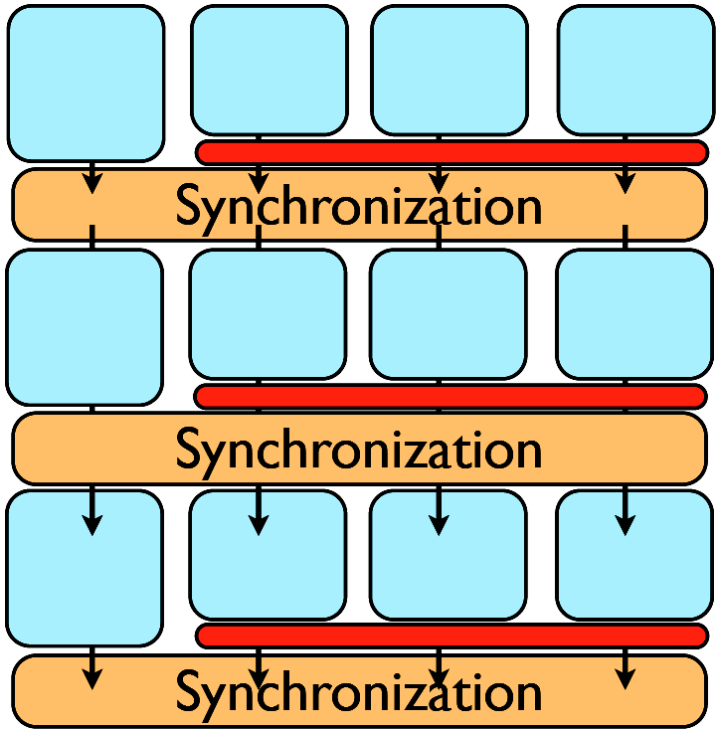
SciNet compute • calcul CANADA

Easy case: serial runs of equal duration

```
#PBS -l nodes=1:ppn=8,walltime=1:00:00
cd $PBS_O_WORKDIR
(cd rundir1; ./dorun1) &
(cd rundir2; ./dorun2) &
(cd rundir3; ./dorun3) &
(cd rundir4; ./dorun4) &
(cd rundir5; ./dorun5) &
(cd rundir6; ./dorun6) &
(cd rundir7; ./dorun7) &
(cd rundir8; ./dorun8) &
wait # or all runs get killed immediately
```

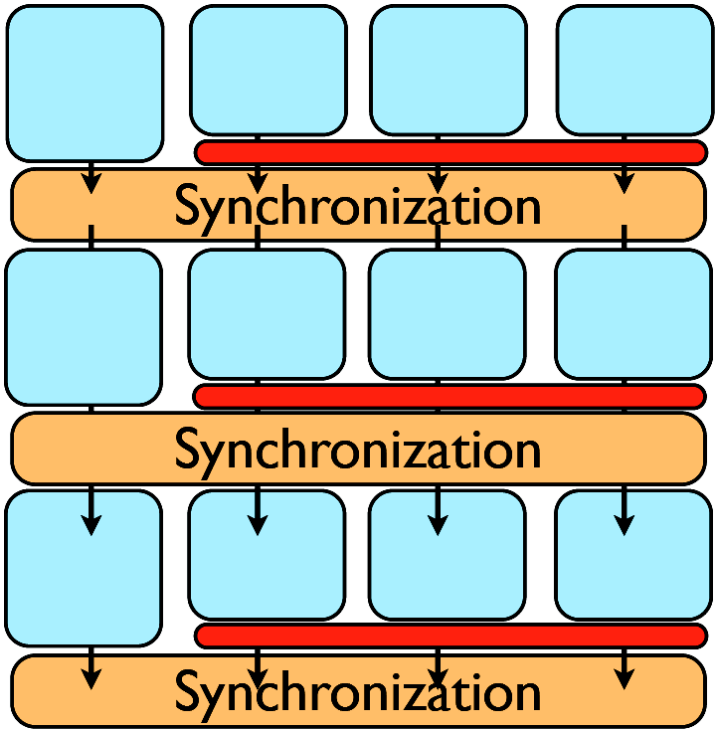
Hard case: serial runs of unequal duration

Different runs may not take the same time: **load imbalance**.



Hard case: serial runs of unequal duration

Different runs may not take the same time: **load imbalance**.



