Parallel Computing with R: infrastructure (and sfCluster)

Jochen Knaus

Institute of Medical Biometry and Medical Informatics, University of Freiburg

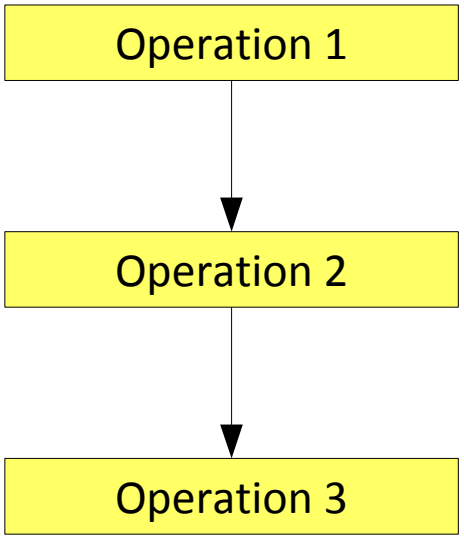
jo@imbi.uni-freiburg.de

January 19, 2011

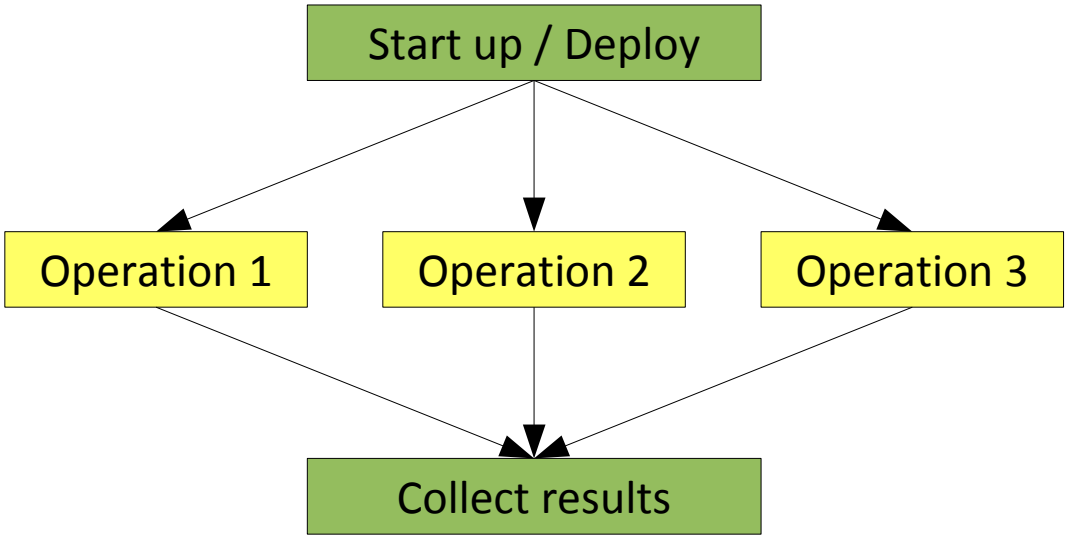
UNIVERSITÄTS KLINIKUM
FREIBURG

Basic idea

Sequentially execution

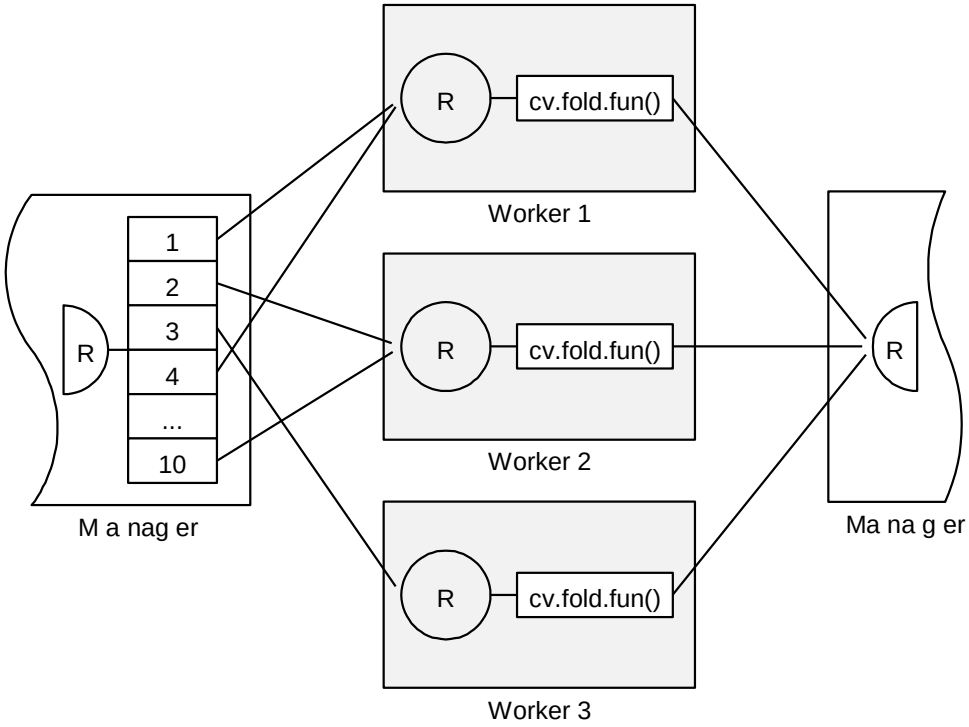


Parallel execution



Basic workflow for most R parallel packages

- Master/Worker setting. Automatic distribution using `apply()` calls:



- There are some alternative approaches (e.g. Map/Reduce, Distributed Shared Memory) – these are designed for and superior in special scenarios.

Different approaches for a technical infrastructure for parallel computing (in general and with R)

Some buzzwords:

- Multicore — using more than one “core” of a modern CPU
- Cluster — using several computers in (mostly) local networks
- Grid — using several computers “all over the world”
- Cloud — rent CPU hours in huge computing centers

There are several possibilities for executing parallel R programs. Either on the infrastructure side and on the R side. There is no solution fitting all needs, but most likely a good solution for each requirement.

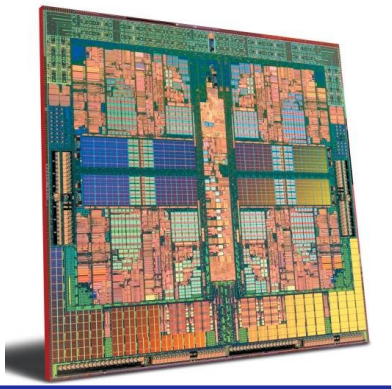
Common traps / pitfalls on parallel computing using plain R

Beside pitfalls on the programming side (random numbers etc.) there are general problems:

- Workgroups: If several people use parallel computing in the same environment, this requires far more coordination as in the sequential case. Can be avoided using management software.
- Memory: Most solutions require data duplicated per worker. This can easily lead to memory overload. Can be avoided with some management software.
