Big Data Analytics and HPC

Matthew J. Denny

mzd5530@psu.edu – www.mjdenny.com – @MatthewJDenny

www.mjdenny.com/ICPSR_Data_Science_2015.html

July 28, 2015
Overview

1. Overview of High Performance Computing/Big Data Analytics in R.
2. Programming Choices.
3. Big Data Example.
4. Hardware.
High Performance Computing

- Make use of low overhead, high speed programming languages (C, C++, Fortran, Java, etc.)
- Parallelization
- Efficient implementation.
- Good scheduling.
Big Data Analytics

- Use memory efficient data structures and programming languages.
- More RAM.
- Databases.
- Efficient inference procedures.
- Good scheduling.
How They Fit Together

High Performance Computing

Big Data
Hardware Constraints

- **RAM** = computer working memory – determines size of datasets you can work on.

- **CPU** = processor, determines speed of analysis and degree of parallelization.
Look At Your Activity Monitor!
2. Programming Choices
Efficient R Programming

- Loops are slow in R, but fast enough for most things.
- Built-in functions are mostly written in C – much faster!
- Subset data before processing when possible.
- Avoid growing data structures.
Loops Are “Slow” in R

system.time({
    vect <- c(1:10000000)
    total <- 0
    #check using a loop
    for(i in 1:length(as.numeric(vect))){
        total <- total + vect[i]
    }
    print(total)
})

[1] 5e+13

user  system elapsed
7.641  0.062  7.701
And Dast in C

```r
system.time({
  vect <- c(1:10000000)
  #use the builtin R function
  total <- sum(as.numeric(vect))
  print(total)
})

[1] 5e+13

user  system elapsed
 0.108   0.028   0.136
```
Summing Over A Sparse Dataset

#number of observations
numobs <- 100000000

#observations we want to check
vec <- rep(0,numobs)

#only select 100 to check
vec[sample(1:numobs,100)] <- 1

#combine data
data <- cbind(c(1:numobs),vec)
Conditional Checking

```r
system.time(
{
    total <- 0
    for(i in 1:numobs){
        if(data[i,2] == 1)
            total <- total + data[i,1]
    }
    print(total)
})
[1] 5385484508
user  system elapsed
199.917 0.289 200.350
```
system.time(
{
  dat <- subset(data, data[,2] ==1)
  total <- sum(dat[,1])
  print(total)
}
)
[1] 5385484508
   user  system elapsed
 5.474   1.497   8.245
2.a. Pre-Allocation
Adding To A Vector vs. Pre-Allocation

```r
system.time({
  vec <- NULL
  for (i in 1:(10^5)) vec <- c(vec,i)
})

  user  system elapsed
18.495  7.401  25.935

system.time({
  vec <- rep(NA,10^5)
  for (i in 1:(10^5)) vec[i] <- i
})

  user  system elapsed
0.144  0.002  0.145
```
Pre-Allocated Vector – Bigger Example

```r
system.time({
    vec <- rep(NA,10^6)
    for (i in 1:(10^6)) vec[i] <- i
})
```

```
user  system  elapsed
1.765  0.040  1.872
```
Adding To A Vector – Bigger Example

system.time({
    vec <- NULL
    for (i in 1:(10^6)) vec <- c(vec, i)
})

Timing stopped at: 924.922 120.322 1872.294

I didn’t feel like waiting...
Pre-Allocation

- Vectors in R can only hold about 2.1 Billion elements.
- Write to over-allocated vector then subset.
- Speedup is exponential in the vector size and number of additions.
2.b. Parallelization
Parallelization Using **foreach**

- Works best when we need to calculate some complex statistic on each row/column of dataset.

- Works just like a regular `for()` loop as long as operations are **independent**.

