# From ~10 Users To ~10 Million In ~6 Weeks

Adam Lev // Tamar Labs // 2017

# A Fair Warning

- Big Data
- Microservices
- Stream Processing
- Real-Time
- Scalabuility

# A Fair Warning

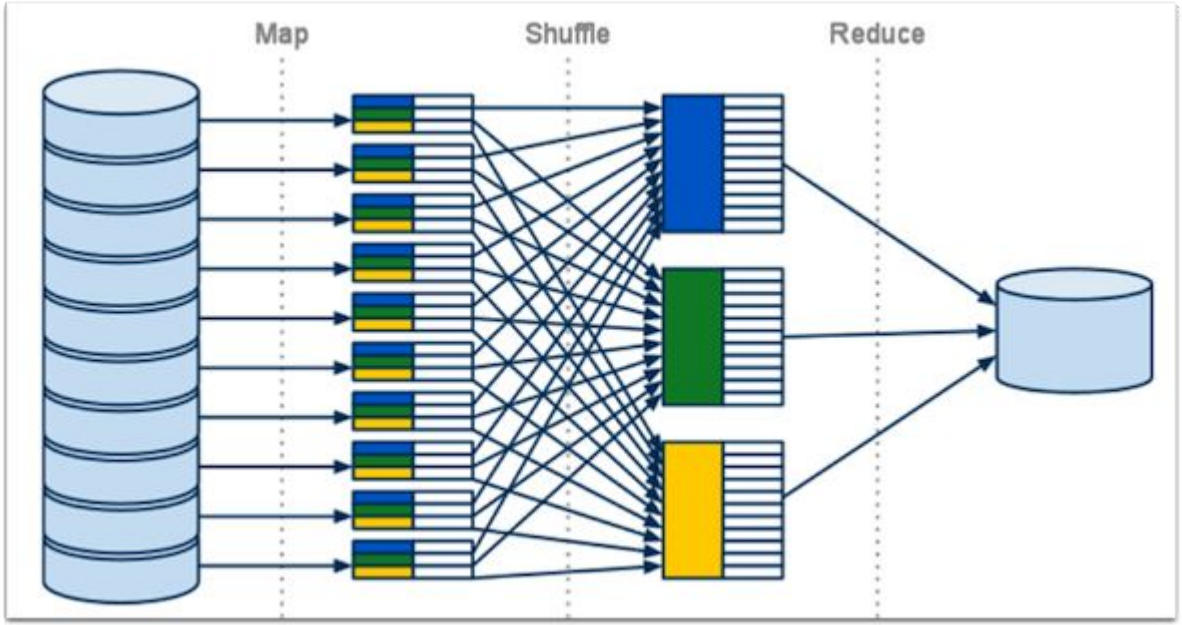
- Big Data
- Microservices
- Stream Processing
- Real-Time
- Scalabuility
- “Redis is fast”

# Big Data

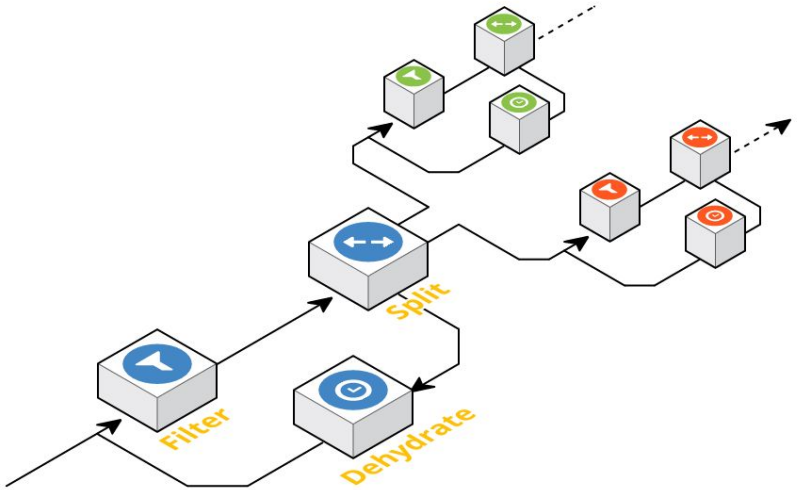
# ~~Big~~ Fast Data

* AKA Stream Processing

# This is Big Data



# This is stream processing



# minimizing state dependency

# Global State is bad

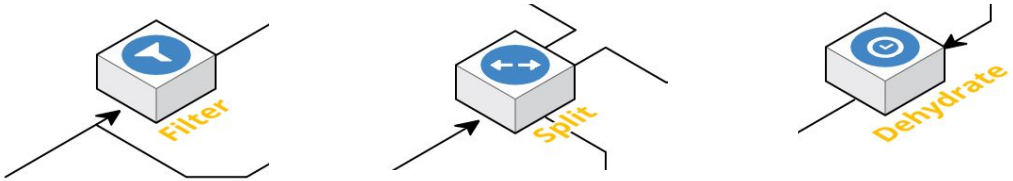
- Inherently requires side-effects
- Optimizes for design simplicity, instead of code simplicity
- Totally unscalable in real-time and near real-time systems.

