State-of-the-art in Parallel Computing with R

Markus Schmidberger
(schmidb@ibe.med.uni-muenchen.de)

Statistical Computing 2009 - 41st Workshop
28.06-01.07.2009, Schloss Reisensburg (Günzburg)
The Future is Parallel

Prof. Bill Dally, Nvidia, 01-2009
Thilo Kielmann, University of Amsterdam, 12-2008
ITRS Roadmap 2005 and 2007
International Technology Roadmap for Semiconductors

Clock Rate (GHz)

2001 2003 2005 2007 2009 2011 2013

2005 Roadmap
2007 Roadmap
2008

Intel single core
Intel multicore
GPU

[D. Patterson, USENIX 2008 keynote]
Fundamental Technology Change

• CPU's will get faster only marginally:
  – limits of integration density
  – energy consumption (proportional to clock rate)

• If you want a faster computer, you need to use multiple CPU's:
  – past: Clock rate has doubled every 18 months
  – future: Number of cores will double every 18 months

• All programs must be parallel to use this new hardware!
New Technologies

• Intel's 80-Core Teraflop Chip Prototype
• Cell microprocessor
• Graphics Processing Units (GPU's)
• Clusters
  – Cloud Computing
  – Grid Computing
New Paper
submitted in December – State-of-the-art at the end of 2008

State-of-the-art in Parallel Computing with R

Markus Schmidberger
Ludwig-Maximilians-Universität
München, Germany

Martin Morgan
Fred Hutchinson Cancer Research Center, WA, USA

Dirk Eddelbuettel
Debian Project,
Chicago, IL, USA

Hao Yu
University of Western Ontario, ON, Canada

Luke Tierney
University of Iowa, IA, USA

Ulrich Mansmann
Ludwig-Maximilians-Universität
München, Germany

Preprint: http://epub.ub.uni-muenchen.de/8991/

- State of development
- Technology
- Fault-Tolerance & Load balancing
- Usability
- Acceptance
- Performance
Parallel Program Design

• Convert serial programs into parallel programs
  • compiler or pre-processor
  • prefer manual to automatically parallelization
    • wrong results may be produced,
    • performance may actually degrade,
    • much less flexible than manual parallelization,
    • code is too complex for automatical parallelization, etc..

• Very manual process of identifying and implementing parallelism

• Analysing the serial Code
  • understand serial code
  • profilers and performance analysis tools exist
  • identify program's hotspots or bottlenecks.
• In the R language:
  • profile R code for memory use and evaluation time
  • ?Rprof
  • CRAN packages proftools and profr
Simple Parallelization

```r
l <- list(a=c(1:10), b=c(2:12), c=c(4:14))

Seriell:
for(i in l)
    print(mean(i))
lapply(l, mean)

Parallel:
library(snow)
cl <- makeCluster(3, type='SOCK')
clusterApply(cl, l, mean)
stopCluster(cl)
```

Simple Parallelization for Statisticians

- Bootstraping: time-consuming and simple to parallelize
- library(boot): generating bootstrap replicates
- Example: generalized linear model fit for data on the cost of constructing nuclear power plants. 999 bootstraps
- Serial: 9.2 sec <-> 3 nodes: 3.1 sec
Parallelization

- **Multiprocessors**
  - the use of two or more central processing units (CPUs) within a single computer system
  - Today: Two/Four-processors are becoming a standard for workstations

- **Multicomputers**
  - different parts of a program run simultaneously on two or more computers that are communicating with each other over a network
  - Computer, network, software
  - Cluster, Grid
Master-Slave Architecture

