Distributed Computation via MPI

02-11-2020

There are two common parallelization models in parallel programming

- Master / Worker (like what we have seen in `parallel`'s `parLapply`)
- Single Program Multiple Data (SPMD)

  SPMD (single program, multiple data) is a technique employed to achieve parallelism; it is a subcategory of MIMD. Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster.

Advantages of SPMD over Master/Worker - It is very close to the serial code. i.e. SPMD is easy to modify from serial. - It is much shorter than the original Master/Worker version. i.e. SPMD is traceable for debugging. - It makes the master as one of workers. i.e. SPMD fully utilizes resources. - It is easy to automatically process large numbers of independent jobs. i.e. SPMD can parallelize by jobs.

MPI

MPI (Message Passing Interface) is one of the most standard parallel computing architectures. In R, there are two MPI packages `Rmpi` and `pbdMPI`.

- `Rmpi` has been here for a long time, so it is more popular to `Rmpi` given its age. However it is not actively maintained now which means that `Rmpi` is not working probably with new versions of MPI (for example, >= openmpi 2.0).
  - `Rmpi` could be used to execuate both master/worker type of jobs or SPMD type of jobs but there is a bug in `Rmpi` with makes `master/worker` mode fails in our cluster.
- `pbdMPI` is a late comer (which means that it is actively maintained) and it claims that it's faster than `Rmpi`

SPMD

The general process in SPMD programs goes something like this:

- initialize communicators
- have each process read in its portion of the data
- perform computations
- communicate results
- shut down the communicators

Our cluster

Our cluster uses SLURM to manage resources. There are two common resource management systems, PBS and SLURM. It's more difficult to adapt to the other system once you know how a system works.

- first ssh into your account on peloton.cse.ucdavis.edu
- upload your R script to the server (either by a GUI software such as Cyber Duck/ WinScp or using `scp`)
- Use the following to download all the demo codes,

```
svn export https://github.com/UCDavis-STA-141C-Winter-2020/sta141c-lectures/branches/master/02-11/demo/
```

Important! - Do not run R in the head node - Be considerate, do not specific more than enough cores. Our class has 128 cores shared betwwen 50 students. Use `sinfo` to query the current status of the cluster and `squeue` to check the running jobs in the cluster.

Basic Communicator Wrangling

Managing a Communicator: Create and destroy communicators - `init()` Initialize communicator - `finalize()` shut down communicator(s) Rank query: Determine the processor's position in the communicator - `comm.rank()` "who am I?"" - `comm.size()` "how many of us are there?"" - Print: printing with control over which processor prints. - `comm.print(x, ...)` - `comm.cat(x, ...)`

To run the script `01-pbdmpi.R`

```
# make site packages such as pbdMPI available
module load R
# use 3 cores and 1 minute limit
# high2 is a partition reserved for our class
srun -p high2 -n 3 -t 1 Rscript 01-pbdmpi.R
# use 3 cores on 2 nodes and 1 minute limit
srun -p high2 -N 2 -n 3 -t 1 Rscript 01-pbdmpi.R
```

You could press `ctrl+c` anytime to interrupt the job.

Communication functions

- `bcast` - A Rank Broadcasts an Object to Every Rank
- `scatter` - A Rank Scatter Objects to Every Rank
- `gather/allgather` - A Rank / All Ranks Gather(s) Objects from Every Rank
- `reduce/allreduce` - A Rank / All Ranks Receive(s) a Reduction of Objects from Every Rank
- `send/recv` - A Rank Sends / Receives (blockingly) an Object to / from the Other Rank
- `isend` - A Rank Sends (non-blockingly) an Object to the Other Rank

Parallel Lapply Functions

- `pbdLapply` - analogue of `lapply()`
- `pbdSapply` - analogue of `sapply()`

There are three modes: `mw`, `spmd` and `dist`.

Batch submission

If you task takes a decent amount of time to complete, you may not want to run it directly via `srun` as any network connection issue may cause data lost. Instead, you could submit your job via `sbatch`.

See 02-11/batch/

```
# change directory to where job.sh is located
sbatch job.sh
```

Use `squeue -u $USER` to show all your current jobs and use `scancel <JOBID>` to cancle your job.

Array job

If you have a lot of tasks to be executed but they absolutely need no communications, you could consider using an array job.

See 02-11/array/

You could also submit an array job via `sbatch`.

```
# change directory to where array.sh is located
sbatch array.sh
```

Reference:

- Data Science with R and pbdR at ORNL https://www.olcf.ornl.gov/wp-content/uploads/2018/07/pbdR_part2.pdf
- A Forgotten MPI Tool For R http://thinkdatascience.com/pbdr-a-forgotten-mpi-tool-for-r/