Using R for Web Data Collection at Scale

peter.meissner@virtual7.de
@peterlovesdata

virtual7

WIR GESTALTEN DIE DIGITALE ZUKUNFT DEUTSCHLANDS.

About me:

Social Scientist by Training

IT Guy by Design

R Textbook Author (Munzert et. al)

Long Time R User

R Package Author (wikipediatrend, robotstxt, tabit, …)

Consultant/Data Scientist @ virtual7

Public Sector Client
Customer Story
About Scaling R

Origin Story
{kafkaesque}
R package
https://github.com/petermeissner/kafkaesque

Mission: Web Scraping at Scale

Public Sector Agency

- High knowledge about data analytics.
- High demand and expertise for transparency and correctness of results.
- A lot of fresh ideas.
- But building and scaling software systems is not a core skill.

Mission: Web Scraping at Scale

Build and **scale** a software system for **collecting** specific **data from internet**

- Web search
- Web page rendering
- Data extraction

Mission: Web Scraping at Scale

… while using R

- *Contra R:*
  - Not known for speed.
  - Not known for scalability
- *Pro R:*
  - Elegant
  - Robust
  - Build in Data Management
  - **Well Known Quantity for Customer**

Let‘s have a look
at the problem.

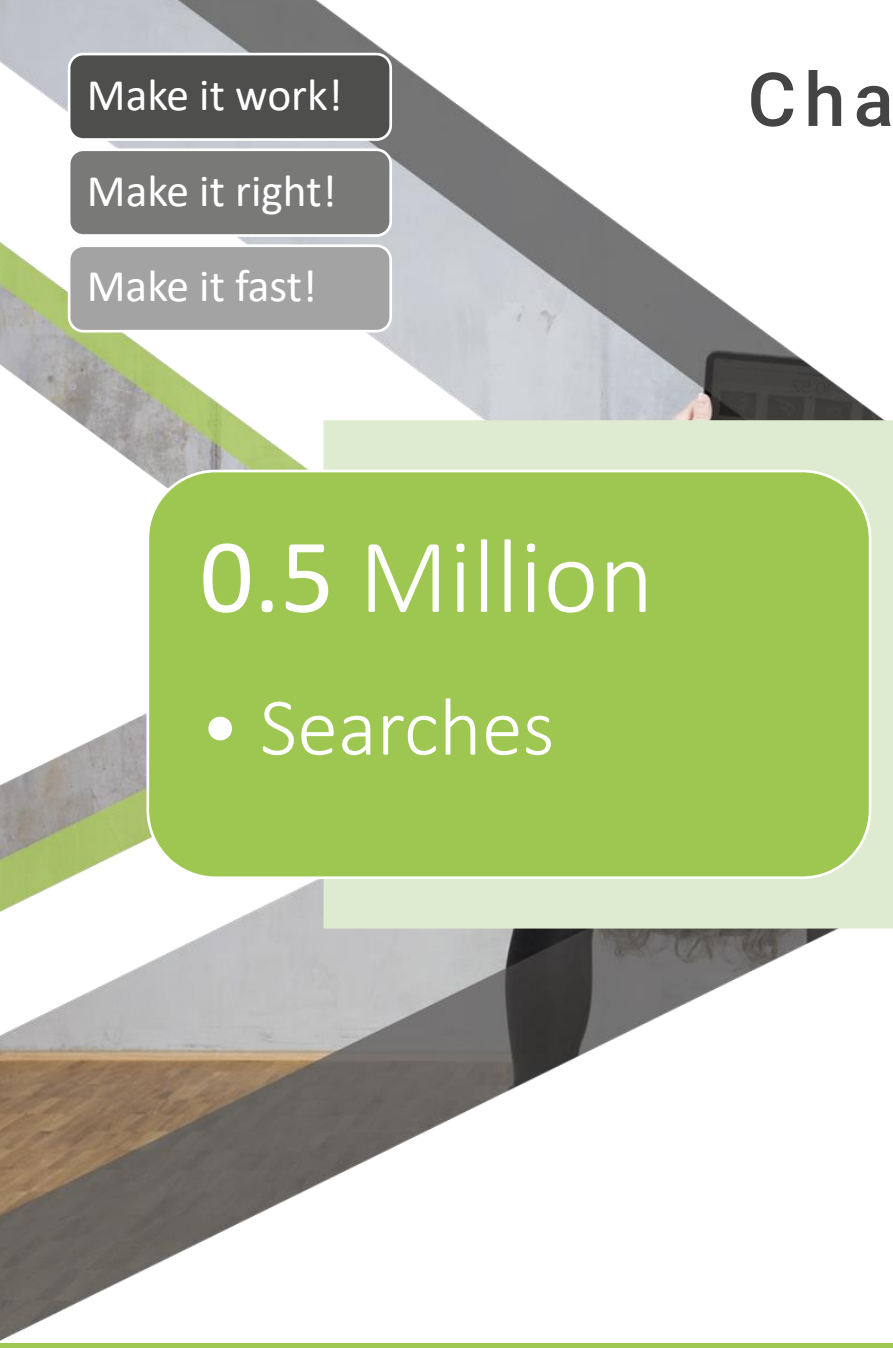
Make it work!

Make it right!

Make it fast!

