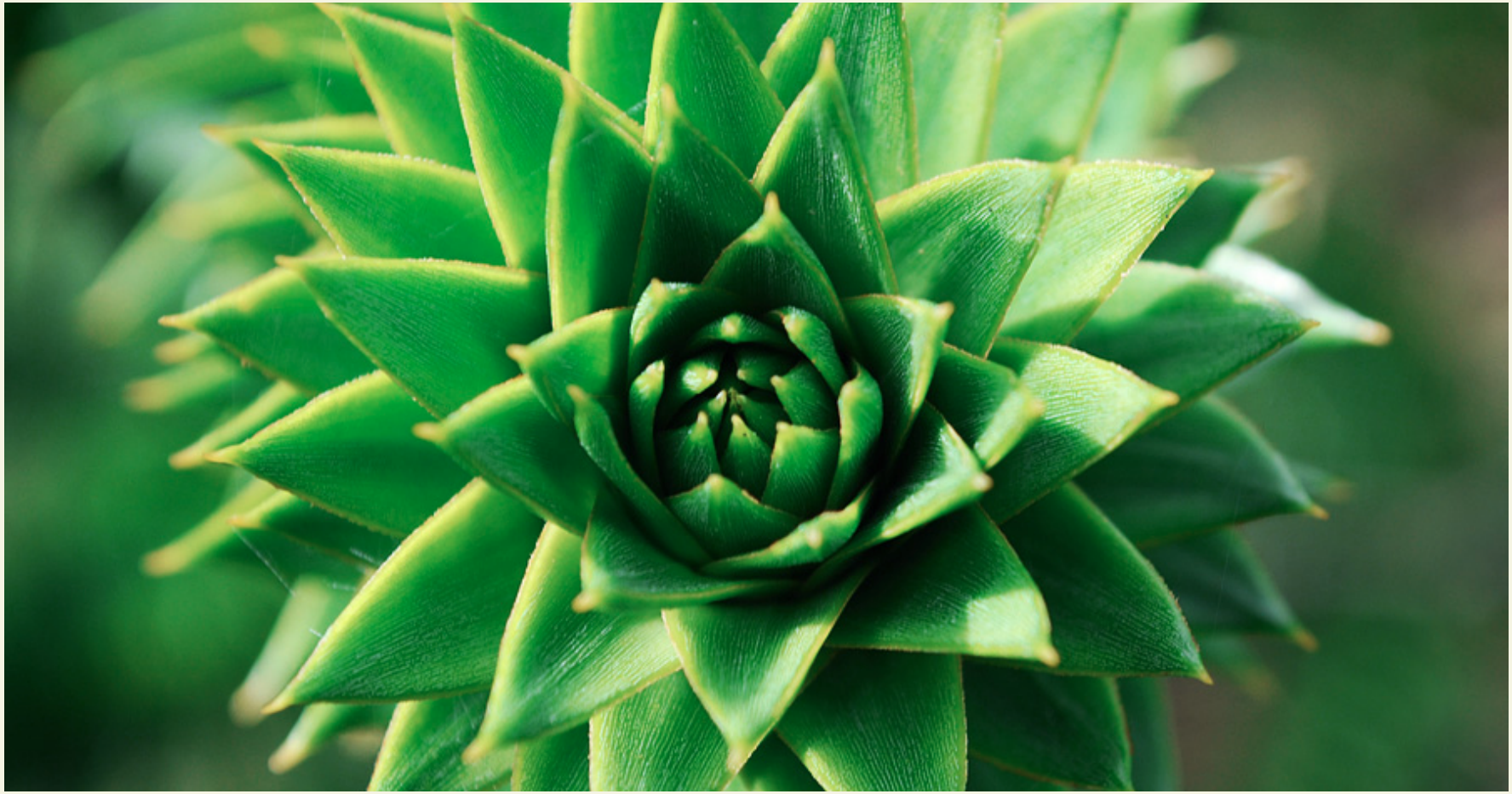
PARALLEL COMPUTING IN R

Theodore Van Rooy

Royalty Consulting

May 23, 2012

THEODORE.VANROOY@GMAIL.COM



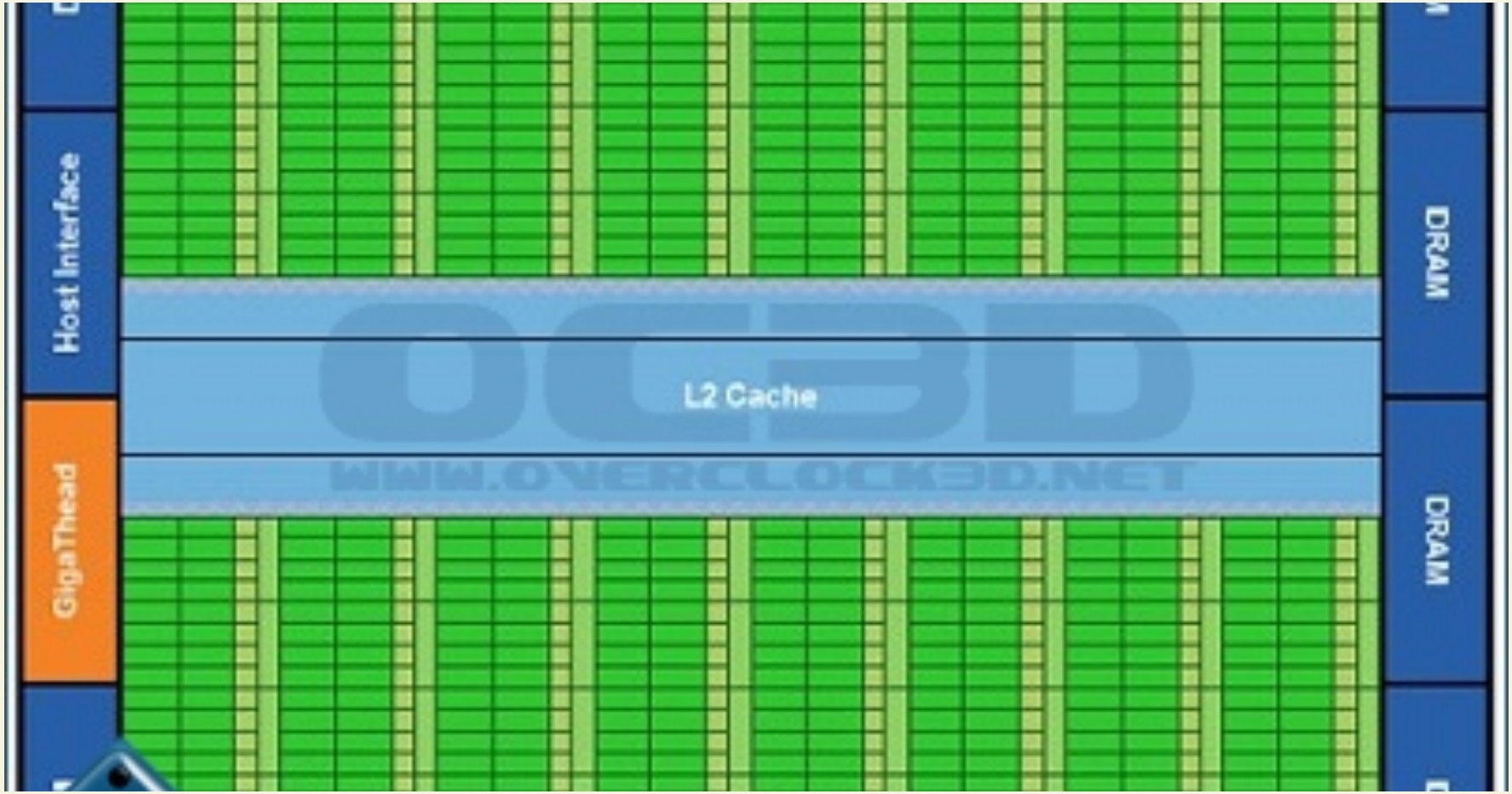
WHY WASTE COMPUTING CYCLES?

- Can spend a lot of time making your code faster:
  - With C or C++
- Or, spend little time with cheap commodity hardware making it parallel.
  - 8 core workstation ~$1000
  - NVIDIA cuda ~$300

CPU VS. GPU

- Both are multi-core
  - CPU's up to 8 cores per chip
  - GPU's hundreds of cores (NVIDIA up to500 cores)
- CPU has more memory, and higher precision, easier to write code for.

