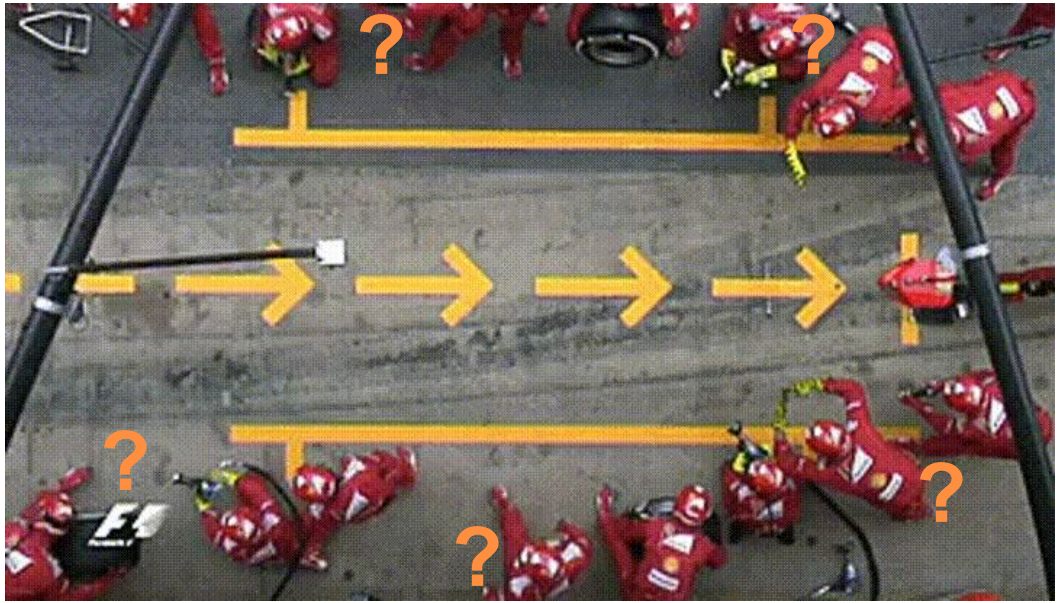
Futureverse: Profile Parallel Code



Where can we improve?

@HenrikBengtsson (University of California, San Francisco)
useR! 2022-06-22 (25 mins)

This talk was extended from 15 to 25 minutes, because one presenter couldn't make our session.

Futureverse: Ecosystem for parallel & distributed computing in R

Core API:

- **future**

Map-reduce API:

- **future.apply**
- **furrr**
- **doFuture**, e.g.
  - **foreach**
  - **plyr**
  - **BiocParallel**

Parallel backends:

- **parallel / parallelly** (local, remote, MPI, cloud)
- **future.callr** (local)
- **future.batchtools** (HPC job schedulers)
- …

Near-live progress updates:

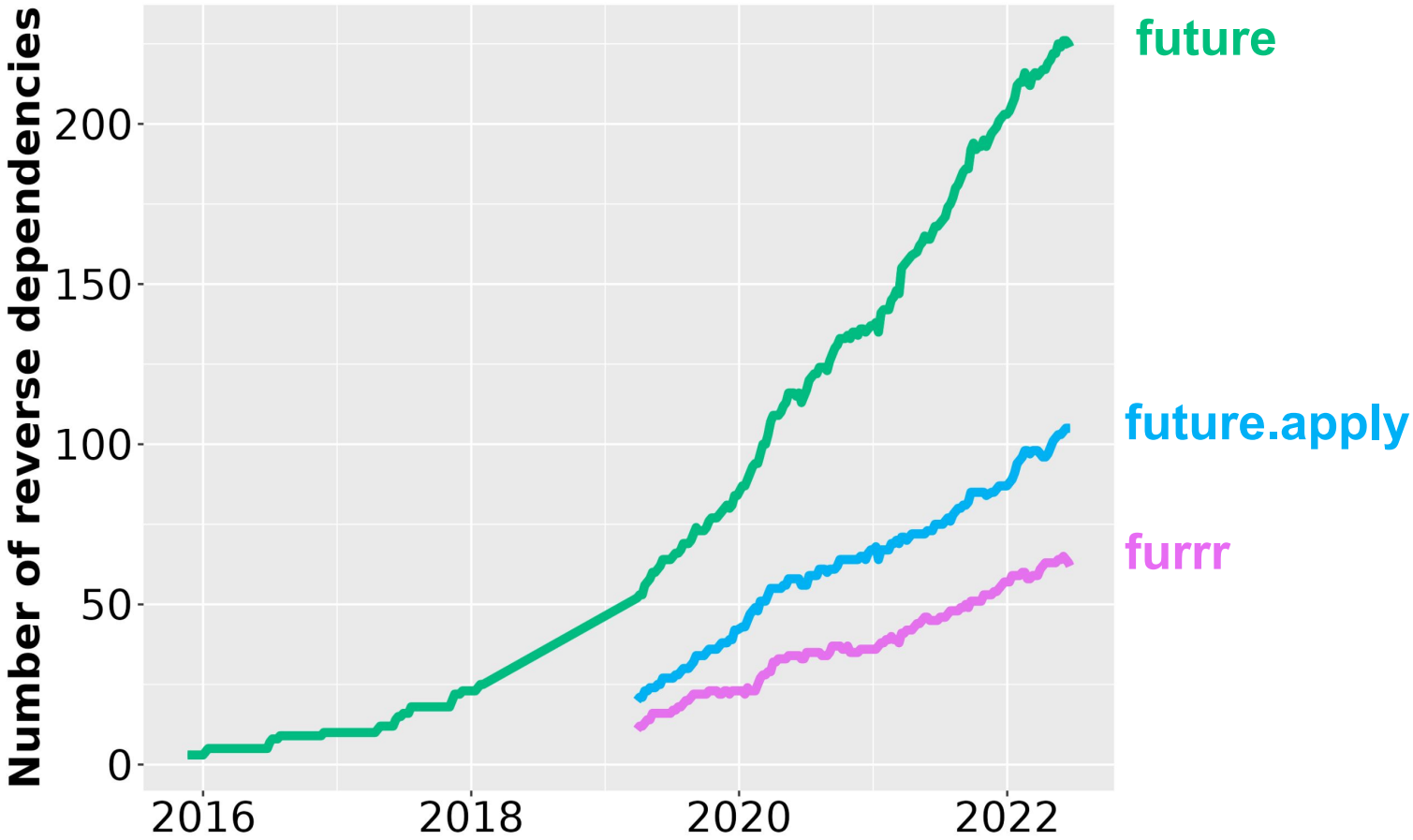
- **progressr**

60s: R package ‘future’

- A simple, unifying solution for parallel APIs
- "Write once, run anywhere"
- 100% cross-platform
- Easy to install (< 0.5 MiB total)
- Well tested, lots of CPU mileage, used in production
- Things should “just work”
- Correctness & reproducibility of the highest priorities

https://www.futureverse.org/

Rapid uptake & top-1% most downloaded



Quick Examples of Parallelizing with Futures

60s: Evaluate R in the Background

```r
# sequentially
x <- 7
y <- slow(x)              # ~ 1 minute
z <- another(x)           # ~ 0.5 minute

# in parallel
library(future)
plan(multisession)
f <- future(slow(x))      # ~ 1 minute (in background)
z <- another(x)           # ~ 0.5 minute
y <- value(f)             # => all done ~ 1 minute
```

60s: Parallel Base-R Apply

```r
# sequentially
x <- 1:20
y <- lapply(x, slow)            # ~ 20 minutes


# in parallel
library(future.apply)
plan(multisession)              # on 4-core laptop
y <- future_lapply(x, slow)     # ~ 5 minutes
```

60s: Parallel Tidyverse Apply

```r
# sequentially
library(purrr)
x <- 1:20
y <- map(x, slow)            # ~20 minutes

# in parallel
library(furrr)
plan(multisession)           # on 4-core laptop
y <- future_map(x, slow)     # ~5 minutes
```

60s: User can parallelize anywhere

```r
# sequentially
plan(sequential)

# On the local machine
plan(multisession)
plan(multisession, workers = 2)

# Ad-hoc cluster of local and remote machines
plan(cluster, workers = c("pi", "remote.server.org"))

# Via an HPC job scheduler (thousands of workers)
plan(batchtools_slurm)
```

Futureverse is unique

- Exports “globals” automatically, i.e. objects and functions that are needed by parallel workers
- Relays output & conditions signaled, i.e. errors, warning, messages, and standard output
- Built-in statistically sound random numbers
- Near-live progress updates - also from remote workers
- You, as a developer, don’t have to think “parallel workers” - just which R expressions to parallelize