Challenge: Bottlenecks

virtual7



0.5 Million

- Searches

7 Million

- Page renderings

7 Million

- Data extractions

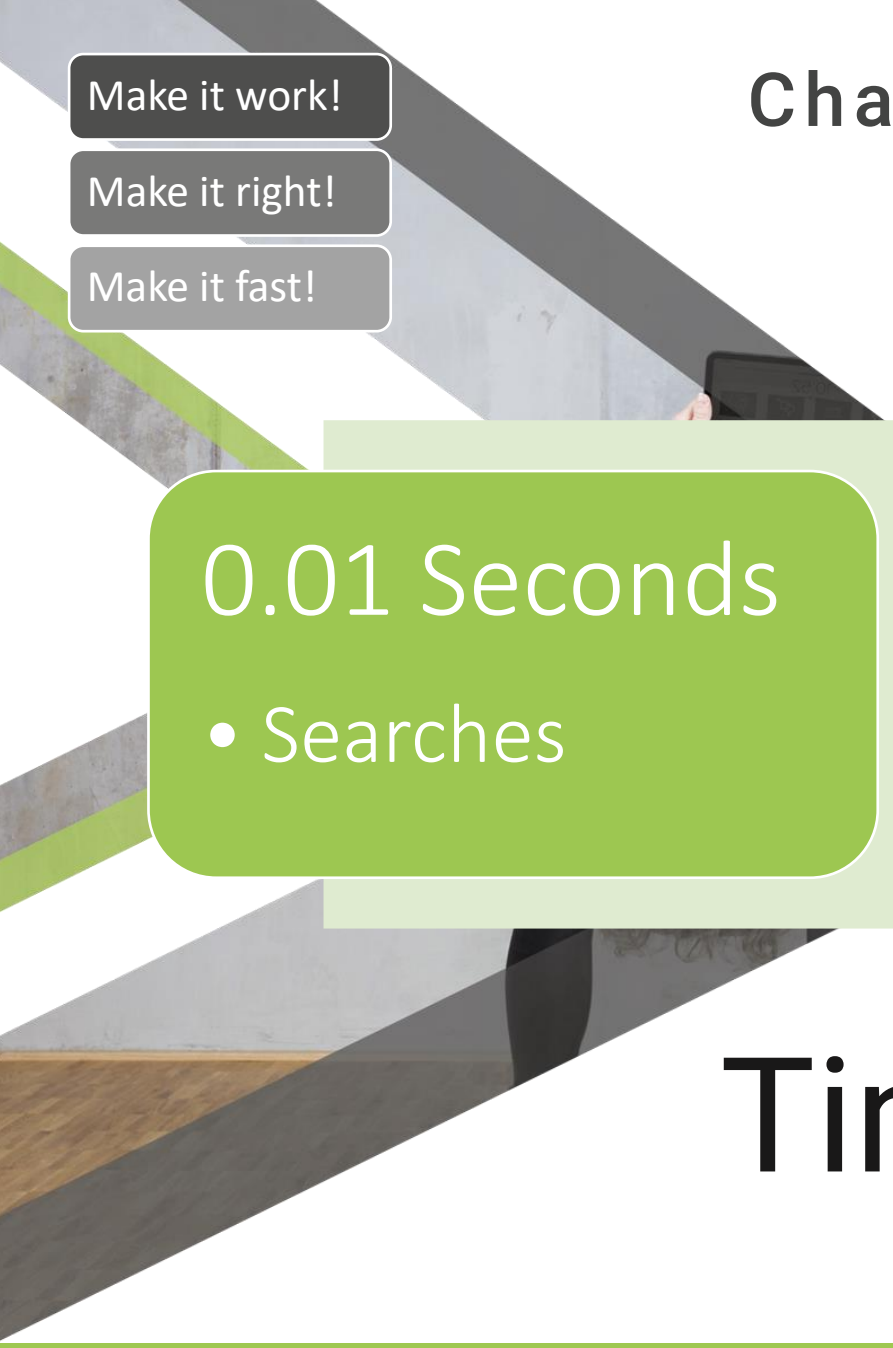
Tasks

Make it work!

Make it right!

Make it fast!

Challenge: Bottlenecks



0.01 Seconds

- Searches

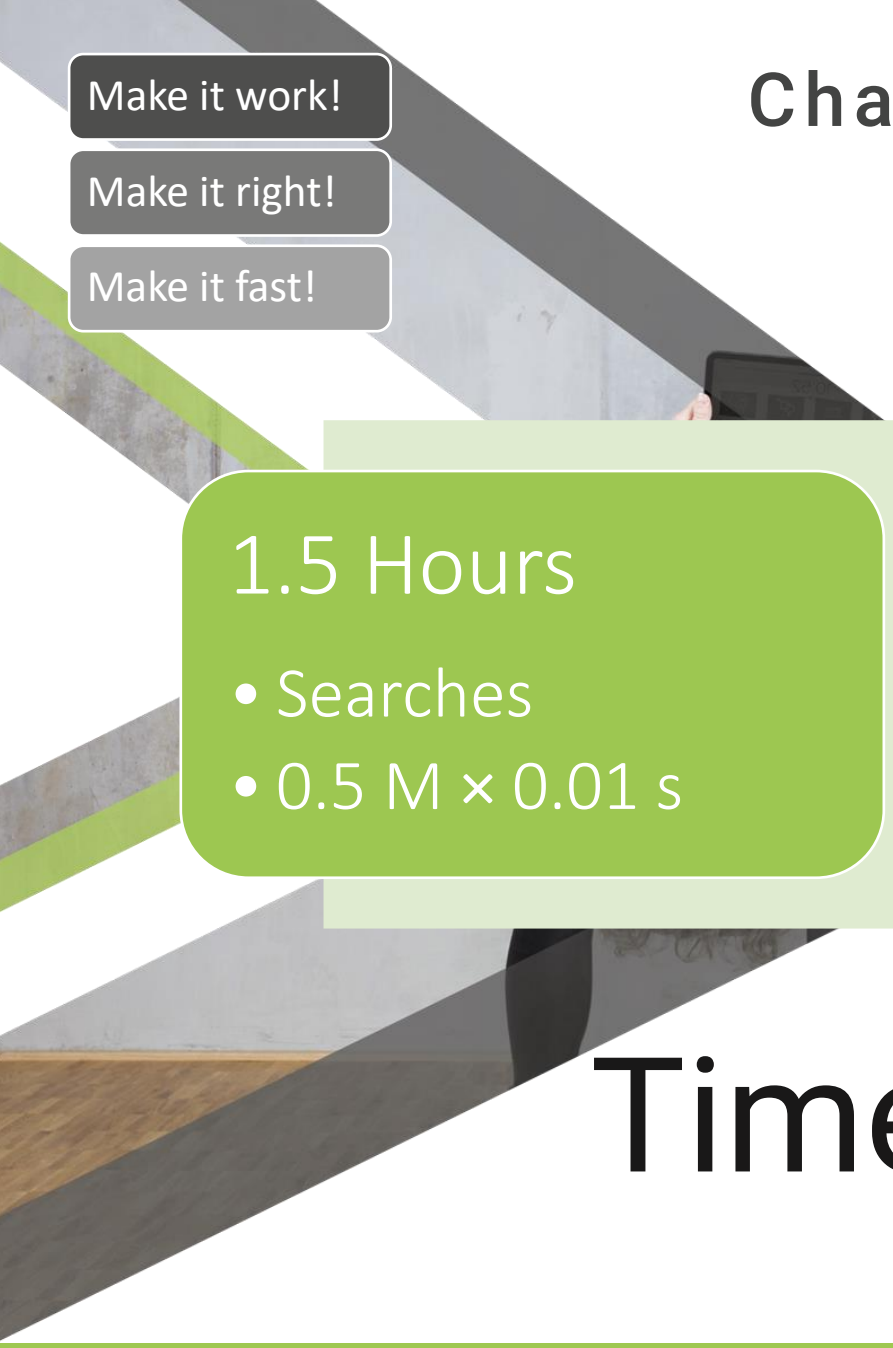
20 Seconds

- Page renderings

0.2 Seconds

- Data extractions

Time per Task



Make it work!

Make it right!

Make it fast!

Challenge: Bottlenecks

virtual7

1.5 Hours

- Searches
- 0.5 M × 0.01 s

4.5 Years

- **Page** renderings
- 7 M × **20** s

16 Days

- **Data** extractions
- 7 M × 0.2 s

Time for all Tasks

Given a naive approach with a single core machine.