# Real-time, Emergent Relevancy & Contextual Completeness

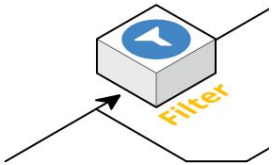
- Eventually you can figure out the answer to any request, but it might take some time. You can’t tell how long it would take, but you can cap it.
- You either know an answer to a request **now**, or not.
- Anything else does not matter.

# thinking
# in
# critical sections

# Handling Uncertainty

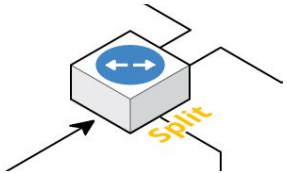


# The Filter



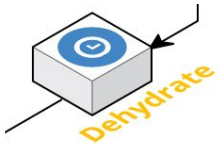

All it knows is what is relevant beyond this point

# The Splitter



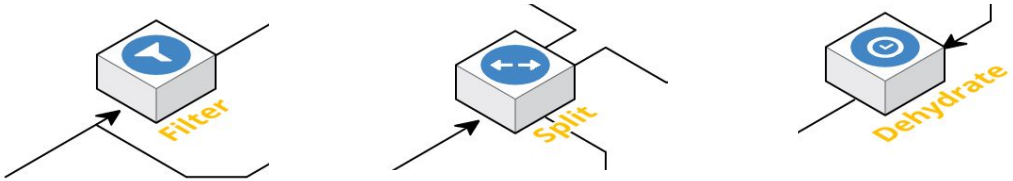

All it knows is where things should go

# The Dehydrator



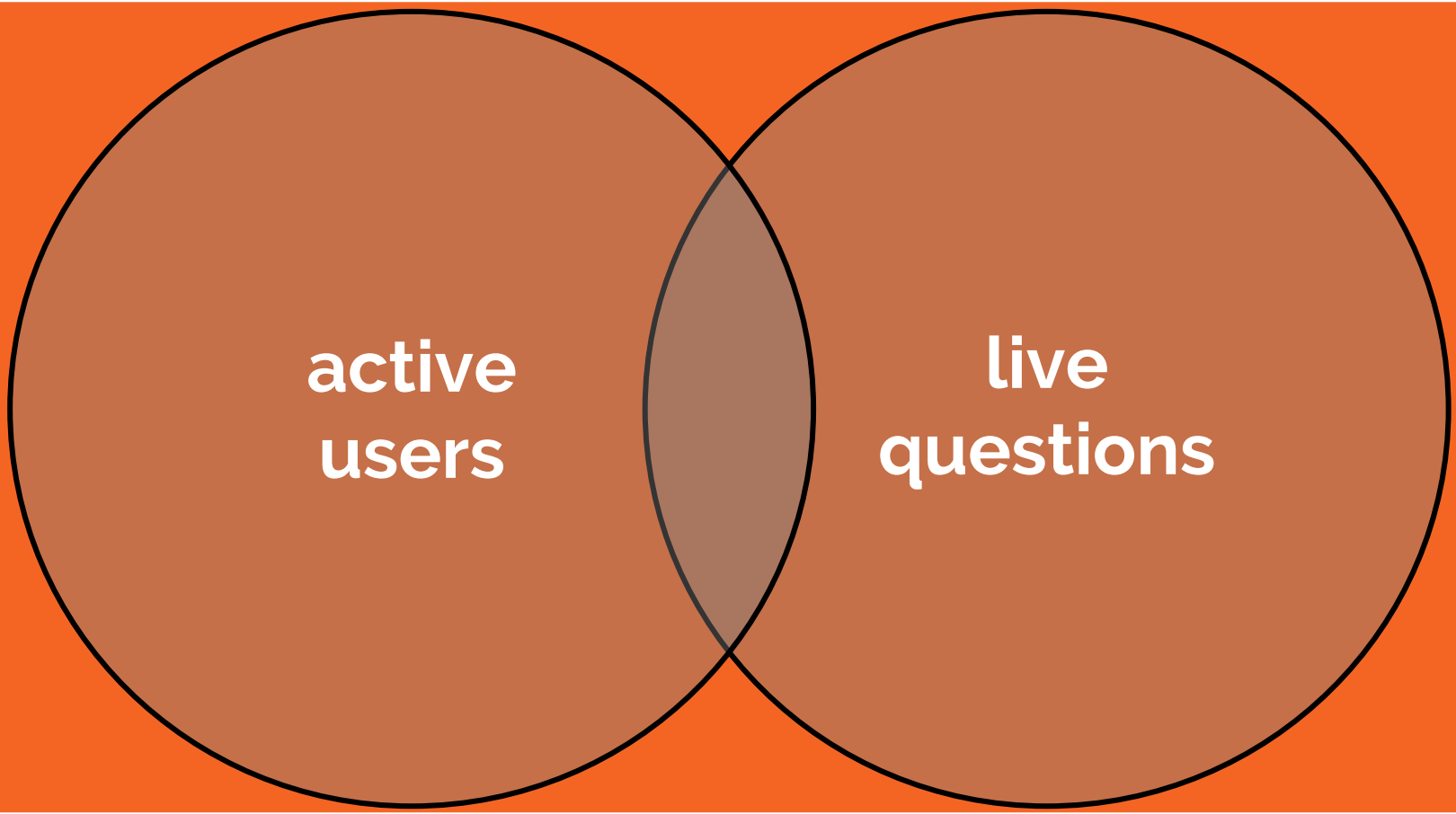

All it knows is when should we retry

# Handling Uncertainty



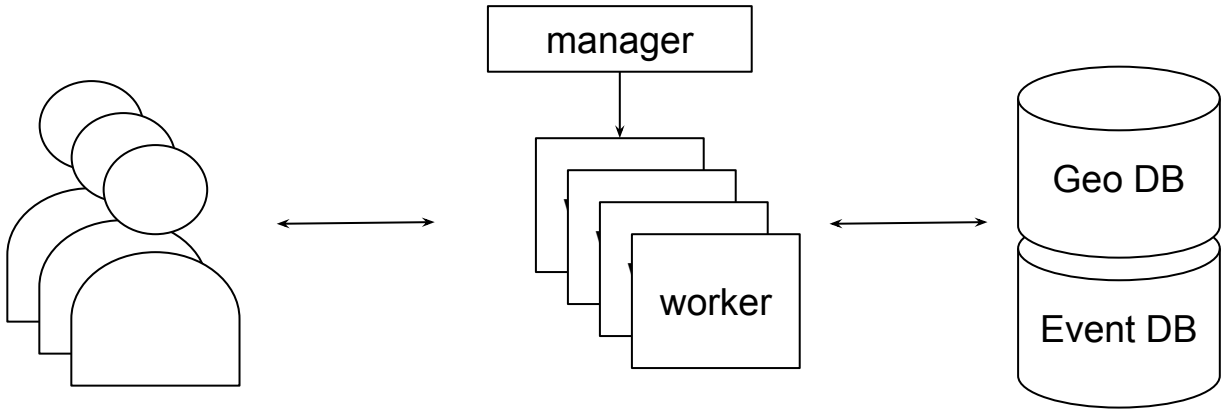
# The Tijuana Problem - a case study

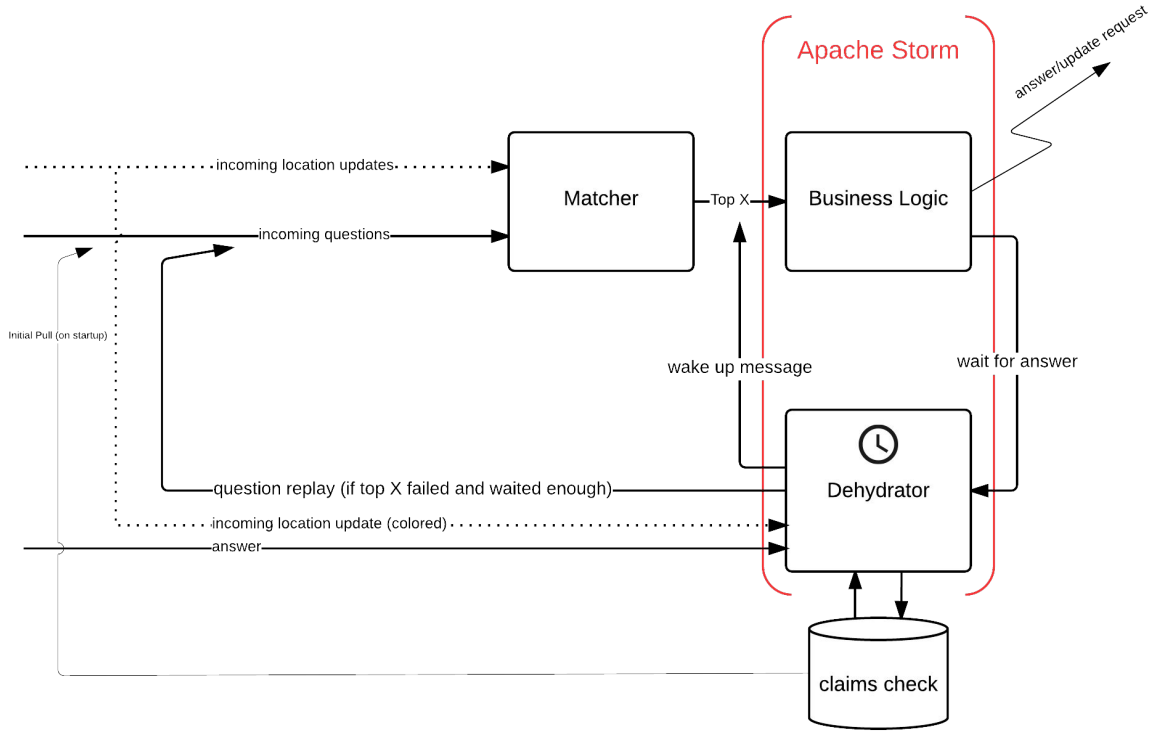
Real time and retro matching of questions and users based on geolocation (as polygon)