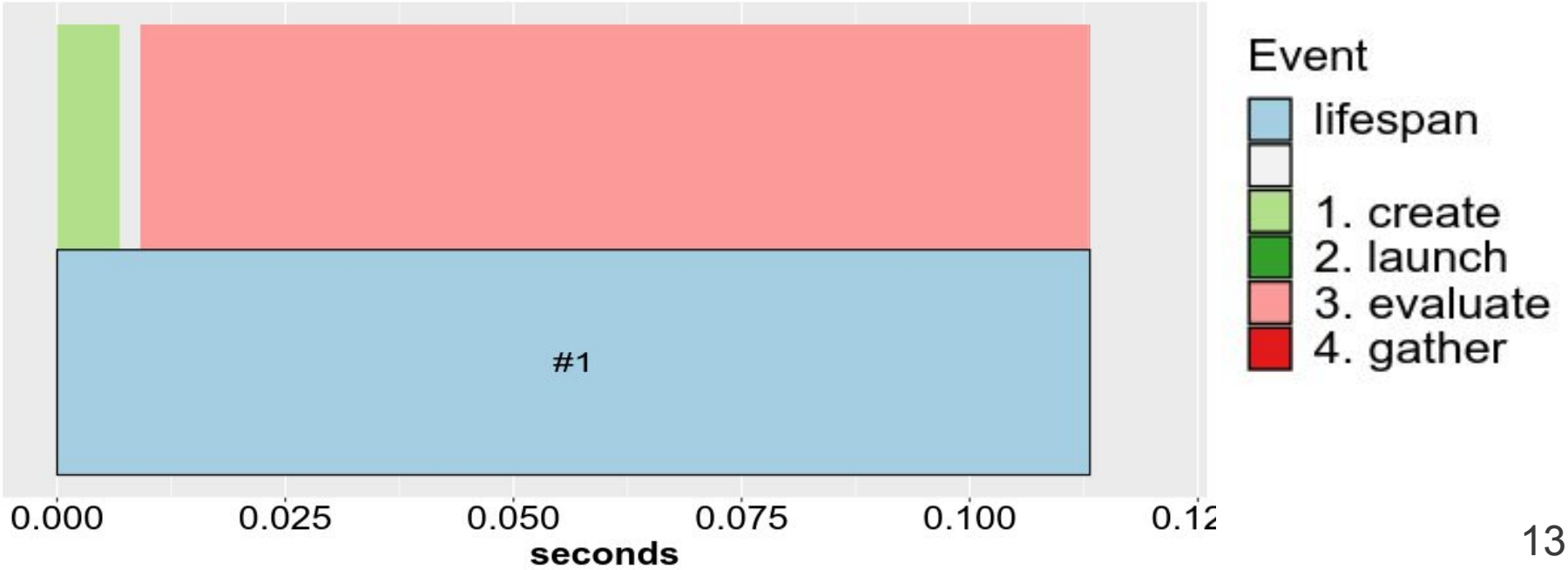
Profiling the parallel framework

Adding a journaling system

- **Log events**, e.g. creation, launching, evaluation, gathering of results
- **Timing information** for now (memory is tricky)
- Work with **any parallel backends**
- **Near-zero overhead** if not used
- Tabular **raw data**
- **Textual & graphical presentation**

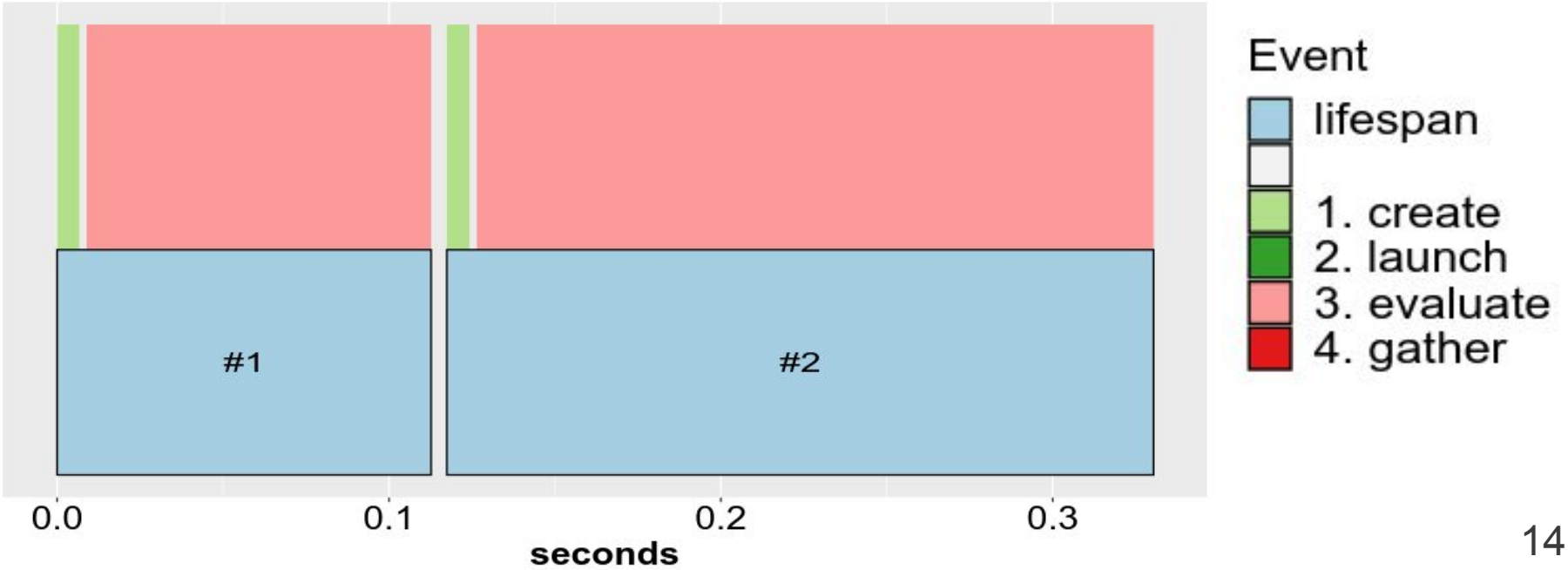
Profiling a sequential future

```
plan(sequential)
f <- future(slow(1))
v <- value(f)
```

