Serverless Computation with OpenLambda

Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani†, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

† Unaffiliated

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

(VMs)

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

(VMs)

compute is evolving

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

(Containers)

compute is evolving

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

(Lambdas)

compute is evolving

Web development in the cloud

CDN: static content (e.g., JavaScript)

Compute: dynamic logic (e.g., Python)

Storage: application data

(Lambdas)

prior to the Lambda model, cloud compute was neither elastic nor pay-as-you-go

Outline

Evolution of compute

Non-conventional virtualization

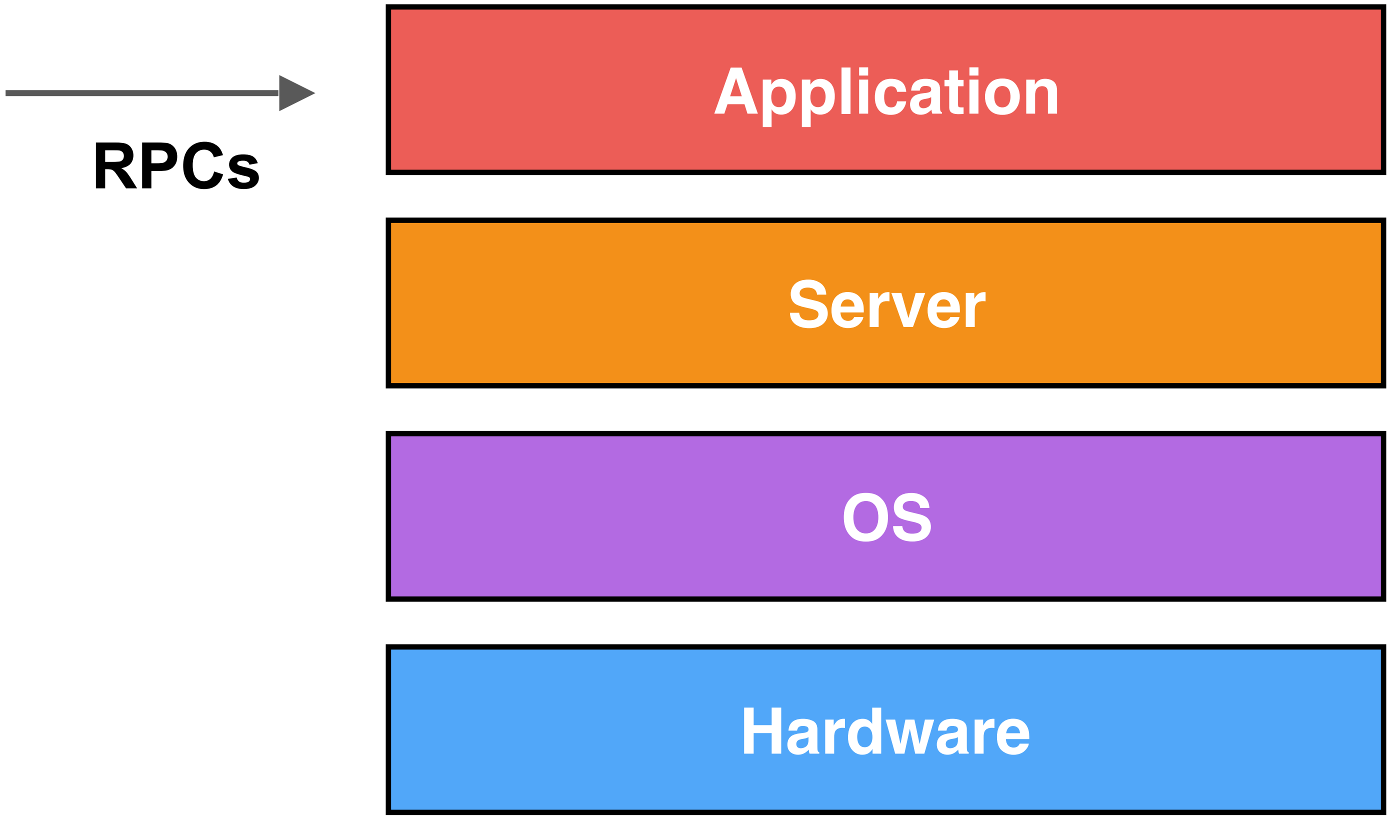
Lambda model

Why OpenLambda?

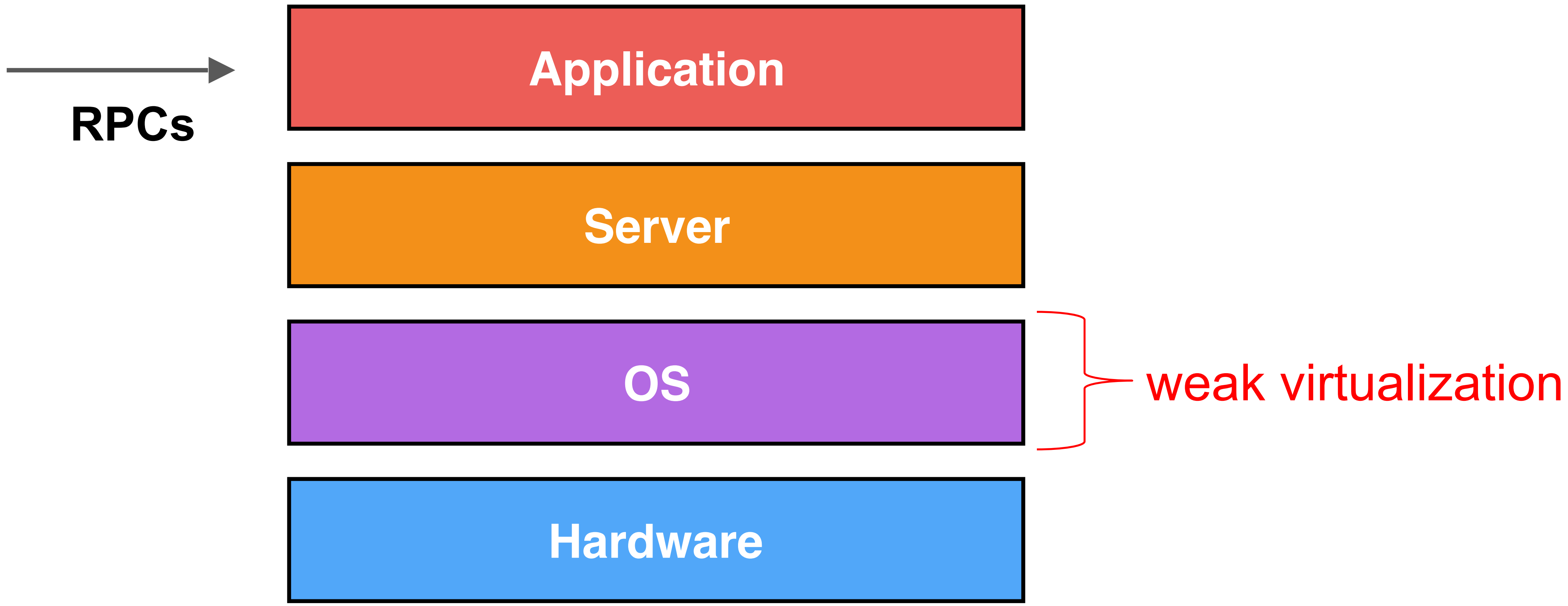
Conclusion

How to virtualize compute?

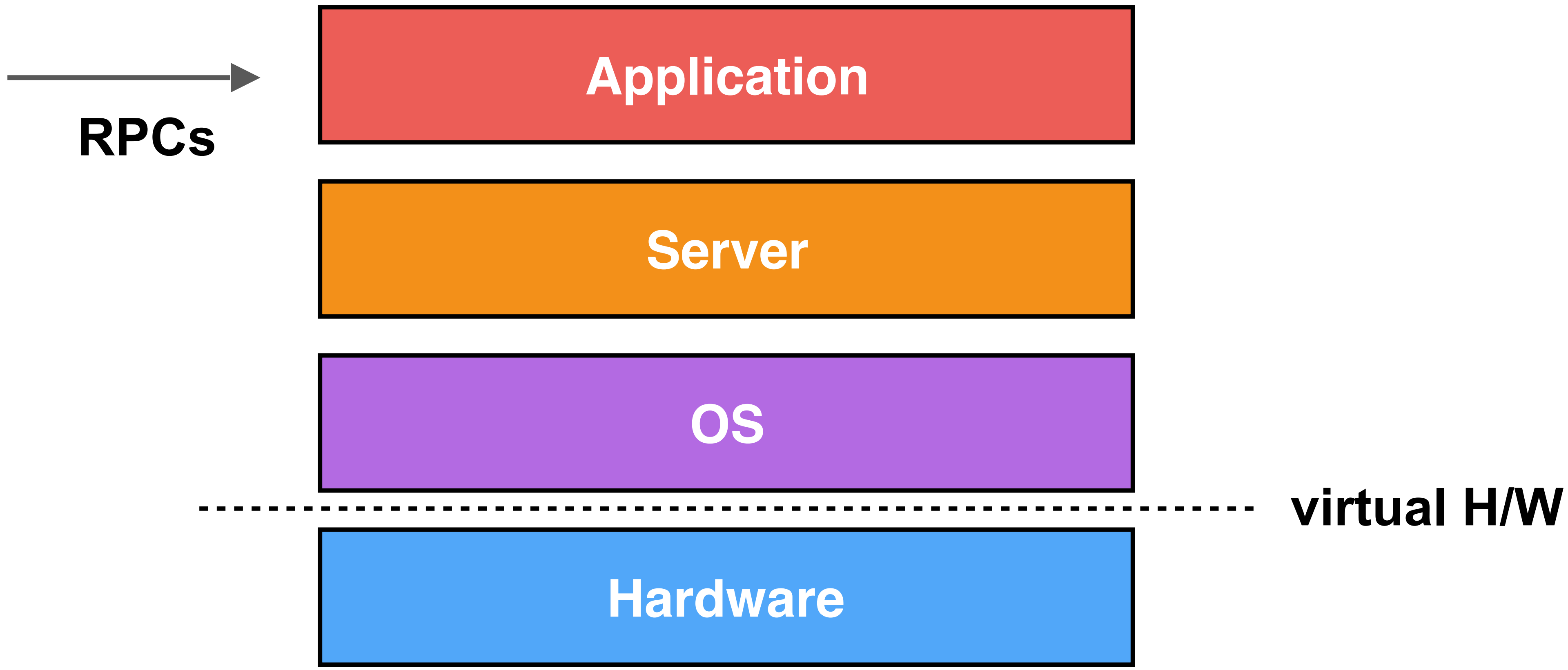
Classic web stack



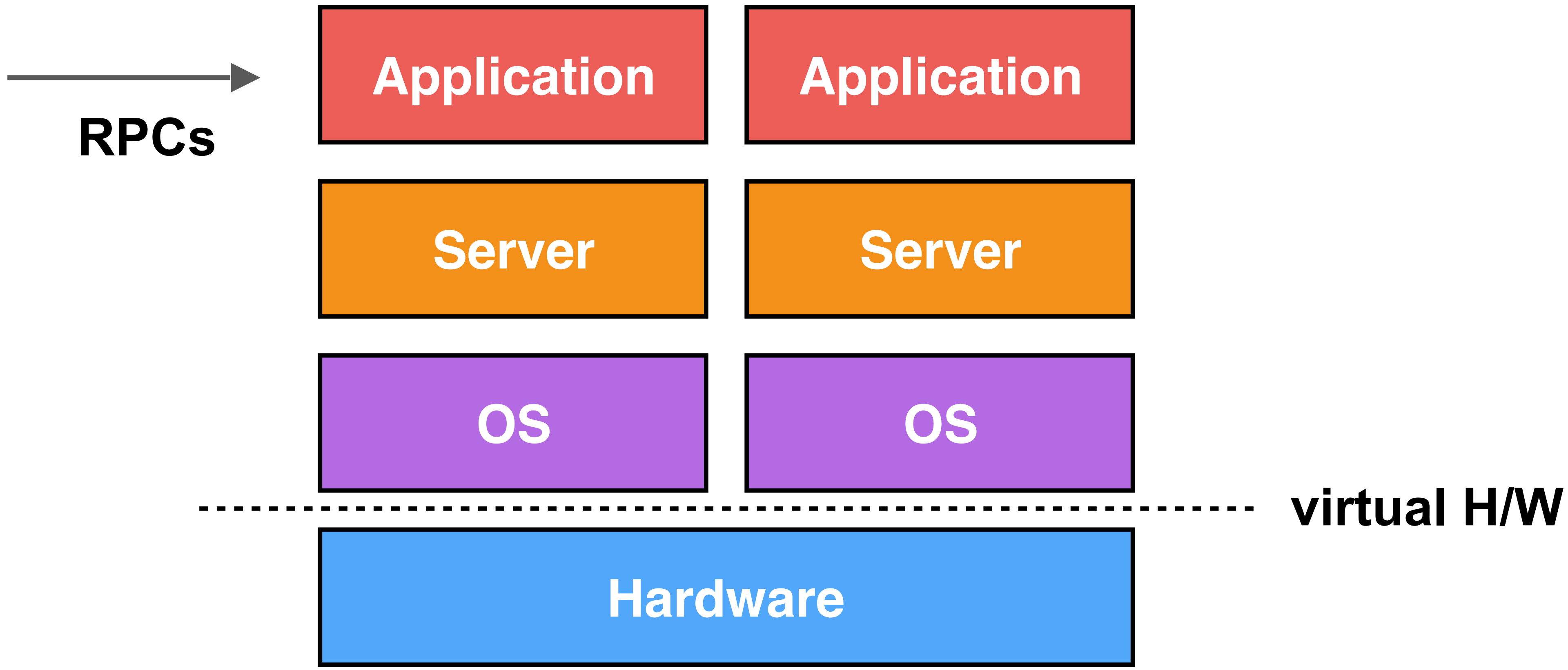
Classic web stack



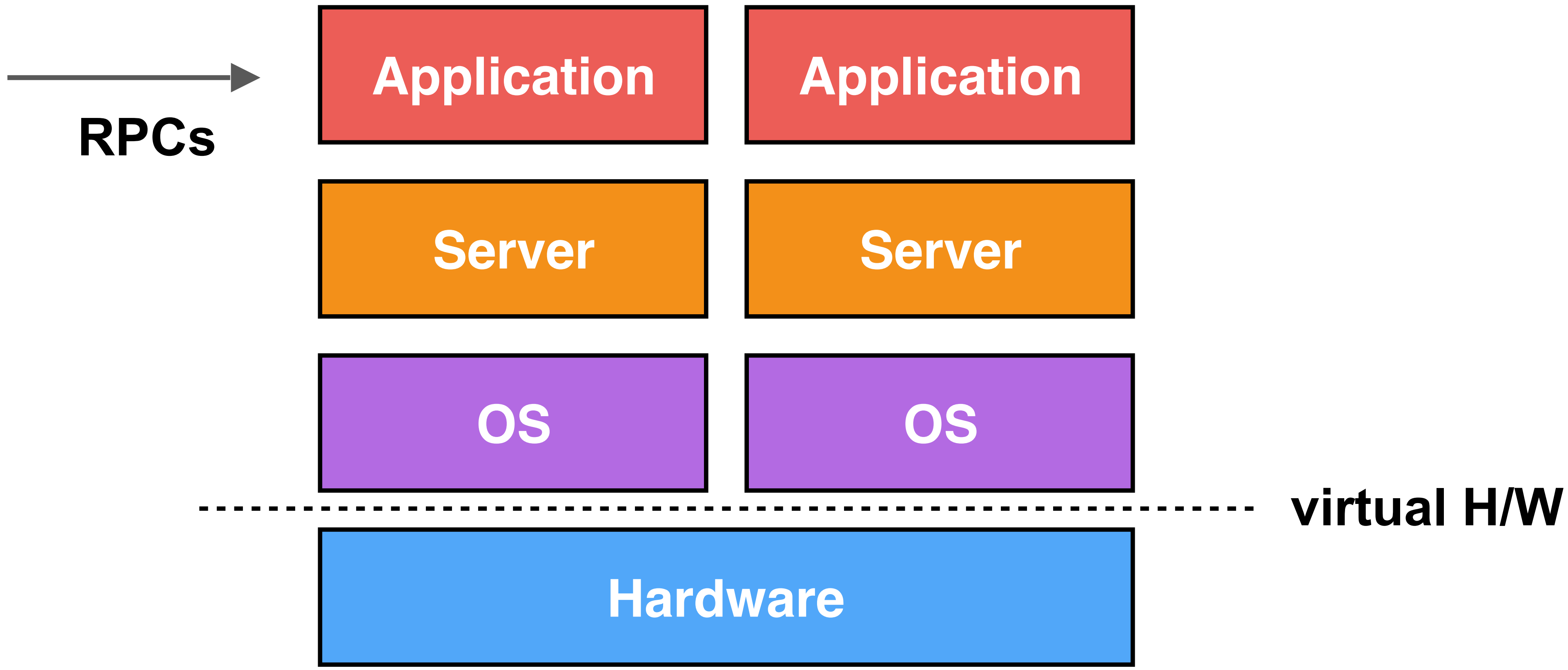
1st generation: virtual machines



1st generation: virtual machines



1st generation: virtual machines



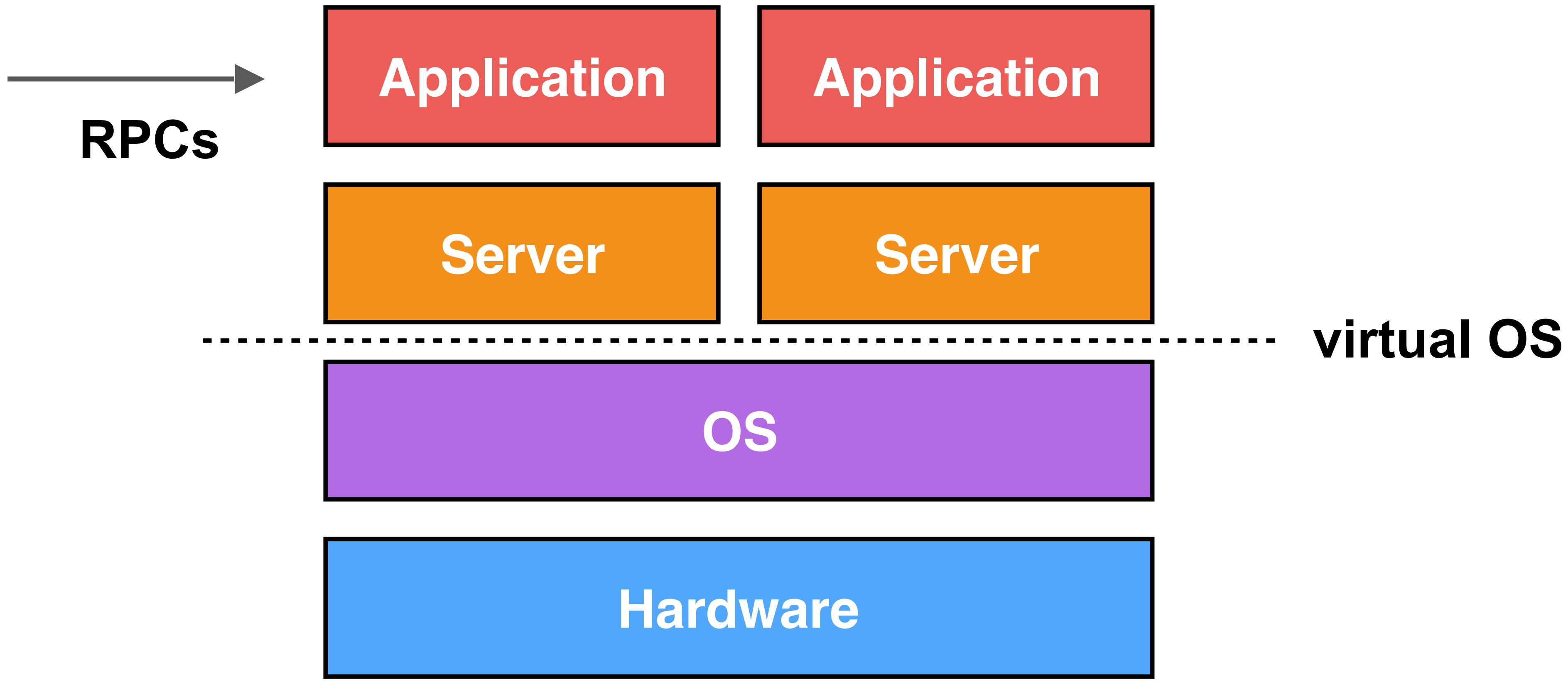
advantages:

- very flexible
- use any OS

problems:

- interposition
- is RAM used? (ballooning)
- redundancy (e.g., FS journal)

2nd generation: containers



advantages:

- centralized view
- init H/W once

Are containers good enough?