- Want to keep all 8 cores on a node busy.
- Or even 16 virtual cores on a node (HyperThreading).
- ⇒ GNU Parallel can do this

SciNet compute • calcul CANADA

GNU Parallel

- GNU parallel is a a tool to run multiple (serial) jobs in parallel. *As parallel is used within a GPC job, we'll call these* **subjobs**.
- It allows you to keep the processors on each 8-core node busy, if you provide enough subjobs.
- GNU Parallel can use multiple nodes as well.

On the GPC cluster:

- GNU parallel is accessible on the GPC in the module gnu-parallel, which you can load in your .bashrc.

```
$ module load gnu-parallel/20121022
```

- There are currently (Nov 2012) three gnu-parallel modules on the GPC. Although for compatibility gnu-parallel/2010 is the default, we recommend using gnu-parallel/20121022.

SciNet compute • calcul CANADA

GNU Parallel Example

SETUP

- A serial c++ code 'mycode.cc' needs to be compiled.

SciNet compute • calcul CANADA

GNU Parallel Example

SETUP

- A serial c++ code 'mycode.cc' needs to be compiled.
- It needs to be run 32 times with different parameters, 1 through 32.

GNU Parallel Example

SETUP

- A serial c++ code 'mycode.cc' needs to be compiled.
- It needs to be run 32 times with different parameters, 1 through 32.
- The parameters are given as a command line argument.

GNU Parallel Example

SETUP

- A serial c++ code 'mycode.cc' needs to be compiled.
- It needs to be run 32 times with different parameters, 1 through 32.
- The parameters are given as a command line argument.
- 8 subjobs of this code fit into the GPC compute nodes's memory.

GNU Parallel Example

SETUP

- A serial c++ code 'mycode.cc' needs to be compiled.
- It needs to be run 32 times with different parameters, 1 through 32.
- The parameters are given as a command line argument.
- 8 subjobs of this code fit into the GPC compute nodes's memory.
- Each serial run on average takes $\sim$ 2 hour.

GNU Parallel Example

```
$
```

GNU Parallel Example

```
$ cd $SCRATCH/example
$
```

SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$
```

SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$ icpc -O3 -xhost mycode.cc -o myapp
$
```

SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$ icpc -O3 -xhost mycode.cc -o myapp
$ cat > subjob.lst
  mkdir run01; cd run01; ../myapp 1 > out
  mkdir run02; cd run02; ../myapp 2 > out
  ...
  mkdir run32; cd run32; ../myapp 32 > out
$
```

SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$ icpc -O3 -xhost mycode.cc -o myapp
$ cat > subjob.lst
  mkdir run01; cd run01; ../myapp 1 > out
  mkdir run02; cd run02; ../myapp 2 > out
  ...
  mkdir run32; cd run32; ../myapp 32 > out
$ cat > GPJob
    #PBS -l nodes=1:ppn=8,walltime=12:00:00
    cd $SCRATCH/example
    module load intel gnu-parallel/20121022
    parallel --jobs 8 < subjob.lst
$
```

SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$ icpc -O3 -xhost mycode.cc -o myapp
$ cat > subjob.lst
  mkdir run01; cd run01; ../myapp 1 > out
  mkdir run02; cd run02; ../myapp 2 > out
  ...
  mkdir run32; cd run32; ../myapp 32 > out
$ cat > GPJob
    #PBS -l nodes=1:ppn=8,walltime=12:00:00
    cd $SCRATCH/example
    module load intel gnu-parallel/20121022
    parallel --jobs 8 < subjob.lst
$ qsub GPJob
    2961985.gpc-sched
$
```

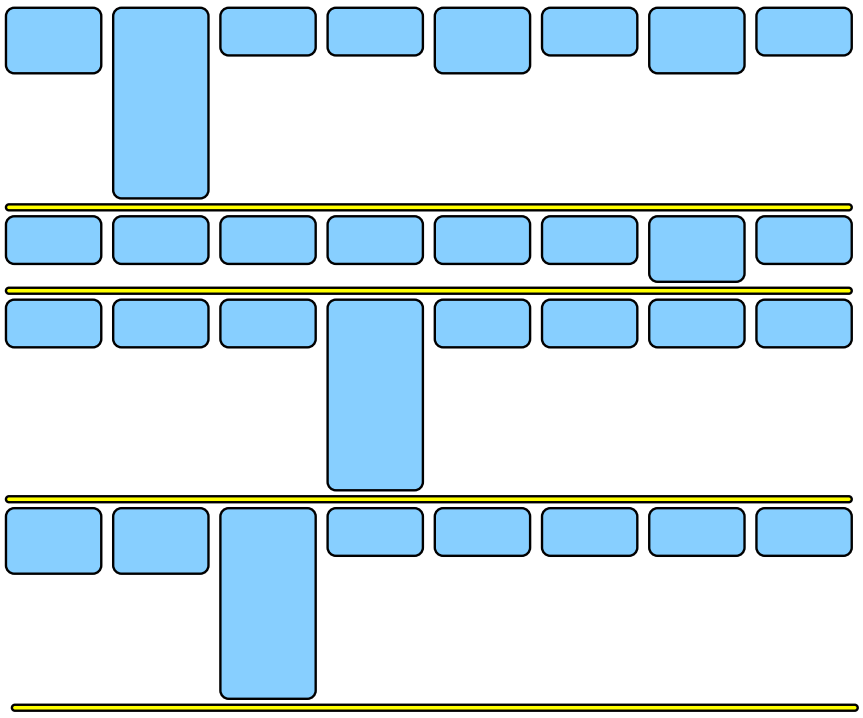
SciNet compute • calcul CANADA

GNU Parallel Example