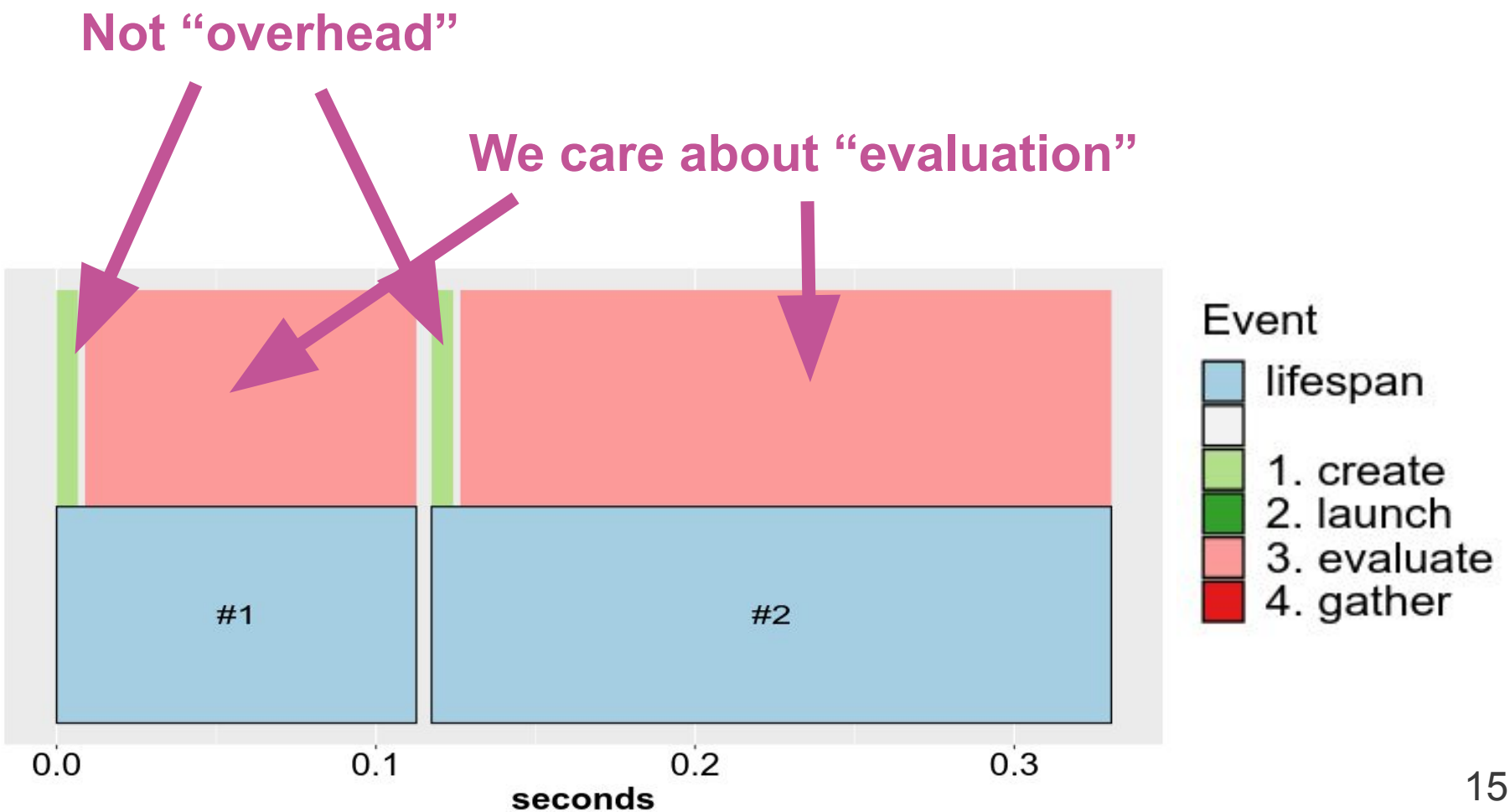


Profiling two sequential futures

```
plan(sequential)
fs <- lapply(1:2, function(x) future(slow(x))
vs <- value(fs)
```