- Works on computer clusters, on multiprocessor machines and in grid computing
- You need underlying technology for communication
  - MPI: Message Passing Interface
  - PVM: Parallel Virtual Machine
  - socket, ssh
# Parallel R Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Websites</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computer Cluster</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rmpi</td>
<td>0.5-6</td>
<td><a href="http://cran.r-project.org/web/packages/Rmpi">http://cran.r-project.org/web/packages/Rmpi</a></td>
<td>MPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.stats.uwo.ca/faculty/yu/Rmpi">http://www.stats.uwo.ca/faculty/yu/Rmpi</a></td>
<td></td>
</tr>
<tr>
<td>rpvm</td>
<td>1.0.2</td>
<td><a href="http://cran.r-project.org/web/packages/rpvm">http://cran.r-project.org/web/packages/rpvm</a></td>
<td>PVM</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.biostat.umn.edu/~nali/SoftwareListing.html">http://www.biostat.umn.edu/~nali/SoftwareListing.html</a></td>
<td></td>
</tr>
<tr>
<td>nws</td>
<td>1.7.0.0</td>
<td><a href="http://cran.r-project.org/web/packages/nws">http://cran.r-project.org/web/packages/nws</a></td>
<td>NWS and socket</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://nws-r.sourceforge.net">http://nws-r.sourceforge.net</a></td>
<td></td>
</tr>
<tr>
<td>snow</td>
<td>0.3-3</td>
<td><a href="http://cran.r-project.org/web/packages/snow">http://cran.r-project.org/web/packages/snow</a></td>
<td>Rmpi, rpvm, nws, socket</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.cs.uiowa.edu/~luke/R/cluster">http://www.cs.uiowa.edu/~luke/R/cluster</a></td>
<td></td>
</tr>
<tr>
<td>snowFT</td>
<td>0.0-2</td>
<td><a href="http://cran.r-project.org/web/packages/snowFT">http://cran.r-project.org/web/packages/snowFT</a></td>
<td>rpvm, snow</td>
</tr>
<tr>
<td>snowfall</td>
<td>1.60</td>
<td><a href="http://cran.r-project.org/web/packages/snowfall">http://cran.r-project.org/web/packages/snowfall</a></td>
<td>snow</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.imbi.uni-freiburg.de/parallel">http://www.imbi.uni-freiburg.de/parallel</a></td>
<td></td>
</tr>
<tr>
<td>papply</td>
<td>0.1</td>
<td><a href="http://cran.r-project.org/web/packages/papply">http://cran.r-project.org/web/packages/papply</a></td>
<td>Rmpi</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://math.acadiau.ca/ACMMAC/software/papply.html">http://math.acadiau.ca/ACMMAC/software/papply.html</a></td>
<td></td>
</tr>
<tr>
<td>biopara</td>
<td>1.5</td>
<td><a href="http://cran.r-project.org/web/packages/biopara">http://cran.r-project.org/web/packages/biopara</a></td>
<td>socket</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://hedwig.mgh.harvard.edu/biostatistics/node/20">http://hedwig.mgh.harvard.edu/biostatistics/node/20</a></td>
<td></td>
</tr>
<tr>
<td>taskPR</td>
<td>0.31</td>
<td><a href="http://cran.r-project.org/web/packages/taskPR">http://cran.r-project.org/web/packages/taskPR</a></td>
<td>MPI (only LAM/MPI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://users.ece.gatech.edu/~gte810u/Parallel-R">http://users.ece.gatech.edu/~gte810u/Parallel-R</a></td>
<td></td>
</tr>
<tr>
<td><strong>Grid Computing</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GridR</td>
<td>0.8.4</td>
<td><a href="http://cran.r-project.org/web/packages/GridR">http://cran.r-project.org/web/packages/GridR</a></td>
<td>web service, ssh, Condor, Globus</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.stefan-rueping.de">http://www.stefan-rueping.de</a></td>
<td></td>
</tr>
<tr>
<td>multiR</td>
<td>-</td>
<td><a href="http://e-science.lancs.ac.uk.uk/multiR">http://e-science.lancs.ac.uk.uk/multiR</a></td>
<td>3 tier client/server architecture</td>
</tr>
<tr>
<td>Biocep-R</td>
<td>NA</td>
<td><a href="http://biocep-distrib.r-forge.r-project.org">http://biocep-distrib.r-forge.r-project.org</a></td>
<td>java 5</td>
</tr>
<tr>
<td><strong>Multi-core System</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pnmath(0)</td>
<td>0.2</td>
<td><a href="http://www.cs.uiowa.edu/~luke/R/experimental">http://www.cs.uiowa.edu/~luke/R/experimental</a></td>
<td>openMP, Pthreads</td>
</tr>
<tr>
<td>fork</td>
<td>1.2.1</td>
<td><a href="http://cran.r-project.org/web/packages/fork">http://cran.r-project.org/web/packages/fork</a></td>
<td>Unix: fork</td>
</tr>
<tr>
<td>R/Parallel</td>
<td>0.6-20</td>
<td><a href="http://www.rparallel.org">http://www.rparallel.org</a></td>
<td>C++, file</td>
</tr>
<tr>
<td>romp</td>
<td>0.1a</td>
<td><a href="http://code.google.com/p/romp">http://code.google.com/p/romp</a></td>
<td>openMP</td>
</tr>
<tr>
<td>multicore</td>
<td>0.1-3</td>
<td><a href="http://cran.r-project.org/web/packages/multicore">http://cran.r-project.org/web/packages/multicore</a></td>
<td>fork</td>
</tr>
</tbody>
</table>
Computer Cluster R Packages

