Running a SPRINT job on the HPC Wales platform

THE UNIVERSITY *of* EDINBURGH

There are other parallel R packages

Building Block Approaches

- Difficult to program
- Bespoke implementation
- Biostatistician needs to be a parallel programmer
  - Rmpi: wrapper around MPI
  - NWS and Sleigh: implement a shared memory system

Task Farm Approaches

- Require substantial changes to existing scripts
- Cannot be used to solve some problems
  - Biopara: Execute R functions remotely via SSH
  - Parallel 'apply' commands, runs the same command on every element in a list
  - SNOW: allows a single expression to be executed on different data segments

THE UNIVERSITY of EDINBURGH

R parallel package

- Parallel has been a standard part of R since R 2.14
  - Built on multicore and snow packages
- Apply
  - As analogues of lapply there are parLapply(cl, x, FUN, ...)
  - mclapply(X, FUN, ..., mc.cores)
- Underlying:
  - Random number generator
  - Load balancing
- Examples of use:
  - Bootstrapping, MCMC.

THE UNIVERSITY *of* EDINBURGH

So a Need for…

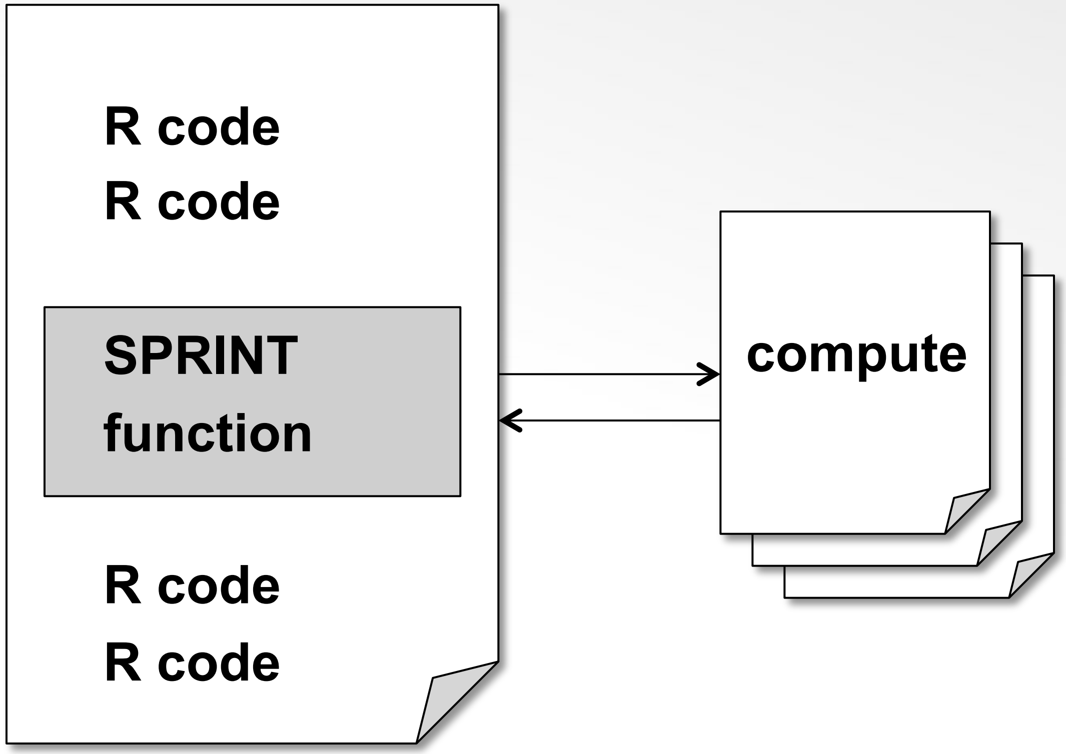
Tool for easy access to HPC:

- User friendly
- Hide the complexity of accessing and programming HPC
- Allow process and analyse large amounts of high throughput post genomic data using R for statistical computing.
- Allow repeated, regular use by biostatisticians
- Implement functions that are not embarrassingly parallel, e.g. partitioning around medoids, Pearson’s correlation.

THE UNIVERSITY *of* EDINBURGH

What does SPRINT do?

- The rest of the R workflow doesn't change



THE UNIVERSITY *of* EDINBURGH

Performance - The Goal of SPRINT

Overcome limitations on data size and analysis time by providing easy access to High Performance Computing for all R users

THE UNIVERSITY *of* EDINBURGH

SPRINT and Data Size

Overcome limitations on data size and analysis time by providing easy access to High Performance Computing for all R users

Input Matrix Size	Output Matrix Size	Serial Run Time	Parallel Run Time
11,000 x 320 **26.85 MB**	0.9 GB	**63.18 secs**	4.76 secs
22,000 x 320 53.7 MB	3.6 GB	Insufficient memory	13.87 secs
35,000 x 320 **85.44 MB**	9.12 GB	Crashed	36.64 secs
45,000 x 320 **109.86 MB**	15.08 GB	Crashed	42.18 secs

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

S. Petrou et al, dCSE NAG Report, www.r-sprint.org.

For example, Pearson's correlation, pcor()

•Enables processing of datasets where the output does not fit in physical memory

•uses R ff package: memory-efficient storage of large data on disk and fast access functions (also available from CRAN).

• ff objects can be created, stored, used and removed, almost like standard R RAM objects.

•ff objects are perfect for reading the same data from many R processes.

