

May 09, 2008

# Creating R code and managing R programs for parallel execution with snowfall/sfCluster

Jochen Knaus ([jo@imbi.uni-freiburg.de](mailto:jo@imbi.uni-freiburg.de))  
IMBI Freiburg  
supported by DFG Forschergruppe FOR 534

# Overview

(Short) introduction to parallel computing in R  
*Principles, terminology, common techniques, existing R packages.*

snowfall R package

*Design, features, functions (basic), comparison to snow.*

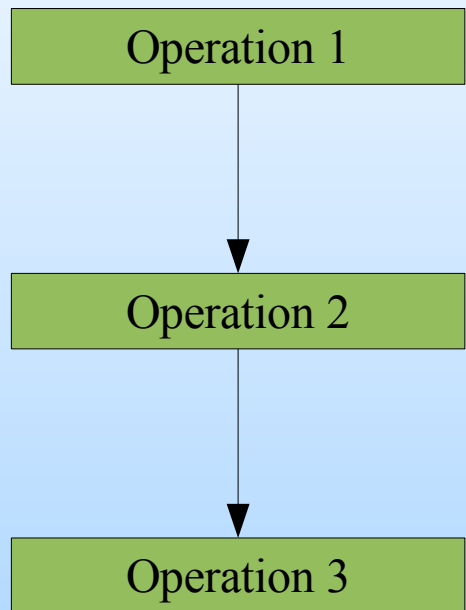
sfCluster management tool

*Design, features, implementation, administration.*

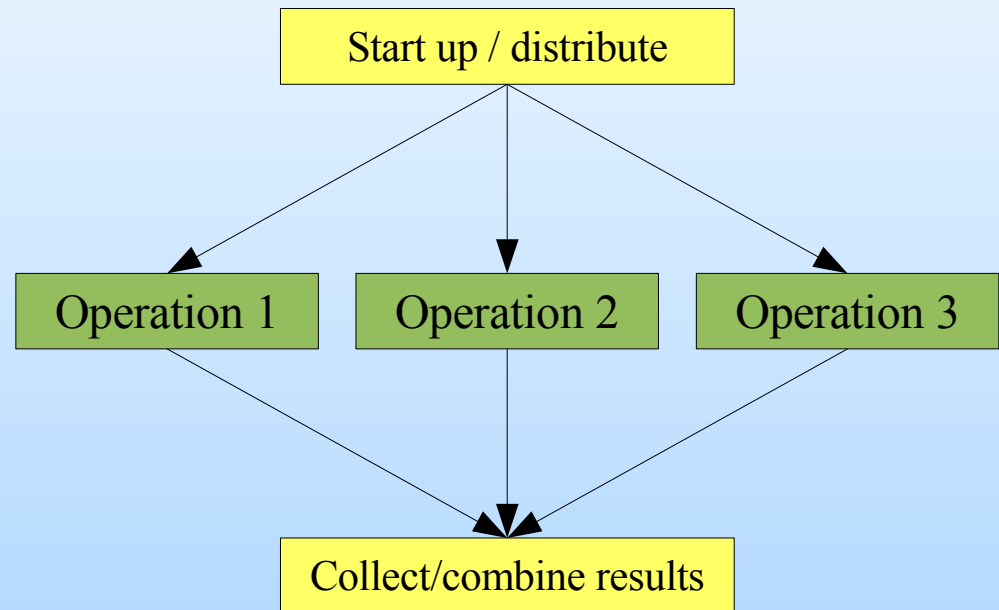
Future additions / outlook / discussion

# Parallel programs

“Conventional” serial program  
Sequential execution



Parallel execution



# About parallelisation

- Goal on parallel execution is to receive a significant speed up on overall computing time or the possibility to work with larger datasets (like high-dimensional data).
- As computers have more and more CPUs, parallel execution gets cheaper and more attractive.
- Parallel algorithms are often not trivial. But many problems in biostatistics are **embarrassingly parallel**, for example bootstrapping or cross-validation.

# Parallel Computing and R

- R is *not* designed to run parallel.
- But there are several packages to connect R to common cluster solutions.
- Cluster means, that several computers can be used together for an R program (but can also used to enhance program runtime on a single multicore computer).