- Good for bootstrapping.
Parallelization Using foreach

# Packages:
require(doMC)
require(foreach)

# Register number of cores
nCores <- 8
registerDoMC(nCores)

# iterations
N <- 100

# Run analysis in parallel
results <- foreach(i=1:N,.combine=rbind) %dopar% {
    result <- function(i)
}

Parallelization Using A snowfall Cluster

- Can run across multiple machines.
- Can run totally different jobs on each thread.
- Requires explicit management by researcher.
Parallelization Using A snowfall Cluster

# Package:
library(snowfall)

# Register cores
numcpus <- 4
sfInit(parallel=TRUE, cpus=numcpus)

# Check initialization
if(sfParallel()){
    cat( "Parallel on", sfCpus(), "nodes.\n" )
} else{
    cat( "Sequential mode.\n" )
}
Parallelization Using A snowfall Cluster

```r
# Export all packages
for (i in 1:length(.packages())){
  eval(call("sfLibrary", (.packages()[[i]],
    character.only=TRUE))
}

# Export a list of R data objects
sfExport("Object1","Object2","Object3")

# Apply a function across the cluster
result <- sfClusterApplyLB(indexes,Function)

# Stop the cluster
sfStop()
```
Parallelization Using `mclapply()`

- **Will not work with Windows machines.**

- **Simple parallelization.**

- **Works well with functions written in Rcpp.**
Parallelization Using `mclapply()`

```r
# Packages:
library(parallel)

# Wrapper Function
run_on_cluster <- function(i){
  temp <- your_function(i,some other stuff)
  return(temp)
}

# Run analysis
indexes <- 1:Iterations
Result <- mclapply(indexes,
                   run_on_cluster,
                   mc.cores = num_cpus)
```
2.c. Memory Efficient Regression
High Memory Regression Using `biglm()`

- Memory efficient implementation of `glm()`
- Can also read in data in chunks from internet or a relational database.
- Will not take data in matrix form, only `data.frame`
# Data must be of data.frame type
data <- as.data.frame(data)

# Use variable names in formula
str <- "V1 ~ V2 + V4"

# Run model
model <- bigglm(as.formula(str),
                data = full_data,
                family = binomial(),
                maxit = 20)
3. Big Data Example
1. 90,867 final versions of bills introduced in the U.S. Congress form 1993-2012.

2. 293,697,605 tokens (370,445 unique).

3. 90 covariates for every bill.

4. Additional data on amendments, cosponsorships, and floor speeches.
This Act may be cited as the ‘‘EPA Science Act of 2014’’.

SEC. 2. SCIENCE ADVISORY BOARD.

(a) Independent Advice.--Section 8(a) of the Environmental Research, Development, and Demonstration Authorization Act of 1978 (42 U.S.C. 4365(a)) is amended by inserting ‘‘independently’’ after ‘‘Advisory Board which shall’’.

(b) Membership.--Section 8(b) of the Environmental Research, Development, and Demonstration Authorization Act of 1978 (42 U.S.C. 4365(b)) is amended to read as follows:

‘‘(b)(1) The Board shall be composed of at least nine members, one of whom shall be designated Chairman, and shall meet at such times and places as may be designated by the Chairman.

‘‘(2) Each member of the Board shall be qualified by education, training, and experience to evaluate scientific and technical ....
<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Substantive</strong></td>
<td>Confers the intent of a piece of legislation or a particular provision.</td>
<td>{restrict abortion}, {reduce the deficit}</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Topical</strong></td>
<td>Confers information about the subject of the Bill.</td>
<td>{alternate academic achievement standards}</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Boilerplate</strong></td>
<td>Gives direction about legal interpretation or implementation.</td>
<td>{Notwithstanding any other provision of this paragraph...}</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>Gives no information about intent, legal interpretation or implementation.</td>
<td>{SECTION}, {(c) Title III.–}, {(1) Part a.–}, {(A) Subpart 1.–}, {to adopt}</td>
</tr>
<tr>
<td><strong>Stopwords</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Lets Look at N-Grams

