Performance and Parallel Evaluation

Martin Morgan (mtmorgan@fredhutch.org)
Fred Hutchinson Cancer Research Center
Seattle, WA, USA

19 June, 2015
Performance & Parallel Evaluation

- R code
  - Correct, then efficient

Parallel evaluation
- Computer
- Cluster
- Cloud
Priorities

1. Correct!
2. Robust – works for most realistic inputs
3. Simple
4. Fast
R code: deadly sins

1. Unnecessary iteration
   ```r
   x <- 1:10000; for (i in seq_along(x)) x[i] = log(x[i])
   ```

2. Copy-and-append iteration
   ```r
   answer <- numeric()
   for (i in 1:10000) answer <- c(answer, 1/i)
   for (i in 1:10000) answer[i] <- 1/i
   ```

3. Unnecessary evaluation
   ```r
   x <- 1:1000000
   for (i in seq_along(x)) x[i] = x[i] * sqrt(2)
   ```

4. Re-implementation
fun1 <- function(n) {
  ## How many sins?
  x <- numeric()
  for (i in 1:n)
    x <- c(x, log(i) * sqrt(2))
  x
}

fun2 <- function(n)
  log(seq_len(n)) * sqrt(2)
**R code: saving graces II**

1. Validation – `identical()`, `all.equal()`

```r
identical(fun1(1000), fun2(1000))
## [1] TRUE
```

2. Timing – `system.time()`, `microbenchmark()`

```r
library(microbenchmark)
microbenchmark(fun1(1000), fun2(1000))
## Unit: microseconds
## expr    min     lq mean    uq median    max neval
## fun1(1000) 1726.480 1746.0545 2518.5444 5122.918 40.6885 63.635 100
## fun2(1000)  36.131   36.8945  41.0648    43.264 40.6885  43.264 100
```
3. ‘Experience’ – available packages & functions
4. Profiling – `Rprof()`
5. Foreign languages – e.g., C, `Rcpp`
Parallel evaluation

- Most often: ‘embarrassingly parallel’ evaluation of iterative `for` loops / `lapply()`

Other packages

- `parallel` – a base package; single computer
- `foreach` – popular ‘for’ loop paradigm
- `BatchJobs` – clusters with job schedulers
- `Rmpi` – classic HPC

**BiocParallel**

- Consistent interface
- Plays well with many Bioconductor packages
Parallel evaluation

```r
library(BiocParallel)

fun <- function(i) {
  Sys.sleep(1)
  i
}

system.time(res1 <- lapply(1:5, fun))
## user  system elapsed
## 0.002 0.000  5.007

system.time(res2 <- bplapply(1:5, fun))
## user  system elapsed
## 0.035 0.024  1.085

identical(res1, res2)
## [1] TRUE
```
Parallel evaluation: **BiocParallel**

- Different *Param() objects for styles of computing, e.g.,
  - SerialParam(): no parallel evaluation
  - MulticoreParam(): separate forked processes on one computer
  - BatchJobsParam(): jobs submitted to a cluster queuing system
- `register()` a param or provide it as an argument for use in `bplapply()`.
- Sensible default values.
Parallel evaluation: processing large genomic files

Restrict input to minimum necessary data

- Select columns or fields of files to import, e.g., colClasses argument to read.table(); ScanBamParam() and ScanVcfParam().
- Use a data base, hdf5, or other file format that allows queries or slices of the data to be imported.

Iterate through files to manage memory use

- File connections in base R
  - BamFile("my.bam", yieldSize=1000000)

GenomicFiles

- Functions to help manage collections of genomic files
Parallel evaluation: extended example

Goal: for a vector of paths to bam files, `fls`, summarize GC content of each aligned read.

```r
library(Rsamtools); library(GenomicFiles)
bfls <- BamFileList(fls, yieldSize=100000)
yield <- function(bfl)  # input a chunk of alignments
  readGAlignments(bfl, param=ScanBamParam(what="seq"))
map <- function(aln) {
  # GC content, bin & cummulate
  gc <- letterFrequency(mcols(aln)$seq, "GC",
                        as.prob=TRUE)
  cumsum(tabulate(1 + gc * 50, 51))
}
reduce <- `+`

gc <- bplapply(bfls, reduceByYield, yield, map, reduce)
```
Summary

- **Correct** first, performance second
- No need to worry about code that doesn’t take very long!
- ‘Embarassingly’ parallel (``lapply()``-like) problems easily parallelized, especially on a single computer.
- Opportunity for very scalable computations, e.g., via AMI & StarCluster.
Acknowledgments

- Core (Seattle): Sonali Arora, Marc Carlson, Nate Hayden, Valerie Obenchain, Hervé Pagès, Paul Shannon, Dan Tenenbaum.

- The research reported in this presentation was supported by the National Cancer Institute and the National Human Genome Research Institute of the National Institutes of Health under Award numbers U24CA180996 and U41HG004059, and the National Science Foundation under Award number 1247813. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or the National Science Foundation.