```
$ cd $SCRATCH/example
$ module load intel
$ icpc -O3 -xhost mycode.cc -o myapp
$ cat > subjob.lst
  mkdir run01; cd run01; ../myapp 1 > out
  mkdir run02; cd run02; ../myapp 2 > out
  ...
  mkdir run32; cd run32; ../myapp 32 > out
$ cat > GPJob
    #PBS -l nodes=1:ppn=8,walltime=12:00:00
    cd $SCRATCH/example
    module load intel gnu-parallel/20121022
    parallel --jobs 8 < subjob.lst
$ qsub GPJob
    2961985.gpc-sched
$ ls
    GPJob  GPJob.e2961985  GPJob.o2961985  subjob.lst
    myapp  run01           run02           run03
    ...
```

SciNet compute • calcul CANADA

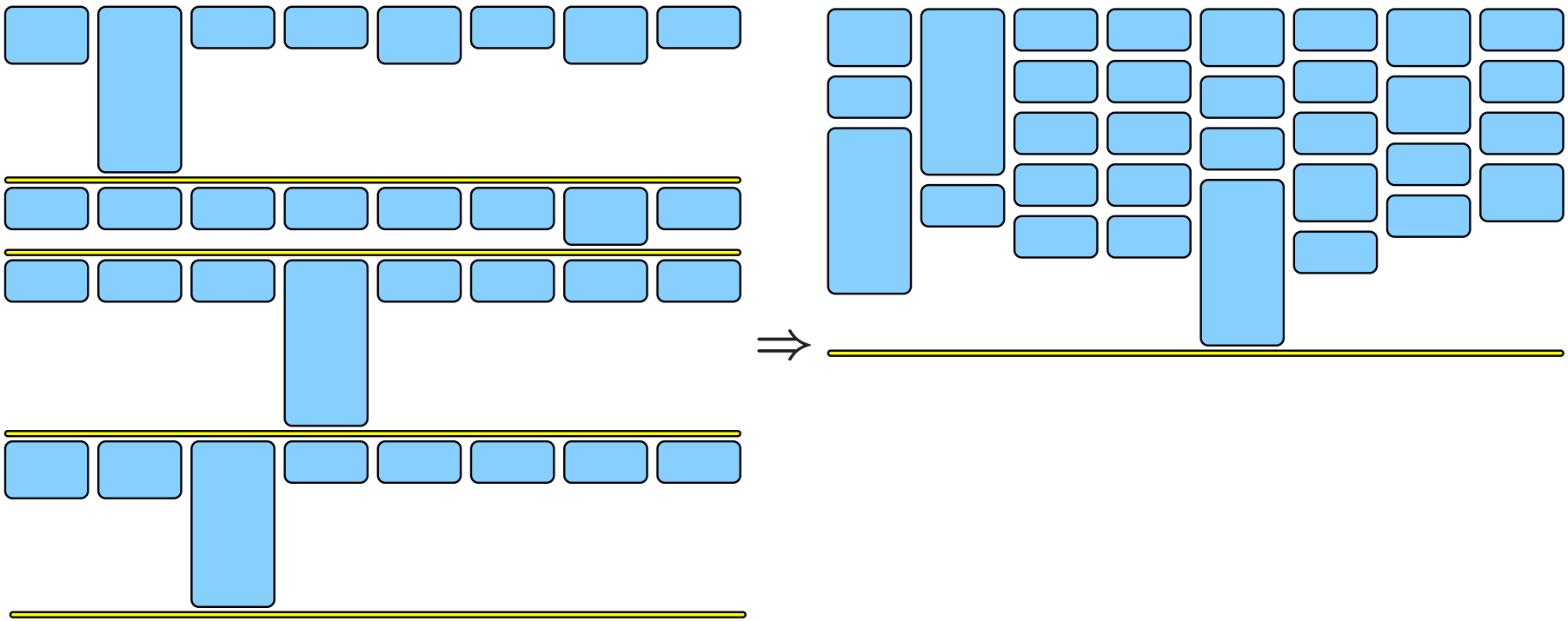
GNU Parallel Example



17 hours

42% utilization

GNU Parallel Example



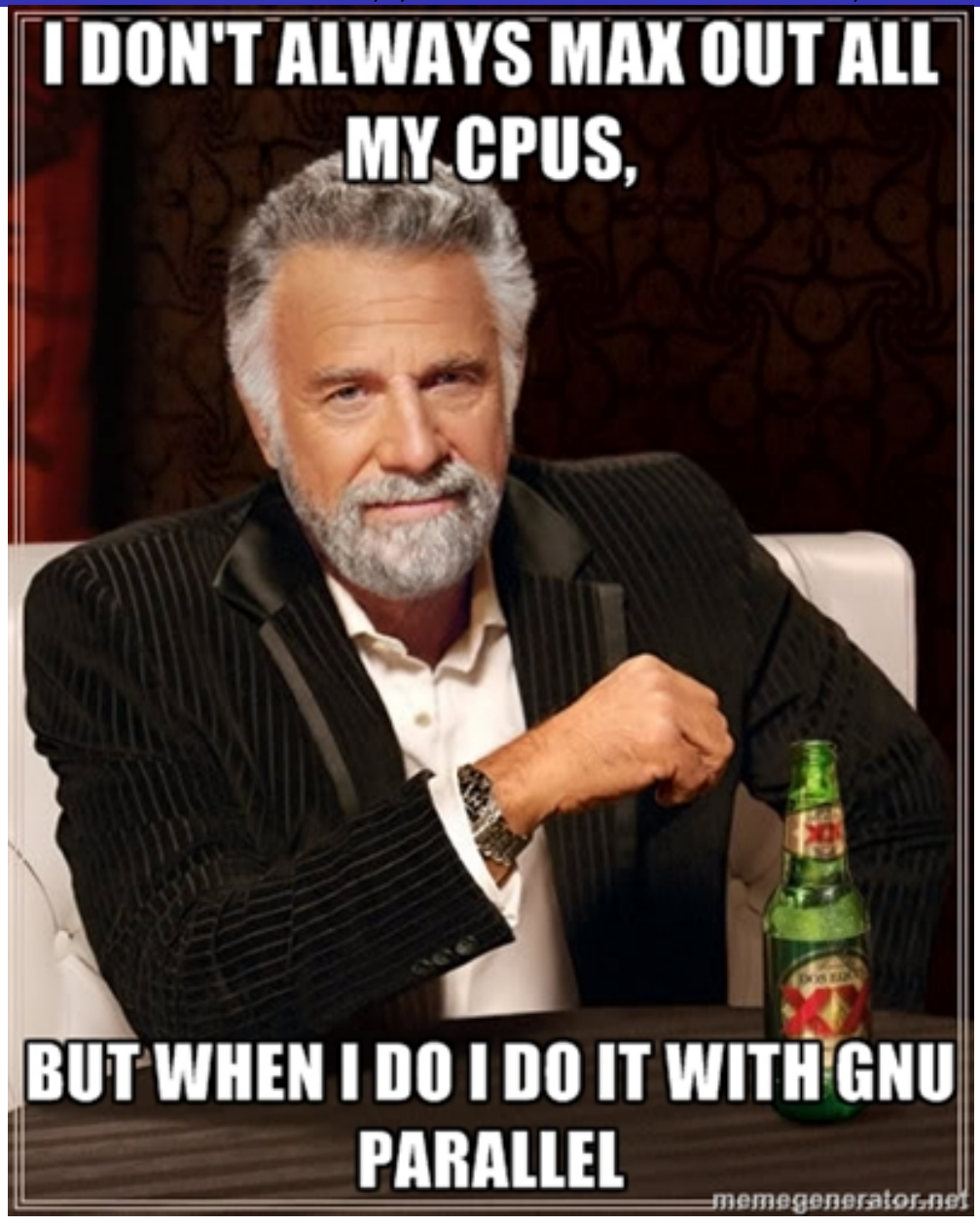
17 hours
42% utilization

10 hours
72% utilization

SciNet
compute • calcul
CANADA

GNU Parallel Testimonial

http://memegenerator.net/instance/22638454



SciNet compute • calcul CANADA

GNU Parallel Syntax

Main modes

1. Reading commands to be run in parallel from an input file.

