Parallel R, Revisited

Norm Matloff
University of California at Davis

UseR! 2012
Vanderbilt University, June, 2012

URL for these slides:
`http://heather.cs.ucdavis.edu/user2012.pdf` (repeated on final slide)

Norm Matloff
University of California at Davis

The Need

The Need

- Data sets getting larger and larger.

The Need

- Data sets getting larger and larger.
- Algorithms becoming more and more complex,

The Need

- Data sets getting larger and larger.
- Algorithms becoming more and more complex,e.g. clustering, machine learning, high-dim methods.

The Need

- Data sets getting larger and larger.
- Algorithms becoming more and more complex,e.g. clustering, machine learning, high-dim methods.
- “Big data” the latest buzzword in the tech world.

The Need

- Data sets getting larger and larger.
- Algorithms becoming more and more complex,e.g. clustering, machine learning, high-dim methods.
- "Big data" the latest buzzword in the tech world.

New York Times, Feb. 11, 2012

New York Times, Feb. 11, 2012

NEWS ANALYSIS

The Age of Big Data

By STEVE LOHR
Published: February 11, 2012 | 82 Comments

GOOD with numbers? Fascinated by data? The sound you hear is opportunity knocking.

Enlarge This Image

Mo Zhou was snapped up by I.B.M. last summer, as a freshly minted Yale M.B.A., to join the technology company's fast-growing ranks of data consultants. They help businesses make sense of an explosion of data — Web traffic and social network comments, as well as software and sensors that monitor

SAS Web page

SAS Web page

NEWS / sascom MAGAZINE

Newsroom

- Press Releases
- Media Coverage
- Analyst Viewpoints
- About SAS
- Awards
- News: SAS Companies

Big data: big challenges, big opportunities

An expert panel discusses how organizations can capitalize on big data to generate new ideas, build new markets and expand existing ones

Participating in a panel discussion at the recent Ideas Economy conference put on by The Economist, SAS Chief Executive Officer Jim Goodnight and other high-tech execs discussed so-called "big data" and the challenges and opportunities companies face in dealing with the ever-growing data deluge.

Oracle Web page

Oracle Web page

The term big data draws a lot of attention, but behind the hype there's a simple story. For decades, companies have been making business decisions based on transactional data stored in relational databases. Beyond that critical data, however, is a potential treasure trove of less structured data: weblogs, social media, email, sensors, and photographs that can be mined for useful information.

Norm Matloff
University of California at Davis

Oracle, cont'd.

Norm Matloff
University of California at Davis

Oracle, cont'd.

But Oracle rocks! :-)

Oracle, cont'd.

But Oracle rocks! :-)

Oracle R Enterprise

Integrates the Open-Source Statistical Environment R with Oracle Database 11*g*
Oracle R Enterprise allows analysts and statisticians to run existing R applications and use the R client directly a
Oracle Database 11*g*—vastly increasing scalability, performance and security. The combination of Oracle Databa
delivers an enterprise-ready, deeply integrated environment for advanced analytics. Users can also use analytic
where they can analyze data and develop R scripts for deployment while results stay managed inside Oracle Da

Where Is Parallel R Today?

Where Is Parallel R Today?

- Tons of packages:

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R

Norm Matloff
University of California at Davis

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R
- Base R now incorporates **snow** (cluster, multicore)

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R
- Base R now incorporates **snow** (cluster, multicore) and **multicore** (multicore).

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R
- Base R now incorporates **snow** (cluster, multicore) and **multicore** (multicore).
- Mainly useful on "embarrassingly parallel" (EP) problems

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R
- Base R now incorporates **snow** (cluster, multicore) and **multicore** (multicore).
- Mainly useful on "embarrassingly parallel" (EP) problems—those dividable into subproblems that need little or no intercommunication.

Where Is Parallel R Today?

- Tons of packages: CRAN Task View: High-Performance and Parallel Computing with R
- Base R now incorporates **snow** (cluster, multicore) and **multicore** (multicore).
- Mainly useful on "embarrassingly parallel" (EP) problems—those dividable into subproblems that need little or no intercommunication.
- What about non-EP apps?

Challenges

Challenges

- Multiplatform desirable:

Norm Matloff
University of California at Davis

Challenges

- Multiplatform desirable:
  - multicore

Norm Matloff
University of California at Davis

Challenges

- Multiplatform desirable:
  - multicore
  - cluster

Challenges

- Multiplatform desirable:
  - multicore
  - cluster
  - GPU

Norm Matloff
University of California at Davis

Challenges

- Multiplatform desirable:
    - multicore
    - cluster
    - GPU (and other coming accelerators?)