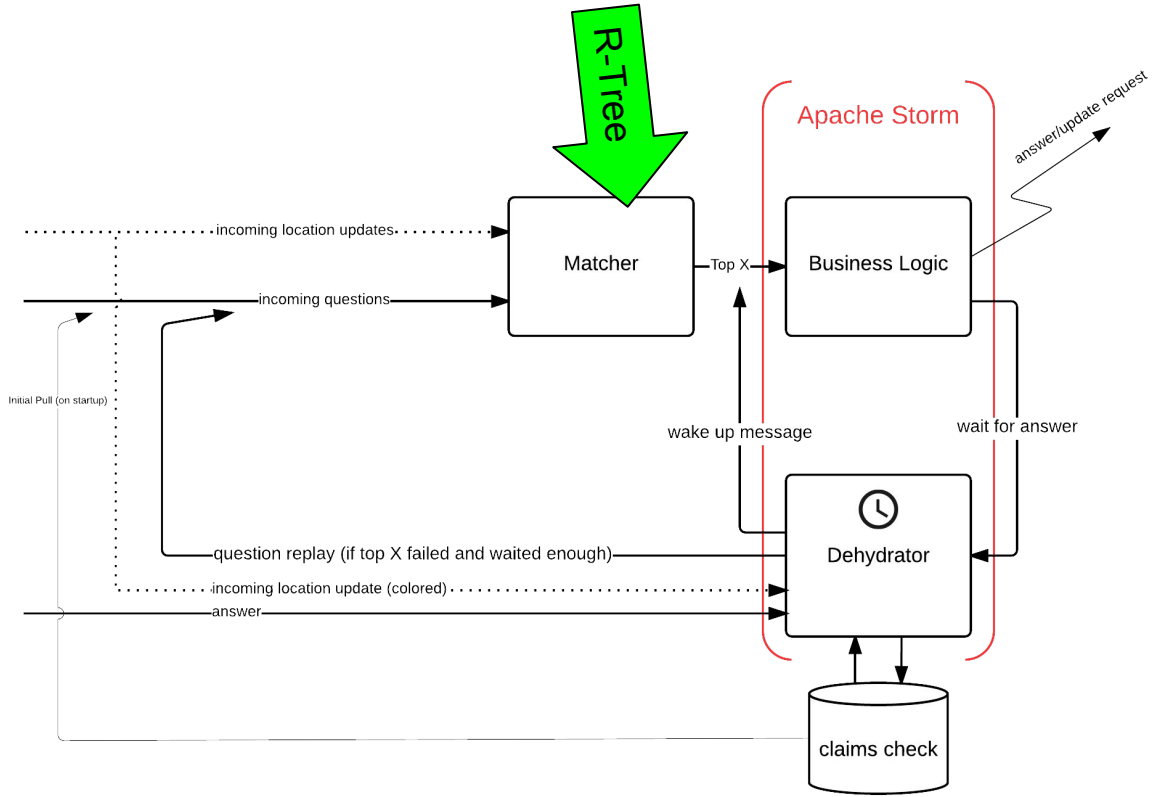

active
users
live
questions

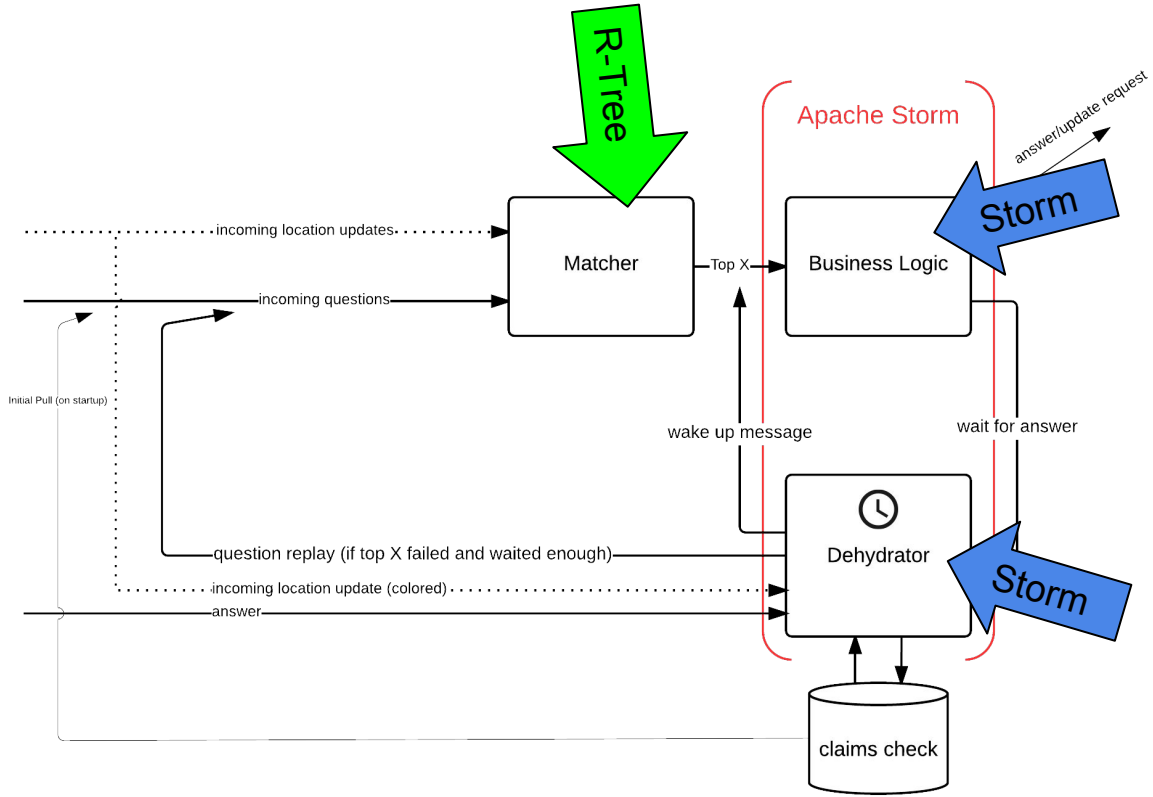
## The Real Problem

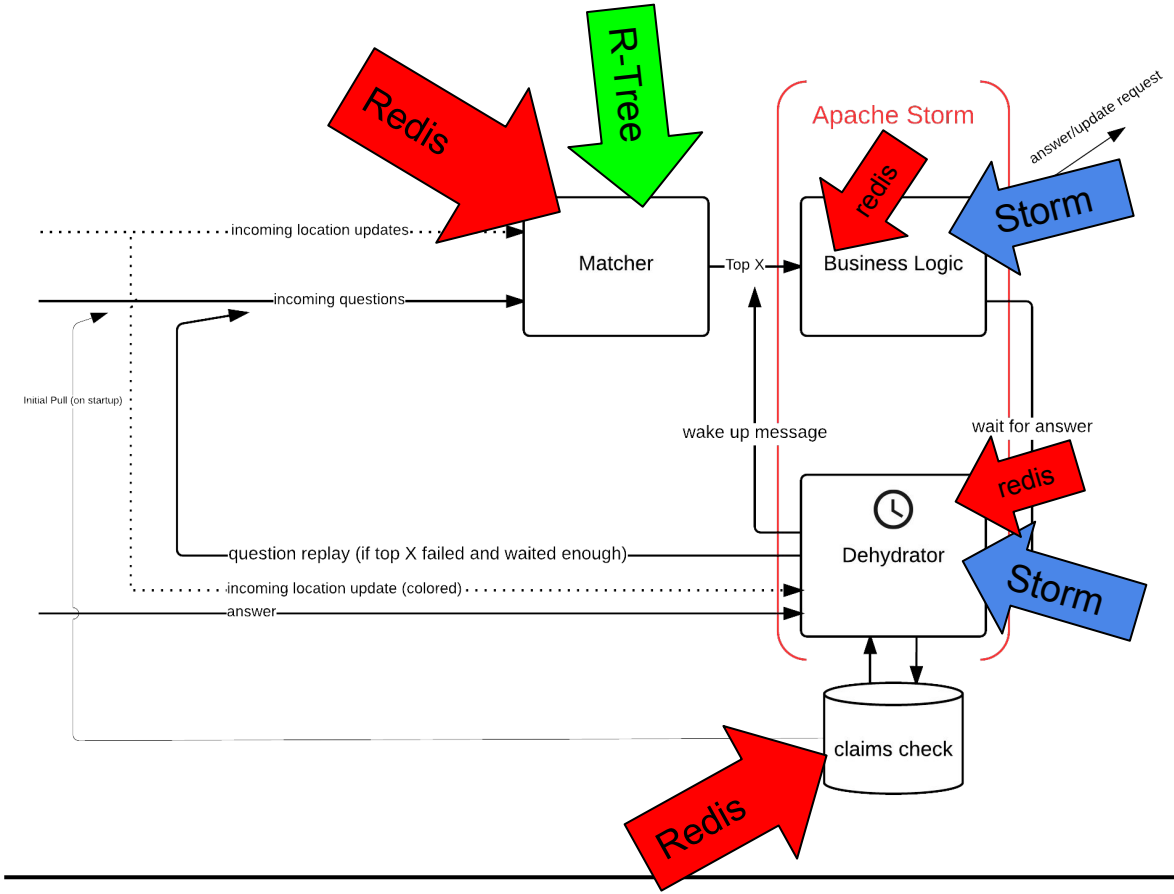
> 1000 users


manager
worker
Geo DB
Event DB


Apache Storm
answer/update request
incoming location updates
incoming questions
Matcher
Top X
Business Logic
Initial Pull (on startup)
wake up message
wait for answer
Dehydrator
question replay (if top X failed and waited enough)
incoming location update (colored)
answer
claims check


R-Tree
Apache Storm
answer/update request
incoming location updates
incoming questions
Matcher
Top X
Business Logic
Initial Pull (on startup)