- MPI
  - Rmpi
  - papply
  - snowFT

- PVM
  - rpvm
  - taskPR

- NWS
  - nws

- SOCKET
  - biopara
  - snow
  - snowfall

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXX</td>
<td>R Package</td>
</tr>
<tr>
<td>XXX</td>
<td>Technology</td>
</tr>
<tr>
<td>XXX</td>
<td>No longer maintained</td>
</tr>
</tbody>
</table>
### Performance evaluation of R packages for computer clusters

<table>
<thead>
<tr>
<th>Component</th>
<th>Rmpi</th>
<th>nws</th>
<th>snow</th>
<th>snowfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component 1</td>
<td>0.91</td>
<td>14.87</td>
<td>10.33</td>
<td>13.99</td>
</tr>
<tr>
<td>Component 2</td>
<td>1.83</td>
<td>3.49</td>
<td>2.00</td>
<td>2.07</td>
</tr>
<tr>
<td>Component 3</td>
<td>3.00</td>
<td>3.03</td>
<td>2.96</td>
<td>2.93</td>
</tr>
</tbody>
</table>

Component 1: Sending Data from the Master to all Slaves (matrix 500 x 500)
Component 2: Distributing a List of Data from the Master to the Slaves (list of matrices 500 x 500)
Component 3: Compute integral of a three-dimensional function (10,000 points)
## Performance evaluation of R packages for computer clusters

<table>
<thead>
<tr>
<th></th>
<th>Component 1</th>
<th>Component 2</th>
<th>Component 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rmpi</td>
<td>29.1</td>
<td>18.6</td>
<td>21.9</td>
</tr>
<tr>
<td>nws</td>
<td>97.3</td>
<td>34.8</td>
<td>21.2</td>
</tr>
<tr>
<td>snow</td>
<td>103.2</td>
<td>20.1</td>
<td>20.5</td>
</tr>
<tr>
<td>MPI</td>
<td>41.2</td>
<td>10.1</td>
<td>20.5</td>
</tr>
<tr>
<td>PVM</td>
<td>86.7</td>
<td>16.0</td>
<td>20.8</td>
</tr>
<tr>
<td>NWS</td>
<td>34.8</td>
<td>9.3</td>
<td>20.2</td>
</tr>
<tr>
<td>Socket</td>
<td>109.6</td>
<td>20.9</td>
<td>20.5</td>
</tr>
<tr>
<td>snowfall</td>
<td>43.0</td>
<td>9.9</td>
<td>20.6</td>
</tr>
<tr>
<td>NWS</td>
<td>88.0</td>
<td>16.3</td>
<td>20.9</td>
</tr>
<tr>
<td>Socket</td>
<td>37.1</td>
<td>9.9</td>
<td>20.3</td>
</tr>
</tbody>
</table>
Performance - Sudoku

- R package: sudoku_2.2
- Generates, plays, and solves Sudoku puzzles.
- Solve 10,000 Sudokus
- Distribute Sudokus equally to all nodes
  - The basic rules of Sudoku are used to fill in missings, then elimination is used to find the TRUE's. If that approach runs out of steam, a guess is made and the program recurses to find either a solution or an inconsistency.
Performance - Sudoku