```
parallel [OPTIONS] < CMDFILE
```

GNU Parallel Syntax

Main modes

1. Reading commands to be run in parallel from an input file.

```
parallel [OPTIONS] < CMDFILE
```

2. Reading command arguments from an input file.

```
parallel [OPTIONS] COMMAND [ARGUMENTS] < ARGFILE
```

This converts each line from the ARGFILE into the argument of **COMMAND** to run in parallel. If present, in the **ARGUMENTS** given to parallel, {} is replaced by each input line.

GNU Parallel Syntax

Main modes

1. Reading commands to be run in parallel from an input file.

```
parallel [OPTIONS] < CMDFILE
```

2. Reading command arguments from an input file.

```
parallel [OPTIONS] COMMAND [ARGUMENTS] < ARGFILE
```

   This converts each line from the ARGFILE into the argument of **COMMAND** to run in parallel. If present, in the **ARGUMENTS** given to parallel, {} is replaced by each input line.

3. Giving command arguments on the command line.

```
parallel [OPTIONS] COMMAND [ARGUMENTS] ::: ARGLIST
```

   Works as case 2, as if the 'words' in the **ARGLIST** were in a file.

GNU Parallel Syntax

Common OPTIONS

- **`--jobs NUM`** sets number of simultaneous subjobs.
  Default: number of virtual cores, i.e. **16** on the GPC.
- **`--joblog LOGFILE`** causes a record for each completed subjob to be written to **`LOGFILE`**, with info on how long they took, their exit status, etc.
- **`--resume`**, combined with **`--joblog`**, continues a gnu-parallel job that was killed prematurely.
- **`--pipe`** splits stdin into chunks given to the stdin of each job.
- **`--block-size NUM`** sets approx. number of bytes per chunk.
- **`--L NUM`** sets number of lines in each chunks.

Resource utility beyond CPU cycles

Apart from CPU cycles, your runs will use other resources as well:

- Network
- Memory
- File system

Of these, network is not too important for the serial runs that we are considering here (no communication).

Memory

Each regular GPC compute node has about 14.5 GB available (16GB - 1.5GB for the operating system)

There is no 'swap' space.

- Do the 8–16 runs fit in memory?
- Use fewer simultaneous runs if really needed.
- Or request one of 80 nodes with 32 GB memory (`-l nodes=1:m32g:ppn=8`).
- Or request one of 2 nodes with 128 GB memory (`-q largemem`).
- Or explore parallization (MPI, OpenMP, pthreads).

SciNet compute • calcul CANADA

File system

One shared file system

- Good: optimized for large files
- Good: your files are everywhere
- Not-so-good: small files (and small file access) can be slow
- Not-so-good: everybody can be affected.

File system

One shared file system

- Good: optimized for large files
- Good: your files are everywhere
- Not-so-good: small files (and small file access) can be slow
- Not-so-good: everybody can be affected.

Guidelines

- Maximize size of files. Large block I/O optimal!
- Minimize number of files. Makes filesystem more responsive!
- Write binary format files: Faster I/O, less space.
- Write and read big chunks at a time, in a linear fashion.

Or use ramdisk

Ramdisk properties

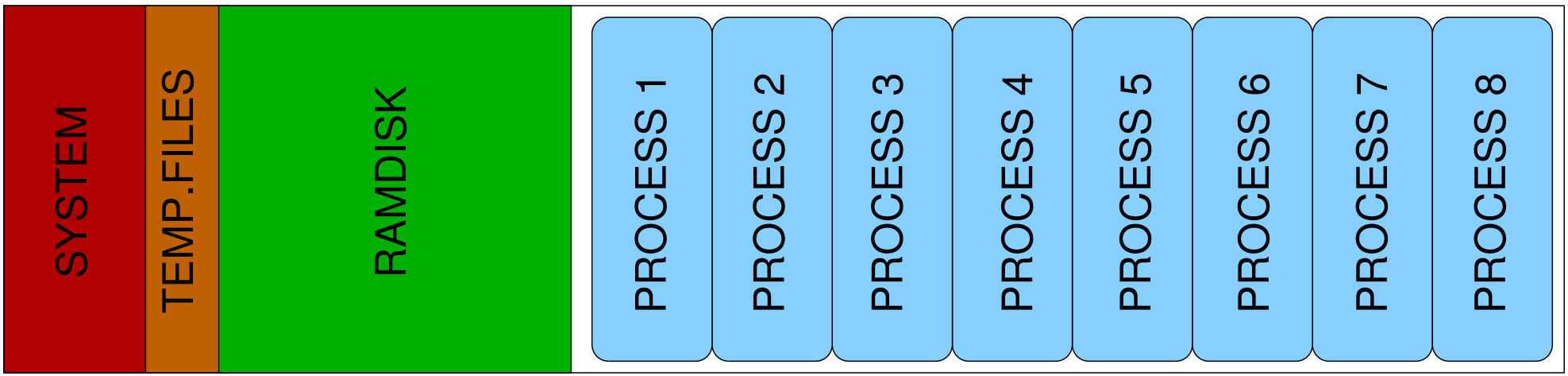
- Good: not shared so not affected by other GPC jobs