Challenges

- Multiplatform desirable:
    - multicore
    - cluster
    - GPU (and other coming accelerators?)
    - **foreach()** multiplatform, but for R code, not C, and does not work on GPU
- R not threaded

Challenges

- Multiplatform desirable:
  - multicore
  - cluster
  - GPU (and other coming accelerators?)
  - **foreach()** multiplatform, but for R code, not C, and does not work on GPU
- R not threaded
  - Very hard, no plans to do it to my knowledge (?).

Challenges

- Multiplatform desirable:
  - multicore
  - cluster
  - GPU (and other coming accelerators?)
  - **foreach()** multiplatform, but for R code, not C, and does not work on GPU
- R not threaded
  - Very hard, no plans to do it to my knowledge (?).
  - **Rdsm**, **bigmemory** threads-like, but not good for parallel computation.

Challenges

- Multiplatform desirable:
    - multicore
    - cluster
    - GPU (and other coming accelerators?)
    - **foreach()** multiplatform, but for R code, not C, and does not work on GPU
- R not threaded
    - Very hard, no plans to do it to my knowledge (?).
    - **Rdsm**, **bigmemory** threads-like, but not good for parallel computation.
- Copy-on-write:

Challenges

- Multiplatform desirable:
  - multicore
  - cluster
  - GPU (and other coming accelerators?)
  - **foreach()** multiplatform, but for R code, not C, and does not work on GPU
- R not threaded
  - Very hard, no plans to do it to my knowledge (?).
  - **Rdsm**, **bigmemory** threads-like, but not good for parallel computation.
- Copy-on-write: Writing to one vector element sometimes causes copying entire vector.

Norm Matloff
University of California at Davis

Challenges, cont'd.

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?

Norm Matloff
University of California at Davis

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
  - Intel Knight's Ferry accelerator:

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
  - Intel Knight's Ferry accelerator: out "next year"—every year

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"—every year
    - CUDA (extension of C) currently GPU dominant software;

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
  - Intel Knight's Ferry accelerator: out "next year"—every year
  - CUDA (extension of C) currently GPU dominant software; same in future?

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
  - Intel Knight's Ferry accelerator: out "next year"—every year
  - CUDA (extension of C) currently GPU dominant software; same in future?
  - OpenCL (for GPU and multicore) growth currently stalled?

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"—every year
    - CUDA (extension of C) currently GPU dominant software; same in future?
    - OpenCL (for GPU and multicore) growth currently stalled?
    - OpenACC;

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"—every year
    - CUDA (extension of C) currently GPU dominant software; same in future?
    - OpenCL (for GPU and multicore) growth currently stalled?
    - OpenACC; for GPUs;

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"—every year
    - CUDA (extension of C) currently GPU dominant software; same in future?
    - OpenCL (for GPU and multicore) growth currently stalled?
    - OpenACC; for GPUs; might become more popular, due to announced connection with OpenMP

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
    - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
    - Intel Knight's Ferry accelerator: out "next year"—every year
    - CUDA (extension of C) currently GPU dominant software; same in future?
    - OpenCL (for GPU and multicore) growth currently stalled?
    - OpenACC; for GPUs; might become more popular, due to announced connection with OpenMP
    - uncertainty abounds

Challenges, cont'd.

"When you come to a fork in the road, take it"—famous baseball player and malapropist Yogi Berra

- Parallel technology in a state of flux:
  - NVIDIA chips currently dominant in the general-purpose GPU (GPGPU) world;true in future?
  - Intel Knight's Ferry accelerator: out "next year"—every year
  - CUDA (extension of C) currently GPU dominant software; same in future?
  - OpenCL (for GPU and multicore) growth currently stalled?
  - OpenACC; for GPUs; might become more popular, due to announced connection with OpenMP
  - uncertainty abounds—so which way should R go?

Norm Matloff
University of California at Davis

Outline of This Talk

Outline of This Talk

- An algorithmic approach:

Outline of This Talk

- An algorithmic approach:
  - Software alchemy:

Outline of This Talk

- An algorithmic approach:
    - Software alchemy: Change non-embarrassingly parallel to EP.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific:

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
  - Assuming parallel R will mainly consist of C/C++ interface.

Outline of This Talk

- An algorithmic approach:
    - Software alchemy: Change non-embarrassingly parallel to EP.
    - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
    - Assuming parallel R will mainly consist of C/C++ interface.
    - But need some multiplatform capability.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
  - Assuming parallel R will mainly consist of C/C++ interface.
  - But need some multiplatform capability.
  - **Rth**: R interface to Thrust.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
  - Assuming parallel R will mainly consist of C/C++ interface.
  - But need some multiplatform capability.
  - **Rth**: R interface to Thrust.
  - Thrust is C++ package for high-level operations, e.g. sort, search, prefix scan.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
  - Assuming parallel R will mainly consist of C/C++ interface.
  - But need some multiplatform capability.
  - **Rth**: R interface to Thrust.
  - Thrust is C++ package for high-level operations, e.g. sort, search, prefix scan.
  - Thrust builds to multiple backends, including GPU and multicore.

