Speeding up R with Parallel Programming in the Cloud

David M Smith
Developer Advocate, Microsoft
@revodavid
Embarassingly Parallel Problems

Easy to speed things up when:

• Calculating similar things many times
  – Iterations in a loop, chunks of data, ...
• Calculations are independent of each other
• Each calculation takes a decent amount of time

Just run multiple calculations at the same time
The Birthday Paradox

What is the likelihood that there are two people in this room who share the same birthday?
pbirthdaysim <- function(n) {
  ntests <- 100000
  pop <- 1:365
  anydup <- function(i)
    any(duplicated(
      sample(pop, n, replace=TRUE)))
  sum(sapply(seq(ntests), anydup)) / ntests
}

bdayp <- sapply(1:100, pbirthdaysim)  About 3 minutes
(on this laptop)
library(foreach)

x <- foreach (n=1:100) %dopar% pbirthdaysim(n)

Looping with the foreach package on CRAN
   – x is a list of results
   – each entry calculated from RHS of %dopar%

Learn more about foreach: cda.ms/tc
Parallel processing with foreach

- Change how processing is done by registering a **backend**
  - registerDoSEQ() sequential processing (default)
  - registerdoMC() local cores via multicore (Mac / Linux)
  - registerdoParallel() local cluster via library(parallel)
  - registerdoFuture() HPC schedulers incl LSF, OpenLava & Slurm
  - registerAzureParallel() remote cluster in Azure Batch

- Whatever you use, call to foreach **does not change**
  - Also: no need to worry about data, packages etc. (mostly)
Local multi-process: foreach + doParallel

library(doParallel)
cl <- makeCluster(2) # local cluster, 2 workers
registerDoParallel(cl)
bdayp <- foreach(n=1:100) %dopar% pbirthdaysim(n)

Launches a separate R process for each task
Works on all platforms (Windows / Mac / Unix)
Single node multicore: doMC on 16-CPU VM

```r
> library(doMC)
> registerDoMC(cluster)

> system.time( bdayp <- foreach(n=1:100) %dopar% pbirthdaysim(n) )
user      system   elapsed
 291.617     0.760   20.743
```

About **11x times faster** than sequential
• 20s vs 220s (using Azure DS5v2 instance)

**Note**: disable multithreaded BLAS to avoid core contention
• In Microsoft R Open: setMKLthreads(1)
Cloud cluster: foreach + doAzureParallel

doAzureParallel: A simple, open-source R package that uses the Azure Batch cluster service as a parallel-backend for foreach

github.com/Azure/doAzureParallel
Birthday simulation: cluster

8-node cluster (standard D2v2: 2 vCPU, 7 Gb)
- specify VM class in cluster.json
- specify credentials for Azure Batch and Azure Storage in credentials.json

library(doAzureParallel)
setCredentials("credentials.json")
cluster <- makeCluster("cluster.json")
registerDoAzureParallel(cluster)

bdyp <- foreach(n=1:100) %dopar% pbirthdaysim(n)
bdyp <- unlist(bdyp)

cluster.json (excerpt):
"name": "davidsmi8caret",
"vmSize": "Standard_D2_v2",
"maxTasksPerNode": 8,
"poolSize": {
  "dedicatedNodes": {
    "min": 8,
    "max": 8
  }
}

45 seconds (more than 5 times faster) on a warm start
Cross-validation with caret

• Most predictive modeling algorithms have “tuning parameters”
• Example: Boosted Trees
  – Boosting iterations
  – Max Tree Depth
  – Shrinkage
• Parameters affect model performance
• Try ‘em out: cross-validate

```r
grid <- data.frame(
  nrounds = ...,  
  max_depth = ..., 
  gamma = ..., 
  colsample_bytree = ..., 
  min_child_weight = ..., 
  subsample = ...) 
)
Cross-validation in parallel

• Caret’s train function will automatically use the registered foreach backend

• Just register your cluster first:
  registerDoAzureParallel(cluster)

• Handles sending objects, packages to nodes

mod <- train(
  Class ~ .,
  data = dat,
  method = "xgbTree",
  trControl = ctrl,
  tuneGrid = grid,
  nthread = 1
)
Low Priority Nodes

- Low-Priority = (very) Low Costs VMs from surplus capacity – up to 80% discount
- Clusters can mix dedicated VMs and low-priority VMs

```
"poolSize": {
    "dedicatedNodes": {
        "min": 3,
        "max": 3
    },
    "lowPriorityNodes": {
        "min": 9,
        "max": 9
    }
}
```
caret speed-ups

- Max Kuhn benchmarked various hardware and OS for local parallel
  - cda.ms/6V

- Let’s see how it works with doAzureParallel

Source: cda.ms/6V
Packages and Containers

• Docker images used to spawn nodes
  – Default: rocker/tidyverse:latest
  – Lots of R packages pre-installed

• But this cross-validation also needs:
  – xgboost, e1071

• Easy fix: add to cluster.json

```json
{
  "name": "davidsmi8caret",
  "vmSize": "Standard_D2_v2",
  "maxTasksPerNode": 8,
  "poolSize": {
    "dedicatedNodes": {
      "min": 4,
      "max": 4
    },
    "lowPriorityNodes": {
      "min": 4,
      "max": 4
    },
    "autoscaleFormula": "QUEUE"
  },
  "containerImage": "rocker/tidyverse:latest",
  "rPackages": {
    "cran": ["xgboost","e1071"],
    "github": [],
    "bioconductor": []
  },
  "commandLine": []
}
```
Id: job20180126022301
chunkSize: 1
enableCloudCombine: TRUE
packages:
  caret;
errorHandling: stop
wait: TRUE
autoDeleteJob: TRUE

Submitting tasks (1250/1250)
Submitting merge task. . .
Job Preparation Status: Package(s) being installed........

Waiting for tasks to complete. . .
| Progress: 13.84% (173/1250) | Running: 59 | Queued: 1018 | Completed: 173 | Failed: 0 |

MY LAPTOP: 78 minutes
THIS CLUSTER: 16 minutes
(almost 5x faster)
Compute costs

• Pay by the minute for VMs used in cluster
• Using D2v2 Virtual Machines
  – Ubuntu 16, 7Mb RAM, 2-core “compute optimized”
• 17 minutes × 8 VMs @ $0.14 / hour
  – about 32 cents (not counting startup or storage)
Thank you!

These slides: cda.ms/ty
Find foreach on CRAN: CRAN.R-project.org/package=foreach
Code for birthday problem simulation: cda.ms/7d
Docs for the foreach package: cda.ms/tc
Get doAzureParallel: cda.ms/7w
Get Azure subscription with $200 free credits: cda.ms/td

David Smith
davidsmi@microsoft.com