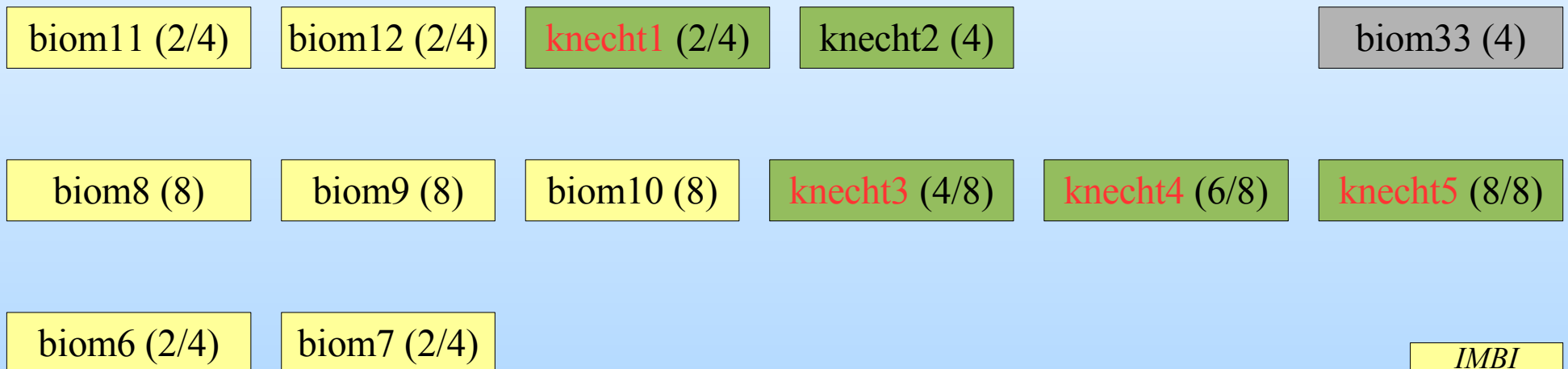
# Cluster terminology

- **CPU** : single calculation unit. Modern hardware CPUs may feature more than one terminological CPU (“**Core**”).
- **Node** : single computer accessible in a cluster. A node can have more than one CPU.
- **Universe** : all machines which can possibly used as a node in a concrete cluster.
- **Cluster** : subset from the universe for specific use.
- **Master** : calculation is started on this node and the control program (start/init) is running here.
- **Slave** : CPU on which a raw parallelized calculation part is running.

# IMBI/FDM computing infrastructure

Computing servers from two institutes useable for parallel programs.

Numbers in brackets show useable CPUs for clustering.



Max. useable at the moment: 56 CPUs (mostly Intel Xeons).

**Red machines** running 64-Bit OS (all machines: Debian Etch, except one w. Ubuntu 7.10).

# Cluster solutions useable with R

To run clusters in a (heterogeneous) machine park there are two basic solutions:

- **PVM** : Parallel Virtual Machine
- **MPI** : Message Passing Interface – standard API definition. Well established implementations are **LAM** (Local Area Multicomputer) or **OpenMPI**.

MPI became the de facto standard during the last years, which is widely supported in many academic and commercial projects.

Our choice for MPI was due to the better support of precise CPU usage.

# Available R packages for parallel computing

- There are several R packages for parallel computing on clusters, like `rpvm` and `Rmpi` on the lower level, `snow`, `snowFT` and `papply` on higher level.
- Low level libraries ask too much from average users.
- All these packages leave cluster setup and management to the user (which means they require a running cluster or setup a basic local cluster for themselves at best).

## First attempt: chose snow

- After evaluation, we chose **snow** as parallel package for our programs.
- We have a heterogeneous infrastructure in our institute, so **MPI** was the first choice.
- **snowFT** was not possible because it is only available for PVM.
- snow features powerful functions and is easy to use if you are keen on cluster techniques.
- snow works very well, but feature minor comfort on some common needs (starting point for snowfall).

# Practical problems using clusters

There are several general pitfalls when developing parallel programs:

- Cluster setup can be problematic with multiple users/user groups, especially in heterogeneous infrastructures.
- Memory usage multiplies.
- Keep track what is going on in the cluster can be difficult (esp. with problems on slave-nodes).
- Network traffic eats up benefit from parallel usage.