Outline of This Talk

- An algorithmic approach:
  - Software alchemy: Change non-embarrassingly parallel to EP.
  - Statistics-specific: Assumes i.i.d. data.
- My new package: **Rth**:
  - Assuming parallel R will mainly consist of C/C++ interface.
  - But need some multiplatform capability.
  - **Rth**: R interface to Thrust.
  - Thrust is C++ package for high-level operations, e.g. sort, search, prefix scan.
  - Thrust builds to multiple backends, including GPU and multicore.
  - So, **Rth** is a tool for easily parallelizing many R operations, usable on both GPU and multicore.

Software Alchemy: Non-EP to EP

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.

Norm Matloff
University of California at Davis

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.
- Old, old idea in parallel processing: Break data into chunks, work on each chunk, then combine results.

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.
- Old, old idea in parallel processing: Break data into chunks, work on each chunk, then combine results.
- But this requires EP to be worthwhile.

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.
- Old, old idea in parallel processing: Break data into chunks, work on each chunk, then combine results.
- But this requires EP to be worthwhile.
- New approach: Exploit the statistical properties.

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.
- Old, old idea in parallel processing: Break data into chunks, work on each chunk, then combine results.
- But this requires EP to be worthwhile.
- New approach: Exploit the statistical properties.
- Key point:

Software Alchemy: Non-EP to EP

- Call it **NEP2EP**.
- Old, old idea in parallel processing: Break data into chunks, work on each chunk, then combine results.
- But this requires EP to be worthwhile.
- New approach: Exploit the statistical properties.
- Key point: Calculate a **statistically equivalent** quantity that lends itself to EP computation.

Advantages of NEP2EP

Advantages of NEP2EP

- Works on R level; no need to resort to C/C++.

Advantages of NEP2EP

- Works on R level; no need to resort to C/C++.
- Fine on either multicore or cluster.

Advantages of NEP2EP

- Works on R level; no need to resort to C/C++.
- Fine on either multicore or cluster.
- Simple to use—e.g. from **snow**.

Advantages of NEP2EP

- Works on R level; no need to resort to C/C++.
- Fine on either multicore or cluster.
- Simple to use—e.g. from **snow**.
- Has a surprising benefit even on unicore.

Advantages of NEP2EP

- Works on R level; no need to resort to C/C++.
- Fine on either multicore or cluster.
- Simple to use—e.g. from **snow**.
- Has a surprising benefit even on unicore.
- Bonus: Automatic generation of standard errors (that you didn't have before).

Norm Matloff
University of California at Davis

How NEP2EP Works

How NEP2EP Works

- Suppose we wish to calculate an estimator $\widehat{\theta}$, say regression coefficients.

How NEP2EP Works

- Suppose we wish to calculate an estimator $\widehat{\theta}$, say regression coefficients.
- Have n data points, r processes (e.g. $r = 2$ for dual core on a single machine).

How NEP2EP Works

- Suppose we wish to calculate an estimator $\widehat{\theta}$, say regression coefficients.
- Have n data points, r processes (e.g. $r = 2$ for dual core on a single machine).
- Break into r chunks of n/r data points each.

How NEP2EP Works

- Suppose we wish to calculate an estimator $\widehat{\theta}$, say regression coefficients.
- Have n data points, r processes (e.g. $r = 2$ for dual core on a single machine).
- Break into r chunks of n/r data points each.
- For $i = 1,...,r$ calculate $\widehat{\theta}$ on chunk i, yielding $\tilde{\theta}_i$.

Norm Matloff
University of California at Davis

How NEP2EP Works

- Suppose we wish to calculate an estimator $\widehat{\theta}$, say regression coefficients.
- Have n data points, r processes (e.g. r = 2 for dual core on a single machine).
- Break into r chunks of n/r data points each.
- For i = 1,...,r calculate $\widehat{\theta}$ on chunk i, yielding $\tilde{\theta}_i$.
- Average all those chunked values:

$$\overline{\theta} = \frac{1}{r} \sum_{i=1}^{r} \tilde{\theta}_i$$

R Code (Snow)

R Code (Snow)

```
wrapper <- function(cls,z,probpars,sfrndmz=F) {
   if (!is.matrix(z)) z <- matrix(z,ncol=1)
   n <- probpars$n
   if (rndmz) z <- z[sample(1:n,n,replace=F),]
   nnodes <- length(cls) obslist <- list()
   chunksize <- n / nnodes
   for (i in 1:nnodes) {
      firstobs <- 1 + (i-1) * chunksize
      lastobs <- firstobs + chunksize - 1
      if (lastobs == n) lastobs <- n
      obslist[[i]] <- z[firstobs:lastobs,]
   }
   thts <- clusterApply(cls,obslist,sf)
   tht <- do.call("+",thts)
   tht / nnodes
}
```

What Does That Give You?