Scaling
is almost always
a variation of a theme known as
„Divide and Conquer“

Scaling: Push Approach

Devide

And

Conquer

Push Approach

- Single Main application, aka a monolith, that …
- Keeps track of state (tasks, workers, …)
- Distributes work and coordinates workes

Pro

- High flexibility in how to distribute and coordinate work.
- Easier to get started.
- No extra bookkeeping needed.
- Might allow to use shortcuts to enhance efficiency.
- No extra software components needed.

Contra

- Coordination and distribution might become performance bottleneck.
- Keeping track of status of the application without fail is hard.
- Single point of failure.
- Knowledge about compute topology needed – e.g. server size, number, workload, …

Scaling: Pull Approach

Devide

And

Conquer

Pull Approach

- No single main application.
- Multiple independent workers.
- That only know how to do tasks.

Pro

- Highly scalable.
- Diverse topologies

Contra

- More planning involved upfront.
- Extra concept for bookkeeping of system state needed.

Pull Approach Based Scaling

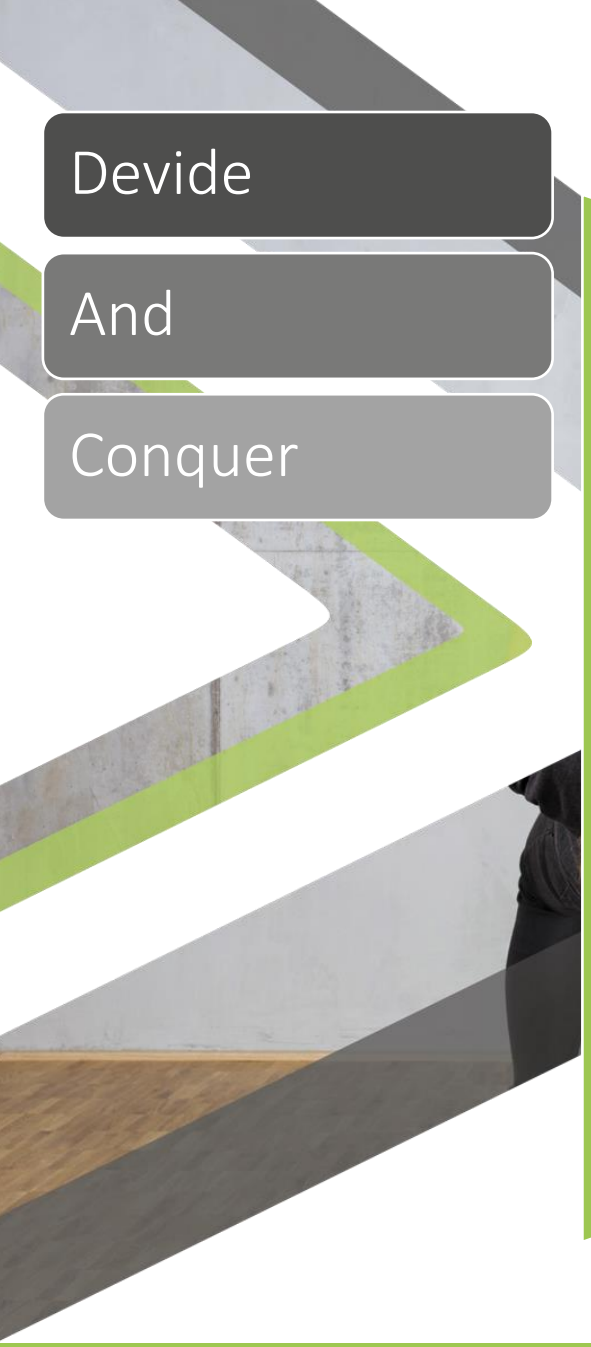
Scaling: Divide and Conquer

virtual7

Devide

And

Conquer



Tasks

- Search → Render → Extract → Search → Render → Extract → Search → Render → Extract → Search → Render → Extract → Search → Render → Extract → Search → Render → Extract

Divide Tasks Horizontal (real independence)

- Search → Render → Extract
- Search → Render → Extract
- Search → Render → Extract

Divide Tasks Vertical (temporal independence)

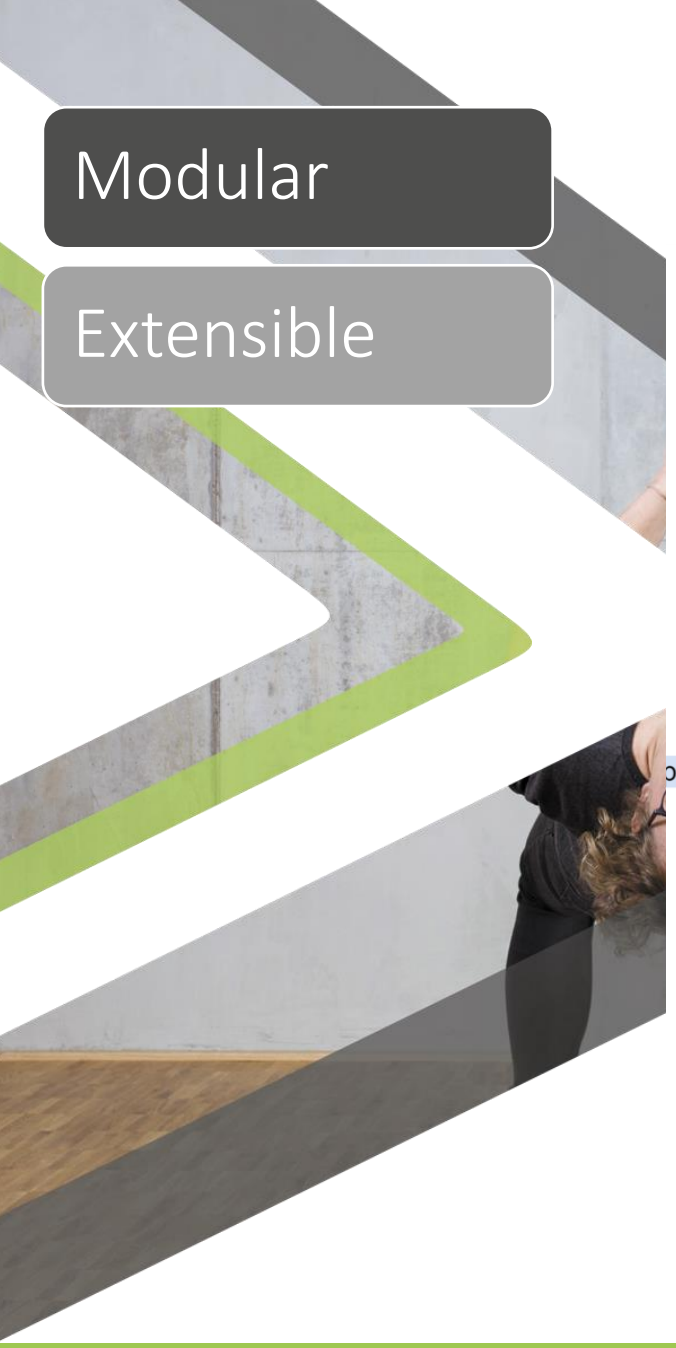
- Search, Search, Search
- Render, Render, Render
- Extract, Extract, Extract
- … + Bookkeeping