Multicore-Machine

IBE Cluster

HLRB2

Speedup vs. Prozessors for different configurations.
State of the Art in Parallel Computing with R

- Multicomputer: **Rmpi** and **snow**
  - acceptable usability
  - wide spectrum of functionality
  - good performance

- Other packages
  - Usability <-> lower functionality

<table>
<thead>
<tr>
<th></th>
<th>Learnability</th>
<th>Efficiency</th>
<th>Memorability</th>
<th>Errors</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rmpi</td>
<td>+</td>
<td>--</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>rpvm</td>
<td>--</td>
<td>--</td>
<td>+</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>nws</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>snow</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>snowFT</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>snowfall</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>papply</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>biopara</td>
<td>--</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>taskPR</td>
<td>+</td>
<td>--</td>
<td>++</td>
<td>0</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Learnability</th>
<th>Efficiency</th>
<th>Memorability</th>
<th>Errors</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Computing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gridR</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R/parallel</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Multi-core System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pnumath(0)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>fork</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multicore</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>
State of the Art in Parallel Computing with R

• Multi-core: in development
  – external and architecture optimized libraries (PBLAS)
    • bottleneck in statistical computation?
  – Multicomputer packages: Rmpi and snow
    • every R instance requires its own main memory!

• Grid Computing: early-stage packages
Which package should I use?

- Depends on your available hardware:
  - Multicore machine: multicore
  - Cluster environment:
    - Snow(-fall) with the available communication mechanism (MPI mostly used)
    - NWS, if you have a lot of global variables
    - Rmpi, for excellent programmer and for high end optimization
  - Grid Computing: gridR, which statistical application is usefull for grid computing?
Tips for Parallel Computing

• Communication is much slower than computation.
  – functions produce large results, reduce results on the worker before returning.
  – additional function parameters can be huge.

bsapply and countPDict Example
R> params <- new("BSPrams", X = Hsapiens, 
                 FUN = countPDict)
R> library(hgu133plus2probe)
R> dict0 <- DNAStringSet(hgu133plus2probe$sequence)
R> pdict0 <- PDict(dict0)
R> bsapply(params, pdict = pdict0)
Tips for Parallel Computing

- Random Generators have to used with care; special-purpose packages `rsprng`, `rlecuyer` (and `snow`) are available.

```r
R> clusterCall(cl, runif, 3)
[[1]]
[1] 0.4351672 0.7394578 0.2008757
[[2]]
[1] 0.4351672 0.7394578 0.2008757
...
[[10]]
[1] 0.4351672 0.7394578 0.2008757
```

- `lexical scoping`: requires some care to avoid transmitting unnecessary data to workers

- Functions used in apply-like calls should be defined in the global environment, or in a package name space.
HELP

• CRAN Task View 'High Performance and Parallel Computing'
  – http://cran.r-project.org/web/views/HighPerformanceComputing.html

• R Mailinglist 'R SIG on High-Performance Computing'
  – https://stat.ethz.ch/mailman/listinfo/r-sig-hpc
Conclusion & Future

- Parallel Computing can help improving performance,
  - but first of all improve your serial code (profiling, vectorization, ...).
  - but be careful with communication costs.
- First Parallel Implementations are easy,
  - but there are a lot of stumbling blocks.
- Parallel Computing with R needs to be improved:
  - Integration of R code into multi-core environments
  - Cloud Computing with R
  - Computing power of graphic processing units
- Flexibility of R package system allows integration of many different technologies.
Acknowledgment

Parallel R
Martin Morgan
Dirk Eddelbuettel
Hao Yu
Luke Tierney
Anthony Rossini

LRZ
HPC Team
Ferdinand Jamitzky

AffyPara Package
Ulrich Mansmann
Esmeralda Vicedo
Klaus Rüschstroer
Robert Gentlemans Group

NEXT
Tutorial: Parallel Computing with R

Dipl.-Tech. Math. Markus Schmidberger
schmidb@ibe.med.uni-muenchen.de
http://ibe.med.uni-muenchen.de

Thanks for your attention