What Does That Give You?

- The result, $\overline{\theta}$ can be proven to have the **same asymp. statistical accuracy** as the original $\widehat{\theta}$.

Norm Matloff
University of California at Davis

What Does That Give You?

- The result, $\overline{\theta}$ can be proven to have the **same asymp. statistical accuracy** as the original $\widehat{\theta}$.
- But the computation of $\overline{\theta}$ is EP even if $\widehat{\theta}$ is non-EP.

What Does That Give You?

- The result, $\overline{\theta}$ can be proven to have the **same asymp. statistical accuracy** as the original $\widehat{\theta}$.
- But the computation of $\overline{\theta}$ is EP even if $\widehat{\theta}$ is non-EP.
- Alchemy! Non-EP $\rightarrow$ EP.

Rough Theoretical Speedup Analysis

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. $r = 2$ for dual core).

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r = 2 for dual core).
- Say basic alg. takes $O(n^c)$ time.

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r = 2 for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$,

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r = 2 for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$, speedup of r^c.

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r = 2 for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$, speedup of r^c.
- For algs. having $c > 1$, speedup is *superlinear* (par. proc. term).

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r = 2 for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$, speedup of r^c.
- For algs. having $c > 1$, speedup is *superlinear* (par. proc. term).Not the usual small stuff like cache effects!

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r $= 2$ for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$, speedup of r^c.
- For algs. having $c > 1$, speedup is *superlinear* (par. proc. term).Not the usual small stuff like cache effects!
- Uniprocessing case: Run time is $rO(n^c/r^c)$, i.e. $O(n^c/r^{c-1})$.

Rough Theoretical Speedup Analysis

- Say n obs., r processes (e.g. r $= 2$ for dual core).
- Say basic alg. takes $O(n^c)$ time.
- So, NEP2EP speedup is (roughly) $O(n^c/r^c)$, speedup of r^c.
- For algs. having $c > 1$, speedup is *superlinear* (par. proc. term).Not the usual small stuff like cache effects!
- Uniprocessing case: Run time is $rO(n^c/r^c)$, i.e. $O(n^c/r^{c-1})$. So, if $c > 1$ NEP2EP gives a speedup with no parallelism!

NEP2EP Timing Experiments

NEP2EP Timing Experiments

- NEP2EP in Snow

NEP2EP Timing Experiments

- NEP2EP in Snow
- multicore machine, 32 threads (2 CPUs x 8 cores x hyperthreading of 2)

NEP2EP Timing Experiments

- NEP2EP in Snow
- multicore machine, 32 threads (2 CPUs x 8 cores x hyperthreading of 2)
- num. cores = 2,4,8,16,24,32;

NEP2EP Timing Experiments

- NEP2EP in Snow
- multicore machine, 32 threads (2 CPUs x 8 cores x hyperthreading of 2)
- num. cores = 2,4,8,16,24,32; sometimes better beyond 32, probably due to cache/VM effects

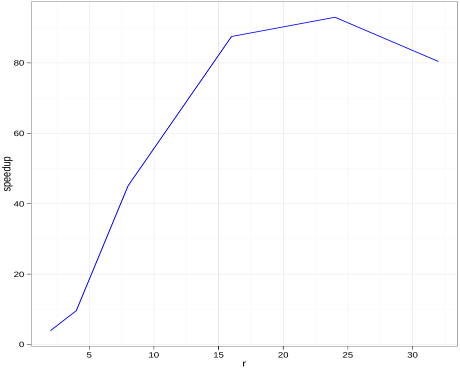
NEP2EP Timing Experiments

- NEP2EP in Snow
- multicore machine, 32 threads (2 CPUs x 8 cores x hyperthreading of 2)
- num. cores = 2,4,8,16,24,32; sometimes better beyond 32, probably due to cache/VM effects
- procedures tried:
    - Kendall's τ
    - quantile regression
    - nonparametric hazard function est.
    - log-concave density est.
    - linear regression (random x)

Kendall's τ

Kendall's τ

n = 10000

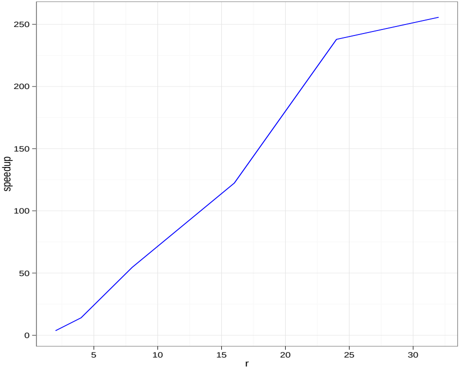


3.92X speedup at 2 threads
93.97X speedup at 24 threads

Kendall's τ, cont'd.