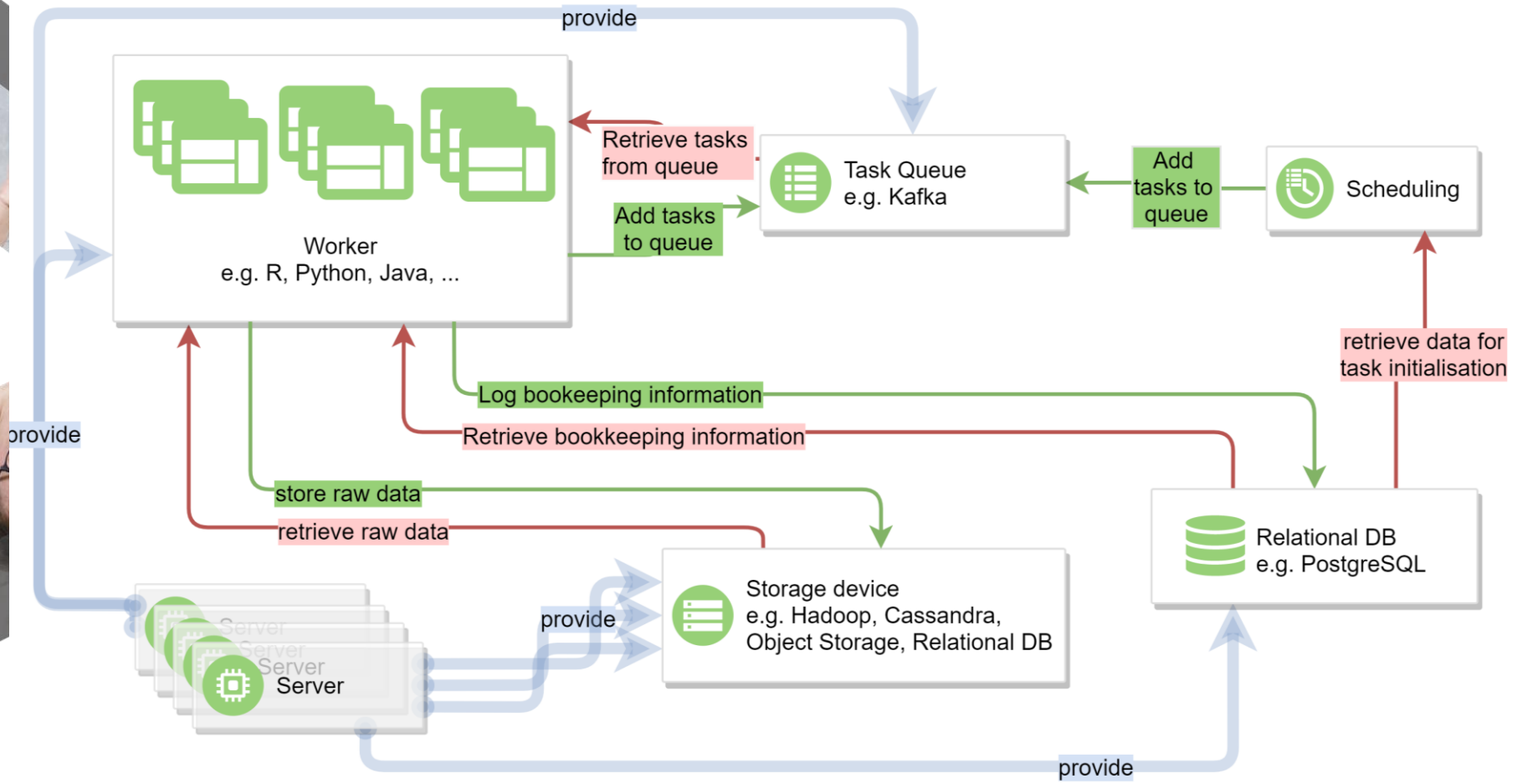
Architecture
&
Implementation

Architechture

Modular

Extensible





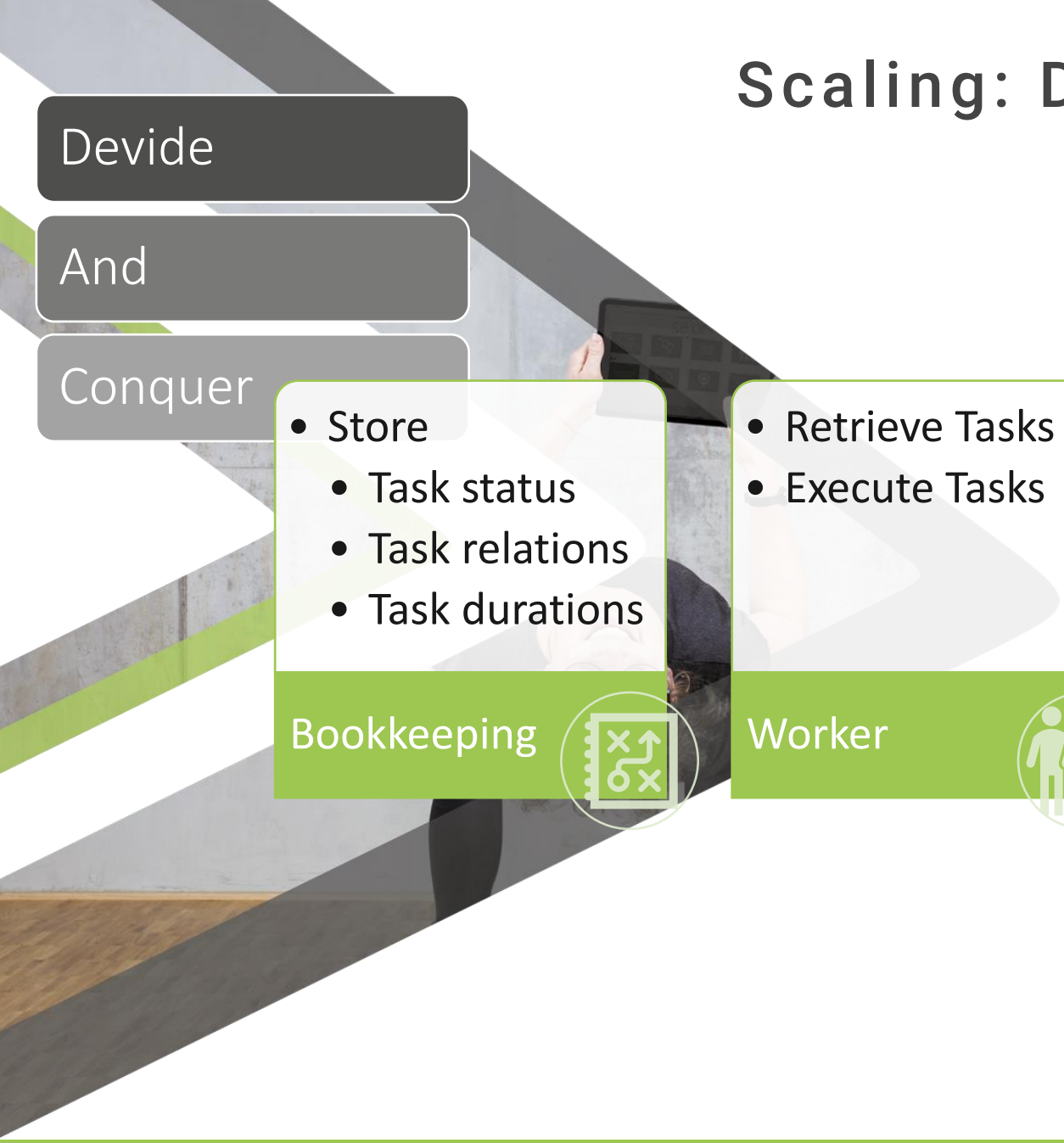
Scaling: Divide and Conquer

virtual7

Devide

And

Conquer



Bookkeeping

- Store
  - Task status