- Administration: Mostly all parallel techniques and frameworks need attentive administration (e.g.: different R package versions on different machines).

Multicore

- Using more than one or all calculation cores from modern CPUs.
- Fast communication between Master and Workers.
- “Copy on write”/shared: extremely useful for large datasets, as this can save a lot of memory: only *modified* data require extra memory for workers.
- No administration needed, R packages are enough.



R packages (examples): multicore (*nix only) - process based.
doMC (Windows too) - threaded.

Summary multicore

PRO

- “R only”: easy to install and use. Zero administration.
- Fast communication.
- Very low memory footprint.

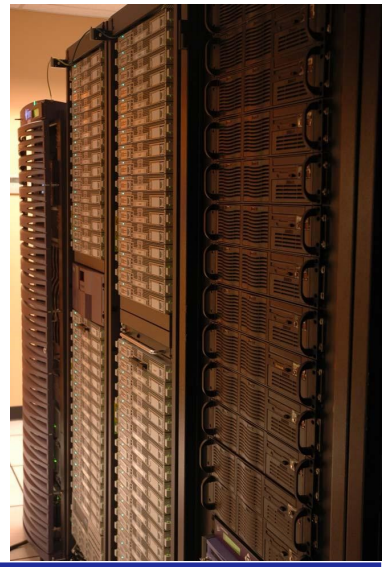
CONTRA

- Only useable on a single machine (if not combined with other techniques).

Conclusion: best and most convenient solution for desktop/laptop machines. Also very useful for programs with huge memory footprint.

Cluster

- Parallel computation can be distributed over several computers (commonly using the “Message Passing Interface” (MPI) today).
- Slower communication (over the network).
- Very established techniques, lots of R packages, even clusters with mixed operating systems possible.
- Screw up is easy if only used from R side (batch/resource manager suggested).
- Adding resources is easy.
- Much administration needed.



R packages (examples): snow, snowfall, papply, doXX, foreach, nws, biopara....

Summary Cluster

PRO

- Flexible. Existing machines can be used together.
- Usage from R side uncomplicated, interactive R shell usable.

CONTRA

- High administration needed.
- If used without additional management systems problems appear quickly.

Conclusion: best solution for smaller amounts of machines. A resource manager is suggested, but (mostly) at the loss of interactive R mode.