<table>
<thead>
<tr>
<th>Unit of Analysis</th>
<th>Topical Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tokens</strong></td>
<td>{Should}, {a}, {Federal}, {agency}, {seek}, {to}, {restrict}, {photography}, {of}, {its}, {installations}, {or}, {personnel}, {it}, {shall}, {obtain}, {a}, {court}, {order}, {that}, {outlines}, {the}, {national}, {security}, {or}, {other}, {reasons}, {for}, {the}, {restriction}</td>
</tr>
<tr>
<td><strong>Bigrams</strong></td>
<td>{Should a}, {a Federal}, {Federal agency}, {agency seek}, {seek to}, {to restrict}, {restrict photography}, {photography of}, {of its}, {its installations}, {installations or}, {or personnel}, {personnel it}, {it shall}, {shall obtain}, {obtain a}, {a court}, {court order}, {order that}, {that outlines}, {outlines the}, {the national}, {national security}, {security or}, {or other}, {other reasons}, {reasons for}, {for the}, {the restriction}</td>
</tr>
</tbody>
</table>
## Syntactic Filtering

### Tag Pattern Example

<table>
<thead>
<tr>
<th>Tag Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN</td>
<td>linear function</td>
</tr>
<tr>
<td>NN</td>
<td>regression coefficients</td>
</tr>
<tr>
<td>AAN</td>
<td>Gaussian random variable</td>
</tr>
<tr>
<td>ANN</td>
<td>cumulative distribution function</td>
</tr>
<tr>
<td>NAN</td>
<td>mean squared error</td>
</tr>
<tr>
<td>NNN</td>
<td>class probability function</td>
</tr>
<tr>
<td>NPN</td>
<td>degrees of freedom</td>
</tr>
</tbody>
</table>

**Verbs capture actions...**

<table>
<thead>
<tr>
<th>Tag Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VN</td>
<td>reduce funding</td>
</tr>
<tr>
<td>VAN</td>
<td>encourage dissenting members</td>
</tr>
<tr>
<td>VNN</td>
<td>restrict federal agencies</td>
</tr>
<tr>
<td>Unit of Analysis</td>
<td>Matches</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Filtered Bigrams</td>
<td>{Federal agency}, {restrict photography}, {court order}, {national security}, {other reasons}</td>
</tr>
<tr>
<td>Filtered Trigrams</td>
<td>NONE</td>
</tr>
<tr>
<td>Noun Phrases</td>
<td>{Federal agency}, {court order}, {national security}, {other reasons}, {other reasons for the restriction}</td>
</tr>
</tbody>
</table>
Constructing a Document-Term Matrix

- Want Document x Vocabulary matrix.
- Take advantage of sparsity.
- Use C++ for indexing.
- Have to chunk and add.
# Constructing A Document-Term Matrix

```r
# gives us simple triplet matrix class
library(slam)

# load in C++ function to generate rows in matrix
Rcpp::sourceCpp('Document_Word_Compiler.cpp')

for(i in 1:chunks){
  dwm <- Generate_Document_Word_Matrix(chunk,..)
  dwm <- as.simple_triplet_matrix(dwm)
  if(j == 1){
    swm <- dwm
  }else{
    swm <- rbind(swm,dwm)
  }
}
```

## Semi-Supervised Major Topic Tags

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Macroeconomics</td>
</tr>
<tr>
<td>2. Civil Rights, Minority Issues, and Civil Liberties</td>
</tr>
<tr>
<td>3. Health</td>
</tr>
<tr>
<td>4. Agriculture</td>
</tr>
<tr>
<td>5. Labor and Employment</td>
</tr>
<tr>
<td>6. Education</td>
</tr>
<tr>
<td>7. Environment</td>
</tr>
<tr>
<td>8. Energy</td>
</tr>
<tr>
<td>9. Immigration</td>
</tr>
<tr>
<td>10. Transportation</td>
</tr>
<tr>
<td>12. Law, Crime, and Family Issues</td>
</tr>
<tr>
<td>13. Social Welfare</td>
</tr>
<tr>
<td>14. Community Development and Housing Issues</td>
</tr>
<tr>
<td>15. Banking, Finance, and Domestic Commerce</td>
</tr>
<tr>
<td>16. Defense</td>
</tr>
<tr>
<td>.....</td>
</tr>
</tbody>
</table>
Filtered Trigrams for Identifying Topical Area