  - Task relations
  - Task durations

Worker

- Retrieve Tasks
- Execute Tasks

Task Queue

- Store
  - Tasks

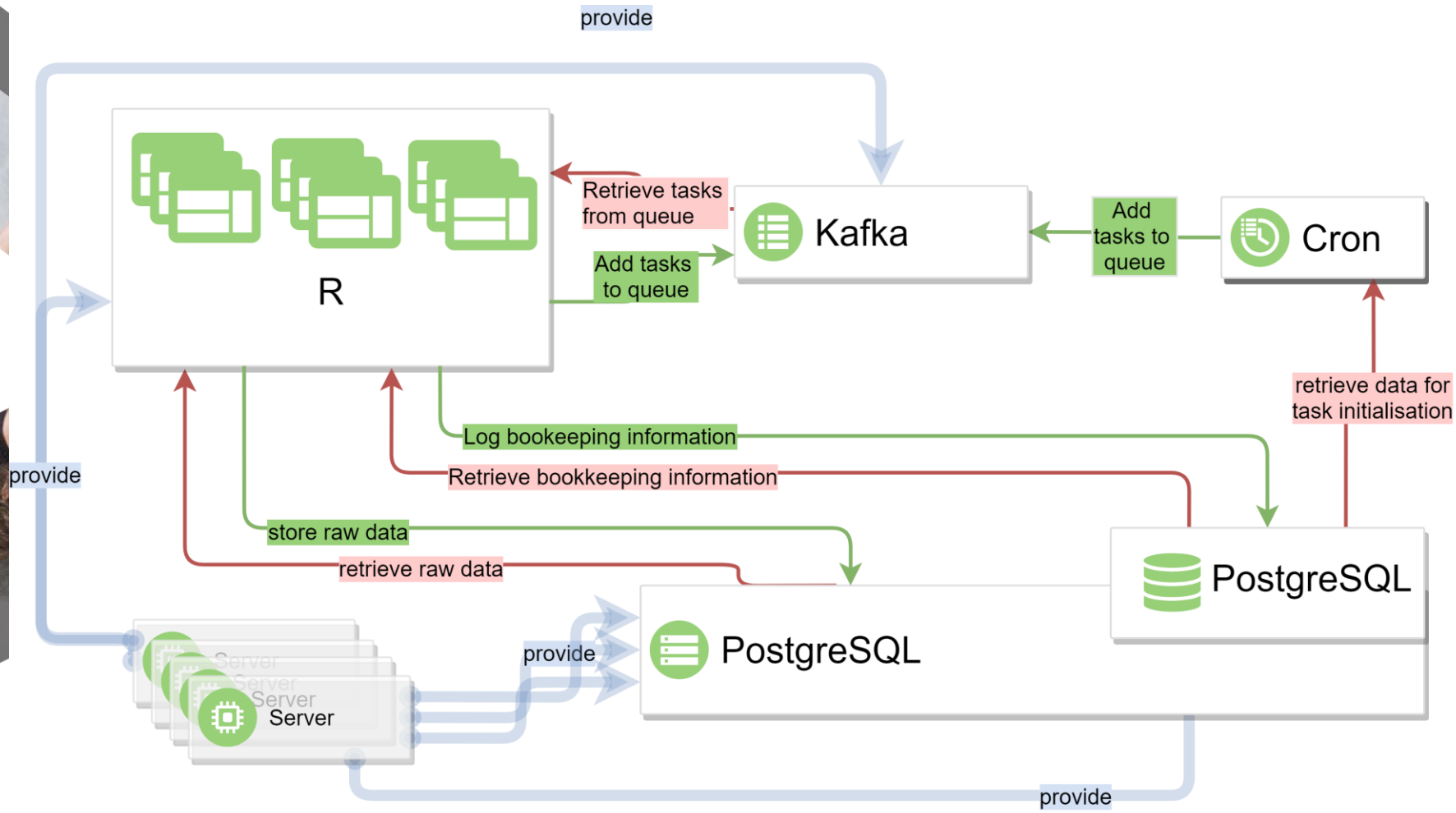
Storage

- Store
  - web pages

Architechture

Open Source

All the way down



Scaling?