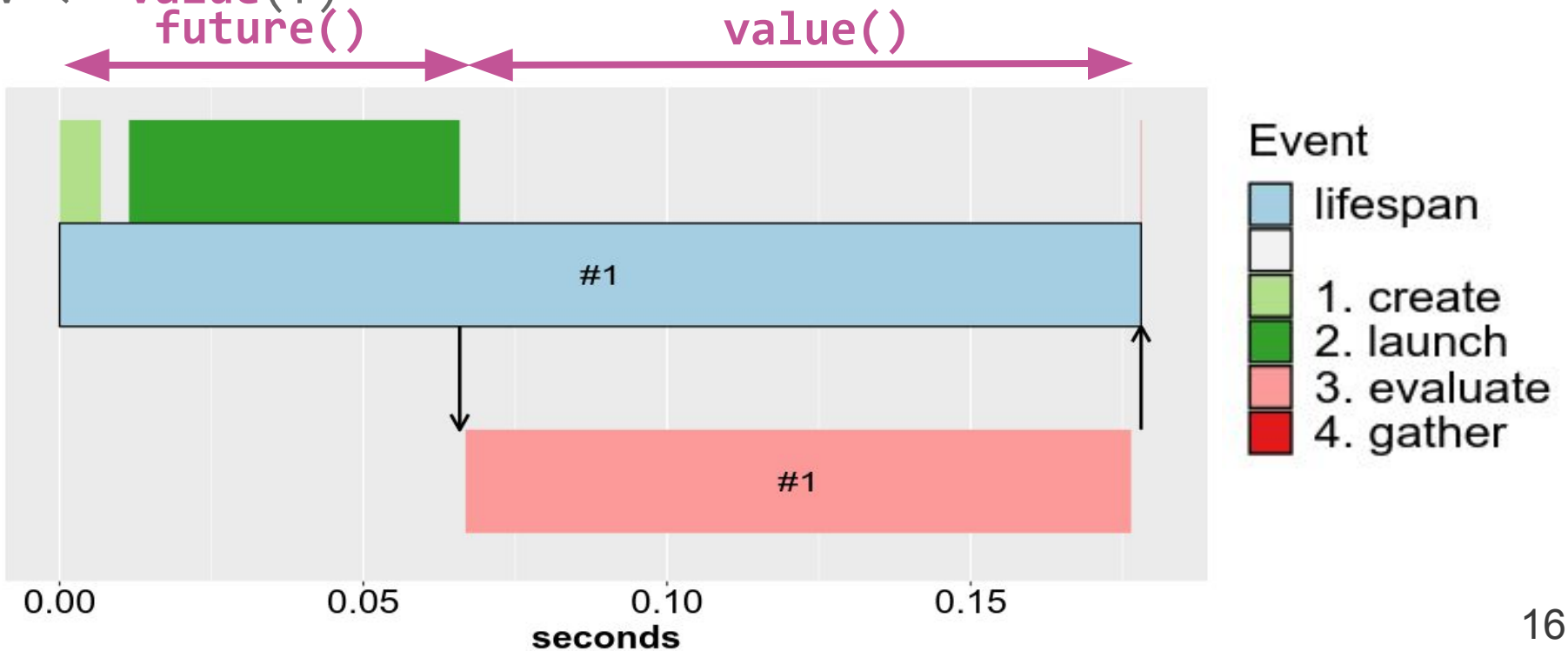


Parallelization comes with overhead



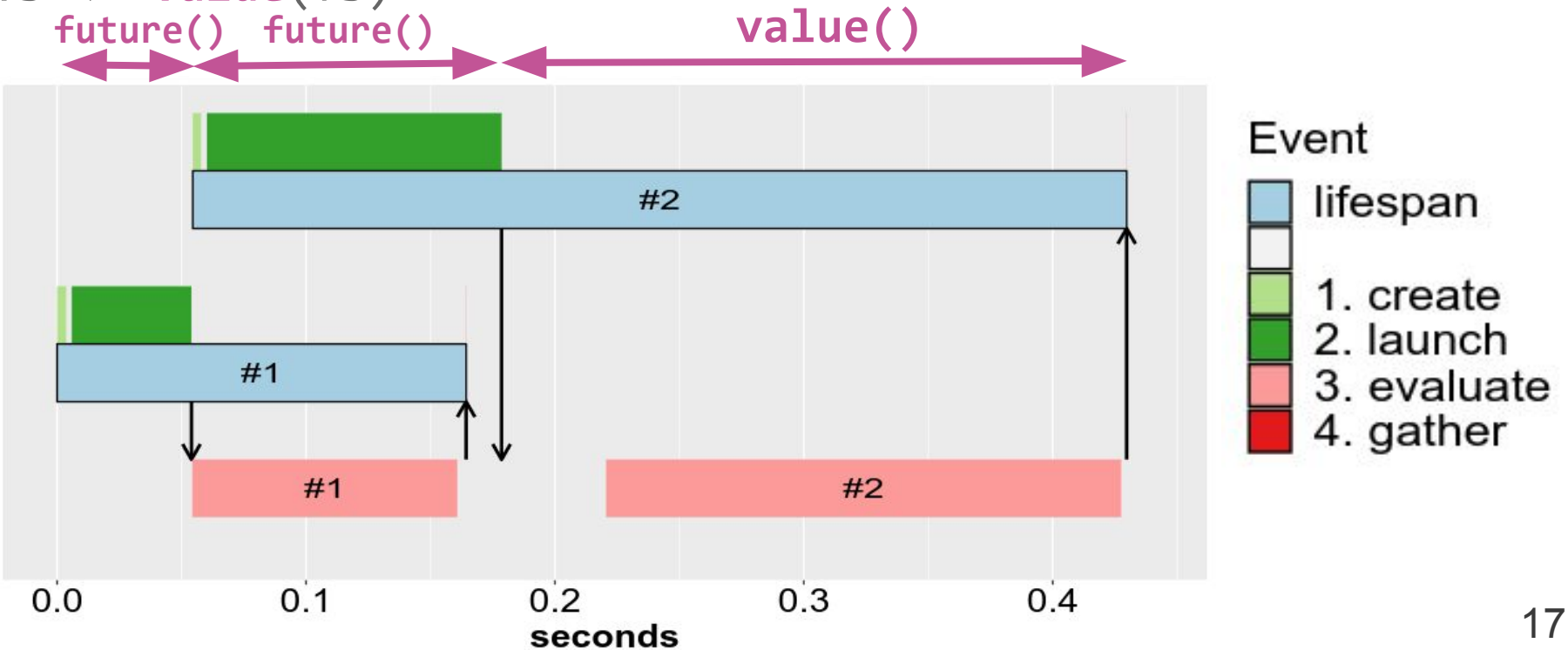
Profiling 1 future with 1 worker

```
plan(cluster, workers = 1)
f <- future(slow(1))
v <- value(f)
```