Kendall's τ, cont'd.

n = 25000



3.36X speedup at 2 threads 255.67X speedup at 32 threads

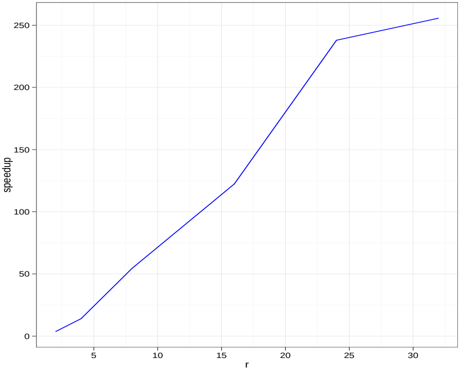
Norm Matloff
University of California at Davis

Quantile Regression

Norm Matloff
University of California at Davis

Quantile Regression

n = 10000, p = 10



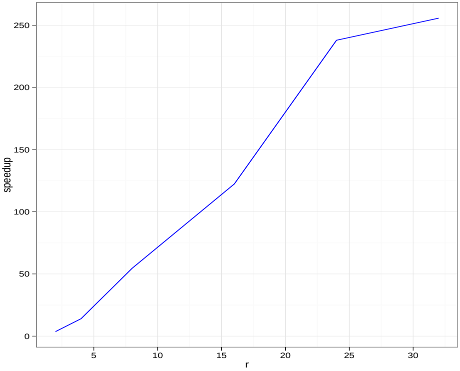
0.86X speedup at 2 threads
1.16X speedup at 8 threads

Norm Matloff
University of California at Davis

Quantile Regression

Quantile Regression

n = 10000, p = 25

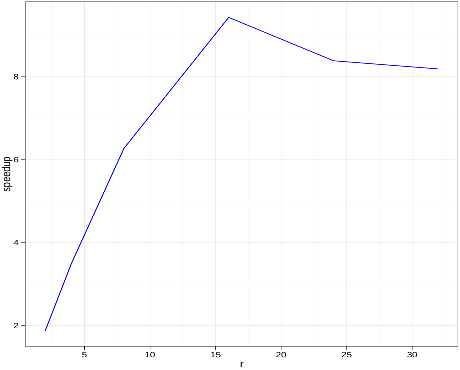


3.36X speedup at 2 threads 255.67X speedup at 32 threads

Hazard Function Estimation

Hazard Function Estimation

n = 25000, p = 0.2 (proportion missing); estimate quantiles 0.2, 0.4, 0.6, 0.8

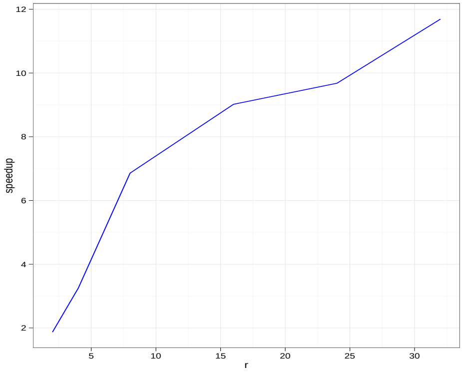


1.87X speedup at 2 threads
9.43X speedup at 16 threads

Hazard Function Estimation

Hazard Function Estimation

n = 50000, p = 0.02 (proportion missing)

