

R in Parallel: From Laptop to Supercomputer

Ott Toomet

Seattle, Nov 14th, 2015

Outline

- 1 Background
- 2 Clusters
 - Multicore
 - Socket
 - MPI
- 3 Data Parallelism
- 4 High-Performance Cluster

Fill the results table

task	single thread	multicore	socket (localhost)	...
10^7	time in s
10^5

What is R

- Tiobe top-20 programming language
- One of the most popular language for data analysis and statistics
- Superb graphics
- * No built-in thread/parallel programming support
- * `parallel`-package for explicit (coarse) parallelism
 - You explicitly call parallel code
- * revolution R for implicit (fine) parallelism
 - The software parallelizes standard constructions automatically
 - Uses parallel libraries like *ScaLAPACK*

What is R

- Tiobe top-20 programming language
- One of the most popular language for data analysis and statistics
- Superb graphics
- * No built-in thread/parallel programming support
- * parallel-package for explicit (coarse) parallelism
 - You explicitly call parallel code
- * revolution R for implicit (fine) parallelism
 - The software parallelizes standard constructions automatically
 - Uses parallel libraries like *ScaLAPACK*

Background

Clusters

Multicore
Socket
MPI

Data
Parallelism

HPC

- just R (command line)
- R-Studio
- ESS (Emacs)
- R CMD Batch
 - loads/saves workspace
- Rscript
 - Less bloated version of R CMD BATCH
 - Does not load/save workspace (see below)

Multicore Parallelism

Background

Clusters

Multicore
Socket
MPI

Data
Parallelism

HPC

- Almost all computers nowadays use multicore processors.
 - Shared memory
 - Fast
 - Cheap
- `mclapply()`
- Let's use it!
- Example: 33 vs 75 seconds on my laptop (4 workers)
- But it does not work on windows ☹
 - Works but does not increase speed
- `browser()` does not work
- Not load balancing

Multicore Parallelism

- Almost all computers nowadays use multicore processors.
 - Shared memory
 - Fast
 - Cheap
- `mclapply()`
- Let's use it!
- Example: 33 vs 75 seconds on my laptop (4 workers)
- But it does not work on windows 😞
 - Works but does not increase speed
- `browser()` does not work
- Not load balancing

Example Program

- Embarrassingly parallel task
- Compute normal density of a long vector
- Find maximum
- How many threads to run?
 - `detectCores()`

Single Core vs Multicore

Background

Clusters

Multicore

Socket

MPI

Data

Parallelism

HPC

Example on 8-core processor:

```
> x <- DGP(N)
```

```
> system.time(search1(nGrid=10))
```

```
Maximum -21427517 at mu = 0.5555556 and  
sigma = 2.222778
```

```
user system elapsed  
158.041    2.590  160.717
```

```
> system.time(search2(nGrid=10))
```

```
Maximum -21427517 at mu = 0.5555556 and  
sigma = 2.222778
```

```
user system elapsed  
138.377    3.505   21.192
```

- open new workers on different computers
 - including on “localhost”
- Access these over internet
- Allows to use multiple computers
- `makePSOCKcluster()`
- Example: 50 vs 75 seconds on my laptop (2 workers)
- Example with 2 computers: 23 vs 75 seconds
- Have to export data
- Communication slow
 - *top* shows the workers only partly (30%) busy with small vectors
- Need password-less ssh connection

- open new workers on different computers
 - including on “localhost”
- Access these over internet
- Allows to use multiple computers
- `makePSOCKcluster()`
- Example: 50 vs 75 seconds on my laptop (2 workers)
- Example with 2 computers: 23 vs 75 seconds
- Have to export data
- Communication slow
 - *top* shows the workers only partly (30%) busy with small vectors
- Need password-less ssh connection

Conclusion So Far

Time in seconds

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100

size	single thread	multicore	socket (localhost)	2 hosts
10^7	75	33	35	23
10^5	74	33	121	30

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100

size	single thread	multicore	MPI	2 hosts
10^7	75	33	49	23
10^5	74	33	353	30

Data Parallelism

- Run the same code on different (chunks of) data
- pbdMPI library
- Works well with a HPC and *mpirun*
- Can be used with distributed data (big data)

Example 1

Two processes execute the same code

- both generate different random numbers
- both print

Example 2

Only master generates data

- Master shares data with all workers

Example 3

Gridsearch example

- Master generates data
- Shares it to workers
- Workers calculate their share
- Master performs the final analysis

- length $10^7 \Rightarrow$ grid 10×10
- length $10^5 \Rightarrow$ grid 100×100
- hyak: $10^7 \Rightarrow$ grid 100×100 , 32 CPUs

size	single thread	multicore	pbdbMPI
10^7	75	33	48
10^5	74	33	46
hyak			244

A Few Commands

`comm.rank()` id of the process (0 = master)

`comm.cat()` parallel-aware printing

`get.jid()` get job list

`task.pull()` parallel tasks (like `lapply()`)

`bcast()` share a value among all workers

`gather()` pull values from all workers

High-Performance Cluster

UW hyak:

- 20,000 cpu cores
- 100TB memory
- MOAB cluster software
- TORQUE scheduler
- use *pbs scripts*
 - Tell the scheduler how much resources you want ...
 - ...and run your stuff ☺
- submit the jobs by *qsub*