Container case studies

Literature: Google Borg

- Internal container platform [1]
- 25 second median startup
- 80% of time spent on package installation
- matters for flash crowds, load balance, interactive development, etc

[1] Large-scale cluster management at Google with Borg.
http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43438.pdf

Container case studies

Literature: Google Borg

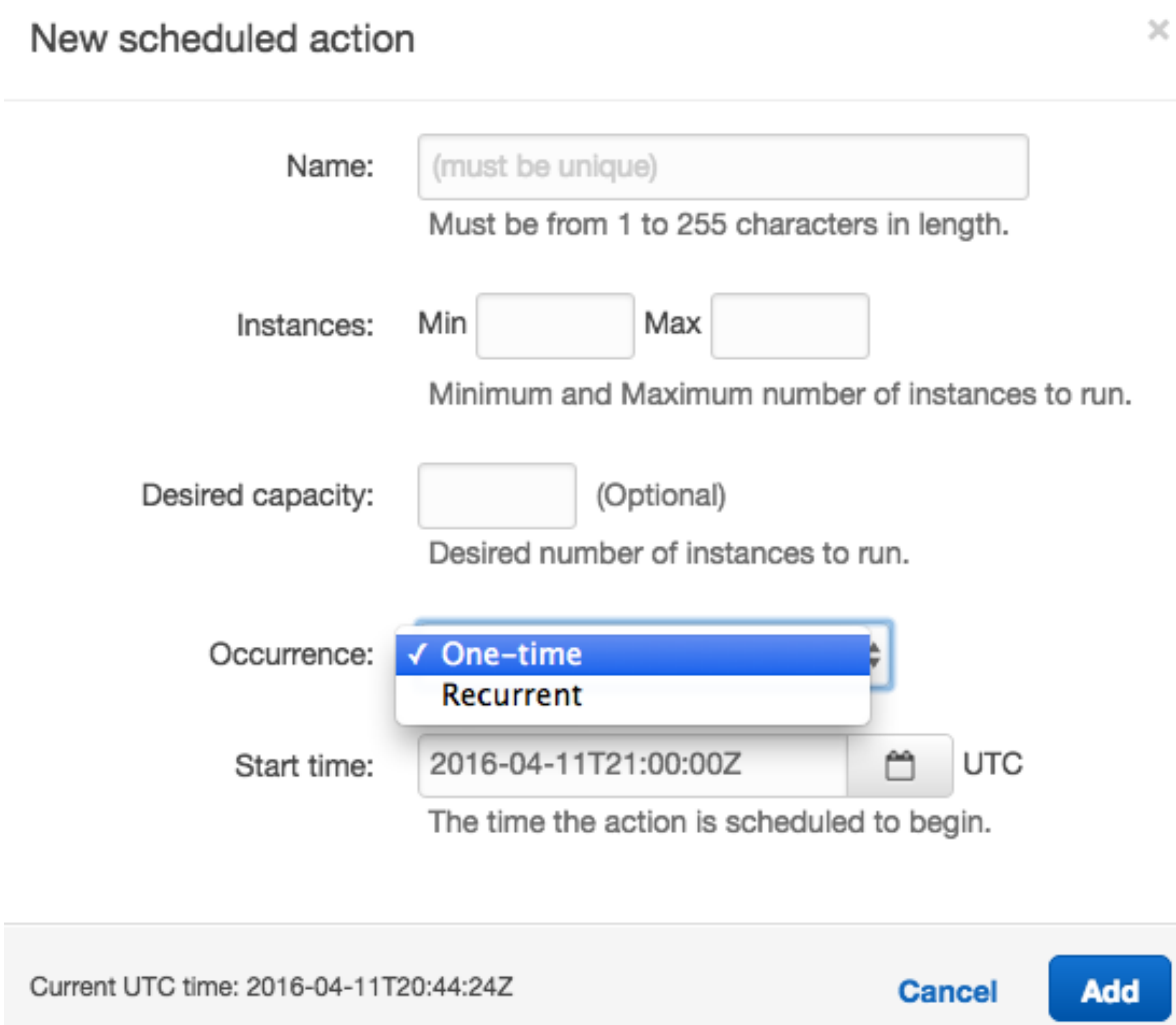
- Internal container platform [1]
- 25 second median startup
- 80% of time spent on package installation
- matters for flash crowds, load balance, interactive development, etc

Experimental: Amazon Elastic Beanstalk

- Autoscaling cloud service
- Build applications as containerized servers, service RPCs
- Rules dictate when to start/stop (various factors)

[1] Large-scale cluster management at Google with Borg.
http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43438.pdf

Interesting "autoscaling" rule

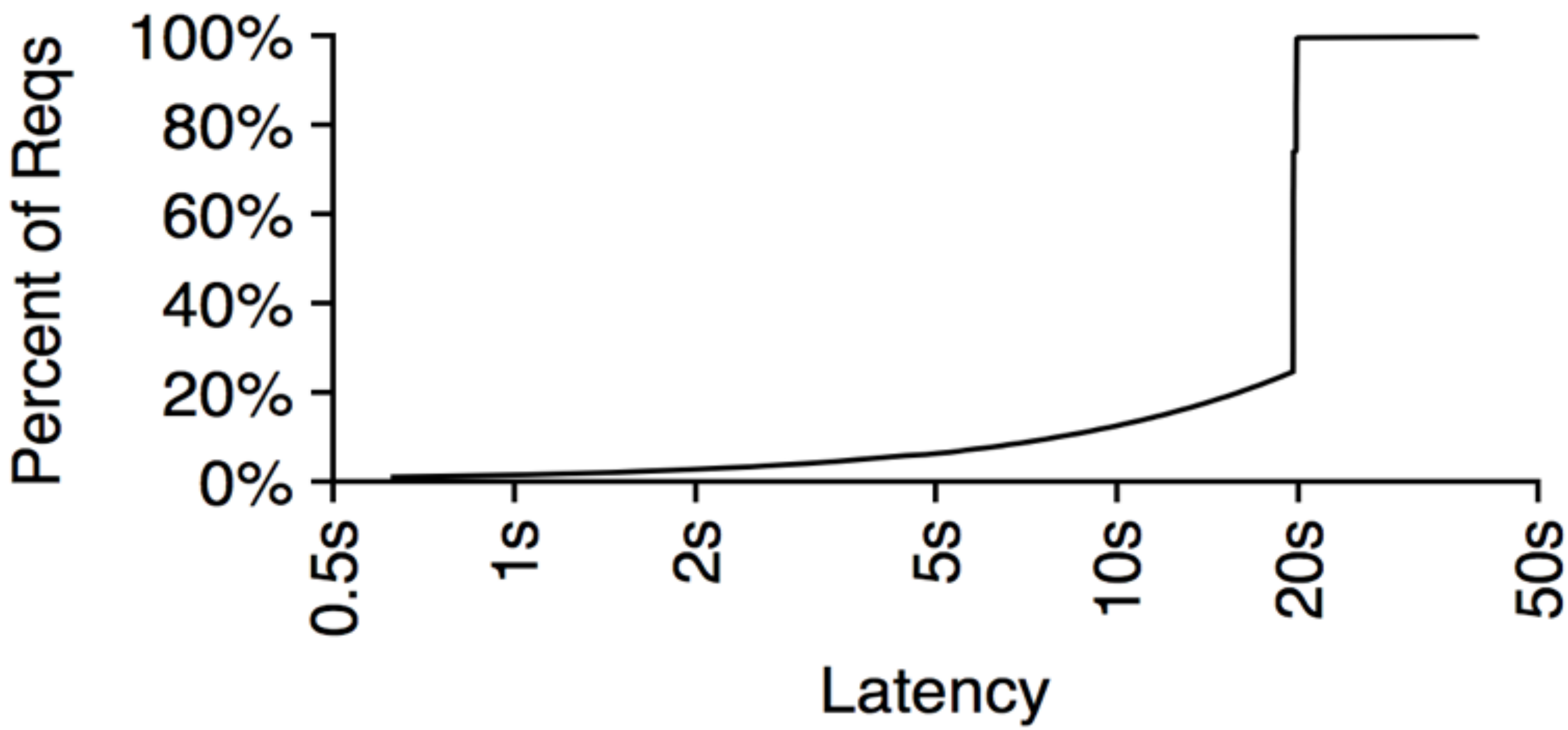


Experiment

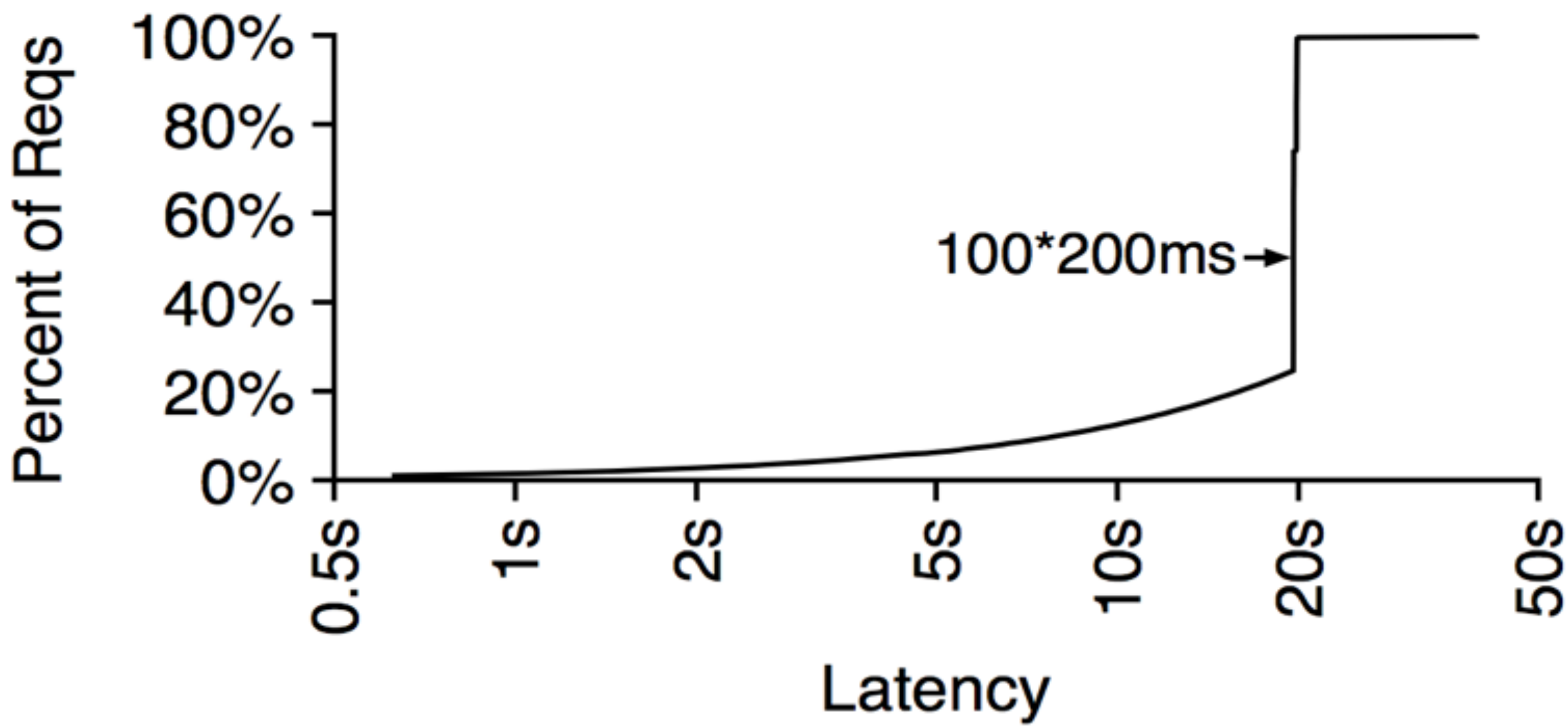
Simulate a small short burst

- Maintain **100 concurrent requests**
- Use **200 ms** of compute per request
- Run for **1 minute**

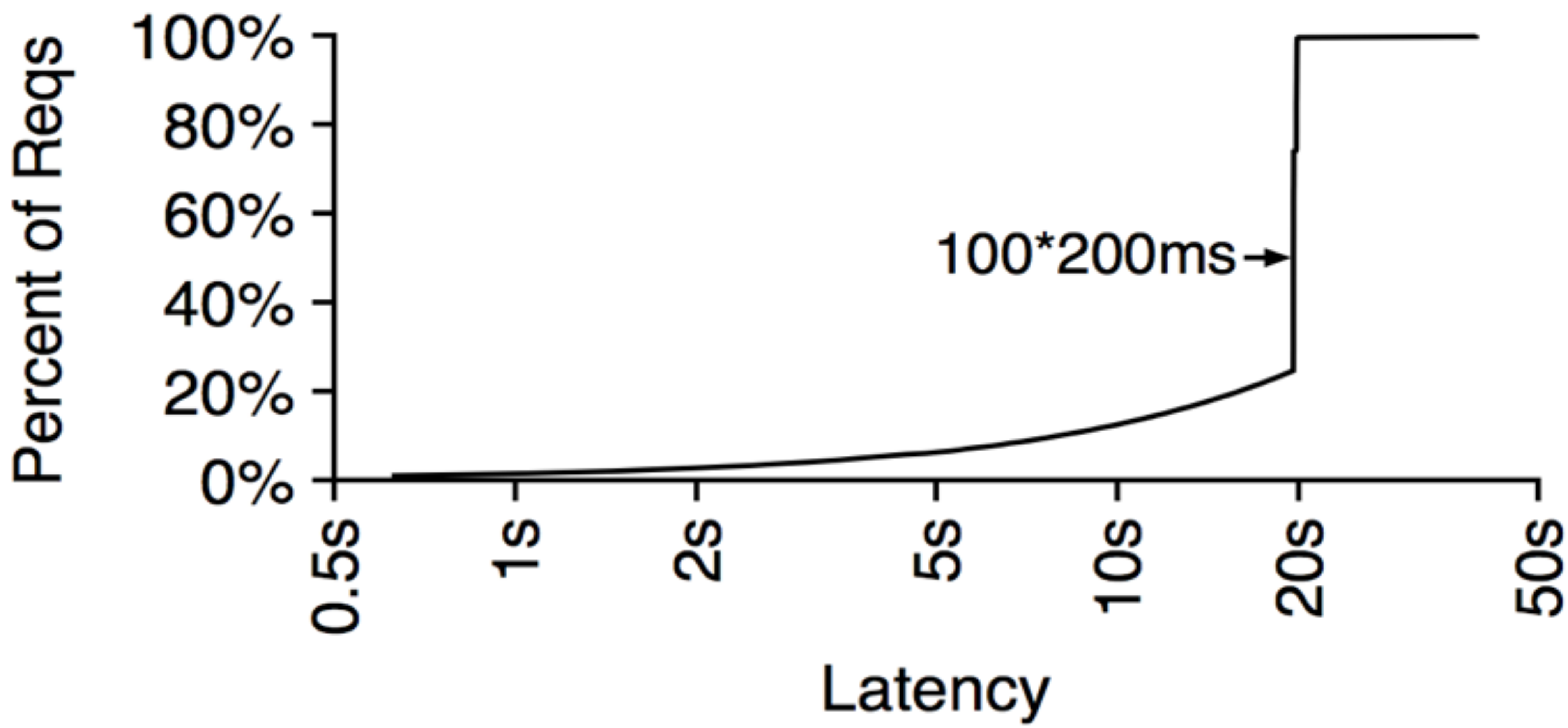
Container Case Study: Elastic Beanstalk



Container Case Study: Elastic Beanstalk

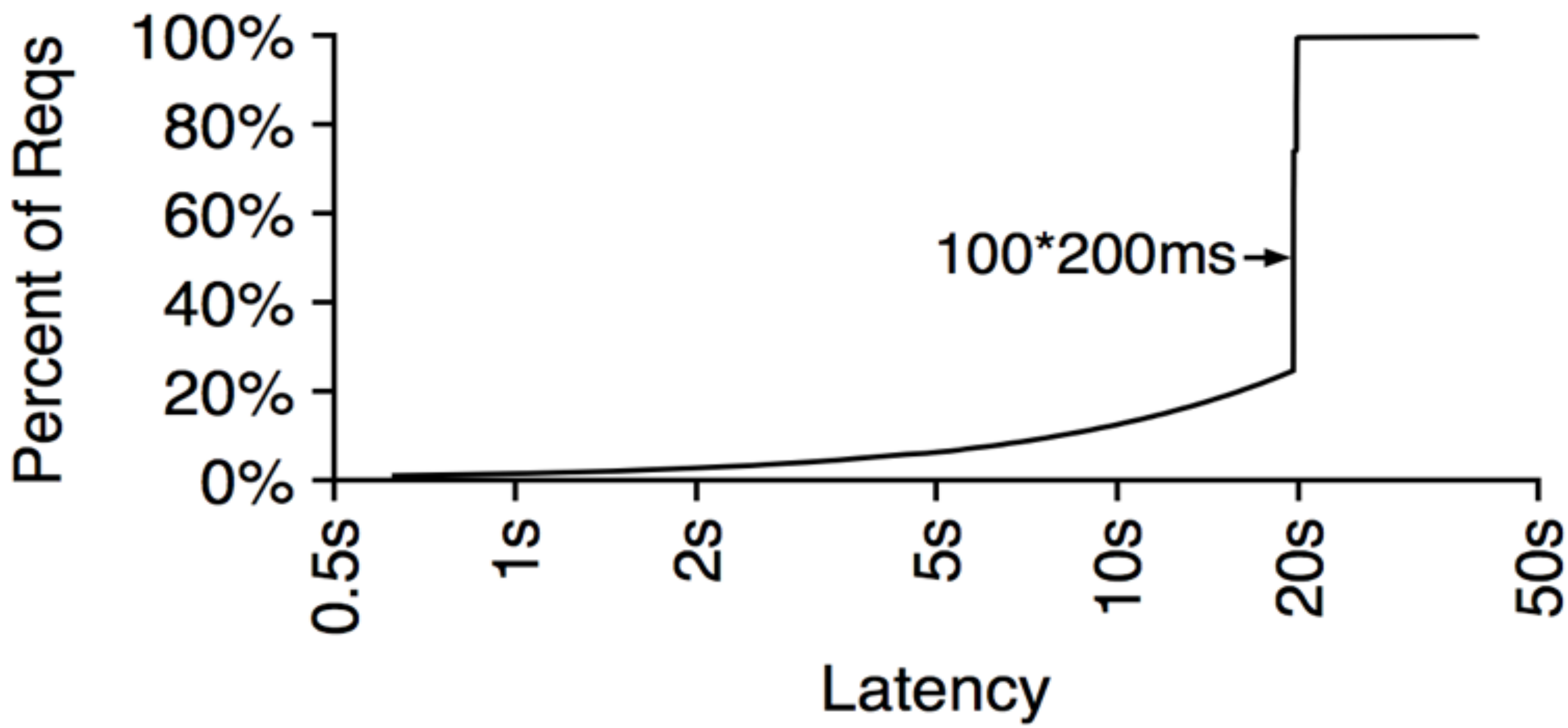


Container Case Study: Elastic Beanstalk



Conclusion: Elastic Beanstalk does not scale quickly enough to handle load bursts.