**And much worse: overload or problems can easily affect (cluster) programs from *all* users!**

# Situation

- In previous solutions the R programmer is responsible for driving past these problems.
- We wanted a little tool solving some of the described problems which can be used by non expert programmers, who just code R as mathematical tool.
- We developed `sfCluster` as this tool and combined it with our snow-abstracting R package `snowfall`, which features for easier access to parallel programming with R.

# Practical solution: sfCluster/snowfall

## snowfall

- R package based on snow (MPI).
- Featuring wrappers for many snow functions, make functions available in sequential execution, extended error handling as well as additional functions for common needs.

## sfCluster

- Unix tool for automatic cluster management and monitoring. Additional features for debugging and comfort.

# Overview

(Short) introduction to parallel computing in R

**snowfall R package**

*Design, features, functions (basic), comparison to snow.*

sfCluster management tool

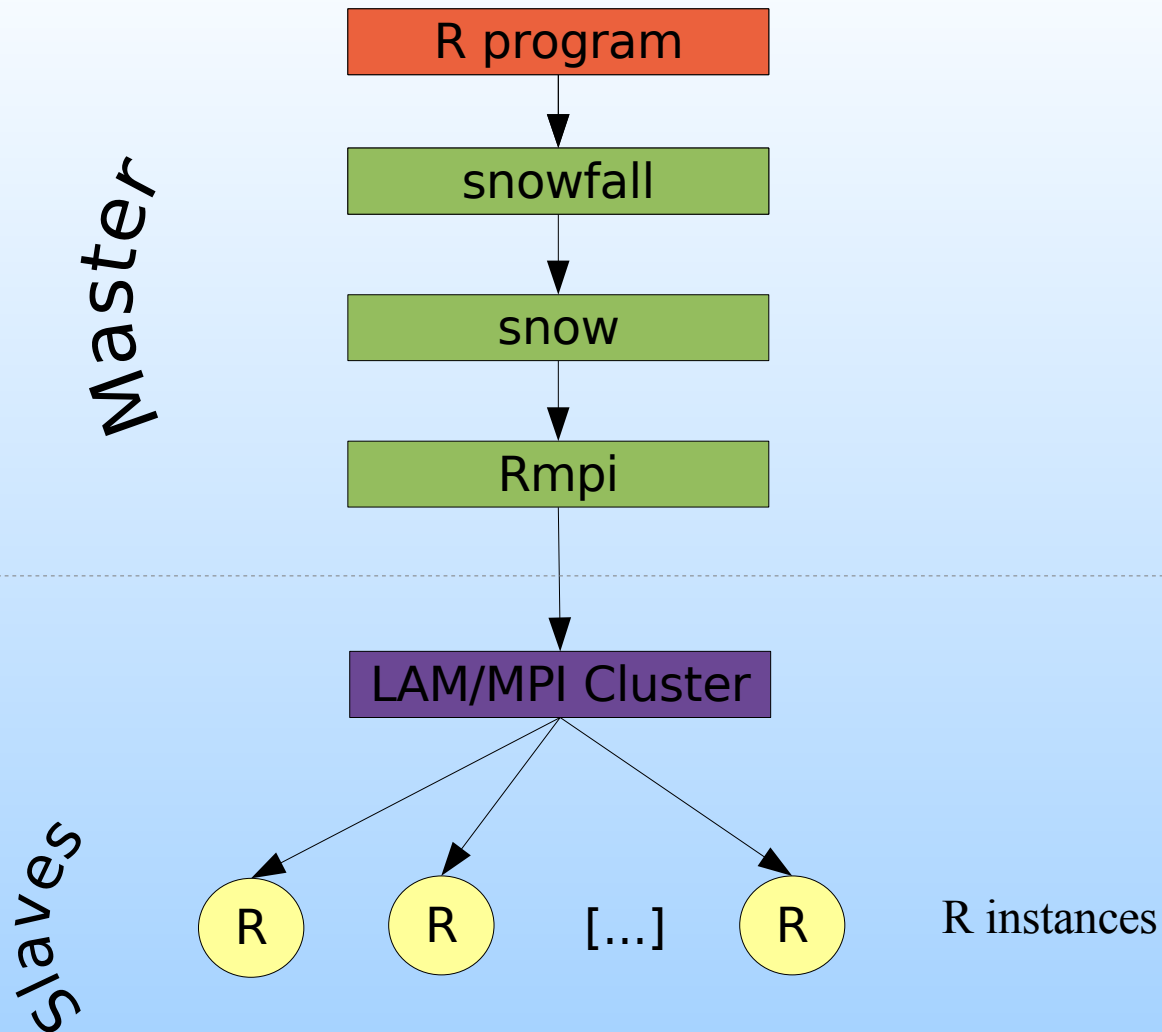
Future additions / outlook / discussion

# snowfall R package

## Design goals:

- Easy access.
- Wrappers for snow functions.
- Additional functions for common tasks.
- Fully supporting sequential execution without any code changes (all wrappers work in sequential mode, too).
- Directly runnable everywhere (even without snow): programs are distributable inside packages.
- Extended error checks.
- Connector to sfCluster.

# Structure of a parallel R program



# snowfall extending snow

- Error handling is much more **strict** and `stop()`s immediately on slave errors. This is easier for inexperienced users. Advanced users can get rid of this default behaviour.
- Wrappers for snow functions work identically in sequential execution, too.
- Implicit cluster handler.
- snow functions can be used directly (porting is easy):

```
library(snowfall)
sfInit()

clusterEvalQ( sfGetCluster(), ls() )
```

- snowfall includes additional functions for more comfort...

# Limitations from snowfall to snow

- Cluster usage limited to MPI (no PVM or socket clusters).
- Uniform Random Number Generation is not wrapped.
