Using SPRINT in the Cloud

Simple Parallel R INTerface

http://www.r-sprint.org/

Terry Sloan
Software Development Group Manager, EPCC
t.sloan@epcc.ed.ac.uk
+44 131 650 5155
SPRINT: a Multi-disciplinary Collaboration

- Division of Pathway Medicine: statistical and biological expertise
  - Peter Ghazal (PI), Thorsten Forster and Muriel Mewissen

- EPCC: parallel computing and software engineering expertise
  - Terry Sloan (PI), Michal Piotrowski and Lawrence Mitchell

- Funding:
  - 2008: Scottish Funding Council edikt2 – prototype development
  - 2009 – 2011: Wellcome Trust Charity – Survey of R community, R package release
  - 2009 – 2011: UK Research Councils – porting to HECToR supercomputer
  - 2011: UK Software Sustainability Institute – Handling of very large datasets

- Publications:
Overview

• The Data Explosion and HPC
• SPRINT
• SPRINT in the Cloud
• Closing Remarks
Data Explosion

Analysing High Throughput, High Dimensional Data

- Microarray gene or exon expression
- Microarray genotyping
- High-throughput sequencing

~50 samples
~2,000 samples
~100ks to ~10Ms raw short reads

~5k to 200k Genes or exons
~260k to 2M SNP
~100ks to ~10Ms raw short reads

High-throughput sequencing
Microarray genotyping
Microarray gene or exon expression
Harnessing the Data Explosion

Requires High Performance Computing (HPC)

Additional computing power such as:
- Multi-core desktops
- Clusters
- GPU
- Cloud
- Supercomputers

How do you choose?
- Access & availability
- Cost
- Tools & applications
- Ease of use & expertise
- Performance
- Type of problem
Current Parallel R modules
Issues

• Difficult to program
• Require biostatistician to also be a parallel programmer!
  – Rmpi: wrapper around MPI
  – NWS and Sleigh: implement shared memory system

• Require substantial changes to existing scripts
• Can’t be used to solve some problems
  – Biopara: Execute R functions remotely via SSH
  – Papply: Parallel ‘apply’ command, runs the same command on every element in a list
  – SNOW: Allows a single expression on different data segments
Simple Parallel R INTerface
(www.r-sprint.org)

SPRINT - Aims

Overcome limitations on data size and analysis time by providing easy access to HPC for all R users
Overcome limitations on data size and analysis time by providing *easy access* to HPC for all R users
SPRINT – Easy Access

- user friendly
- survey of R users - biologists, biostatisticians, ...
- familiar R interface - minimal changes to user’s R script

```r
data(golub)
smallgd <- golub[1:100,]
classlabel <- golub.cl

resT <- mt.maxT(smallgd, classlabel, test="t", side="abs")
quit(save="no")

library("sprint")
data(golub)
smallgd <- golub[1:100,]
classlabel <- golub.cl

resT <- pmaxT(smallgd, classlabel, test="t", side="abs")
pterminate()
quit(save="no")
```
Overcome limitations on data size and analysis time by providing easy access to **HPC for all R users**
SPRINT – scalable & portable

- scalable & portable from multi-core desktop to HPC to the Cloud

<table>
<thead>
<tr>
<th>HPC</th>
<th>Multi-core desktops</th>
<th>Servers</th>
<th>Shared Memory Machines</th>
<th>Network of Workstations</th>
<th>GPU</th>
<th>Cloud</th>
<th>supercomputers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRINT Compatibility</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
SPRINT – Open to contributions

• Library of parallelised R functions
  • complex functions & popular functions

• Implemented in R, C & MPI

• Open to contributions – R-forge
SPRINT Architecture

Master Node

R Runtime

- Load SPRINT
- R script invokes parallel function
- SPRINT takes over the execution
- Results return to R
- Shutdown SPRINT
- Exit R

Init

Worker Node

R Runtime

- Load SPRINT

Init

C/MPI Runtime

- Broadcast function signature
- Compute function in parallel
- Broadcast shutdown
- Finalize

Wait for function command

Invoke SPRINT function

Optionally evaluate in R

Exit R
SPRINT Performance

**pcor**

<table>
<thead>
<tr>
<th>Input Matrix Size</th>
<th>Output Matrix Size</th>
<th>Serial Run Time</th>
<th>Parallel Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>11,000 x 320</td>
<td>26.85 MB</td>
<td>63.18 secs</td>
<td>4.76 secs</td>
</tr>
<tr>
<td>22,000 x 320</td>
<td>53.7 MB</td>
<td>Insufficient memory</td>
<td>13.87 secs</td>
</tr>
<tr>
<td>35,000 x 320</td>
<td>85.44 MB</td>
<td>9.12 GB</td>
<td>36.64 secs</td>
</tr>
<tr>
<td>45,000 x 320</td>
<td>109.86 MB</td>
<td>15.06 GB</td>
<td>Crashed</td>
</tr>
</tbody>
</table>

