Futureverse P2P: Peer-to-Peer Parallelization in R

- Share compute among friends across the world

Henrik Bengtsson

University of California, San Francisco
R Foundation, R Consortium
@HenrikBengtsson

useR! 2025, Durham, NC, USA (2025-08-10)

Future … what?

An R assignment:

```
a <- 1 + 2
```

A future-value assignment:

```
f <- future({ 1 + 2 })
a <- value(f)
```

```
x <- 1:10
b <- slow_sum(x)
```

```
x <- 1:10
f <- future({ slow_sum(x) })
b <- value(f)
```

Future … why? (bc enables parallel processing)

Two calculations:

```
1 x_a <- 1:10
2 x_b <- 11:20
3 a <- slow_sum(x_a) # 1 min
4 b <- slow_sum(x_b) # 1 min
5 c <- a + b
```

Total time: 2 mins

Two futures:

```
1 x_a <- 1:10
2 x_b <- 11:20
3 f_a <- future( slow_sum(x_a) ) # 0 sec
4 f_b <- future( slow_sum(x_b) ) # 0 sec
5
6 a <- value(f_a)                # 1 min
7 b <- value(f_b)
8 c <- a + b
```

Total time: 1 min

=> futures are the core building block for parallel processing

Futureverse allows you to stick with your favorite coding style

Parallel alternatives to traditional, sequential functions:

```
ys <- lapply(xs, slow_sum)                    # base R
ys <- future_lapply(xs, slow_sum)             # {future.apply}
```

```
ys <- map(xs, slow_sum)                       # {purrr}
ys <- future_map(xs, slow_sum)                # {furrr}
```

```
plan(multisession)
plan(future.callr::callr)
plan(future.mirai::mirai_multisession)
plan(cluster, workers = c("n1", "desktop", "server.myuniv.org"))
plan(future.batchtools::batchtools_slurm)
```

Let’s go back in time …

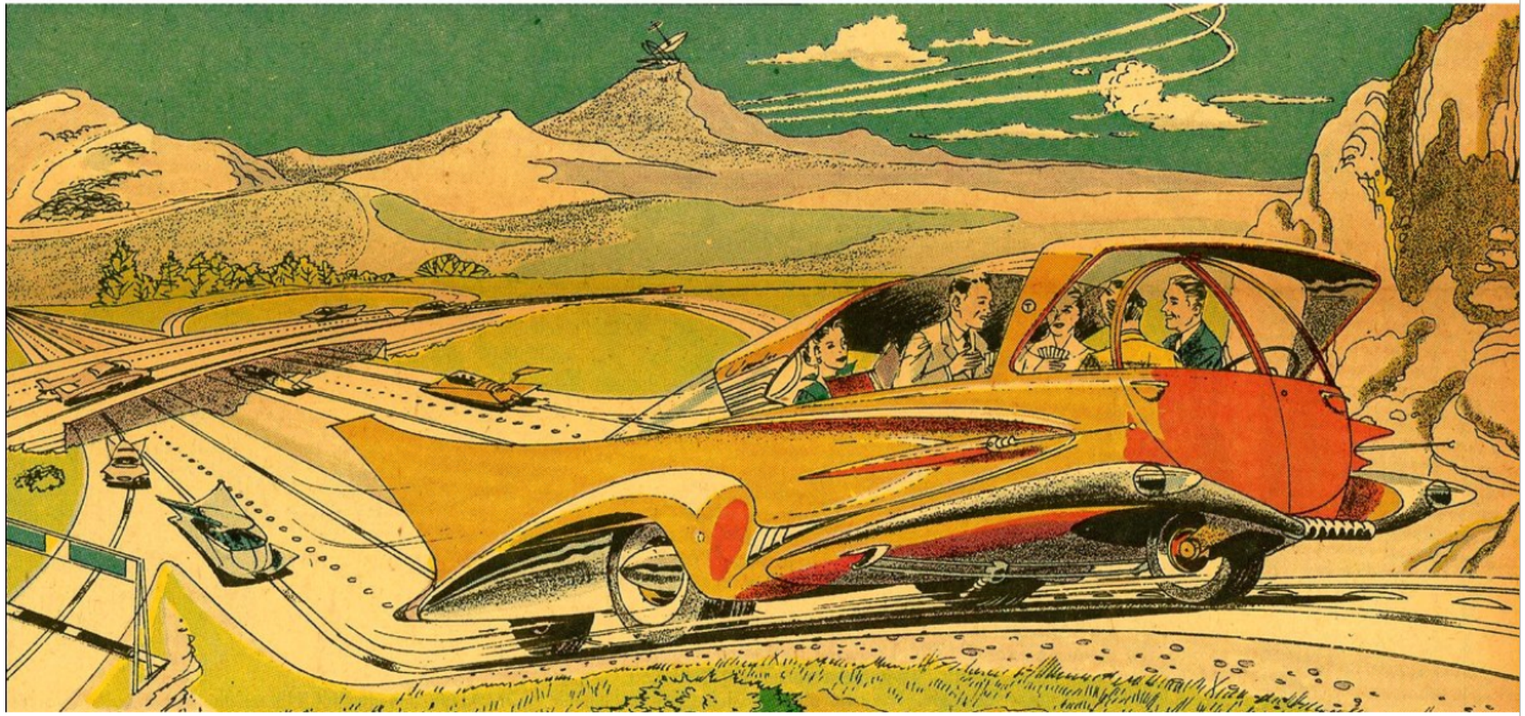
Ten-Year Anniversary (future 0.6.0 released June 2015)