- Some functions are not wrapped.

# Example using R package snowfall

```
sfInit(parallel=TRUE, cpus=2)

require(mvna)
data(sir.adm)

sfExport("sir.adm")
sfLibrary(cmprsk)

wrapper <- function(a) {
  index <- sample(1:nrow(sir.adm),
                 replace=TRUE)
  temp <- sir.adm[index, ]
  fit <- crr(temp$time, temp$status,
            temp$pneu, failcode=1, cencode=0)
  return(fit$coef)
}

result <- sfLapply(1:100, wrapper)
sfStop()

mean(unlist(rbind(result)))
```

Initialisation

Export data needed on all slaves

Load library on all slaves

Parallel call. wrapper() is called for each index. Calls are distributed on slaves

... more in Christine Porzelius talk

# Further snowfall functions

- Additional functions in snowfall:

- `sfLibrary` Load library on slaves
- `sfSource` Source file on slaves
- `sfClusterApplySR` parallel lapply with intermediate saving of results and possibility to restore results after interruption.
- `sfExport` Export variables (extended implementation)
- `sfExportAll` Export all variables
- `sfRemove` Remove specific objects from slaves
- `sfRemoveAll` Remove anything from slave memory
- `sfTest` Unit tests (also use able for slave R running tests)
- ... and some more.

# Overview

(Short) introduction to parallel computing in R

snowfall R package

**sfCluster management tool**  
*Design, features, implementation, administration.*

Future additions / outlook / discussion

# sfCluster management tool

## Design goals:

- Hide cluster handling, setup and shutdown from user.
- Make use as safe as possible even for inexperienced users (do not allow user to run into pitfalls affecting other users programs).
- Implementation as Unix command line tool.
- Using only open source tools.

# sfCluster features (1)

- Automatic resource allocation, depending on current usage of universe.
- One LAM cluster per program (means: multiple clusters per user): clusters are independent.
- Monitoring the execution of parallel R programs with detection of problems.
- Safe shutdown of cluster sessions.
- Works even if LAM cluster itself is gone.
- Local user installation is possible if LAM is present (no root rights needed for installation).