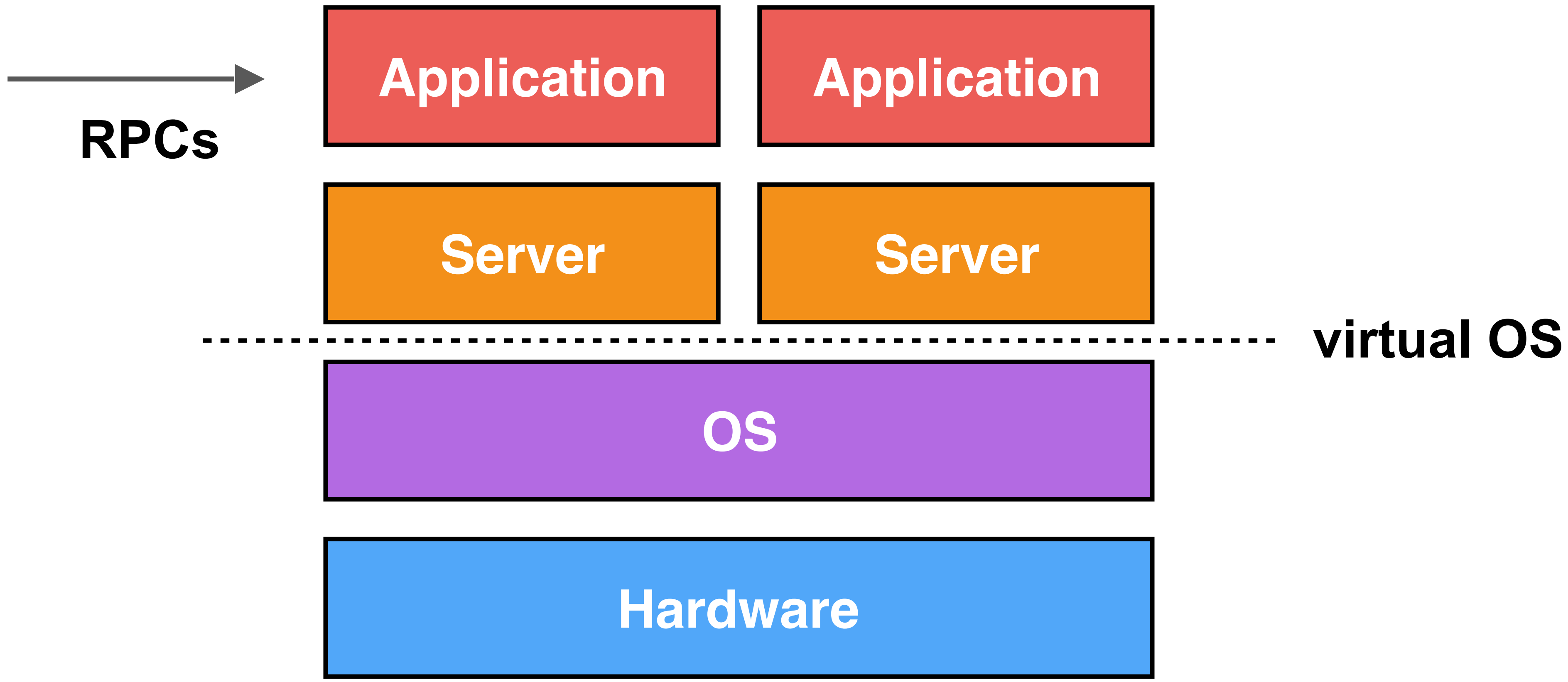
Container Case Study: ~~Elastic Beanstalk~~ Elastic BS



Conclusion: Elastic Beanstalk does not scale quickly enough to handle load bursts.

Why should it take minutes (or even seconds) to execute scripts that are <1000s of LOC?

2nd generation: containers



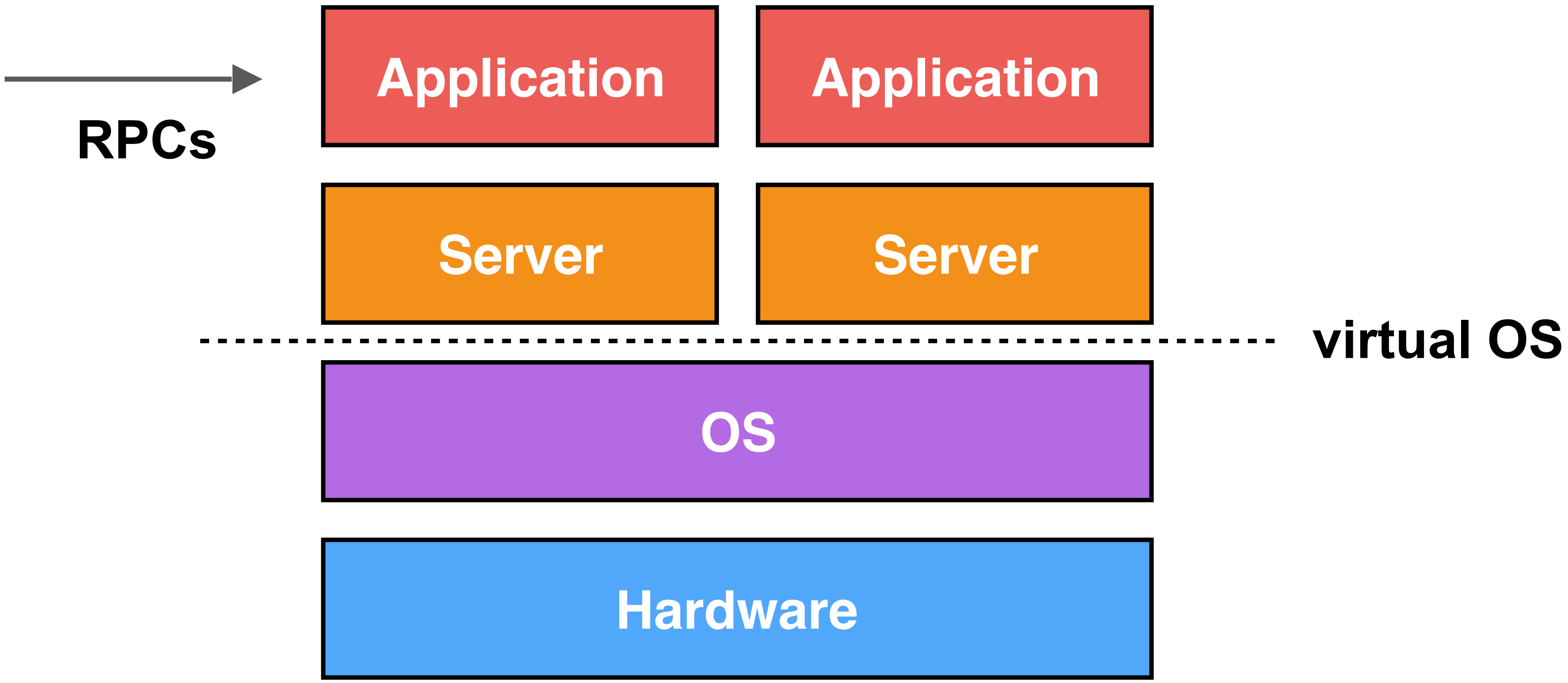
advantages:

- centralized view
- init H/W once

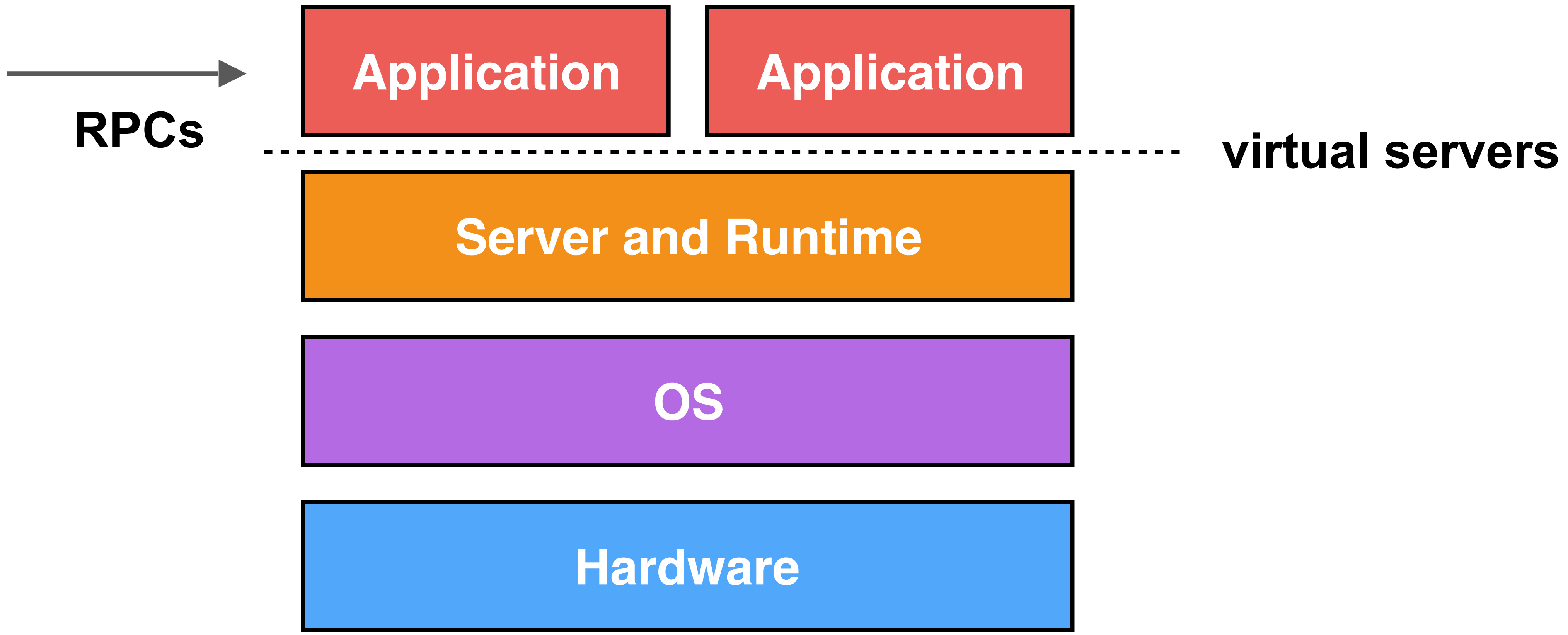
problems:

- large deployment bundle
- server spinup

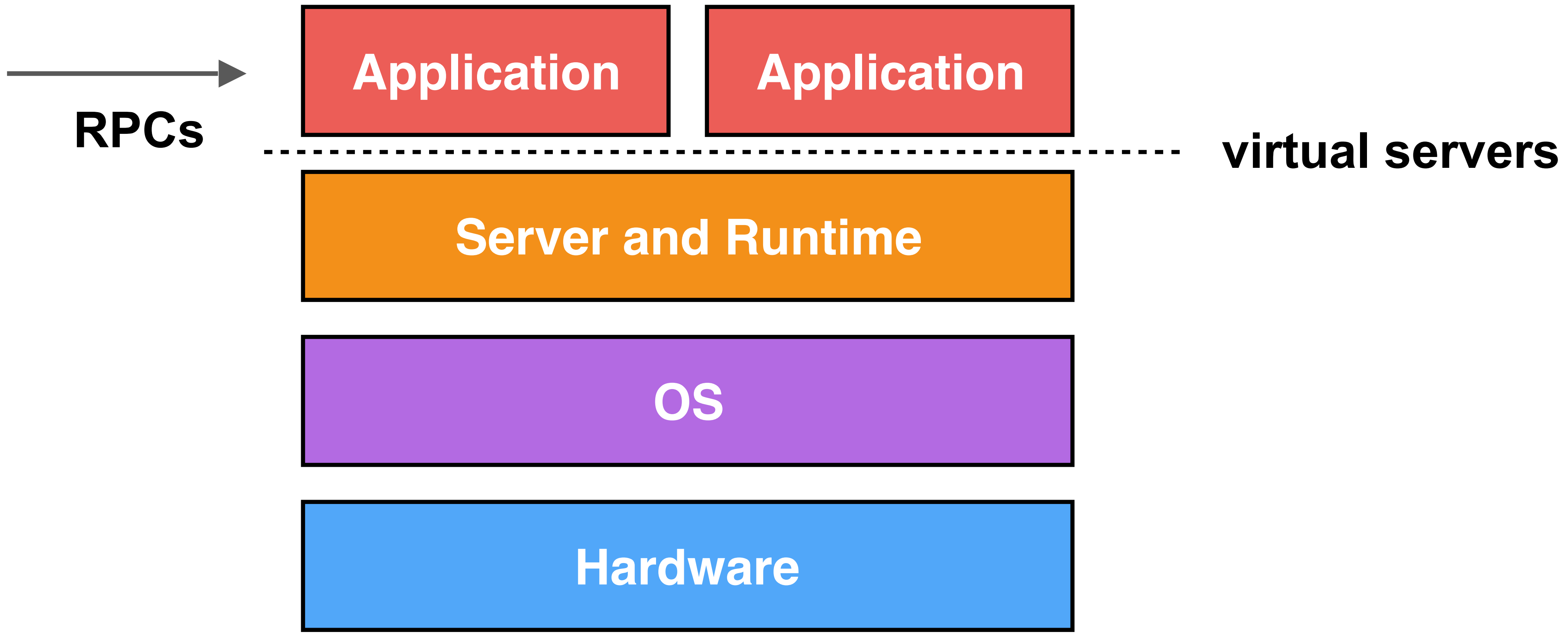
2nd generation: containers



3rd generation: Lambdas

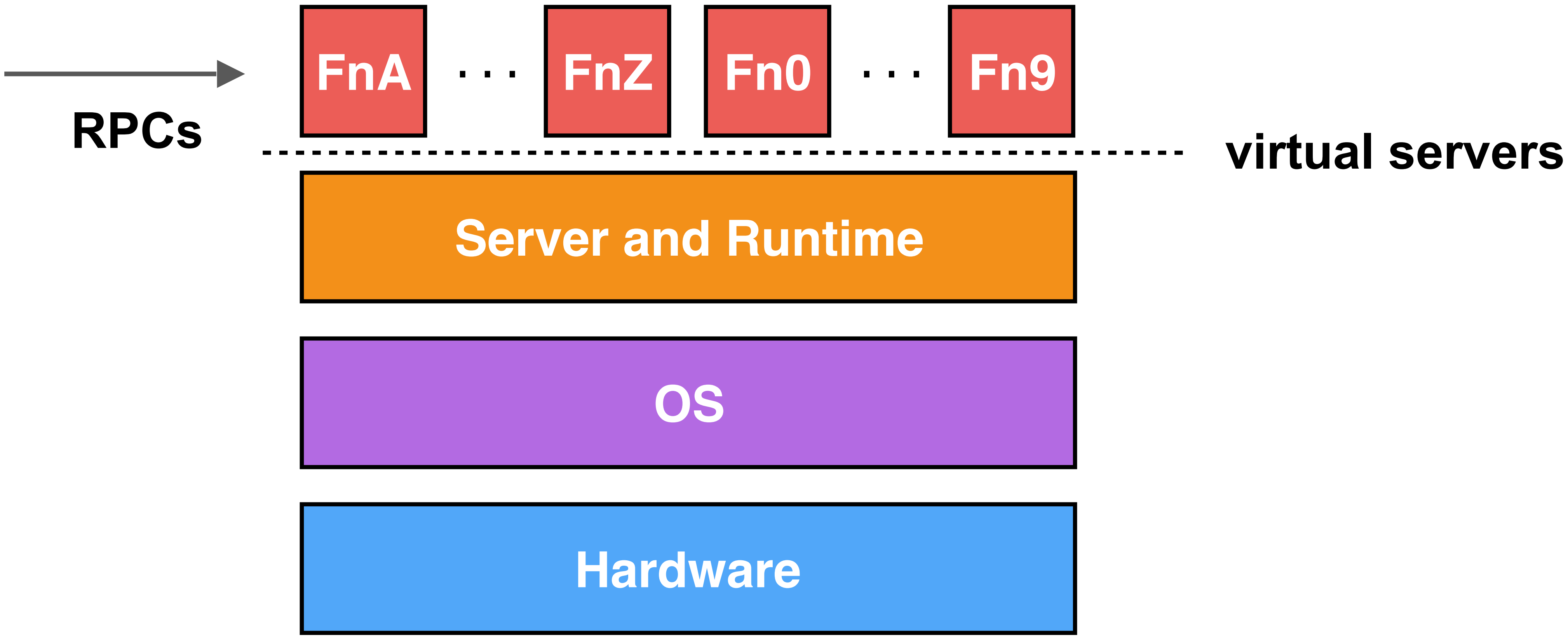


3rd generation: Lambdas



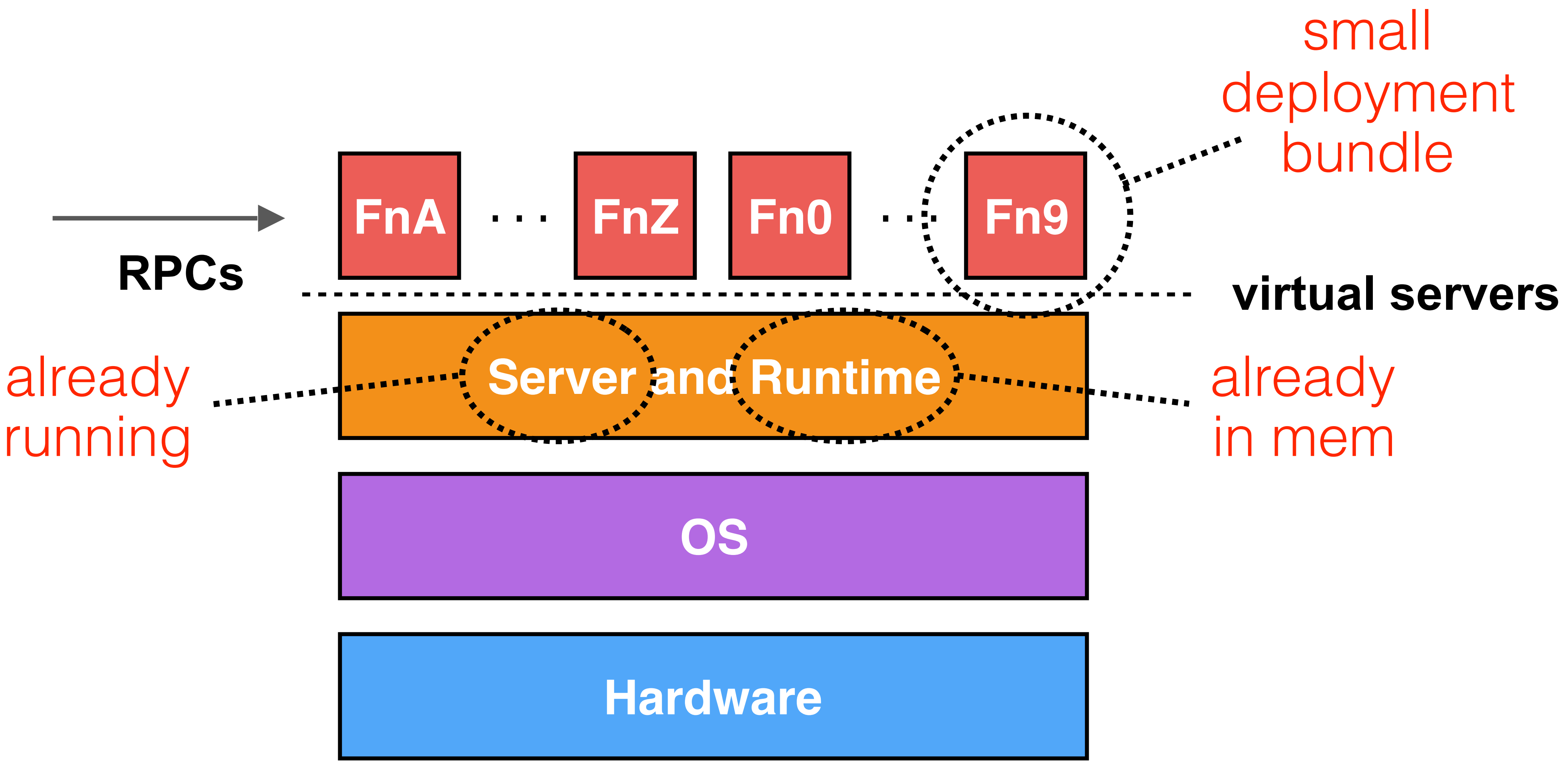
serverless computing

3rd generation: Lambdas



decompose application

3rd generation: Lambdas



advantages:

- very fast startup
- agile deployment
- share memory

problems:

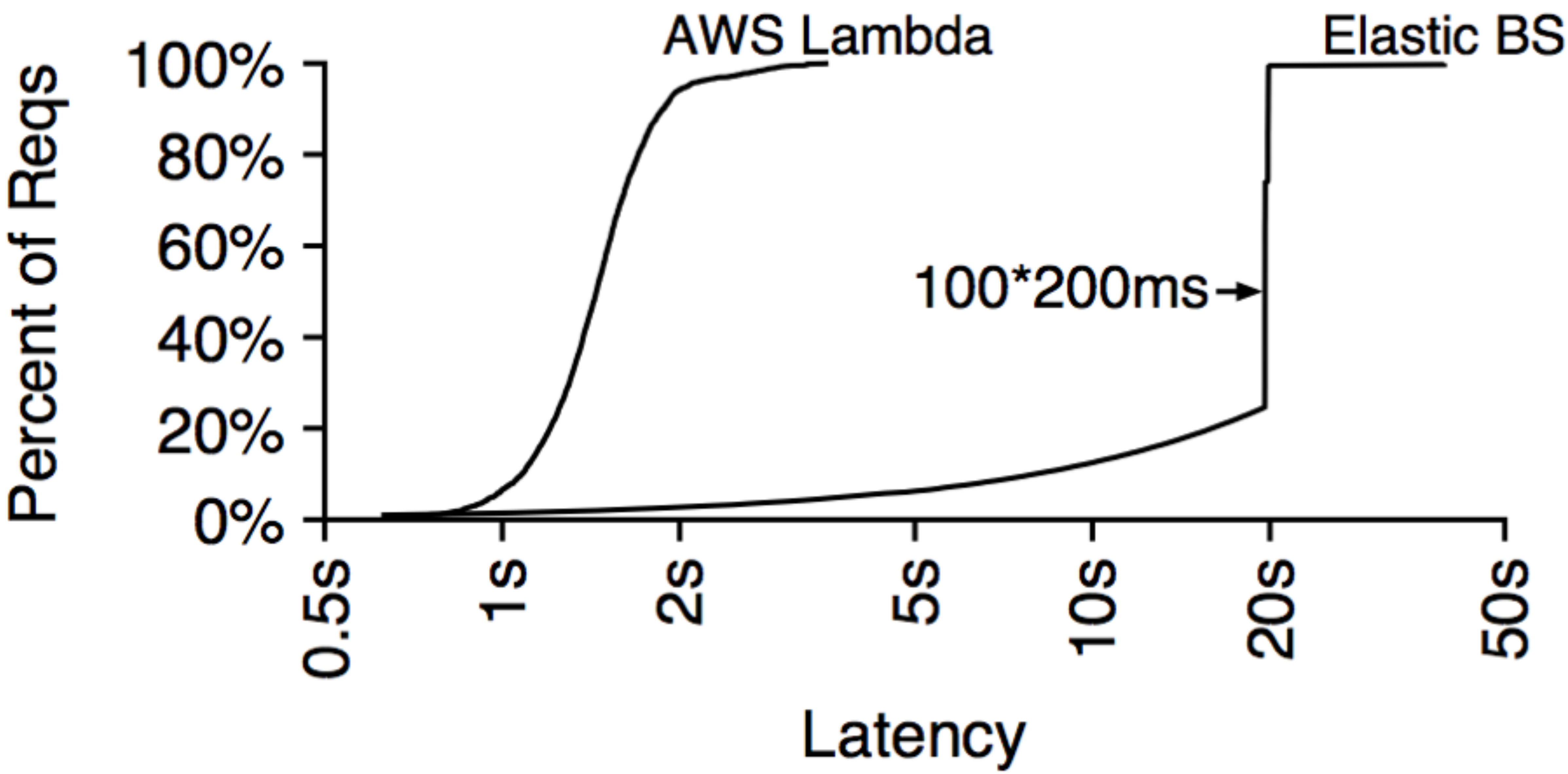
- not flexible

Lambda elasticity

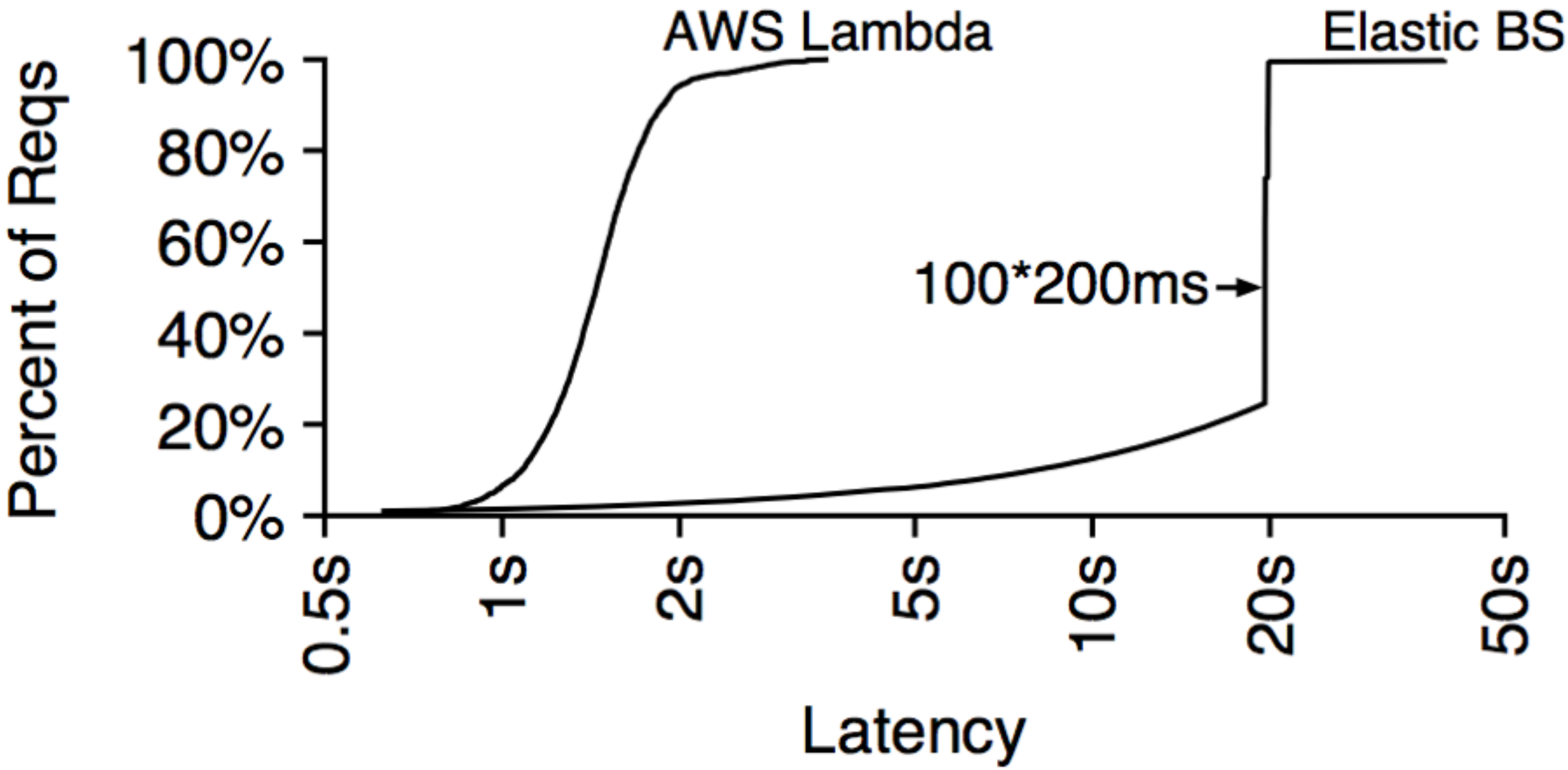
Repeat ElasticBS experiment

- Maintain **100 concurrent requests**
- Spin **200 ms** per request
- Run for **1 minute**

Lambda elasticity

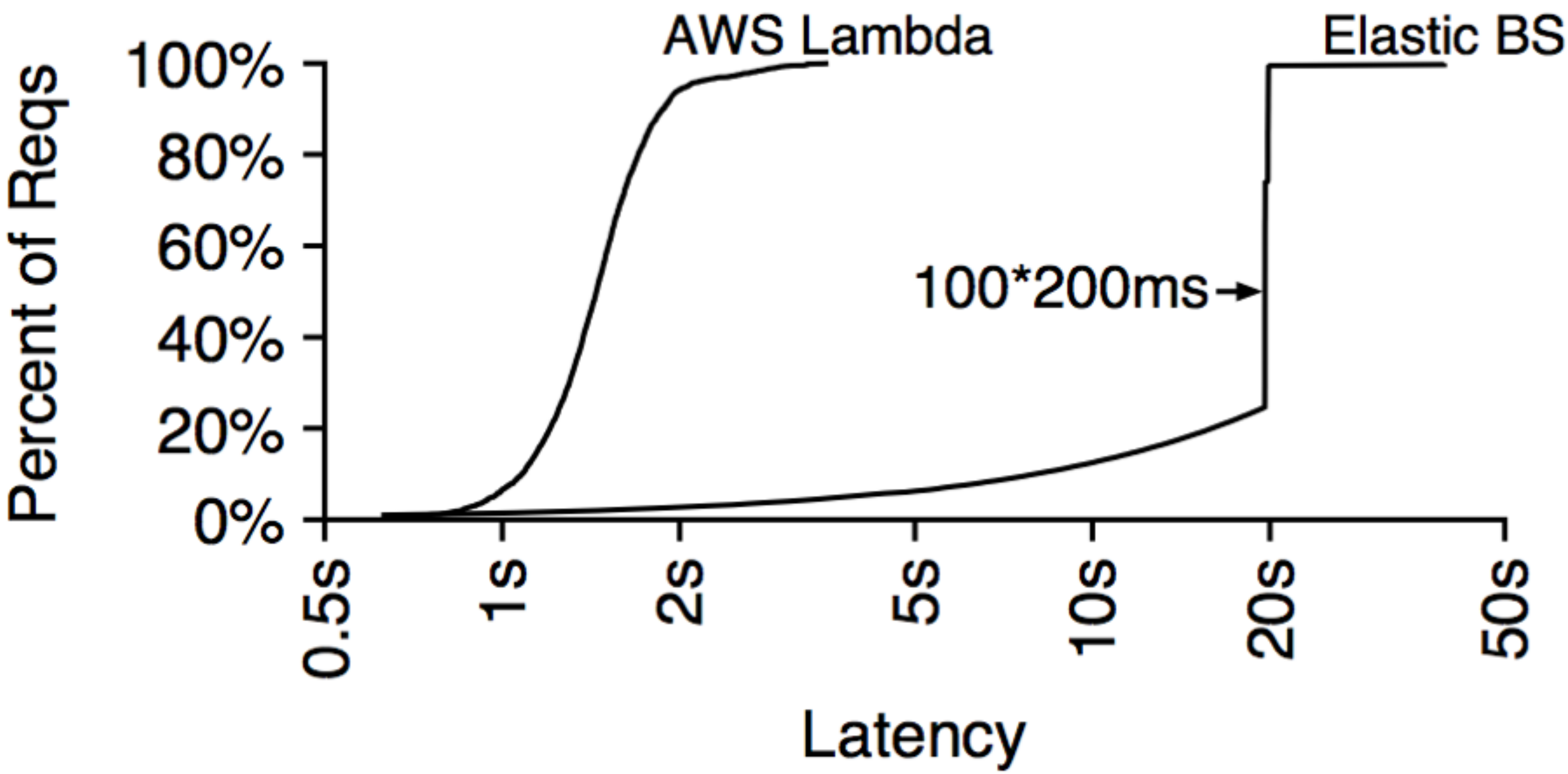


Lambda elasticity



Conclusion: Lambdas are highly elastic

Lambda elasticity



Conclusion: Lambdas are highly elastic
(though a little slow)

Outline

Evolution of compute

Non-conventional virtualization

Lambda model

Why OpenLambda?