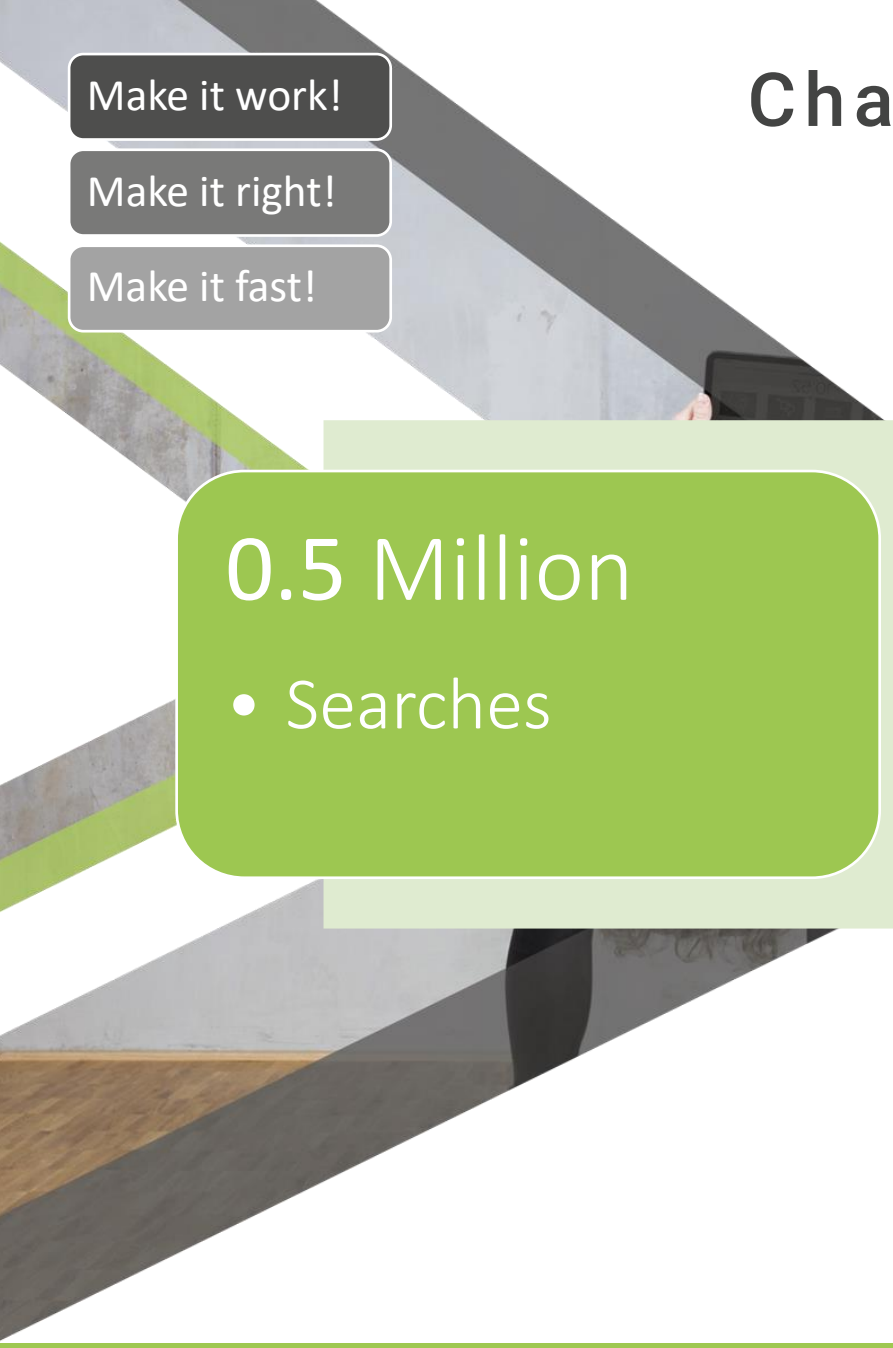
Make it work!

Make it right!

Make it fast!

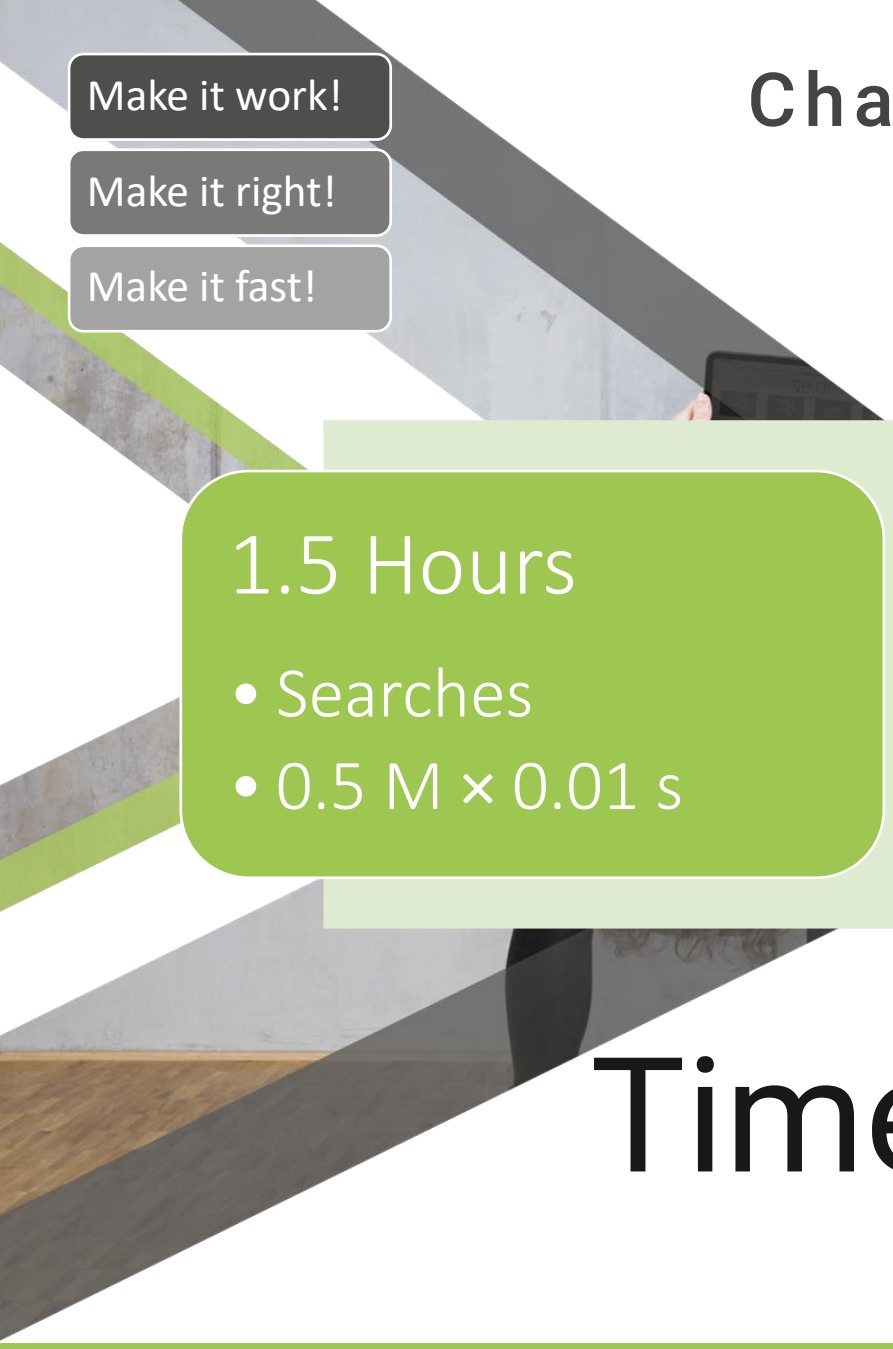
Challenge: Bottlenecks

virtual7



0.5 Million	**7 Million**	7 Million
• Searches	• **Page renderings**	• **Data** extractions

Tasks



Make it work!

Make it right!

Make it fast!