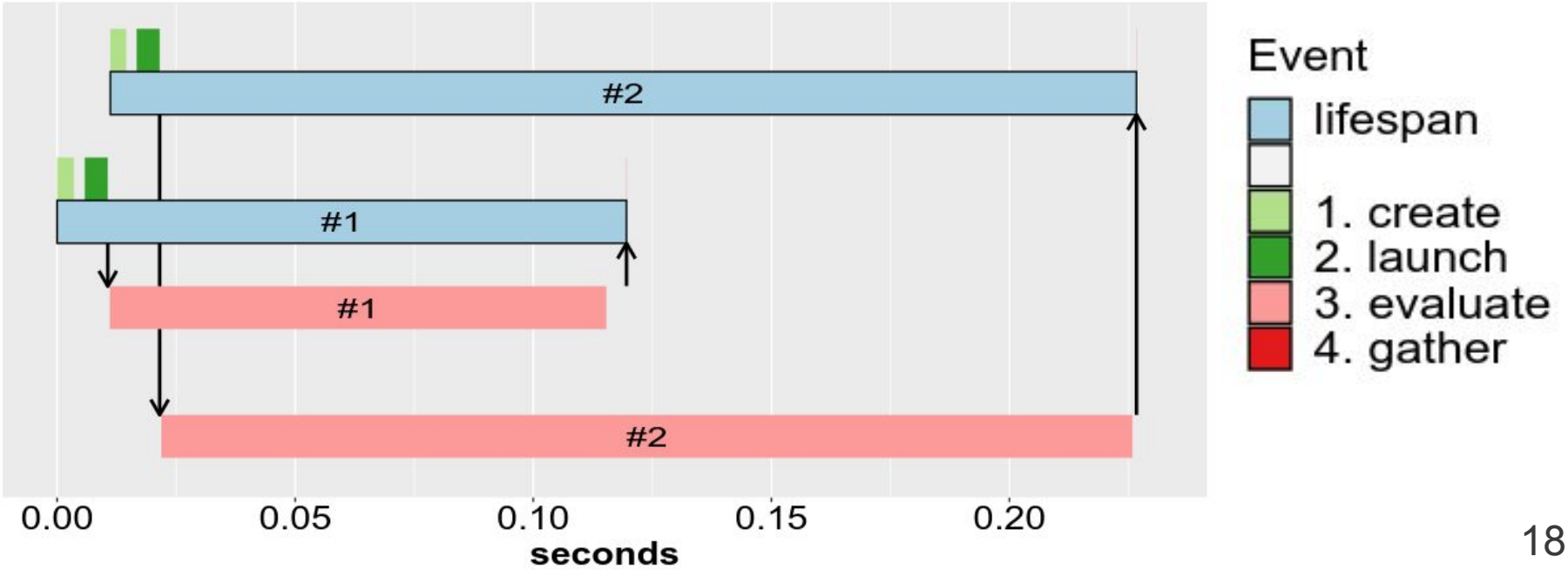
Profiling 2 futures with 1 worker

```
plan(cluster, workers = 1)
fs <- lapply(1:2, function(x) future(slow(x))
vs <- value(fs)
```



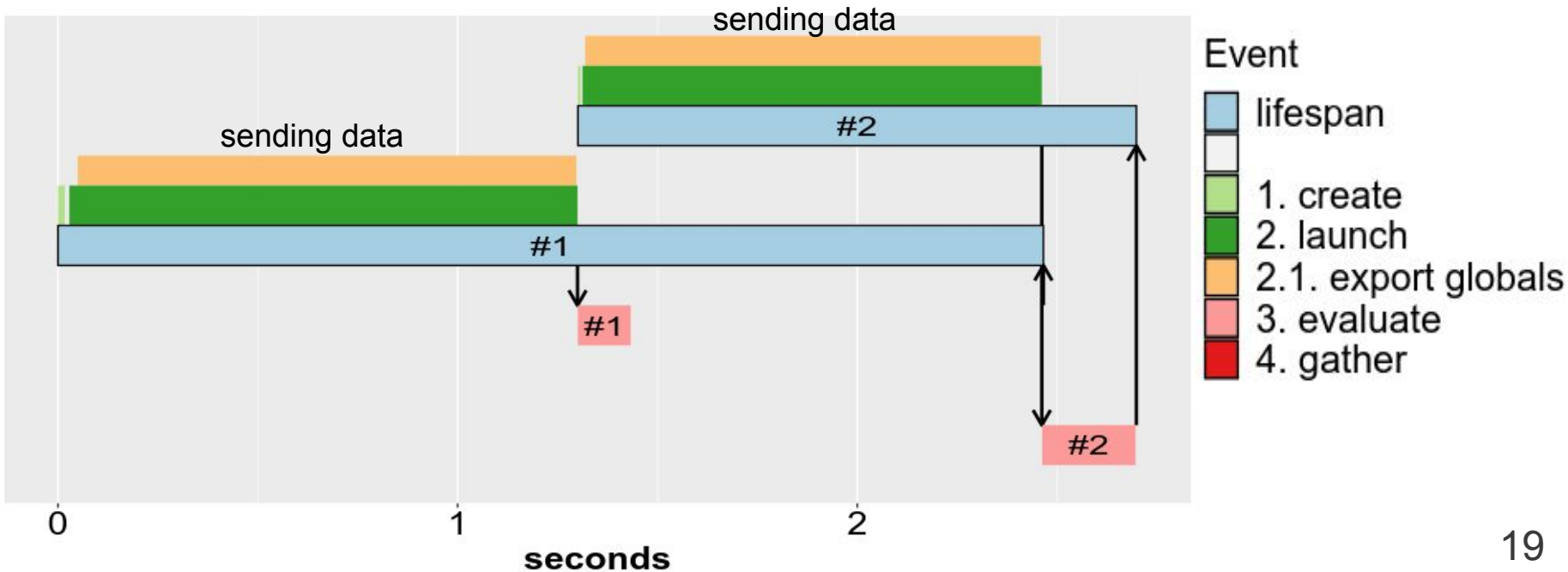
Profiling 2 futures with 2 workers

```
plan(cluster, workers = 2)
fs <- lapply(1:2, function(x) future(slow(x))
vs <- value(fs)
```