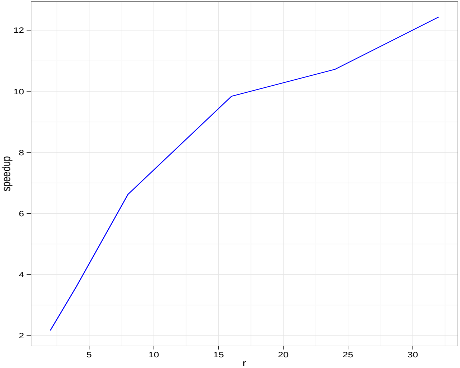


1.87X speedup at 2 threads
11.69X speedup at 32 threads

Log Concave Density Estimation

Log Concave Density Estimation

n = 200000



2.17X speedup at 2 threads
12.43X speedup at 32 threads

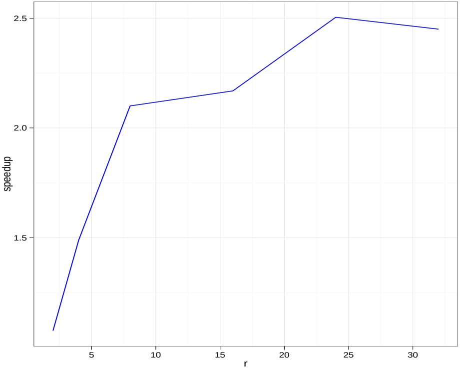
Linear Regression

Linear Regression

n = 50000, p = 50; should expect less here, $O(n, p^3)$

Linear Regression

n = 50000, p = 50; should expect less here, $O(n, p^3)$



0.90X speedup at 2 threads 1.97X speedup at 32 threads

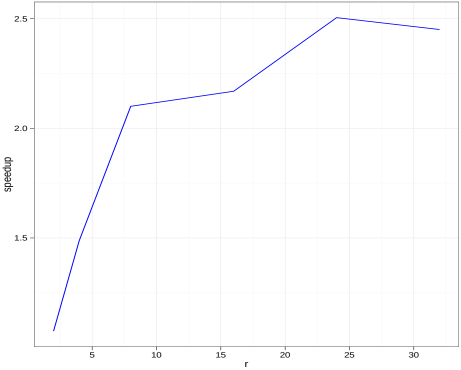
Norm Matloff
University of California at Davis

Linear Regression

Norm Matloff
University of California at Davis

Linear Regression

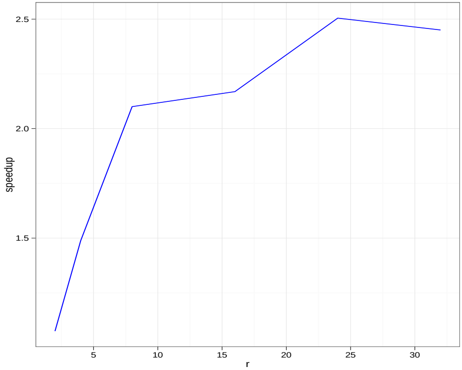
n = 50000, p = 100



1.08X speedup at 2 threads
2.50X speedup at 24 threads

Linear Regression

n = 50000, p = 100



1.08X speedup at 2 threads
2.50X speedup at 24 threads

Put in context: Seligman (2010) found GPU provides speedup only if $r > 1000$.

Norm Matloff
University of California at Davis

What About the Large-Sample Nature?

What About the Large-Sample Nature?

- One can prove that NEP2EP works asymptotically, i.e. gives the same statistical accuracy as the original estimatator.

What About the Large-Sample Nature?

- One can prove that NEP2EP works <u>asymptotically</u>, i.e. gives the same statistical accuracy as the original estimatator. Is that large-n requirement an issue?

What About the Large-Sample Nature?

- One can prove that NEP2EP works asymptotically, i.e. gives the same statistical accuracy as the original estimatator. Is that large-n requirement an issue?
- No, not an issue:

What About the Large-Sample Nature?

- One can prove that NEP2EP works asymptotically, i.e. gives the same statistical accuracy as the original estimatator. Is that large-n requirement an issue?
- No, not an issue: Since we're talking about settings where parallel computing is needed, we're working with large samples by definition

Norm Matloff
University of California at Davis

What About the Large-Sample Nature?

- One can prove that NEP2EP works asymptotically, i.e. gives the same statistical accuracy as the original estimatator. Is that large-n requirement an issue?
- No, not an issue: Since we're talking about settings where parallel computing is needed, we're working with large samples by definition—the large n is the reason we need parallel computing!

What About the Large-Sample Nature?

- One can prove that NEP2EP works asymptotically, i.e. gives the same statistical accuracy as the original estimatator. Is that large-n requirement an issue?
- No, not an issue: Since we're talking about settings where parallel computing is needed, we're working with large samples by definition—the large n is the reason we need parallel computing!
- NEP2EP gives essentially the same values as the original.

Norm Matloff
University of California at Davis

Accuracy

Accuracy

Absolute differences, r = 16:

Norm Matloff
University of California at Davis

Accuracy

Absolute differences, r = 16:

app.	prob. size	rel. diff.
Kendall	n = 1000	0.005849463
quant. reg.	n = 10000, p = 10	0.001274819
haz. ftn.	n = 25000, p = 0.2	0.007422595
log conc. dens.	n = 25000	0.0003593208
lin. reg.	n = 50000, p = 100	0.0001207394

Norm Matloff
University of California at Davis

Rth

Rth

Motivations:

- Parallelizing R will need to rely in part on C/C++ code.