Challenge: Bottlenecks

virtual7

1.5 Hours

- Searches
- 0.5 M × 0.01 s

4.5 Years

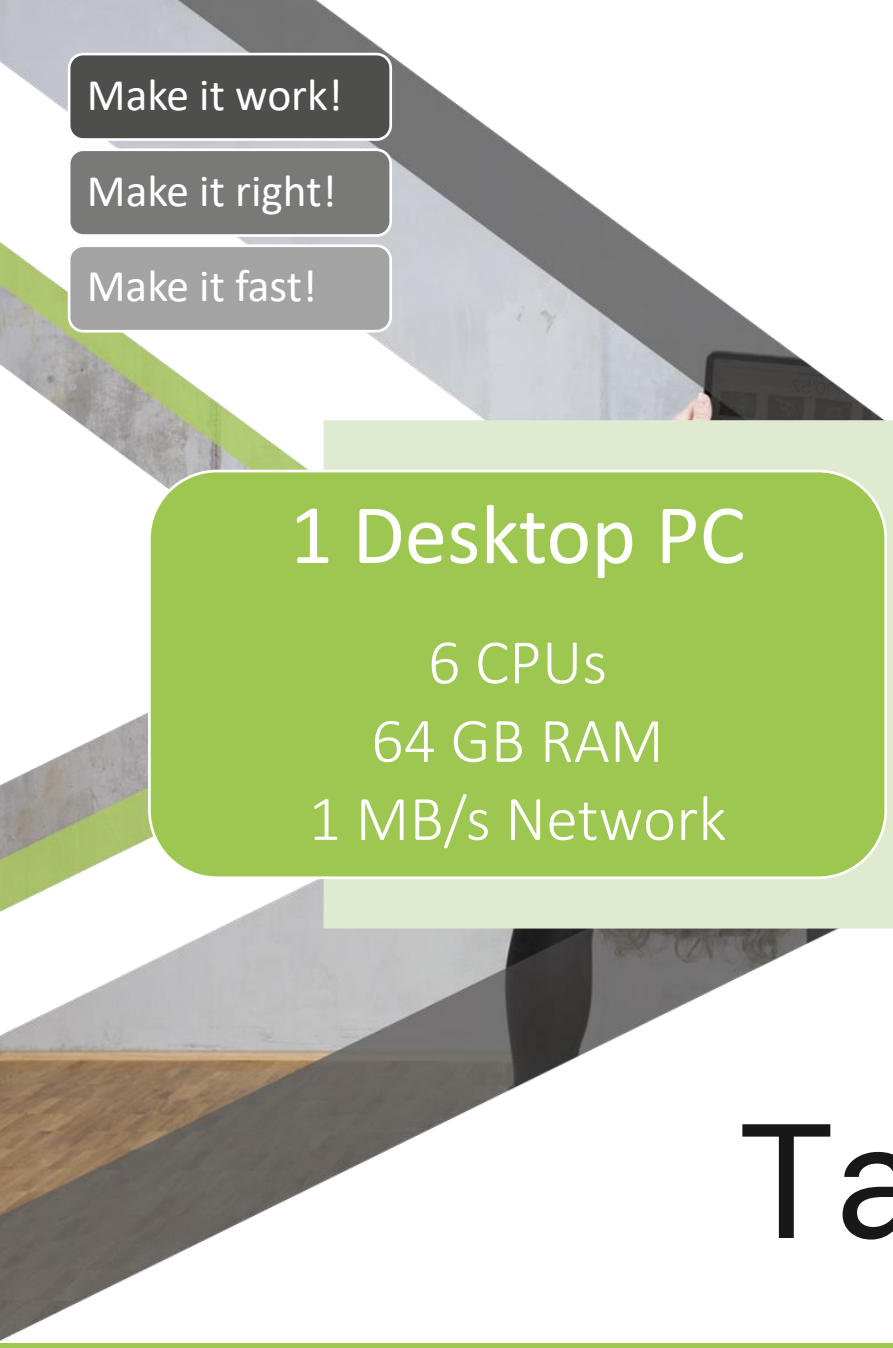
- **Page renderings**
- **7 M × 20 s**

16 Days

- Data extractions
- 7 M × 0.2 s

Time for all Tasks

Given a naive approach with a single core machine.



Make it work!

Make it right!

Make it fast!

Challenge: Done

virtual7

1 Desktop PC

6 CPUs
64 GB RAM
1 MB/s Network

Everything done within 55 days

For about 1’500 €uro once

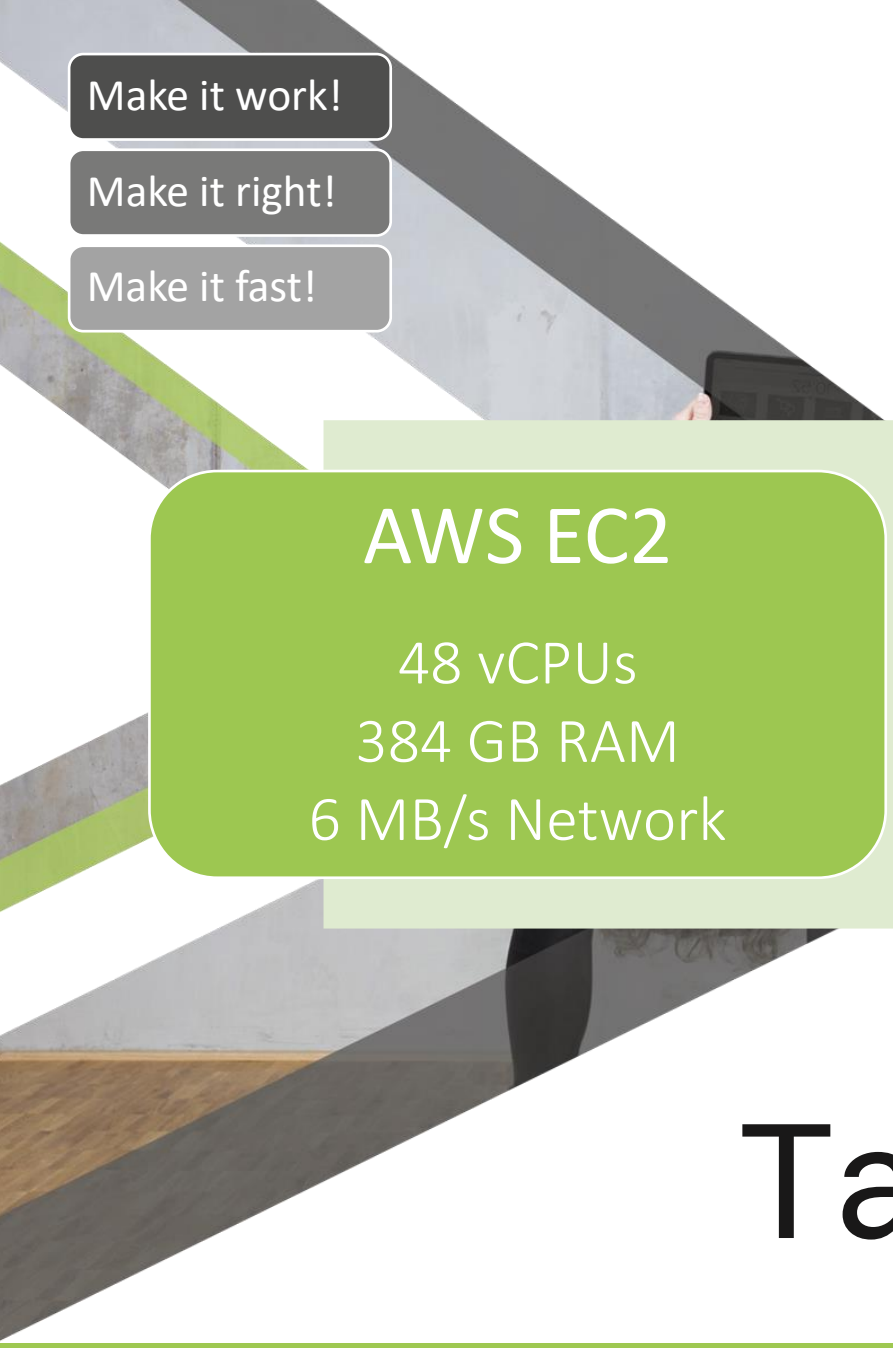
Task Completed

Make it work!