Conclusion

Lambda model

Run user handlers in response to events

- web requests (RPC handlers)
- database updates (triggers)
- scheduled events (cron jobs)

Pay per function invocation

- actually pay-as-you-go
- no charge for idle time between calls
- e.g., charge `actual_time * memory_cap`

Share everything

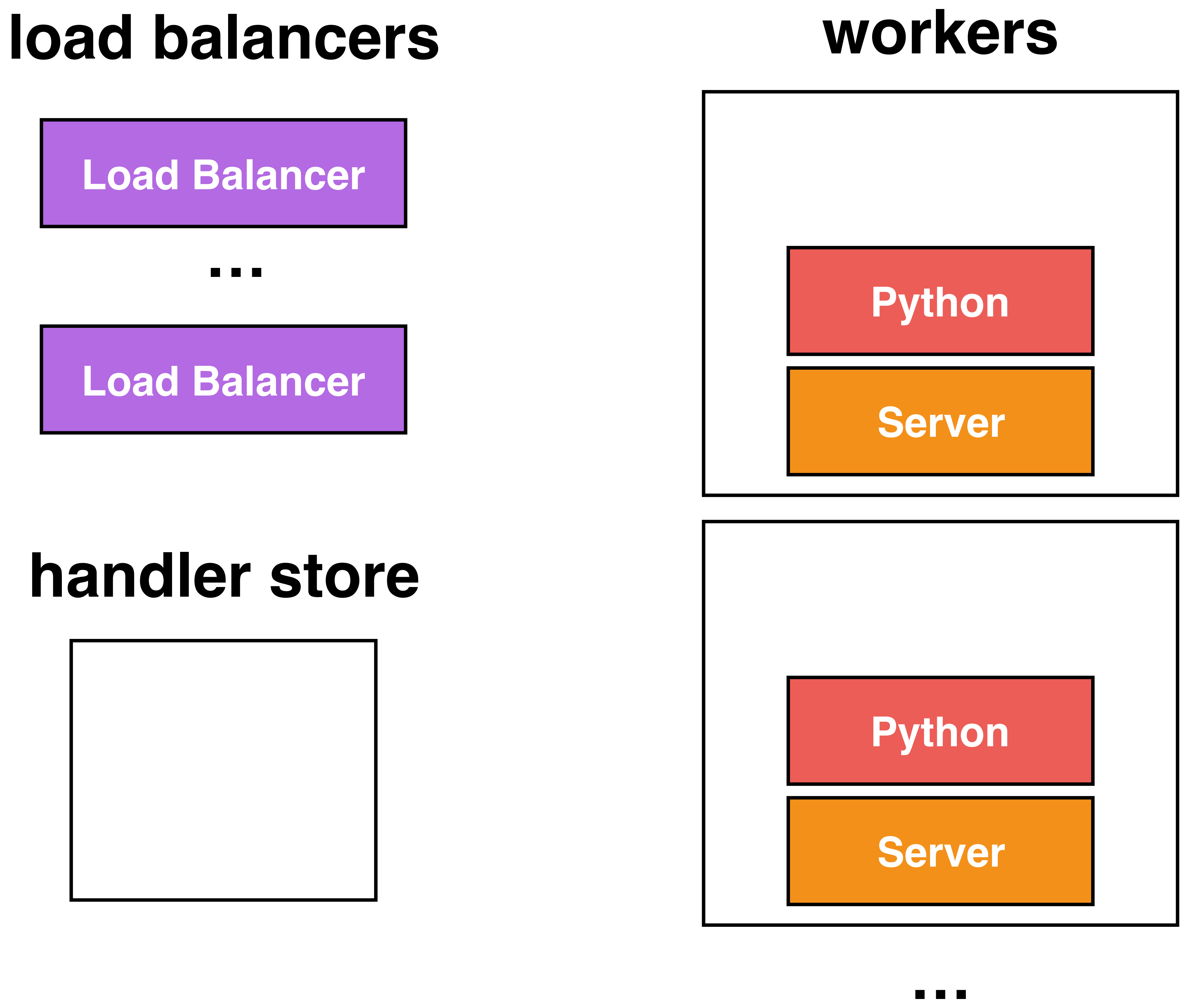
Share server pool between customers

- Any worker can execute any handler
- No spinup time
- Less switching

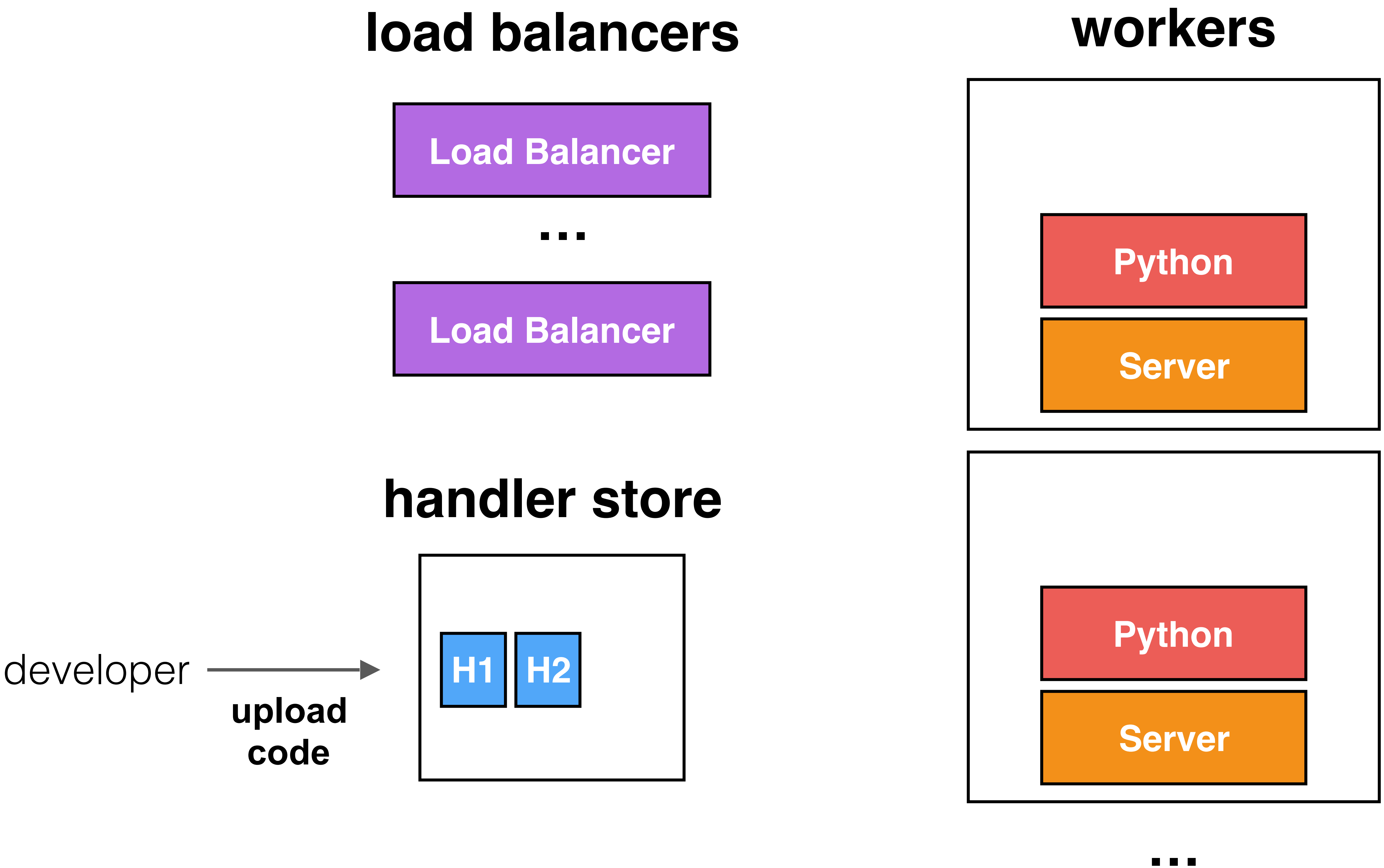
Encourage specific runtime (C#, Node.JS, Python)