**pmaxT**

<table>
<thead>
<tr>
<th>Input Matrix Size</th>
<th># Permutations</th>
<th>Serial Run Time (estimated)</th>
<th>Parallel Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>36,612 x 76</td>
<td>500,000</td>
<td>6 hrs</td>
<td>73.18 secs</td>
</tr>
<tr>
<td>36,612 x 76</td>
<td>1,000,000</td>
<td>12 hrs</td>
<td>146.64 secs</td>
</tr>
<tr>
<td>36,612 x 76</td>
<td>2,000,000</td>
<td>23 hrs</td>
<td>290.22 secs</td>
</tr>
<tr>
<td>73,224 x 76</td>
<td>500,000</td>
<td>10 hrs</td>
<td>148.46 secs</td>
</tr>
<tr>
<td>73,224 x 76</td>
<td>1,000,000</td>
<td>20 hrs</td>
<td>294.61 secs</td>
</tr>
<tr>
<td>73,224 x 76</td>
<td>2,000,000</td>
<td>39 hrs</td>
<td>591.48 secs</td>
</tr>
</tbody>
</table>

**ppam**

<table>
<thead>
<tr>
<th>Input Data Size</th>
<th># Clusters</th>
<th>Serial Run Time Pam()</th>
<th>Parallel Run Time Ppam()</th>
</tr>
</thead>
<tbody>
<tr>
<td>2400</td>
<td>12</td>
<td>11.3 secs</td>
<td>1.1 secs</td>
</tr>
<tr>
<td>2400</td>
<td>24</td>
<td>52.5 secs</td>
<td>2.2 secs</td>
</tr>
<tr>
<td>4800</td>
<td>12</td>
<td>83.3 secs</td>
<td>4.4 secs</td>
</tr>
<tr>
<td>4800</td>
<td>24</td>
<td>434.7 secs</td>
<td>15.9 secs</td>
</tr>
<tr>
<td>10000</td>
<td>12</td>
<td>17 mins</td>
<td>22.3 secs</td>
</tr>
<tr>
<td>10000</td>
<td>24</td>
<td>99 mins</td>
<td>77.1 secs</td>
</tr>
<tr>
<td>22374</td>
<td>24</td>
<td>Insufficient memory</td>
<td>270.5 secs</td>
</tr>
</tbody>
</table>

Benchmark on HECToR - UK National Supercomputing Service on 256 cores.

Benchmark on a shared memory cluster with 8 dual-core 2.6GHz AMD Opteron processors with 2GB of RAM per core.
SPRINT Roadmap

• V0.3 April 2011
  – pcor, pmaxt, ppam

• V0.4 Jun 2011 ?
  – pboot, prandomforest, prankprod, papply

• V0.5 Oct 2011 ?
  – prma, ....
• To use HPC, you no longer have to *own* it
  – buy cycles across the Internet
  – your jobs will run according to some agreed level of service

• On-demand HPC is a reality (eg. Amazon EC2)
  – can be highly effective for users with bursty use patterns
  – however, data privacy and security are still overriding concerns for some
“The Amazon EC2 instances are connected using a virtual ethernet network with no guarantees on bandwidth or latency and therefore performs poorly when the process count increases”, S. Petrou, 2010
...not all Clouds have a silver lining

- Comparing SPRINT PAM implementation with a cloud-specific PAM implementation based on Map-Reduce

<table>
<thead>
<tr>
<th>MPI on Hector 8 cores</th>
<th>MapReduce 9 EC2 Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>53 sec</td>
<td>269 min</td>
</tr>
</tbody>
</table>


- Raw SPRINT
  - performance
  - generic scalability in a single framework

- Cloud-specific
  - fault tolerance
  - Simplified programming
Cloud costs

• Plethora of pricing models
  “EC2 VM: 4 regions x 4 purchase options x 10 instance types x 2 OS => over 300 prices”
  Ali Khajeh-Hosseini, University of St. Andrews, Summer 2010

• Variable performance
  “A job on the cloud can severely be impacted by the load in which the cloud is under. For SPRINT the time varied between 20 minutes and 1 hour.”
  Gary McGilvary, The University of Edinburgh, March 2011

  – Using SPRINT R scripts to investigate difference in cloud costs and performance for different parts of the globe

• Getting the data onto the cloud and storing it there
  – commercial clouds are businesses
Closing Remarks

• High Performance Computing can help with the life sciences data explosion

• SPRINT offers R users easy, scalable, portable access to HPC

• The cloud is one way for R users to get access to HPC but not every cloud has a silver lining