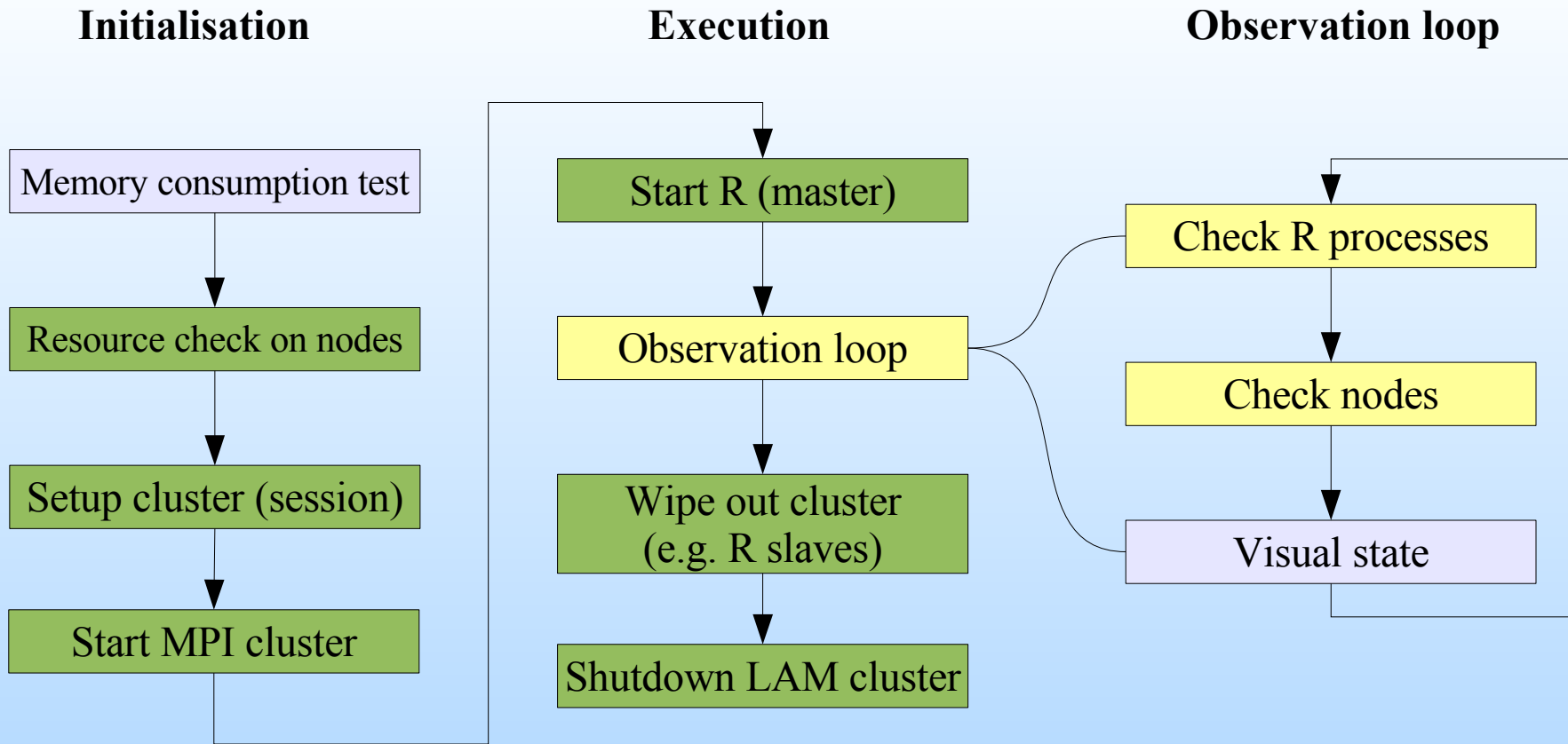
## sfCluster features (2)

- Able to use specific R version for parallel R program.
- Able to use common R shell with set up cluster.
- Monitoring mode with “live” informations about the current cluster state, R processes, access to slave logfiles and R output.
- Administration options (for user and administrator) like viewing running clusters or print currently free resources.
- Safe killing of sfCluster sessions and leftover parts (even if LAM itself died).