- Minimize network copying
- Code will be in resident in memory

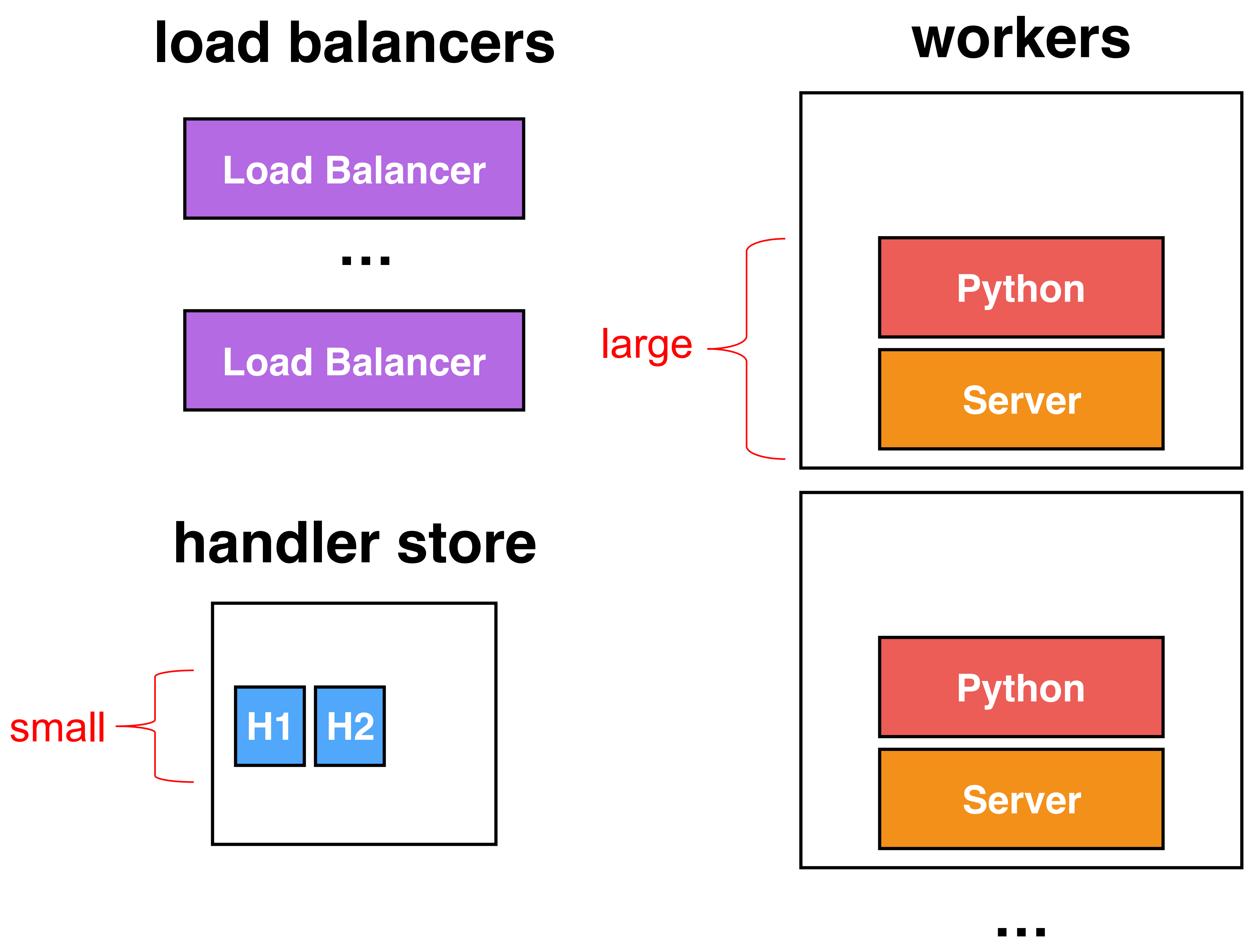
Multi-node architecture



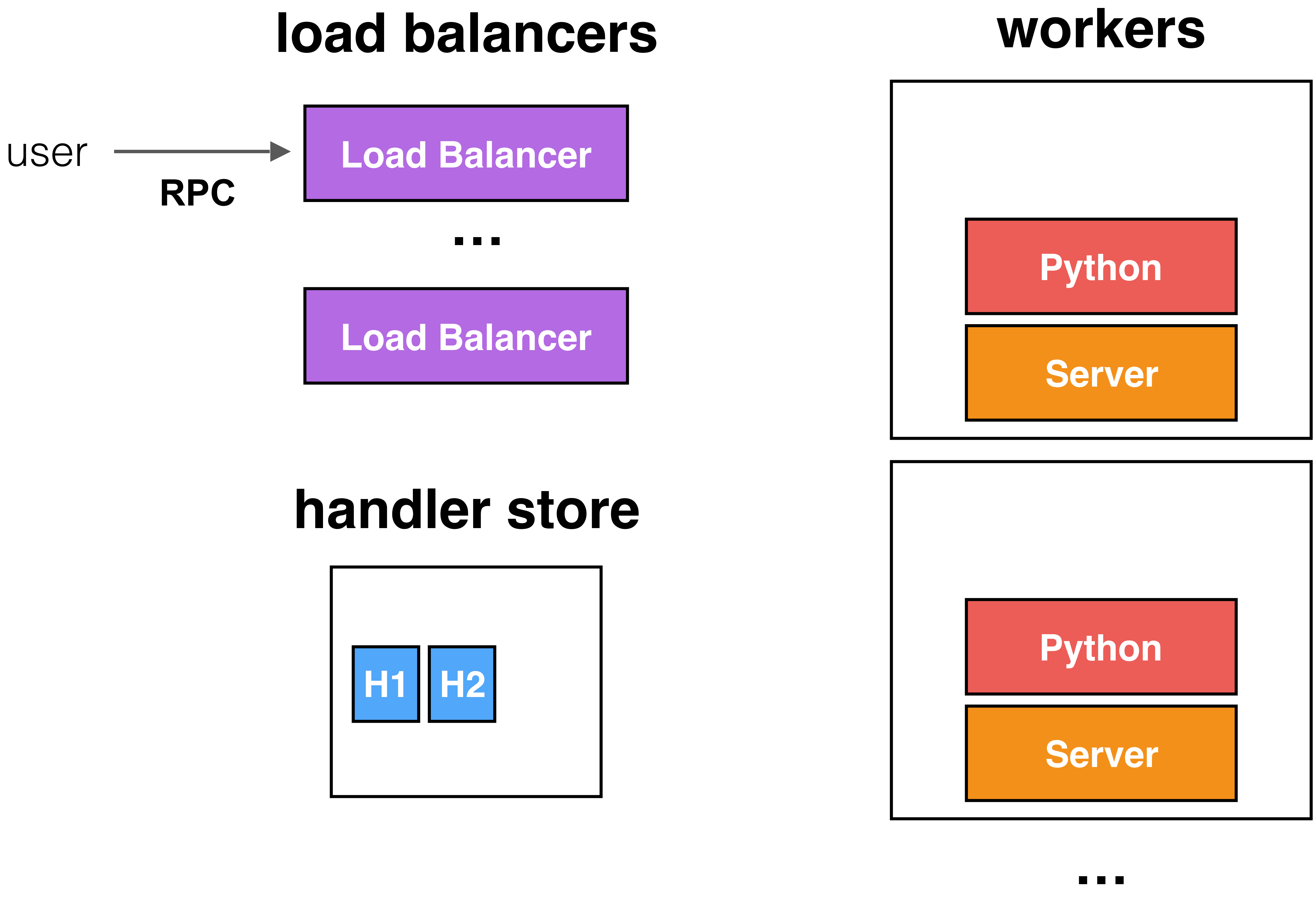
Multi-node architecture



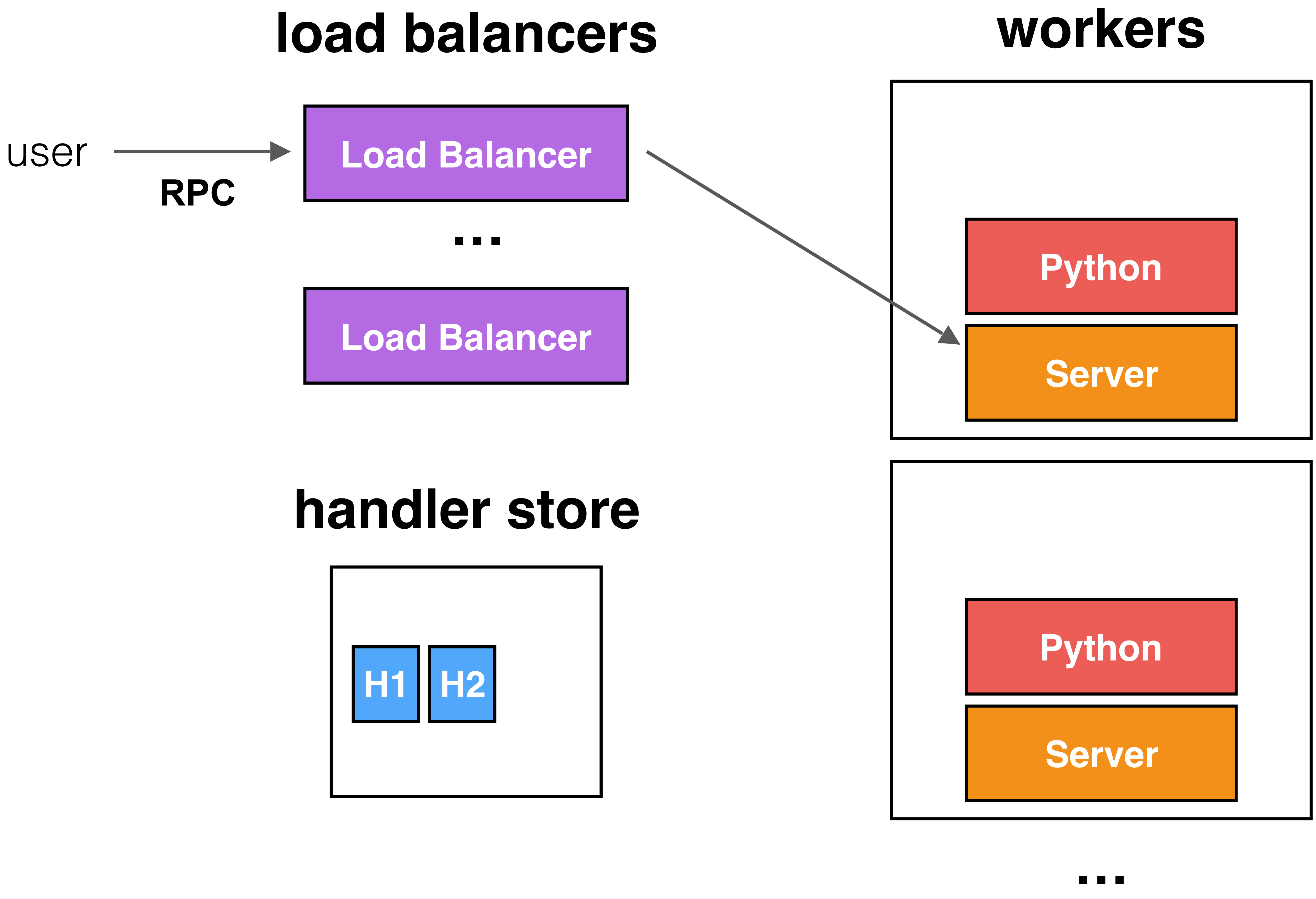
Multi-node architecture



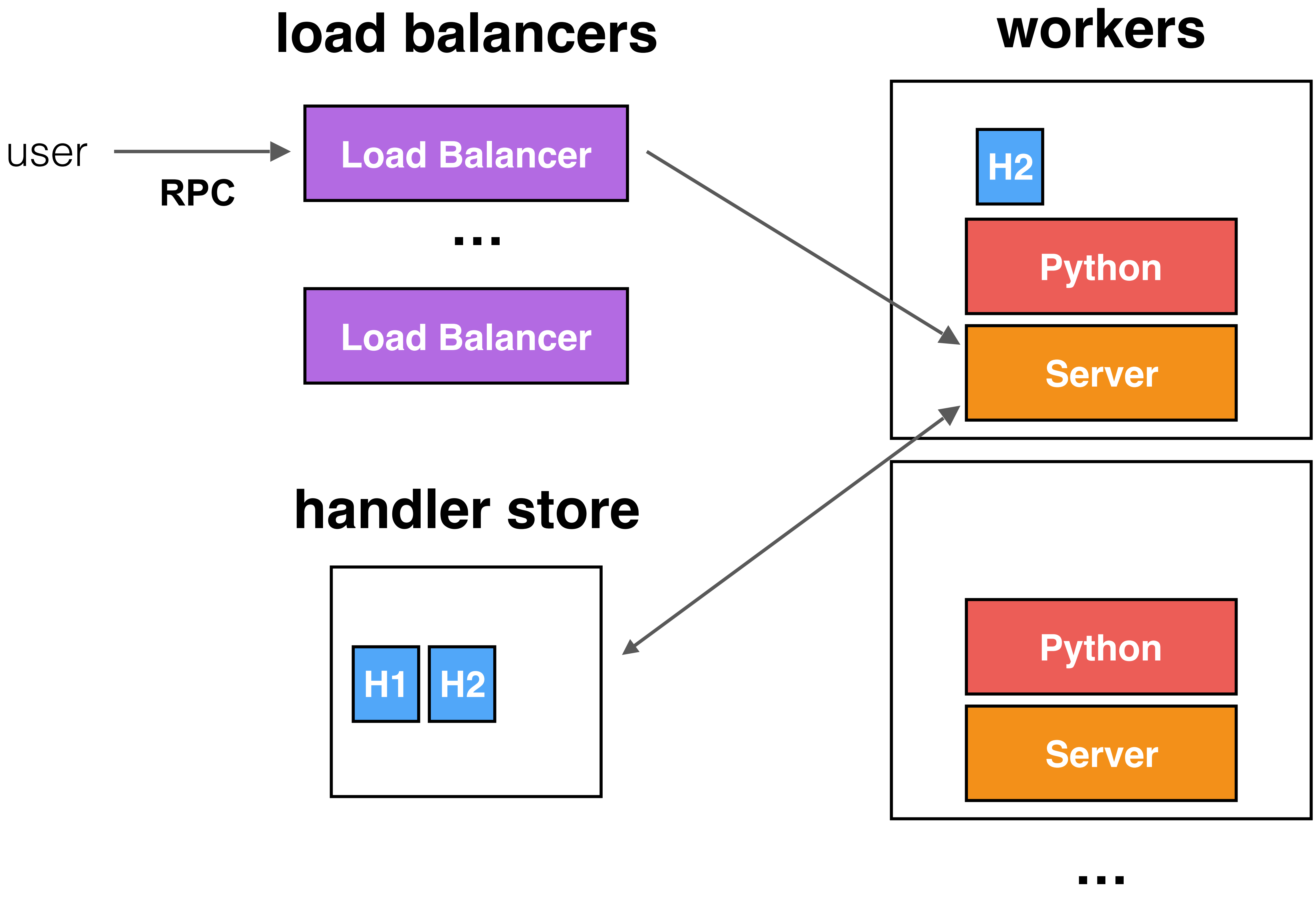
Multi-node architecture



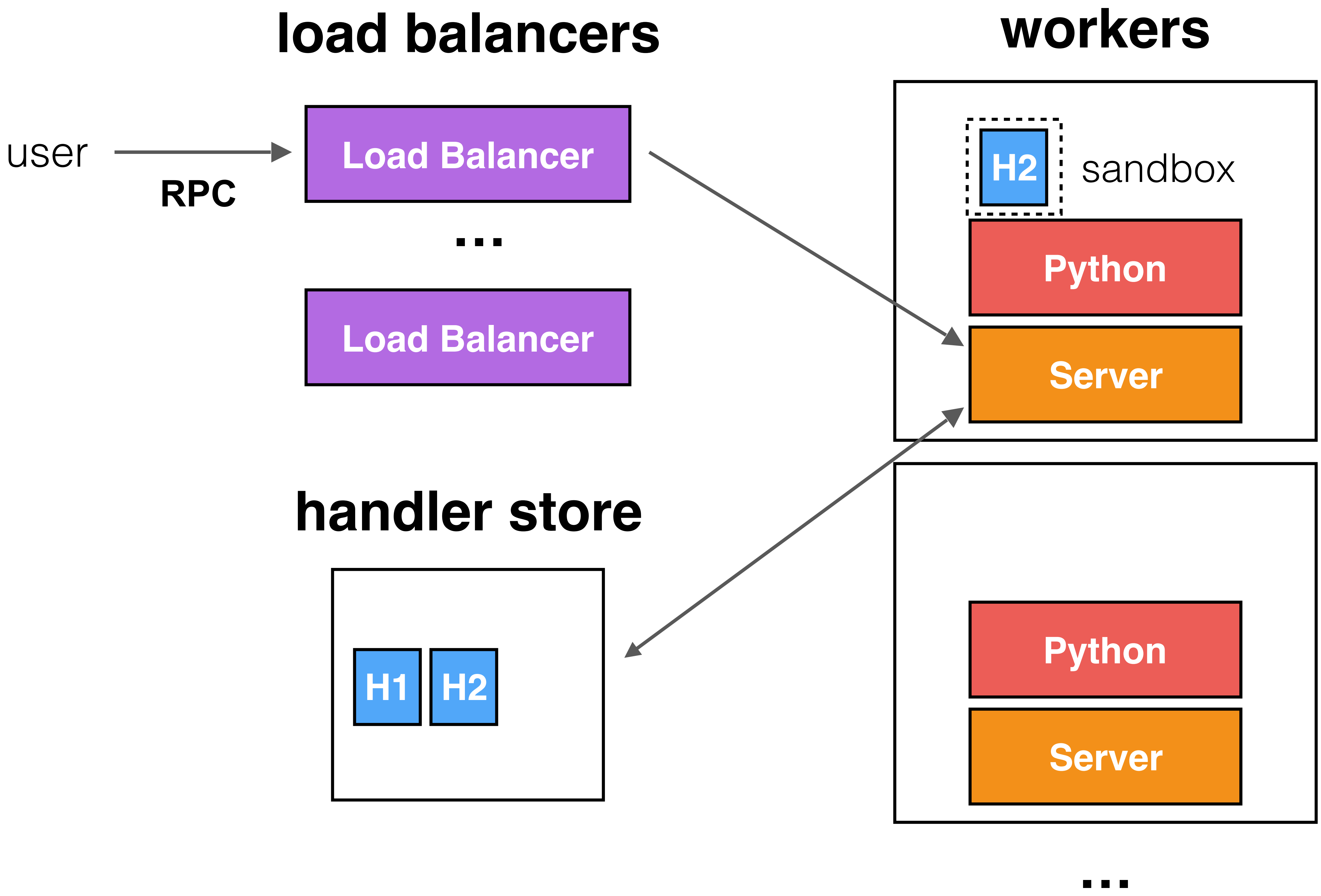
Multi-node architecture



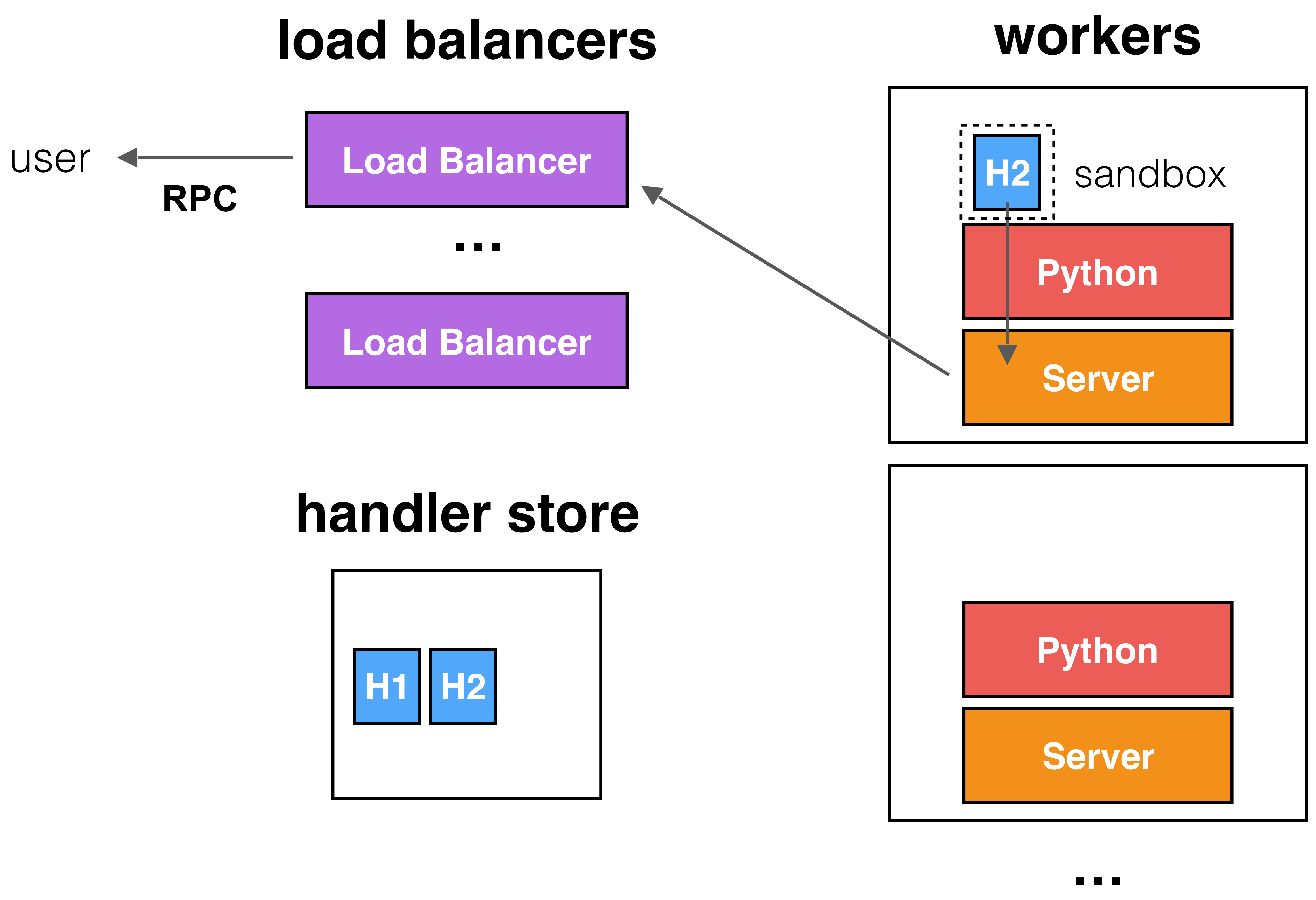
Multi-node architecture



Multi-node architecture



Multi-node architecture



Outline

Evolution of compute

Non-conventional virtualization

Lambda model

Why OpenLambda?

Conclusion

Need for open source serverless

Many research areas

- Applications, tools, distributed systems, execution engines
- Evaluate ideas by **building**, not just simulating

Need for open source serverless

Many research areas

- Applications, tools, distributed systems, execution engines
- Evaluate ideas by **building**, not just simulating

First implementations are proprietary

AWS Lambda

Google Cloud Platform

Google Cloud Functions

Need for open source serverless

Many research areas

- Applications, tools, distributed systems, execution engines
- Evaluate ideas by **building**, not just simulating

First implementations are proprietary

AWS Lambda

Google Cloud Platform

Google Cloud Functions

OpenLambda: explore further-reaching techniques

- Goal: enable academic research on Lambdas
- Storage awareness, kernel support, RPC inspection
- …

Need for open source serverless

Many research areas

- Applications, tools, distributed systems, execution engines
- Evaluate ideas by **building**, not just simulating

First implementations are proprietary

AWS Lambda   Google Cloud Platform   Google Cloud Functions

OpenLambda: explore further-reaching techniques

- Goal: enable academic research on Lambdas
- Storage awareness, kernel support, RPC inspection
- …

Other recent open-source implementations

Azure Functions   IBM OpenWhisk

OpenLambda research topics

Workloads

- Workload studies
- Benchmarks
- Versioning+dependencies
- Code characteristics
- Package management