Grid

- Grid computing is basically an extension of cluster computing, often with machines from (many) different computing centers involved. E.g. many universities in Germany participate in nationwide Grids (D-Grid, bwGRID).
- Grids usually have huge amounts of processors (e.g. LHC Grid: >100.000).
- Only batch usage. Few R solutions, you need to know what you are doing.
- Certifications/Virtual Organisation membership required on most, can be complicated.
- Often unhandy/impossible to install own packages as some Grids are based on conservative software stacks (old R versions, C Compilers, Libraries etc.).
- Problem: sensible data can be transferred and saved in different locations.

Products R related: GridR (Condor/Atlas), Rsge (Sun Grid Engine)

Conclusion Grid

PRO

- Very high amounts of resources.
- Completely managed: (if it is working), just start jobs.

CONTRA

- High administration.
- Certfications/VOs complicated.
- Only batch mode.
- Data stored somewhere.

Conclusion: if you have access to a Grid, it may feature high amounts of resources. Downside is complicated handling and limitations on the software.

Cloud computing

- Extremely scalable: you only pay the CPU hours you are using and huge amounts of CPUs can be ordered at the same time.
- Machines are basically “virtual machines” which are cloned. After starting the cloud instances, it can behave like a cluster.
- Interactive R shell possible (Remote Login).
- Administration: depends. You can use preconfigured virtual machines with R. For specific demands some extra effort might be required. If it runs: very little.
- Costs: From 0,1 to 2,5 dollars / CPU hour (depends e.g. on RAM).

Products (R related) examples: Amazon EC2 AMI with plain R (Open Data Group). (Older) EC2 AMI with R and Bioconductor (M. Aryee)

Summary Cloud computing

PRO

- Can be very cheap compared to dedicated hardware.
- Extremly flexible.
- Usage from R side can be used like an exclusive cluster.
- Simple administration of machines via browser.

CONTRA

- Can become more expensive as real hardware.
- Data is stored somewhere.
- Few R support (like automatically shutdown instance after job finish).

Conclusion: best if you need computing power rarely or sometimes need extreme computing power for a short time. Setup can be tricky if you need special software.

Part 2: Batch / Resource Managers for Clusters

- There are dozens of established batch systems / resource managers for clusters (e.g. PBS, Moab, slurm, Condor...). But all are for general use and sometimes inconvient when used with R.

sfCluster/snowfall (Knaus et al, 2009)

- Managing clusters for R usage.
- R interactive mode useable (managed).
- In conjunction with the R package snowfall, user do not have to change their R program for running with different parallelisation options.
- Completely hides cluster setup from user (with automatically resource allocation).
- Support for user groups, monitoring, can stop clusters if single worker exceed memory etc.

Example using sfCluster/snowfall

```
jo@imbi9:~$ sfCluster -i --cpus=32 --mem=2.5G
Session-ID  : 9SO6CK5o_R
imbi1.imbi.privat: 8 CPUs assigned (8 possible).
imbi9.imbi.privat: 5 CPUs assigned (5 possible).
  [...]
imbi3.imbi.privat: 1 CPUs assigned (8 possible).

ASSIGNED 32 cpus on 7 machines (32 requested).

-- /usr/local/bin/sfCluster: START R-interactive session --

> library( snowfall )
Loading required package: snow
> sfInit()
Loading required package: Rmpi
Library Rmpi loaded.
        32 slaves are spawned successfully. 0 failed.
Startup Lockfile removed: /h/jo/.sfCluster/SFINIT_jo_9SO6CK5o_R_2113_110118
R Version:  R version 2.11.1 (2010-05-31)

snowfall 1.84 initialized (using snow 0.3-3): parallel execution on 32 CPUs.

> q()
Save workspace image? [y/n/c]: n

-- /usr/local/bin/sfCluster: INTERACTIVE session finished. --
```

Literature / Links

- M. Eugster, J. Knaus, C. Porzelius, M. Schmidberger and E. Vicedo: “Hands-on tutorial for parallel computing with R”, Computational Statistics, 2010.
- CRAN Task View: High-Performance and Parallel Computing with R: http://cran.r-project.org/web/views/HighPerformanceComputing.html
- M. Schmidberger, M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney and U. Mansmann: “State of the Art in Parallel Computing with R”, Journal of Statistical Software 2009.
- J. Knaus, C. Porzelius, H. Binder and G. Schwarzer: “Easier parallel computing in R with snowfall and sfCluster”, The R Journal, 2009.
- sfCluster/snowfall Website: http://www.imbi.uni-freiburg.de/parallel (tutorials, materials, additional information).