future logo: ***Dan LaBar****, ggplot2 logo balloon wall:* ***Greg Swinehart*** *&* ***Hadley Wickham***

useR! 2016 at Stanford, CA, USA

A Future for R

Henrik Bengtsson, UC San Francisco



useR 2016, Stanford, CA, 2016-06-28

https://future.p2p.futureverse.org

useR! 2016 at Stanford, CA, USA

R package: future

- A simple **unified API**
- Works the same on **all platforms**
- **Easy to install**
- **Lightweight** (~300 kB incl. dependencies)
- **Vignettes**
- **Extendable** by anyone

5 / 18

https://future.p2p.futureverse.org

useR! 2016 at Stanford, CA, USA

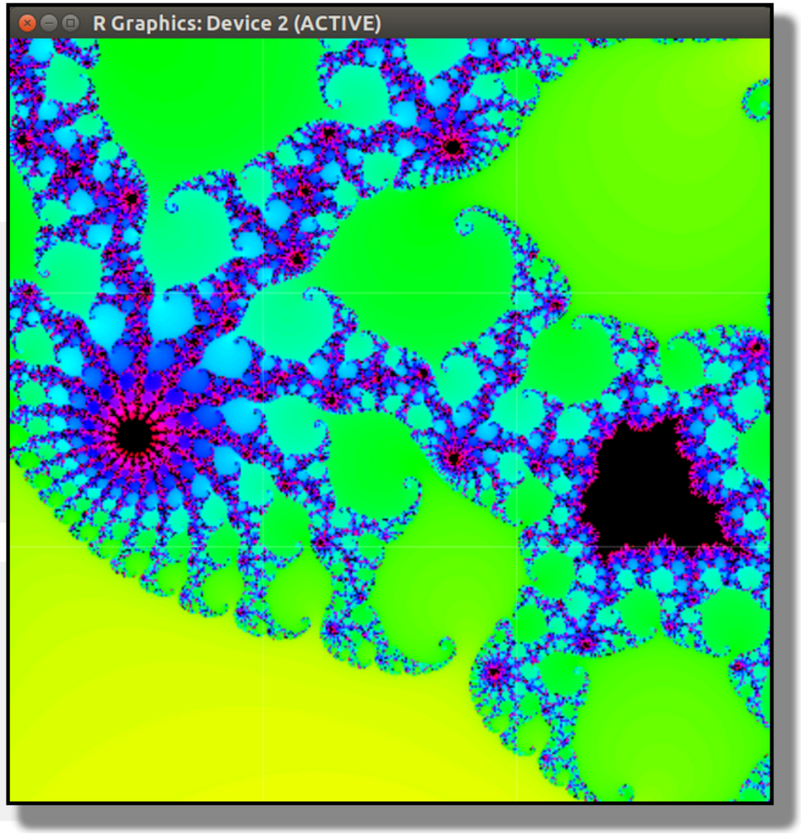
Consistent futures everywhere ☆

- Unix
- macOS
- Windows

```
> library("future")
> plan(multiprocess)
> demo("mandelbrot")

Calculating and plotting
Mandelbrot regions ...
```

```
- Region 1 done
- Region 2 done
- Region 7 done
- Region 5 done
- ...
```



7 / 18

https://future.p2p.futureverse.org

useR! 2016 at Stanford, CA, USA

future.BatchJobs: Futures for HPC

future.BatchJobs:	Job scheduler:
batchjobs_slurm	Slurm
batchjobs_sge	Sun Grid Engine
batchjobs_torque	TORQUE / PBS
batchjobs_lsf	Load Sharing Facility
batchjobs_openlava	OpenLava

```
library("future.BatchJobs")