Distributed systems

- Databases
- Load balancing
- Scatter gather patterns
- Sessions and streams

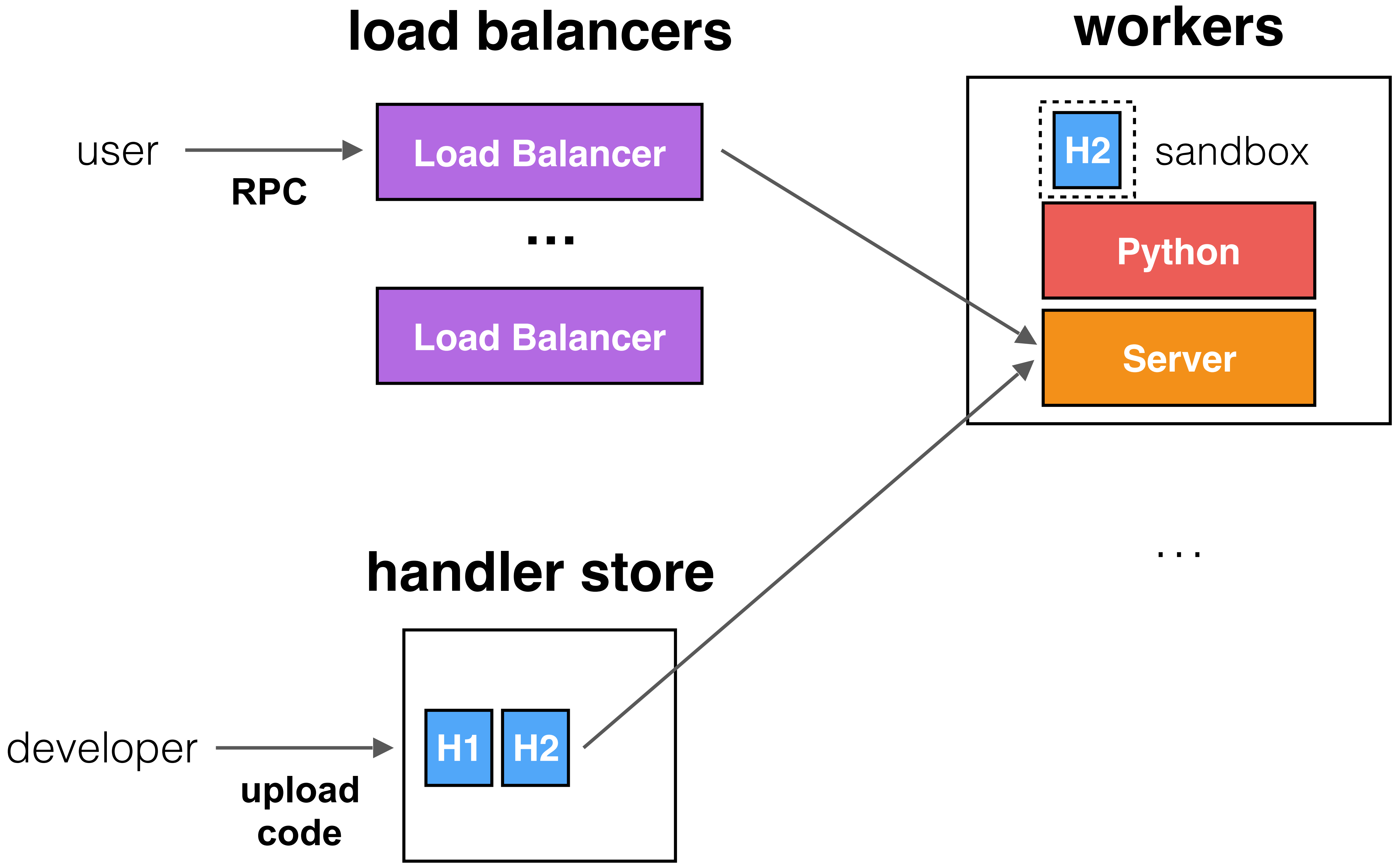
Tools

- Debugging
- Monetary cost optimization
- Porting legacy applications

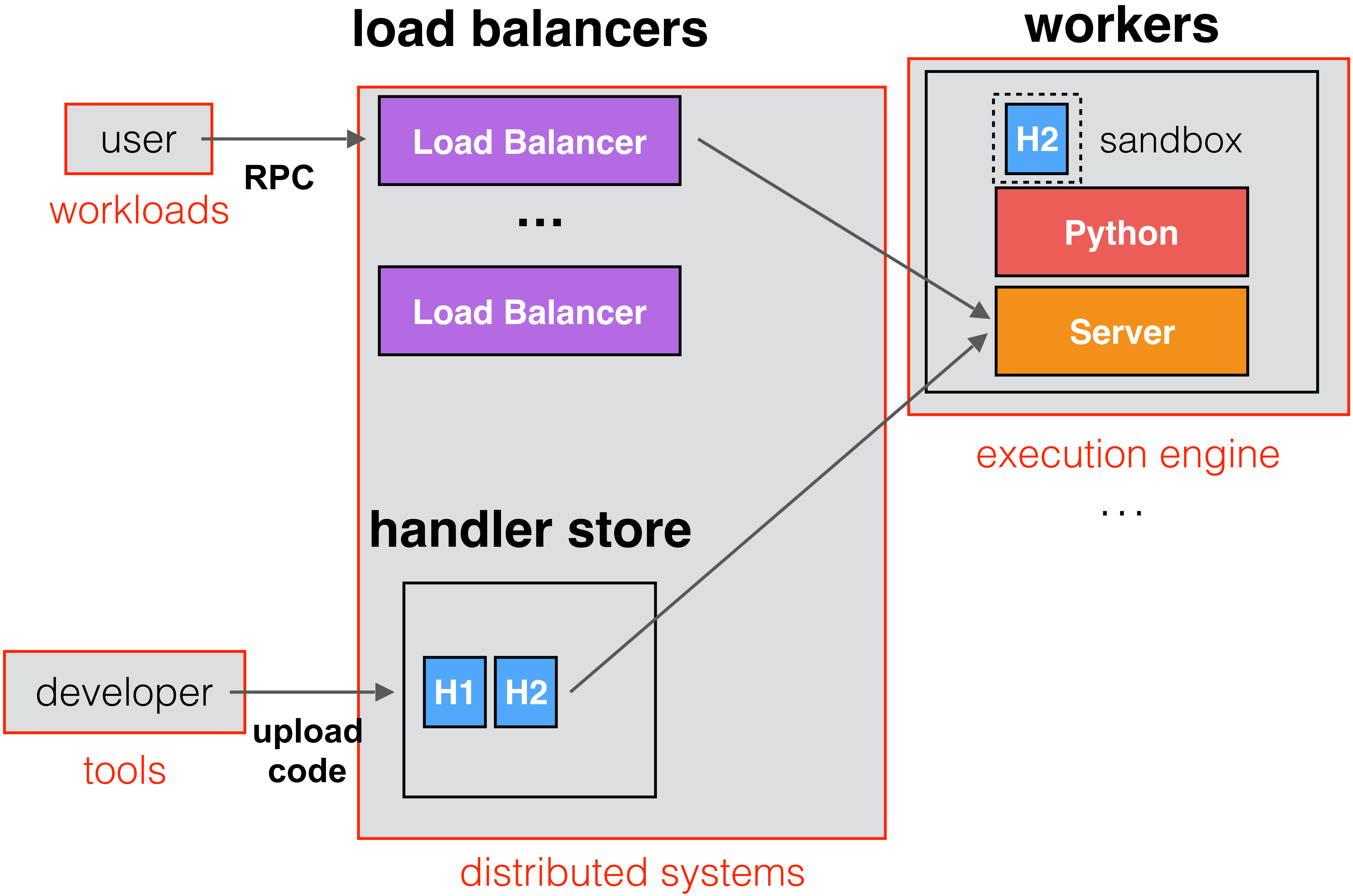
Execution engines

- Sandboxing
- Containers
- Just-in-time interpreters

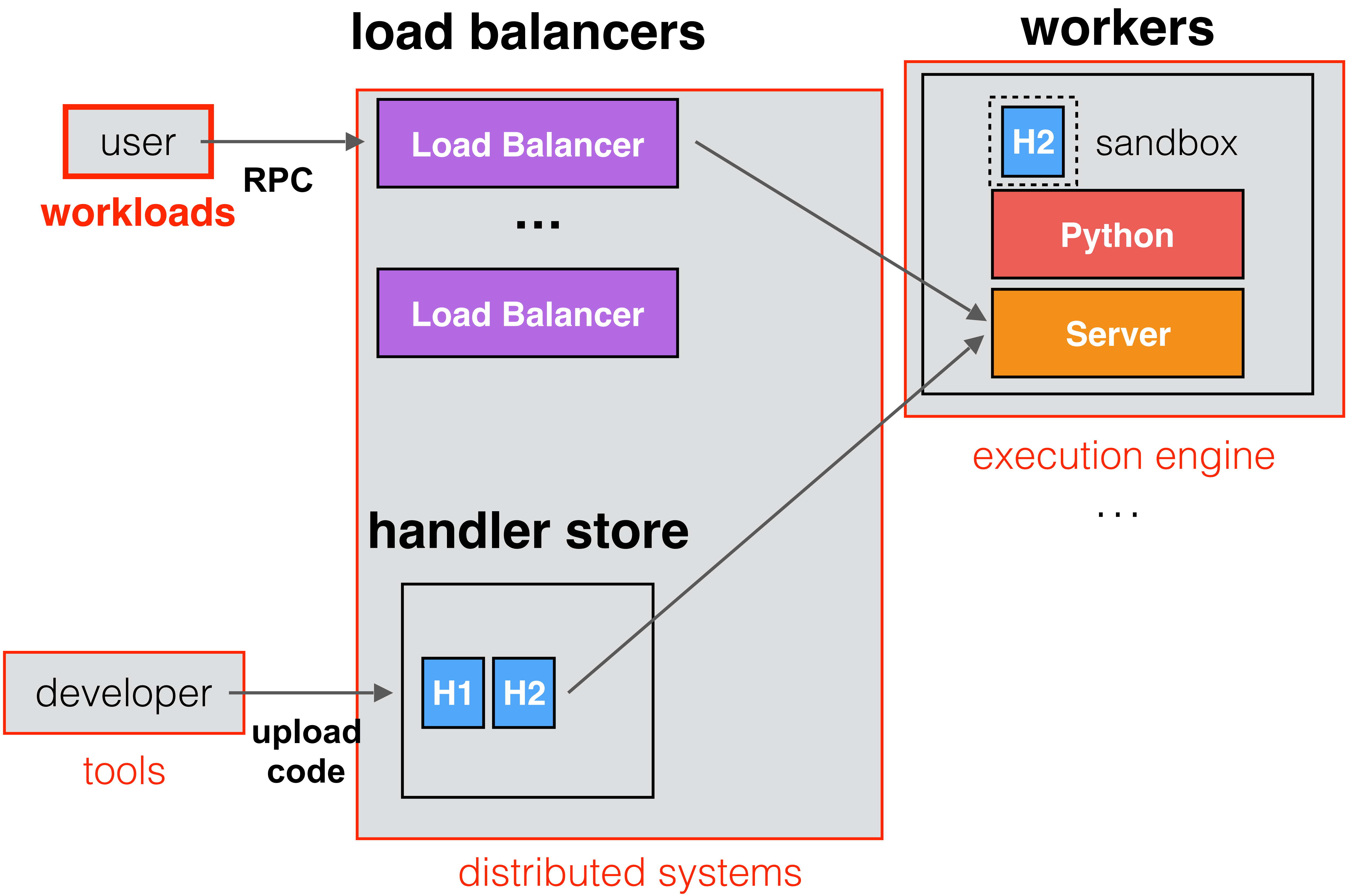
OpenLambda research topics



OpenLambda research topics



OpenLambda research topics



Understanding Lambda workloads

Collaborate with industry, measurement studies

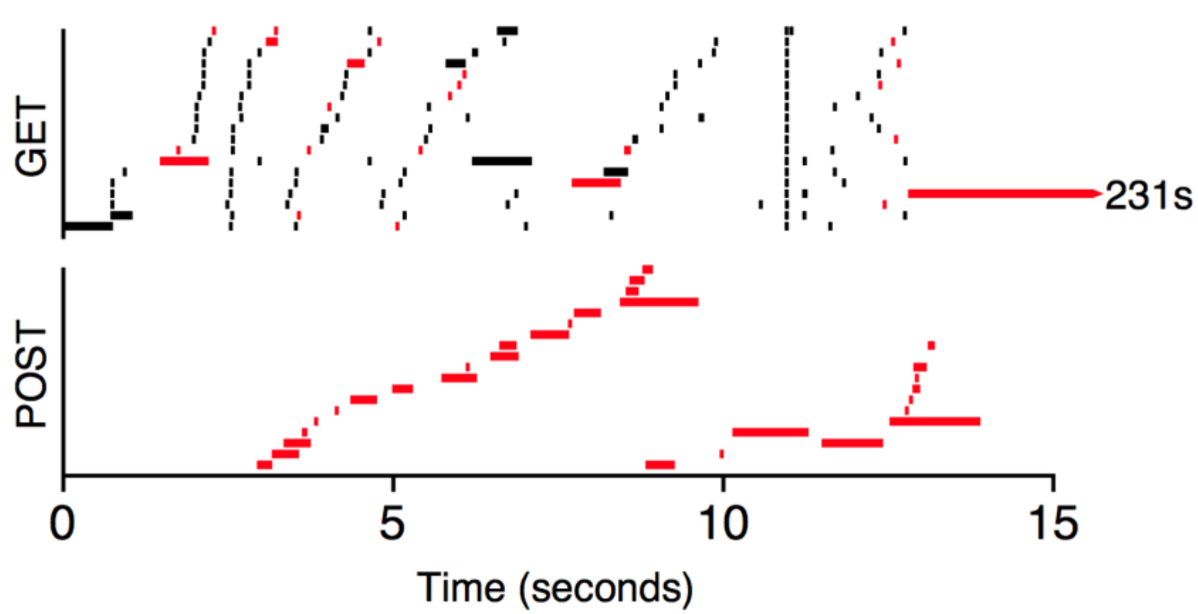
- e.g., Azure Functions

Build LambdaBench

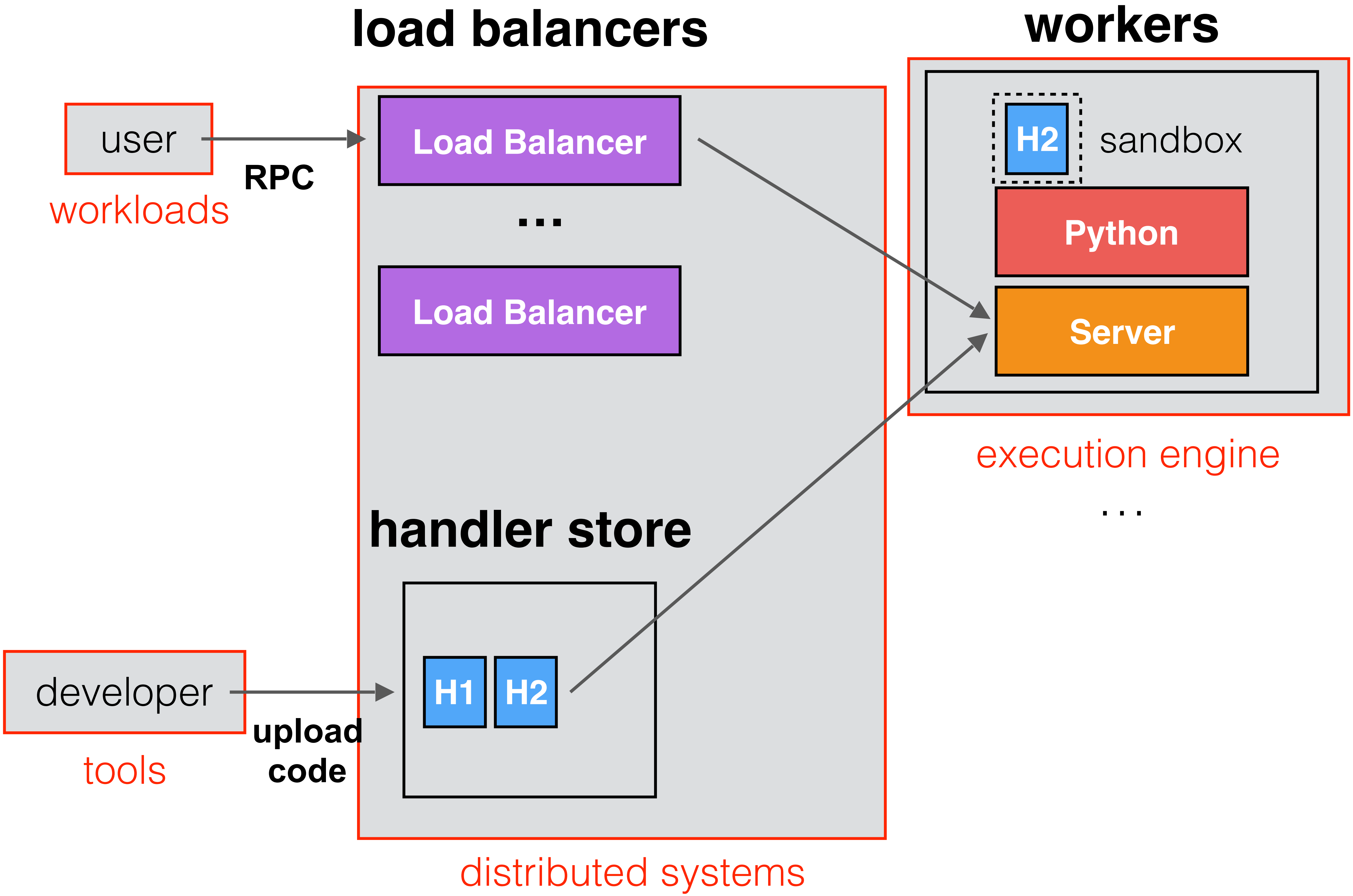
- Everybody joining builds an application
- Ticketing, calendar, autocomplete, OCR, flash card, stock alert, blog, and scientific compute applications

Trace RPC calls (e.g., AJAX) of existing apps

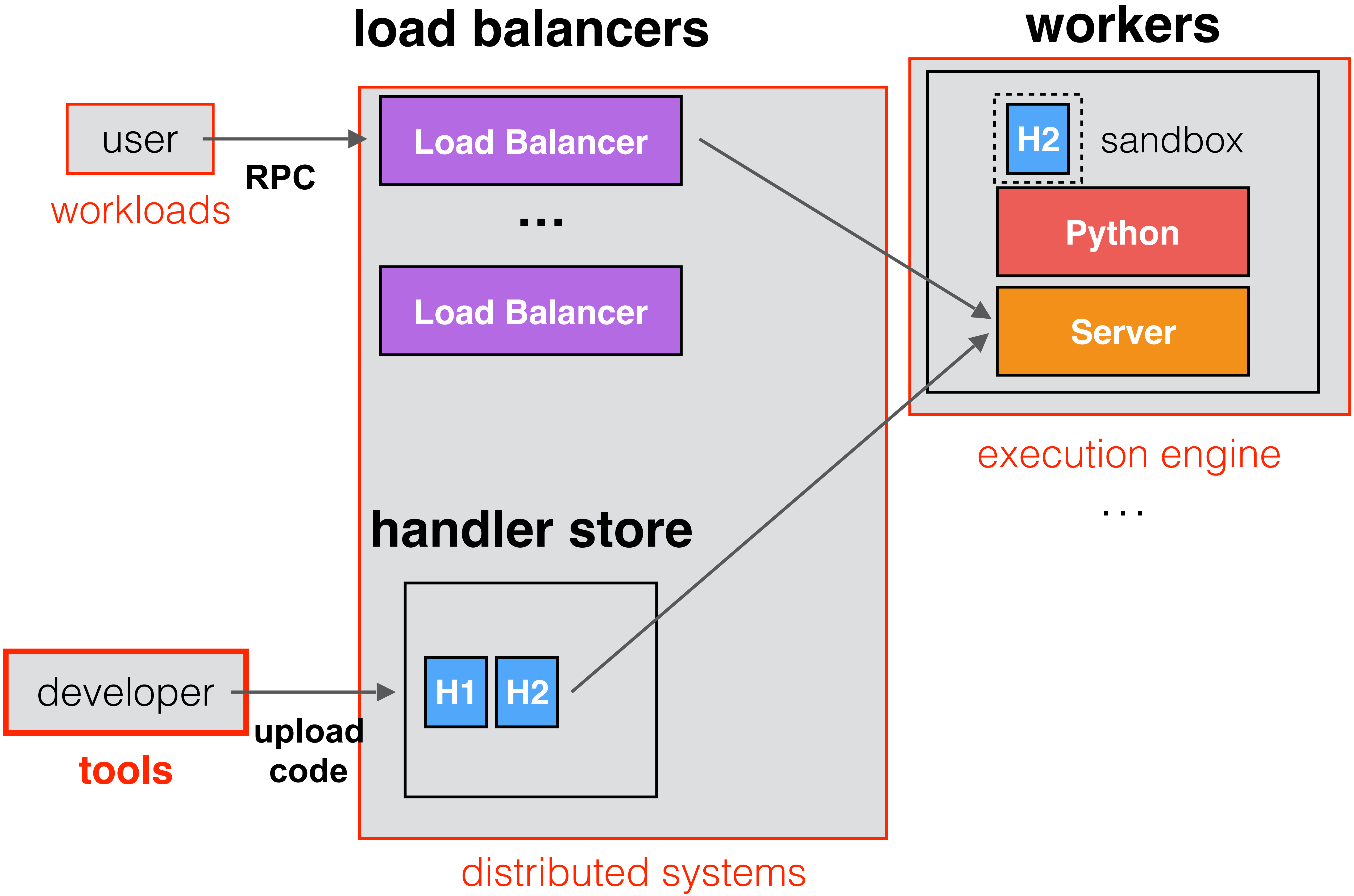
Gmail:



OpenLambda research topics



OpenLambda research topics



Developer tools

Portability

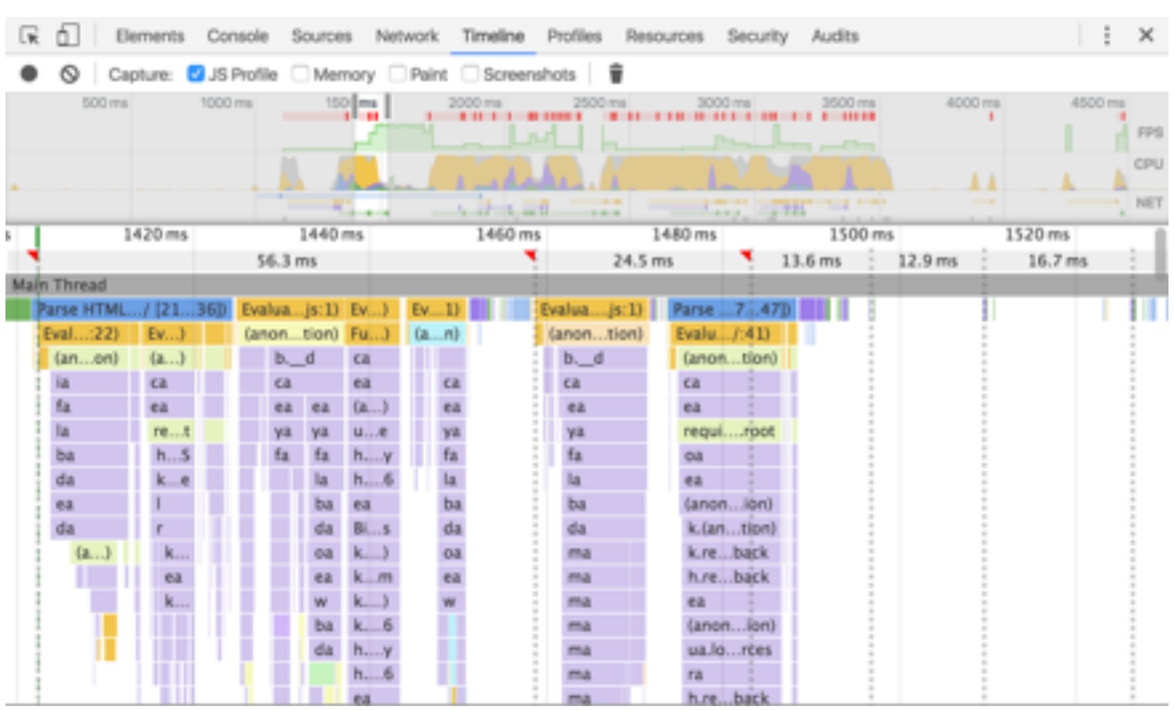
- E.g., can Django apps run on Lambdas?