- Good: does not care about small files
- Good: low latency and high bandwidth
- Not-so-good: not shared so you have to stage files in and out
- Not-so-good: takes memory from your application

Up to 11GB of the memory of the node can be used as a local storage: ramdisk.

SciNet compute • calcul CANADA

Ramdisk

Memory real estate



If your GPC job runs out of memory, it simply crashes!

SciNet
compute • calcul
CANADA

Ramdisk

Suggested use cases:

- Your app creates many temp files that you do not need.
  Put them on ramdisk instead and forget about them
- Your app accesses data through random reads (e.g. database)
  Put input data on ramdisk
- Your application needs many small input files
  Keep them in a tarball on `$SCRATCH` *and untar into ramdisk*
- Your application creates many small output files
  Write to ramdisk and tar result into a tarball on `$SCRATCH`

All provide you have enough memory real estate.

Ramdisk

Using ramdisk in a single-node GPC job is easy enough:

- The ramdisk is located at the directory `/dev/shm/`.
- Anything put in that directory is actually in memory.
- Memory is shared on the node: all subjobs see the same content.
- Regular `cp` or `tar` commands in your job script can stage files in and out.

Ramdisk

Using ramdisk in a single-node GPC job is easy enough:

- The ramdisk is located at the directory `/dev/shm/`.
- Anything put in that directory is actually in memory.
- Memory is shared on the node: all subjobs see the same content.
- Regular `cp` or `tar` commands in your job script can stage files in and out.

Example

```
mkdir /dev/shm/work/
cp database.db subjob.lst /dev/shm/work/
cd /dev/shm/work
parallel --jobs 8 --joblog $SCRATCH/log < subjob.lst
tar cf $SCRATCH/result.tar *.out
```

GNU Parallel On Multiple Nodes

- If you have hundreds of serial subjobs that you want to run concurrently, the approach above would require tens of scripts to be submitted separately.
- Luckily, GNU Parallel can distribute jobs over a number of nodes.

GNU Parallel On Multiple Nodes

How do we do it?

- GNU Parallel can be passed a file with the list of nodes to which to ssh.
- Such a file is in fact automatically generated by the Moab/Torque scheduler and its name is made available in the environment variable `$PBS_NODEFILE`.
- Important: Properly set the working directory on the other nodes. Otherwise, the run tries to start from the wrong place (`$HOME`) and will likely fail.

GNU Parallel On Multiple Nodes

Relevant options

- **`--sshloginfile FILE`**: **`FILE`** contains the hostnames to use. You should give the option **`--sshloginfile $PBS_NODEFILE`**.
- **`--jobs NUM`**: Run **`NUM`** simultaneous subjobs per node.
- **`--workdir DIR`**: The working directory on all nodes is **`DIR`**.
- **`--env VAR`** copies environment variable **`VAR`** to other nodes. Useful for threaded applications (**`OMP_NUM_THREADS`**).
- **`--basefile FILE`** copies **`FILE`** to all nodes before starting subjobs. Use an absolute path.
- **`--nonall`** execute the given command on each node.

GNU Parallel On Multiple Nodes

Example (Twenty-five node GNU Parallel Job)