wake up message
wait for answer
Dehydrator
question replay (if top X failed and waited enough)
incoming location update (colored)
answer
claims check


R-Tree
Apache Storm
answer/update request
Storm
incoming location updates
incoming questions
Matcher
Top X
Business Logic
Initial Pull (on startup)
wake up message
wait for answer
Dehydrator
question replay (if top X failed and waited enough)
incoming location update (colored)
answer
Storm
claims check


Redis
R-Tree
Apache Storm
answer/update request
redis
Storm
incoming location updates
Matcher
Top X
Business Logic
incoming questions
Initial Pull (on startup)
wake up message
wait for answer
redis
Dehydrator
Storm
question replay (if top X failed and waited enough)
incoming location update (colored)
answer
claims check
Redis

## The meat on the bone

# >20,000,000 concurrent active user events

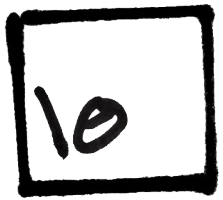
# What we learned from that project

- Soft & Fast ≫ Hard & Slow
- Good Enough ≫ Perfect
- Dehydration ≫ Computation *

* and is is the best HPC cheat of all times

# More Resources

- Dehydrator Module: https://tamarlabs.github.io/ReDe
- The Article "Fast Data": https://goo.gl/DDFFPO
- The Null Terminator Blog: www.nullterminator.org




null
terminator
blog

NullTerminatorBlog.com

adam@tamarlabs.com