Rth

Motivations:

- Parallelizing R will need to rely in part on C/C++ code.
- Nice to have the same parallel code work on multicore and GPU systems.

Norm Matloff
University of California at Davis

Rth

Motivations:

- Parallelizing R will need to rely in part on C/C++ code.
- Nice to have the same parallel code work on multicore and GPU systems. PGP—Pretty Good Parallelism.

Rth

Motivations:

- Parallelizing R will need to rely in part on C/C++ code.
- Nice to have the same parallel code work on multicore and GPU systems. PGP—Pretty Good Parallelism.
- Nice to have code for high-level operations available (sort, search, prefix scan, etc.).

Rth

Motivations:

- Parallelizing R will need to rely in part on C/C++ code.
- Nice to have the same parallel code work on multicore and GPU systems. PGP—Pretty Good Parallelism.
- Nice to have code for high-level operations available (sort, search, prefix scan, etc.).
- Hopefully make it (somewhat) easy for users to write their own parallel code.

Some Existing Possibilities

Some Existing Possibilities

These work on both multicore and GPUs:

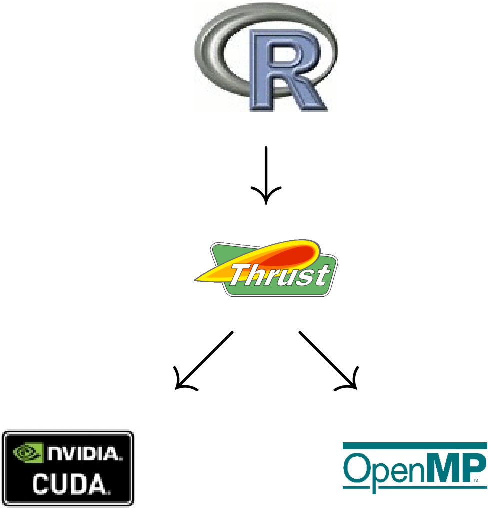
- OpenCL: Extension of C.
- Magma: Matrix routines.
- OpenACC: Like OpenMP for GPUs.

But OpenCL and OpenACC do not provide high-level ops, and Magma is narrow.

Norm Matloff
University of California at Davis

Rth

Rth



Norm Matloff
University of California at Davis

Goals

Goals

- Provide parallel Thrust code called from R,

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.
- Facilitate sophisticated user writing own parallel code.

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.
- Facilitate sophisticated user writing own parallel code.
- Currently just at very early stage of project.

What is Thrust?

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.
- Facilitate sophisticated user writing own parallel code.
- Currently just at very early stage of project.

What is Thrust?

- C++ package, modeled on STL.

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.
- Facilitate sophisticated user writing own parallel code.
- Currently just at very early stage of project.

What is Thrust?

- C++ package, modeled on STL.
- Can compile to either GPU or multicore backend.

Goals

- Provide parallel Thrust code called from R,
- Thrust transparent to the ordinary user.
- Parallelize a number of R operations in Thrust.
- Facilitate sophisticated user writing own parallel code.
- Currently just at very early stage of project.

What is Thrust?

- C++ package, modeled on STL.
- Can compile to either GPU or multicore backend.
- Provides high-level operations, e.g. sort, search, prefix scan, foreach, reduction, etc.

Norm Matloff
University of California at Davis

Example: sorting

Example: sorting

R interface code:

```
rthsort <- function(x) {
   dyn.load("rthsort.so")
   n <- length(x)
   tmp <- .C("rthsort",as.double(x),
      as.integer(n),tmpres=double(n))
   return(tmp$tmpres)
}
```

Example: sorting

R interface code:

```
rthsort <- function(x) {
   dyn.load("rthsort.so")
   n <- length(x)
   tmp <- .C("rthsort",as.double(x),
      as.integer(n),tmpres=double(n))
   return(tmp$tmpres)
}
```

Sorting 10000000 numbers: R 4.78 sec, **Rth** 1.52sec.

Norm Matloff
University of
California at
Davis

sorting, cont'd.

sorting, cont'd.