plan(batchjobs_slurm)
```

```
bam <- listenv()
for (i in seq_along(fastq)) {
  bam[[i]] %<-% DNAseq::align(fastq[i])
}
```

12 / 18

https://future.p2p.futureverse.org

useR! 2016 at Stanford, CA, USA

A7. Futures I'd like to see

- `plan(r32)`
  - e.g. `RODBC::odbcConnectAccess()` works only on 32-bit R.
- `plan(p2p)`
  - Private and / or community-based peer-to-peer computer cluster
- `plan(rhelp)`
  - Post R scripts to R-help and ask for the results `:P`

Back to the present

future.p2p: sharing compute among friends - it's easy!

```
library(future.apply)

plan(future.p2p::cluster, cluster = "alice/friends")

y <- future_lapply(xs, slow_sum)
```

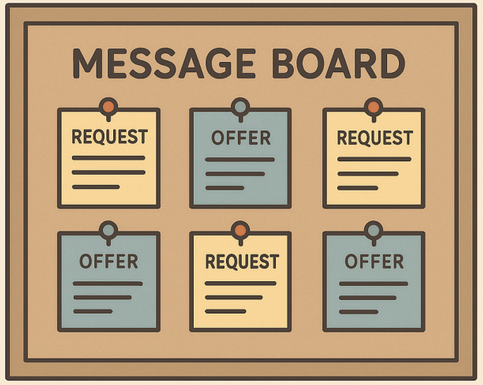
https://future.p2p.futureverse.org

How to get started …

A P2P cluster has two components

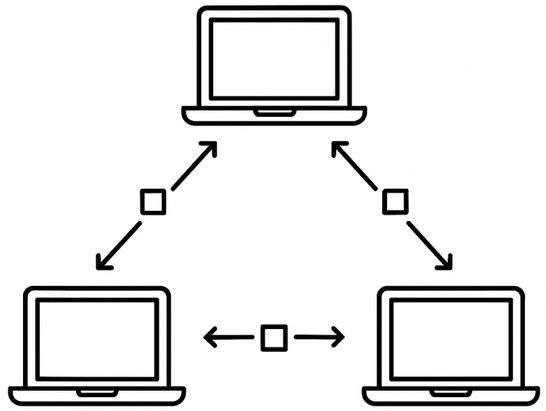
Message Board

Used to announce futures and offers to do work
(centralized; lightweight - only metadata)



P2P file-transfer protocol

Used to send futures to workers and receive results
(peer-to-peer; full-size data transfers)



To join a P2P cluster you need an account

All P2P cluster users need a pico.sh account to access the message board:

1. `ssh-keygen`

```
{carol}$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/carol/.ssh/id_ed25519):
Created directory '/home/carol/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/carol/.ssh/id_ed25519
Your public key has been saved in /home/carol/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:f211XJctII1KUe6HCq7zFA5gtS9smVsBxccNANY1s7g carol@hb-x1-2025
The key's randomart image is:
+--[ED25519 256]--+
|    ==o+*=oo.    |
|   o oo.+=o...  o|
|  o . oo...   ..+|
| . o + o.. .   oo|
|    B E S o .   +|
|   . B o o . . ..|
|    . + . . . o  |
|    .o     . .   |
|     o           |
```

2. `ssh pico.sh` => pick a username

```
pico.sh • signup

Welcome to pico.sh's management TUI!

By creating an account you get access to our pico services.  We
have free and paid services.  After you create an account, you can
go to our docs site to get started:

  https://pico.sh/getting-started

pubkey: SHA256:f211XJctII1KUe6HCq7zFA5gtS9smVsBxccNANY1s7g

 signup
> carol

 SIGNUP


^c quit • esc prev page • tab focus • enter create user
```

That’s it!

Alice hosts a P2P cluster

Alice sets up P2P cluster and gives ‘bob’ and ‘carol’ access:

```
[alice]> future.p2p::host_cluster("alice/friends", users=c("bob", "carol"))
```

This is basically setting up a shared message board.

Q. What happens if Bob tries to use the P2P cluster?

```
[bob]> plan(future.p2p::cluster, cluster = "alice/friends")
[bob]> y <- future_lapply(xs, slow_sum)
```

Nothing - it will get stuck! Why?

Contributing P2P workers is easy!

```
[alice]> future.p2p::worker("alice/friends")
```

```
[ bob ]> future.p2p::worker("alice/friends")
```

```
[carol]> future.p2p::worker("alice/friends")
```

```
[bob]> plan(future.p2p::cluster, cluster = "alice/friends")
[bob]> y <- future_lapply(xs, slow_sum)
```

=> Processed on three P2P workers

You can set it up from the terminal

Setting up a P2P cluster:

```
{alice}$ Rscript -e future.p2p::host_cluster --cluster=alice/friends \
         --users=bob,carol
```

Starting P2P workers:

```
{alice}$ Rscript -e future.p2p::worker --cluster=alice/friends &
{alice}$ Rscript -e future.p2p::worker --cluster=alice/friends &
```

```
{ bob }$ Rscript -e future.p2p::worker --cluster=alice/friends &
```

```
{carol}$ Rscript -e future.p2p::worker --cluster=alice/friends &
{carol}$ Rscript -e future.p2p::worker --cluster=alice/friends &
{carol}$ Rscript -e future.p2p::worker --cluster=alice/friends &
```

=> A shared P2P cluster with 6 workers. More can be added at any time!

Pros and cons

High latency but also high throughput

- High latency:
  - round-trip takes time, because p2p file transfers take time
  - Example: `1+2` takes 2-10 seconds to send, evaluate, and return
  - Just a prototype, so this will be improved
- High throughput:
  - any number of users and workers can join
  - a public P2P cluster could have thousands of P2P workers

⚠️ P2P computing requires mutual trust ⚠️

What if?

```
f <- future(system("erase-harddrive"))
```

Some technical mitigations:

- End-to-end encryption
- Run workers in sandboxed environments, e.g.
  - in a Linux container, or
  - in a virtual machine

  See **parallelly** package for some examples

=> I encourage work on these topics - it's challenging, but important

The Future is bright

… and fabulous

Stay tuned for exciting Futureverse improvements

On the horizon

- Custom random number generators (RNG) [together with Ralf Stubner]
- Resource specifications, e.g.
  - avoid memory overuse, e.g. `memory=2*GiB`
  - distribute to proper machines, e.g. `memory=2*GiB` and `gpu`

In the near future

- If you think **furrr** and **future.apply** are neat - just wait!

Thank you and may the future be with you!

- It’s easy to get started - just try it
- Support: https://github.com/HenrikBengtsson/future/discussions
- Tutorials: https://www.futureverse.org/tutorials.html
- Blog posts: https://www.futureverse.org/blog.html
- More features on the roadmap
- I love feedback, ideas, and bug reports 💜