```
#PBS -l nodes=25:ppn=8,walltime=1:00:00
cd $SCRATCH/example
module load gnu-parallel/20121022
parallel --jobs 8 --sshloginfile $PBS_NODEFILE \
--joblog progress.log --workdir $PWD < subjob.lst
```

This runs 200 subjobs at the same time on 25 nodes.
There should be at least a couple thousand subjobs in subjob.lst for this to make sense.

GNU Parallel On Multiple Nodes

Example (Twenty-five node GNU Parallel Job)

```
#PBS -l nodes=25:ppn=8,walltime=1:00:00
cd $SCRATCH/example
module load gnu-parallel/20121022
parallel --jobs 8 --sshloginfile $PBS_NODEFILE \
--joblog progress.log --workdir $PWD < subjob.lst
```

This runs 200 subjobs at the same time on 25 nodes.
There should be at least a couple thousand subjobs in subjob.lst for this to make sense.

How about ramdisk?

GNU Parallel On Multiple Nodes with Ramdisk

- Ramdisks not shared among nodes.
- Stage-in and stage-out become a bit more complicated.
- Default GNU Parallel staging (e.g. **`--basefile`**) does not account well for "master node" being a "compute node" too (**`--basefile`** removes file afterward even on the master node).
- **`parallel --sshloginfile $PBS_NODEFILE --nonall COMMAND`** very useful for stage-in and stage-out.

SciNet compute • calcul CANADA

GNU Parallel On Multiple Nodes with Ramdisk

GNU Parallel On Multiple Nodes with Ramdisk

Stage in a file to each node

```
parallel --sshloginfile $PBS_NODEFILE --nonall \
cp $SCRATCH/f /dev/shm/f
```

GNU Parallel On Multiple Nodes with Ramdisk

Stage in a file to each node

```
parallel --sshloginfile $PBS_NODEFILE --nonall \
cp $SCRATCH/f /dev/shm/f
```

Untar a tar file to /dev/shm on each node

```
parallel --sshloginfile $PBS_NODEFILE --nonall \
tar xf $SCRATCH/data.tar -C /dev/shm
```

SciNet compute • calcul CANADA

GNU Parallel On Multiple Nodes with Ramdisk

Stage in a file to each node

```
parallel --sshloginfile $PBS_NODEFILE --nonall \
cp $SCRATCH/f /dev/shm/f
```

Untar a tar file to /dev/shm on each node

```
parallel --sshloginfile $PBS_NODEFILE --nonall \
tar xf $SCRATCH/data.tar -C /dev/shm
```

Tar results to a tar file on scratch, one for each node

```
NDS=$(uniq $PBS_NODEFILE|wc -l)
seq $NDS|parallel --sshloginfile $PBS_NODEFILE -j1 \
tar cf $SCRATCH/result{}.tar -C /dev/shm \*.out
```

Notes

- These were just examples of what you could do with parallel.
- Submitting several bunches to single nodes is more failsafe, since a node failure would only affect one node, rather than all.
- If memory allows it, run upto 16 runs per node. This utilizes the HyperThreading capabilities of the Nehalem chips, often leading to better throughput (but test!).
- Shorter forms for parallel's command line options

```
--sshloginfile  --slf
--jobs          -j
--workdir       --wd
--block-size    --block
--basefile      --bf
```

SciNet compute • calcul CANADA

Links

Serial Subjobs on the GPC:

wiki.scinethpc.ca/wiki/index.php/User_Serial

Using Ramdisk on the GPC:

wiki.scinethpc.ca/wiki/index.php/User_Ramdisk

Using GNU Parallel

- www.gnu.org/software/parallel
- www.gnu.org/software/parallel/man.html
- www.youtube.com/playlist?list=PL284C9FF2488BC6D1
- O. Tange, GNU Parallel – The Command-Line Power Tool, ;login: The USENIX Magazine, February 2011:42-47.