<table>
<thead>
<tr>
<th>Word</th>
<th>PMI</th>
<th>Local</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>stay</td>
<td>1.754</td>
<td>24425</td>
<td>26428</td>
</tr>
<tr>
<td>start</td>
<td>1.689</td>
<td>12395</td>
<td>14321</td>
</tr>
<tr>
<td>enhance</td>
<td>1.684</td>
<td>22142</td>
<td>25707</td>
</tr>
<tr>
<td>providers</td>
<td>1.684</td>
<td>51656</td>
<td>59976</td>
</tr>
<tr>
<td>mining</td>
<td>1.679</td>
<td>15221</td>
<td>17751</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filtered Trigram</th>
<th>PMI</th>
<th>Local</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>health insurance plan</td>
<td>1.340</td>
<td>868</td>
<td>1528</td>
</tr>
<tr>
<td>health benefit plan</td>
<td>1.306</td>
<td>1125</td>
<td>2049</td>
</tr>
<tr>
<td>term care insurance</td>
<td>1.292</td>
<td>3564</td>
<td>6577</td>
</tr>
<tr>
<td>health plan means</td>
<td>1.110</td>
<td>261</td>
<td>578</td>
</tr>
<tr>
<td>alternative dispute resolution</td>
<td>1.079</td>
<td>923</td>
<td>2108</td>
</tr>
</tbody>
</table>
Big Data Challenges

1. Extending R vectors/matrices beyond 2.1 billion elements.

2. More low level data structures – linked list, queue, stack, etc.


4. Hardware
Classes of Hardware

1. Supercomputers
2. Mainframes
3. Cluster Computing Resources
4. Servers
5. HPC Workstations
6. Consumer Desktops
7. GPGPU
Supercomputers

- Used when all computing resources are needed to solve one problem.
- Physics, engineering, materials science
Mainframes

- Used for large database applications.
- Business analytics, healthcare.
Cluster Computing Resources

- Flexible, used for parallel and high memory tasks.
- General purpose academic computing infrastructure.
Servers

- Most often used for hosting websites.
- Can be useful for long jobs, high memory.
HPC Workstations

- Personal mid-size high memory and parallel computing.
- For people who moderate resources constantly.
Desktop

- Everything, depending on how long you are willing to wait.
- Will run 95% of what you want to do.
General Purpose GPU Computing

- Problems that break down to small, interdependent parts.
- Bootstrapping, complex looping, optimization.
Pricing Tiers

- **Cluster Access**: Usually free through your institution but often requires application/faculty sponsorship.

- **HPC Workstation**: $8,000-$15,000 – Not a good investment for most.

- **Desktop**: $700-$2000 – Often a very good investment.
My Suggestion

- Ask a faculty member for access. **TRY THIS FIRST.**

- Investing in a desktop with 4C/8T (Intel i7) and 16GB+ of RAM is often a smart idea if it will not get in the way of conference attendance.

- Access a university cluster only once you are confident a desktop can no longer meet your needs.
Upgrades

- Old computers are good for HPC tasks that simply take a while to run.

- Locate computer in academic office for free electricity/internet/easier remote access.

- Relatively cheap upgrades can dramatically improve performance.

- BENCHMARK
Know Your Motherboard

- How many RAM Slots?
- Peripherals, CPU, GPU slots.
RAM

- Work with larger datasets.
- 16GB Kits – ($100-150)
- 32GB Kits – ($200-300)
Solid State Hard Drive

- General system performance, data I/O.
- $0.40-$0.80 per GB
- Leave 15-20% free.
Check Review Sites

[AnandTech]

[tom's hardware]
Things To Remember

- Get an Uninterrupted Power Supply (UPS) for stable power.

- Put a sign on your computer that says don’t touch.
Don’t buy it unless you absolutely need it.

Most resources can be borrowed/had for free.

More powerful resources require more time to learn.