Make it right!

Make it fast!

Challenge: Done



AWS EC2

48 vCPUs
384 GB RAM
6 MB/s Network

Everything done within 10 days

For about 4’000 €uro per month

Task Completed

Make it work!

Make it right!

Make it fast!

Challenge: Done

virtual7

The Sky is the limit.

System can scale across clusters.

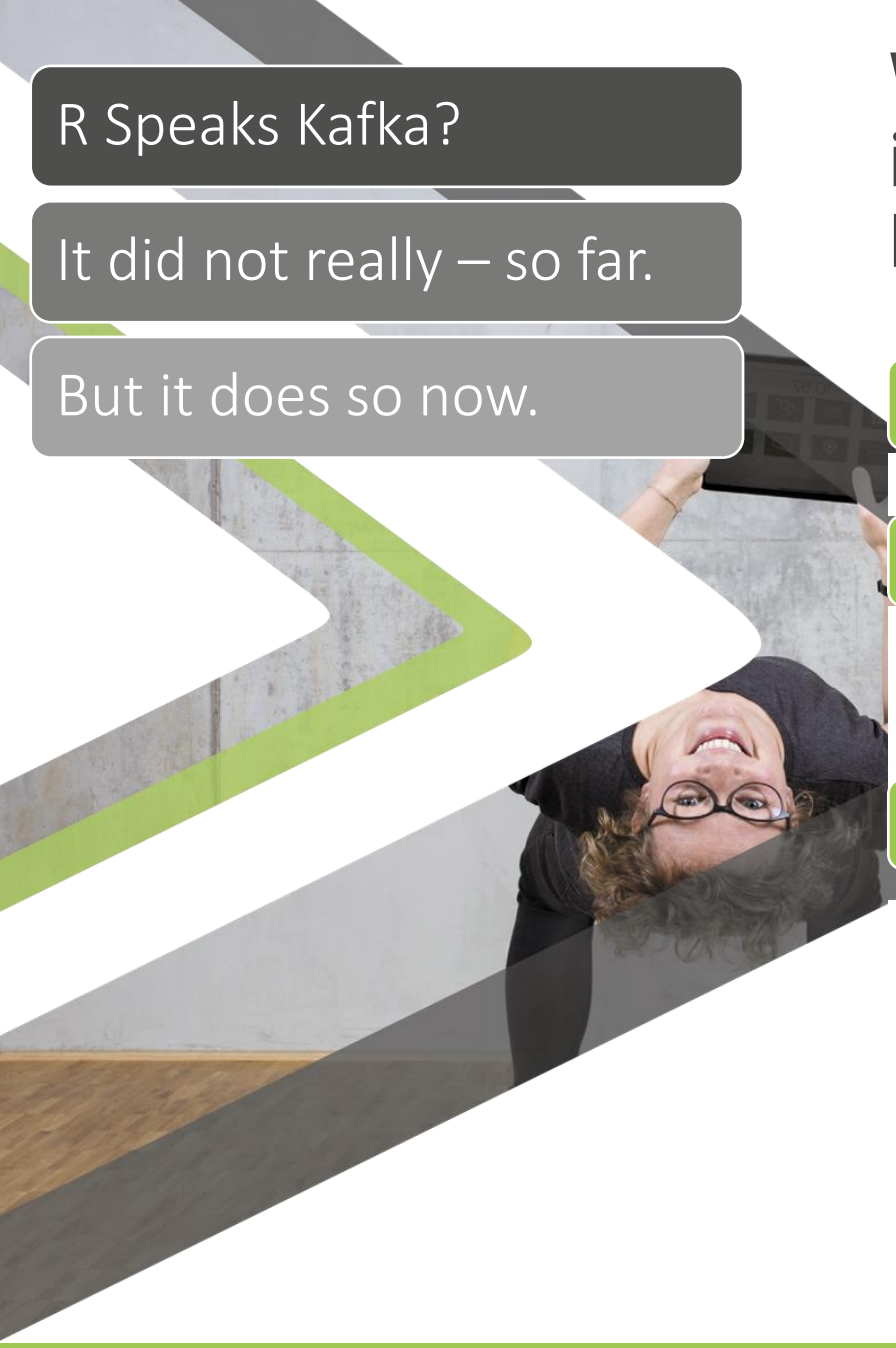
Task Completed

{kafkaesque}

R Speaks Kafka?

It did not really – so far.

But it does so now.



We open sourced an improved version of the R-Kafka bindings

Kafka is written in Java

{rJava} for R – Java bindings

- Data: R to Java
- Data: Java to R
- R: execute Java code within JVM

{kafkaesque}

- Our experience from customer work
- Give back to community
- Access to most of **admin**, **consumer** and **producer APIs**
- ***"User friendly Big Data for mere mortals"***
- https://github.com/petermeissner/kafkaesque

End

Questions & Comments

Technical Details

What are Workers?

- R scripts
- Can be spawned and stopped – scaled up/down while system is running
- Infinite loops that …
  - Ask for new tasks
  - Execute tasks
  - Encapsulate task execution into try-catch-blocks
  - Report start, end and error status to book keeping

Why use Kafka?

- In fact propably any message queue that can run as a standalone service could have done.
- Kafka has some nice properties though.
  - Scalable.
  - Message retention.
  - Kafka is very much build around the idea that consumers pull messages which in turn is very much in line with our own model of scalability.
- We had a rudimentary package that proofed that R-bindings for Kafka worked at some earlier point in time.

Why use RelationalDB for Bookkeeping?

- Via indexing and joins it‘s easy to get various information or do analytics.

Why use PostgreSQL?

- Open Source
- well known
- performant
- industry standard

Are there any reasons not to use R?

- No.
- Except there are other languages which would have come with Kafka bindings already available.
- Otherwise R does not pose any problem here:
  - Its production ready.
  - Predictable.
  - Easy to package and deploy.
  - Performance: The main time consuming task is **web page rendering** that is done not within R but within **headless browsers.**
  - Good data manage capabilities.
  - Good data base conectivity.