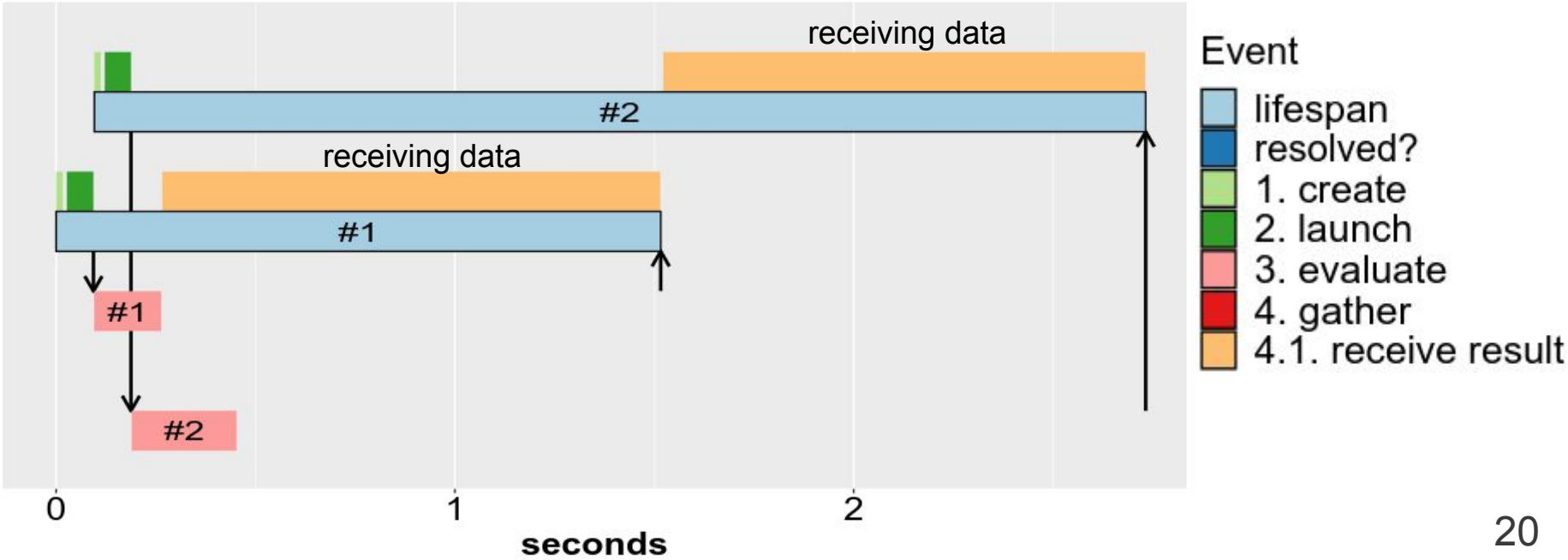
Exporting 800-MB object

```
plan(cluster, workers = 2); huge <- rnorm(100e6)
fs <- lapply(1:2, function(x) future(slow(x, huge))
vs <- value(fs)
```



Returning 800-MB object

```
plan(cluster, workers = 2)
fs <- lapply(1:2, function(x) future(slow_huge_value(x))
vs <- value(fs)
```



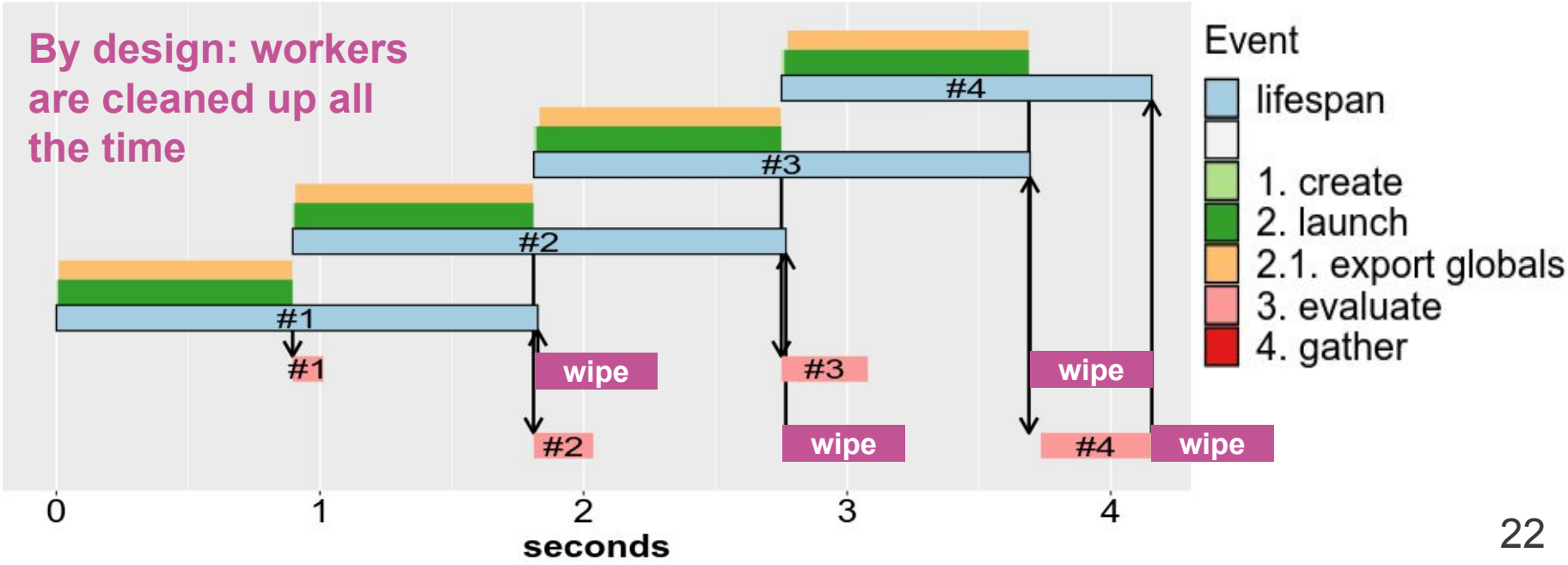
Profiling ⇒ Improving Futureverse

On the roadmap:

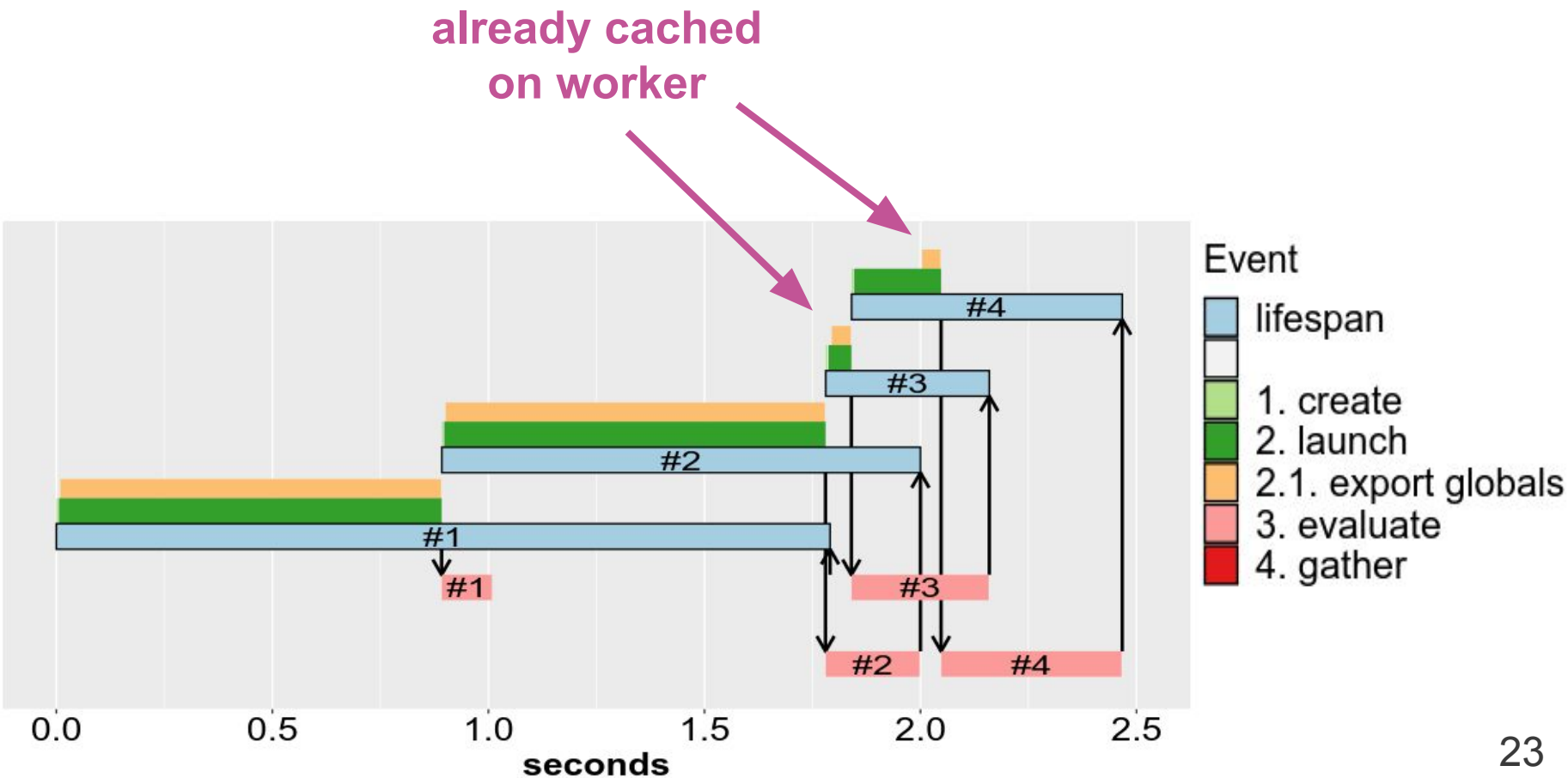
- caching of large globals on parallel workers
- caching of large globals in a central, shared cache

No cache: 800-MB export

```
plan(cluster, workers = 2); huge <- rnorm(100e6)
fs <- lapply(1:4, function(x) future(slow(x, huge))
vs <- value(fs)
```



With cache: 800-MB export



Thank you

More information:

- Website & blog: https://www.futureverse.org
- CRAN: https://cran.r-project.org/package=future
- GitHub: https://github.com/HenrikBengtsson/future
- Twitter: @HenrikBengtsson

Sponsored by:

- Essential Open-Source Software program of the Chan Zuckerberg Initiative (CZI EOSS #4)
- R Consortium ISC grant (past)