A GPU


Host Interface
GigaThead
L2 Cache
DRAM
DRAM
OC3D
WWW.OVERCLOCK3D.NET

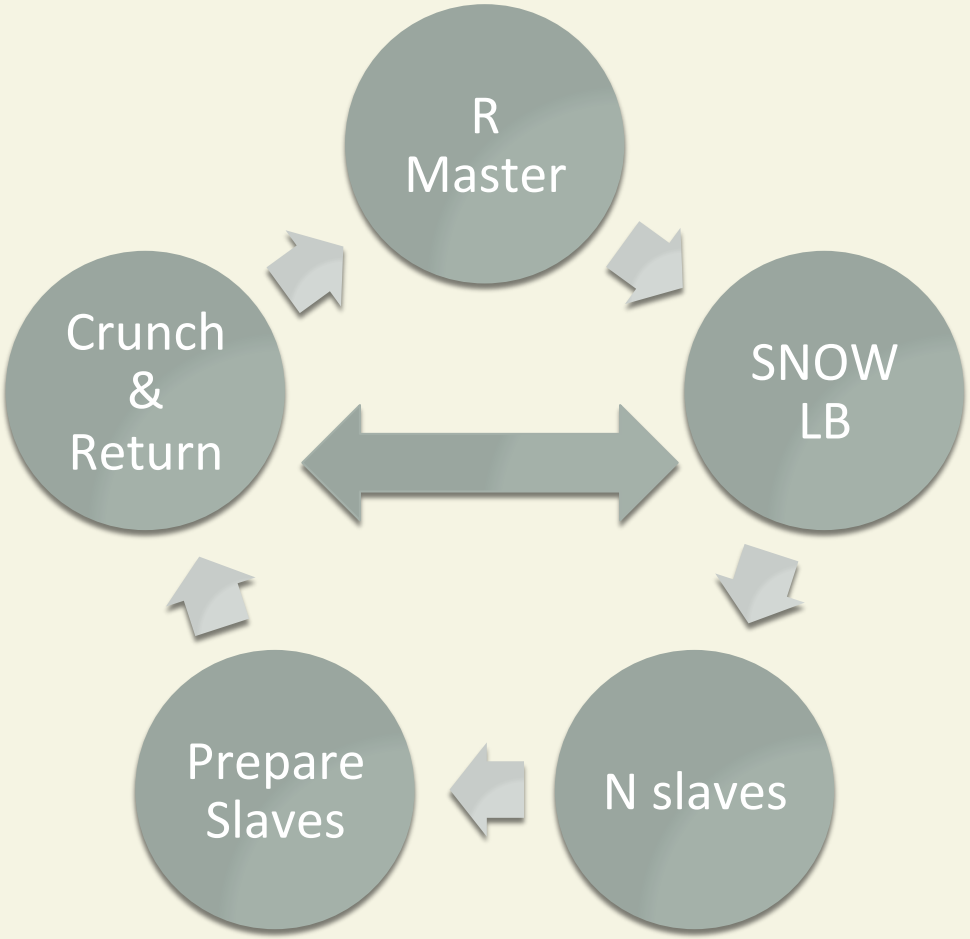
R CAN DO BOTH

- SNOW
  - Socket or openMPI + passwordless SSH
- Rgpu (and some others)

LINUX

- openMPI + passwordless SSH for multiple boxes, or make a socket cluster for 1 box.
- SLURM – Cluster resource manager
- Ganglia – Cluster manager frontend
- Webmin – to manage the cluster
- Many other packages too.

HOW IT WORKS



SOME CODE

```
getData <- function (getID) {

  cat('getting ',as.character(getID), " ")
  url = paste('https://graph.facebook.com/', getID, '/.....
  urlpage = paste('https://graph.facebook.com/', getID, '/?.....
  a = try(fromJSON(getURL(url)))
  b = try(fromJSON(getURL(urlpage)))


  #do some parsing

  # return some data

  return(list(data=foo))

}
```

SOME HELPER FUNCTIONS

```
prepareSlave = function(){
  library(RCurl)
  library(rjson)
}

getDataTry = function(getID){
  res = try(getData(getID))

  if(class(res)!="try-error"){
    return(res)
  }else{
    return("try-error")
  }
}
```

THE PARALLELIZED CODE

```
library(snow)

IDs = read.csv('MyFacebookPages.csv')

#   make sure the data retrieve and parse works as intended
#   res = lapply(X=IDs[sample(1:nrow(IDs), 50),"id"], FUN=getDataTry)

#   setup the cluster
cl = makeCluster(8, type="SOCK")
clusterExport(cl, c("getData", "getDataTry", "prepareSlave"))
clusterCall(cl, prepareSlave)

#   full scale data retrieval and parsing
res = clusterApplyLB(cl=cl, x=as.list(IDs[,"id"]), fun=getDataTry)
stopCluster(cl=cl)
```

DEBUGGING

- Lapply is your friend…
- Most often errors come from not having the correct slave environment.
- Wrap your eval code in a try catch function (and send back the error)
- Flag for debug or cluster compute to make it easier in a production environment.

OPTIMIZATION

- Profile your code (Rprof)
- Watch system resources (memory)
- Be smart about what data you export to the slaves.
  - If smaller dataset then export upfront
  - If a large dataset then save memory by exporting smaller chunks

RGPU

- Prerequisites:
  - NVIDIA GPU card (http://developer.nvidia.com/cuda-downloads)
  - Installed and working NVIDIA toolkit
  - Installed and working Dev drivers
  - Installed and working CUDA SDK
  - Installed and working cuBLAS
  - Patience and perseverence
- RGPU not on CRAN - https://trac.nbic.nl/rgpu/
- Linux (not on a VM)

RGPU NOTE

- GPU is parallel, but better for a single operation type problem.
- For example: RNG, or sum, or solving Big Matrix problems.
- New “parallel” functions have to be C coded in CUDA first.

RGPU EXAMPLE

```
library(rgpu)

rgpudevices()

meangpu(rnorm(1000000))
```

HOW BIG COULD IT GET?

- Went up to 80 cores, worked great…
- The main thing is debugging in parallel, really hard to do
- Might want to customize snow for error reporting, try catch mechanisms
- Code profiling shows that parallel overhead isn't the big part of the equation, but it's significant enough..
  - (use the R code profiler to make sure it makes sense)

RELATED PACKAGES

- Multicore
- New in R 2.14 parallel
- cloudRmpi – utilizes the Amazon EC2 service
- Rreval – remote evaluation of r functions over ssh
- Bigmemory – possible share objects in memory without R's memory issues
- HadoopStreaming

THANKS!