## sfCluster features (3): “Subuniverses”

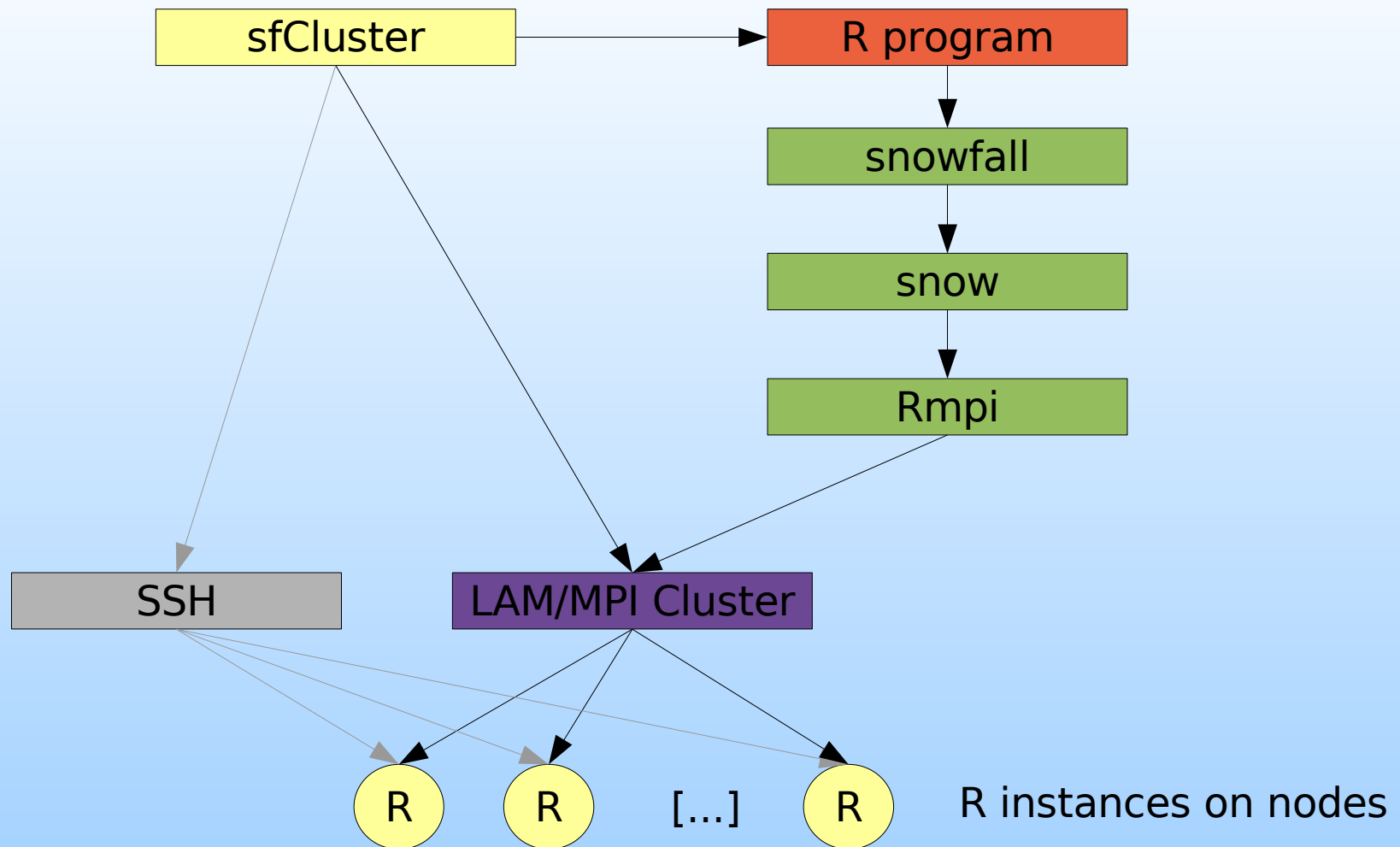
- sfCluster allows the definition of “subuniverses” in the whole cluster universe, which are accessible to specific user groups.
- Each node can be accessible in multiple groups (groups are not hierarchical).
- Each user sees only machines belonging to his/her group.
- This means certain users have probably no access to some nodes in the universe (as they have no login for these nodes or some machines are reserved for specific users).
- Groups can be defined centralized.

# sfCluster workflow



(optional) stop on error      optional step

# Communication structure



# sfCluster execution modes

There are three main execution modes for running sfCluster:

- **batch** (-b)                      like “R CMD BATCH”. *Default.*
- **interactive** (-i)                interactive R shell with cluster.
- **monitor** (-m)                    batch + debugging informations.
  