Thrust code:

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.h>

void rthsort(double *x, int *nx, double *xout)
{  int n = *nx;
   // set up device vector and copy x to it
   thrust::device_vector<double> dx(x,x+n);
   // sort, then copy back to x
   thrust::sort(dx.begin(), dx.end());
   thrust::copy(dx.begin(), dx.end(),xout);
}
```

General Pattern

General Pattern

Sort example was straight wrapper. What about other cases?

Norm Matloff University of California at Davis

General Pattern

Sort example was straight wrapper. What about other cases?

- Put together the appropriate Thrust ops.

General Pattern

Sort example was straight wrapper. What about other cases?

- Put together the appropriate Thrust ops.
- For most Thrust ops, write app-specific function to be called.

Example: convolution

Example: convolution

Thrust code:

```
void rthconv(double *x, int *nx,
   double *y, int *ny, double *z)
{  int nxx = *nx, nyy = *ny, nzz = nxx + nyy −
   thrust::device_vector<double> dx(x,x+nxx);
   ...
   thrust::counting_iterator<int> seqb(0);
   thrust::counting_iterator<int> seqe = seqb -
   thrust::for_each(seqb,seqe,do1i(dx.begin(),
      dy.begin(),dz.begin(),nxx,nyy));
   thrust::copy(dz.begin(), dz.end(),z);
}
```

Key line:

```
thrust::for_each(seqb,seqe,do1i(dx.begin(),...
```

Norm Matloff
University of California at Davis

convolution, cont'd.

convolution, cont'd.

User supplies "foreach" function, in the form of a

```
struct do1i  {  // "do 1 i"
...
   __device__
   void operator()(const int i)
   {  int j;  // handle 1 i in i,j loop
      int rpi = rndperm[i];
      double xdi = xd[rpi];
      for (j = 0; j < ny; j++)
         zd[rpi+j] += xdi * yd[ny-j-1];
   }
};
```

convolution, cont'd.

User supplies "foreach" function, in the form of a

```
struct do1i  {  // "do 1 i"
...
   __device__
   void operator()(const int i)
   {  int j;  // handle 1 i in i,j loop
      int rpi = rndperm[i];
      double xdi = xd[rpi];
      for (j = 0; j < ny; j++)
         zd[rpi+j] += xdi * yd[ny-j-1];
   }
};
```

A callable struct.

Performance

Performance

Performance

- **Rth**'s **rthconv()** orders of magnitude faster than R's **convolve()**.

Norm Matloff
University of California at Davis

Performance

- **Rth**'s **rthconv()** orders of magnitude faster than R's **convolve()**.
- Not fair to R's **convolve()**;

Performance

- **Rth**'s **rthconv()** orders of magnitude faster than R's **convolve()**.
- Not fair to R's **convolve()**; latter written in C, but works via FFTs, slow.

Performance

- **Rth**'s **rthconv()** orders of magnitude faster than R's **convolve()**.
- Not fair to R's **convolve()**; latter written in C, but works via FFTs, slow.
- Also: R's **convolve()** runs out of space on problems than **rthconv()** can handle (multcore).

Kendall's Tau

Kendall's Tau

```
void rthkendall(float *xy,int *nxy,float *tau)
{  int i,n = *nxy, n2 = 2*n,totcount;
   thrust::counting_iterator<int> seqa(0);
   thrust::counting_iterator<int>
      seqb =  seqa + n - 1;
   doubvec dxy(xy,xy+n2);
   intvec tmp(n-1);
   thrust::transform(seqa,seqb,tmp.begin(),
      calcgti(dxy,n));
   totcount=thrust::reduce(tmp.begin(),tmp.end(
   *tau = totcount / (0.5*n*(n-1));
}
```

Key calls: **transform(), reduce()**; can combine using transform iterator

Example: submatrix ops, select()

Example: submatrix ops, select()

- Not implemented yet.

Example: submatrix ops, select()

- Not implemented yet.
- Easy version: Specific numerical indices.

Example: submatrix ops, select()

- Not implemented yet.
- Easy version: Specific numerical indices.
- More elaborate: Dynamic parse of user R expression, sent off to Thrust code.

Some Thrust Ops

Some Thrust Ops

- sort, search
- reduce, min/max
- permute (e.g. for matrix transpose)
- partition, prefix scan
- foreach, transform, copyif
- set ops
- more are being added

Summary

Summary

- "Software alchemy" parallelizes i.i.d. stat apps, any platform.

Summary

- "Software alchemy" parallelizes i.i.d. stat apps, any platform. Often get superlinear speedup.

Summary

- “Software alchemy” parallelizes i.i.d. stat apps, any platform. Often get superlinear speedup.
- **Rth** provides a way to easily parallelize many other opps.

Norm Matloff
University of California at Davis

Misc.

Misc.

URLs:

- these slides: http://heather.cs.ucdavis.edu/user2012.pdf
- my online book on parallel programming: http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf
- **Rth**: http://heather.cs.ucdavis.edu/~matloff/rth.html

thanks to:

- Prof. Hao Chen (use of large multcore machine)
- Prof. Bill Hsu (use of fast GPUs)
- the audience :-)