THE UNIVERSITY *of* EDINBURGH

SPRINT and Analysis Time

Overcome limitations on data size and *analysis time* by providing easy access to High Performance Computing for all R users

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

Benchmark on HECToR - UK National Supercomputing Service on 256 cores.
S. Petrou et al, HPDC 2010 & CCPE, 2011.

For example, permutation testing, pmaxT()

- Parallel implementation of mt.maxT() from multtest package (available from CRAN)

THE UNIVERSITY *of* EDINBURGH

SPRINT Data Size and Analysis Time

Overcome limitations on data size and analysis time by providing easy access to High Performance Computing for all R users

Input Data Size	# Clusters	Serial Run Time Pam()	Parallel Run Time Ppam()
2400	12	11.3 secs	1.1 secs
2400	24	52.5 secs	2.2 secs
4800	12	83.3 secs	4.4 secs
4800	24	434.7 secs	15.9 secs
10 000	12	17 mins	22.3 secs
10 000	24	99 mins	77.1 secs
22 374	24	Insufficient memory	270.5 secs

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

M. Piotrowski et al, BILIS 2011.

For example, clustering with partitioning around medoids, ppam()

- Parallel implementation of pam() from cluster package (available from CRAN)
- Optimisation of serial version through memory and data storage management
- Increased capacity by using external memory (i.e. ff objects)

THE UNIVERSITY *of* EDINBURGH

How you can use SPRINT

- Install SPRINT
- Modify R script
- Execute script in parallel
- Execute script on a supercomputer

Example

```
library("sprint”)

my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- cor( t(my.matrix) )


quit(save="no”)
```

THE UNIVERSITY *of* EDINBURGH

Example

```
library("sprint")

my.matrix <- matrix(rnorm(500000,9,1.7),
nrow=20000, ncol=25)

genecor <- pcor( t(my.matrix) )

pterminate()

quit(save="no")
```

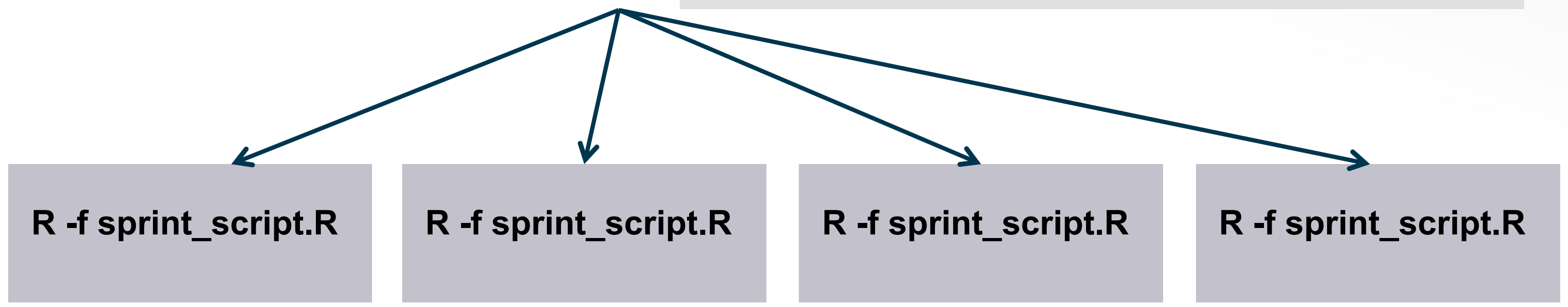
THE UNIVERSITY *of* EDINBURGH

Run using MPI

sprint_script.R

```
library("sprint”)

my.matrix <- matrix(rnorm(500000,9,1.7), nrow=20000, ncol=25)

genecor <- pcor( t(my.matrix) )

pterminate()

quit(save="no”)
```

```
$ mpiexec -n 4 R -f sprint_script.R
```



```
R -f sprint_script.R    R -f sprint_script.R    R -f sprint_script.R    R -f sprint_script.R
```

THE UNIVERSITY *of* EDINBURGH

Running on HPC Wales

2 files:

– R script calling SPRINT functions (sprint_script.R).

– Job submission script (sub.q).

–A request for time and processors on the supercomputer.

–The commands needed to execute your script

Sub.q

```
#!/bin/bash --login
# ! Edit number of processors to fit your job
#BSUB -n 8
# ! Redirect stdout to the file filename
#BSUB -o sprint_test.o.%J
# ! Redirect sterr to the file filename
#BSUB -e sprint_test.e.%J
# ! Edit the job name to identify separate job
#BSUB -J sprint_test
# ! Edit time to fit your job
#BSUB -W 0:10

module purge
module load SPRINT

mpirun -n 8 R --no-save --quiet -f sprint_test.R
```

Submit the job

Log in to hpc wales

```
$ ssh username@login.hpcwales.co.uk
```

Log in to one of the compute clusters

```
ssh ab-log-001
```

Create the 2 files described previously - sprint_script.R and sub.q

Submit the job

```
$ bsub < sub.q
```

The job will then join a queue and be run when resources become available.

THE UNIVERSITY *of* EDINBURGH

Check the results

To see if your job is waiting in the queue, running or finished, run:

```
$ qstat -u $USER
```

If this returns ‘No matching job found’ then your job is finished and the output of running the code will be in a {filename}. o{job_number} file.

```
more sprint_test.o{job_number}
```

Any error messages will be in a {filename}. e{job_number} file.

```
more sprint_test.e{job_number}
```