- **sequential** (-s): serial/sequential execution without cluster.  
Can be used with any of the main modes.

## sfCluster: user options

- Request specific number of CPUs.
- Request specific R version for execution.
- Send mail at success or failure (to any number of addresses).
- Set nice level of all R instances (default: 19)
- Ignore specific nodes from universe.
- Show usage of resources in universe.
- Show available R-versions.

# sfCluster: administration options

- **Installation test** (check nodes for working sfCluster installation).
- Show current **running sessions** (per user or all users).
- Seek for lost clusters/daemons (per user or all users).
- Forced **session shutdown** (kill). Also kill parts of crashed sessions/clusters (zombie R processes, LAM daemons etc.). Can be used by (administration user) *root*.

## sfCluster administrator side

- Behaviour can be widely configured in Unix style configuration files (e.g. definition of event “memory shortage” and behaviour on it's occurrence).
- All users are in group “1”. Must be redefined only if a user is not in this group. Groupfile is plain textfile, too.
- Hosts are configured analogue to host configuration of LAM, with additional setting `maxload` and `groups`.

# Host configuration

- Host configuration is very easy and (mostly) like the original LAM:

```
biom8.imbi.uni-freiburg.de  cpu=8 maxload=8 group=1,3 sched=yes  
knecht2.fdm.uni-freiburg.de  cpu=4 maxload=2 group=3    sched=yes  
biom7.imbi.uni-freiburg.de  cpu=4 maxload=2 group=2,3 sched=yes
```

- Memory usage is always limited by physical RAM.
- Order in the hostfile describes the priority of taking the machine for calculation.
- Overload ( $\text{maxload} > \text{cpu}$ ) is not scheduled correctly at the moment.

# sfCluster implementation

- Perl script (main script plus some classes), ~13000 lines.
- Uses some open source Perl libraries (for system introspection):
  - Proc::ProcessTable, Proc::Simple, Proc::Killfam
  - Mail::Sendmail, [Curses]
  - Sys::Hostname::FQDN, Sys::Info, Linux::SysInfo, Linux::MemInfo
- Uses some Unix tools:
  - renice, tail, stat...
- Uses SSH and LAM commandline programs.
- For visual monitoring mode **ncurses** is used (optional).
- Code is well documented and can be changed easily.

# What do I need to start?

- Base installation as presented.
- Passwordless SSH authentication between nodes (handy for LAM/MPI, too).
- Linux. Works well on Debian Etch/Lenny and Ubuntu Gusty Gibbon (7.10). Does not work on Ubuntu Hardy Heron (8.04) at the moment due to unfixed Rmpi/LAM bug.
- Other Unixes should work after replacement for system introspection libraries.
- Windows does *not* work (CygWin untested).

# Overview

(Short) introduction to parallel computing in R

snowfall R package

sfCluster management tool

Future additions / outlook / discussion

# Future additions

- Installer for sfCluster and all needed components/libraries.
- Port to OpenMPI.
- Adding a simple batchmode (queuing of programs, which are executed if resources are available).
- Probably connections to commercial batch systems.
- Load balanced sfClusterApplySR.

# Future additions: scalability/scheduling

- Problem: the current quad core processors do not scale linear on real life applications.
- This varies on hardware architectures (e.g. Intel Xeon scales weak, AMD Opteron scales better). The reason for this is the limited bandwidth in current shared memory architectures, which often becomes a bottleneck.
- Dual Core processors scale better at the moment.
- => Scheduling should be first on max. 2/3 load on all machines, then use other 1/3 of machines.
- Another scheduling issue: overload (if wanted) is not scheduled correctly.

# Summary

- For skilled users many other solutions work well, too.
- Our target group are average R users without further technological skills.
- sfCluster works very well in small to medium sized clusters, but probably does not work very well in larger clusters (>250 CPUs).
- Automatic cluster setup and resource management approved very practical in everyday use.
- snowfall adds comfort to parallel R programming and includes many ideas from our users.

Fine.

Get sfCluster and snowfall at:

<http://www.imbi.uni-freiburg.de/parallel>

*thanks to cp and sc*