Debugging

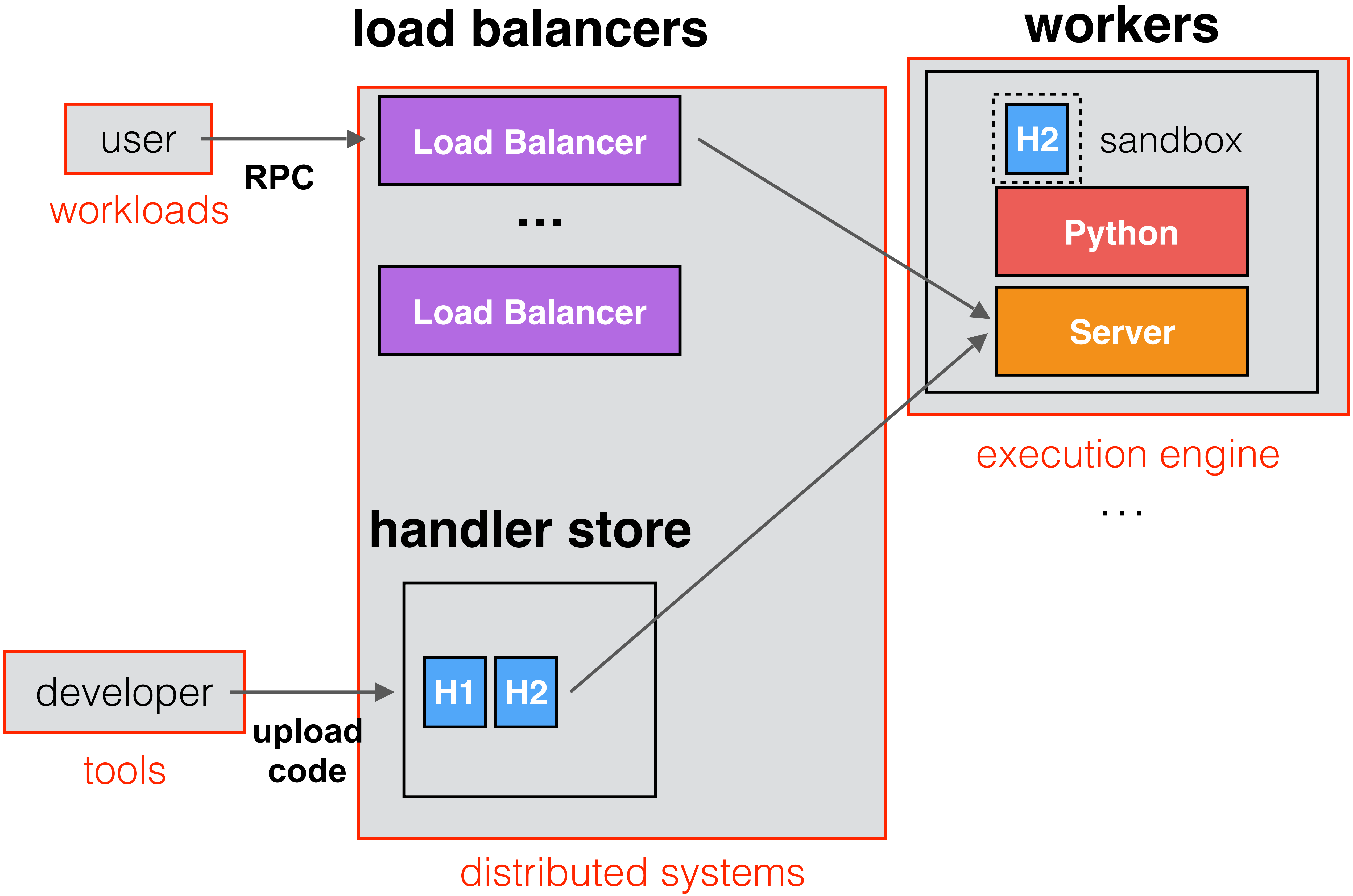
- Understand Lambda flows, may be a complex graph

Optimizing expense

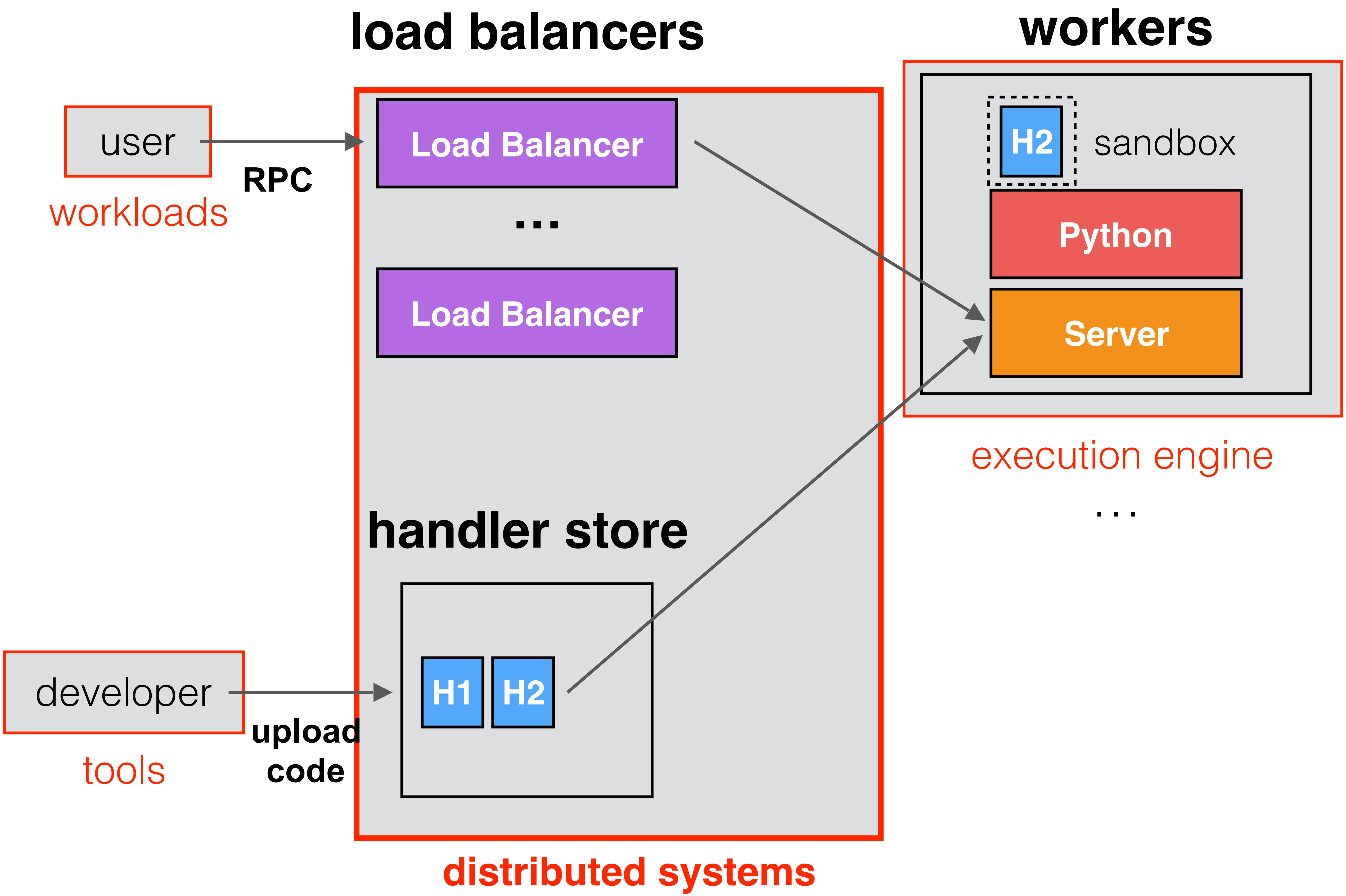
- Hard with containers: how to share 1-hour server time across requests?
- With Lambdas: know cost of every RPC and query
- Show where money is going



OpenLambda research topics



OpenLambda research topics



Building locality-aware Lambdas

Use deep inspection of RPCs for routing

- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Building locality-aware Lambdas

Use deep inspection of RPCs for routing

- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors

- code locality
- data locality
- session locality

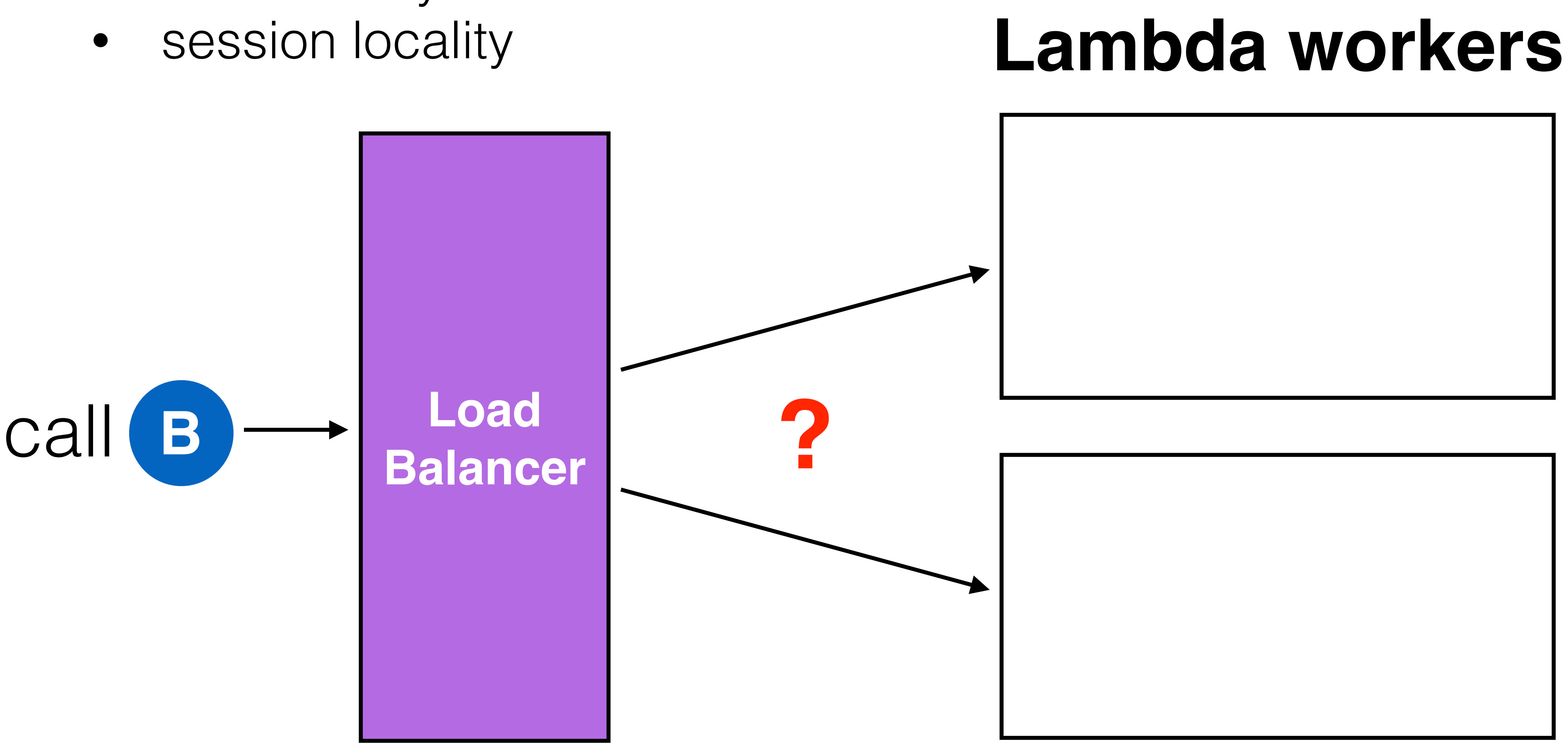
Building locality-aware Lambdas

Use deep inspection of RPCs for routing

- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors

- code locality
- data locality
- session locality



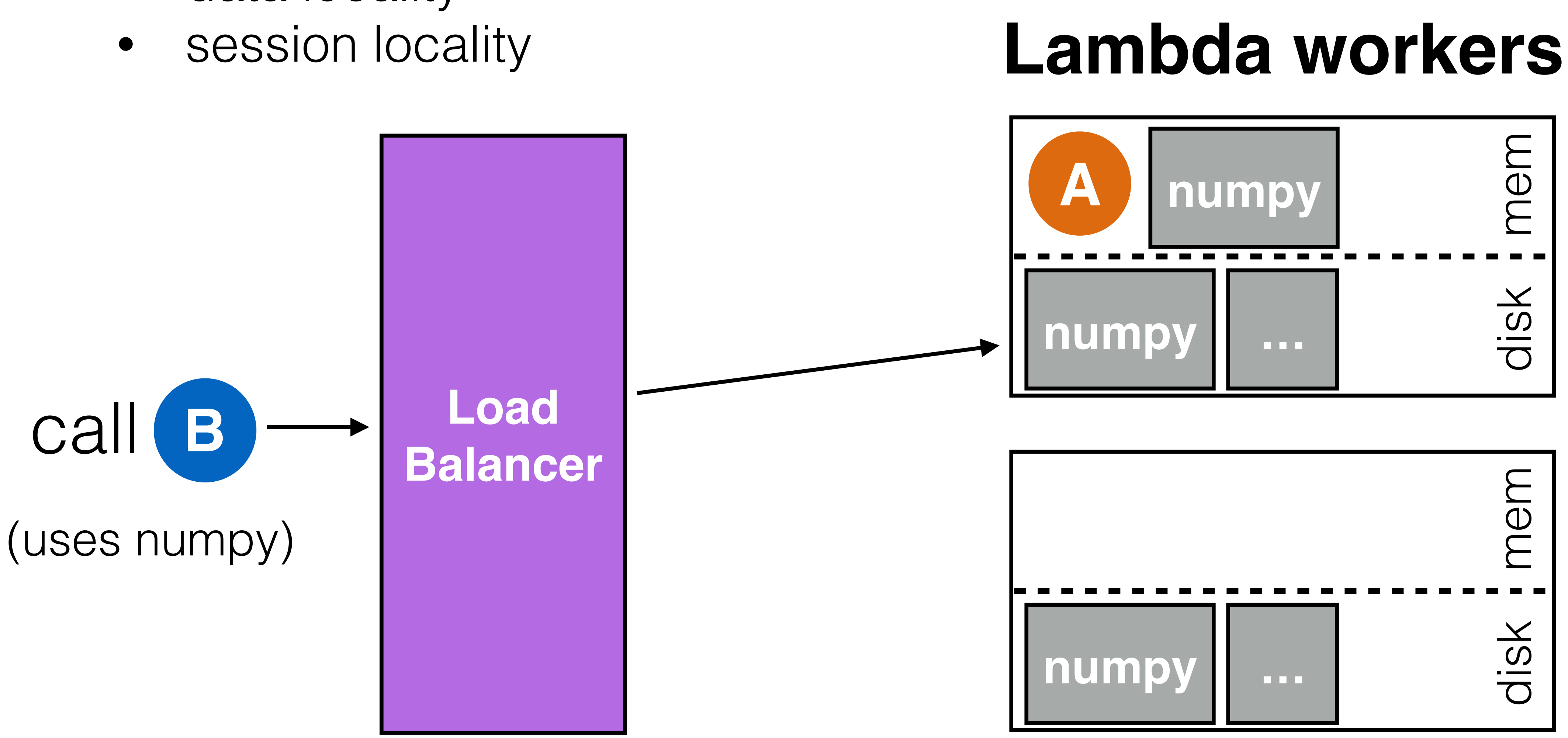
Building locality-aware Lambdas

Use deep inspection of RPCs for routing

- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors

- **code locality**
- data locality
- session locality



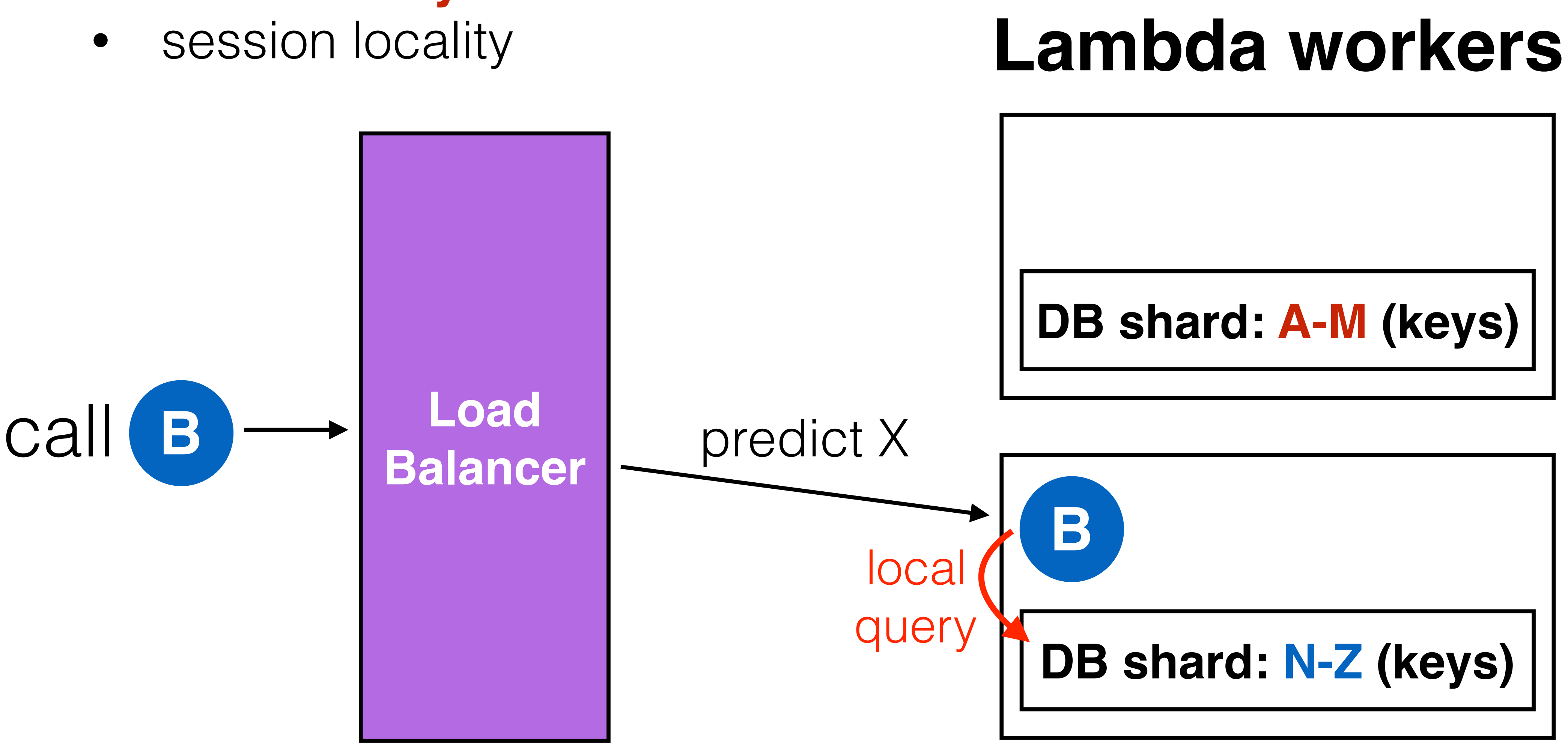
Building locality-aware Lambdas

Use deep inspection of RPCs for routing

- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors

- code locality
- **data locality**
- session locality



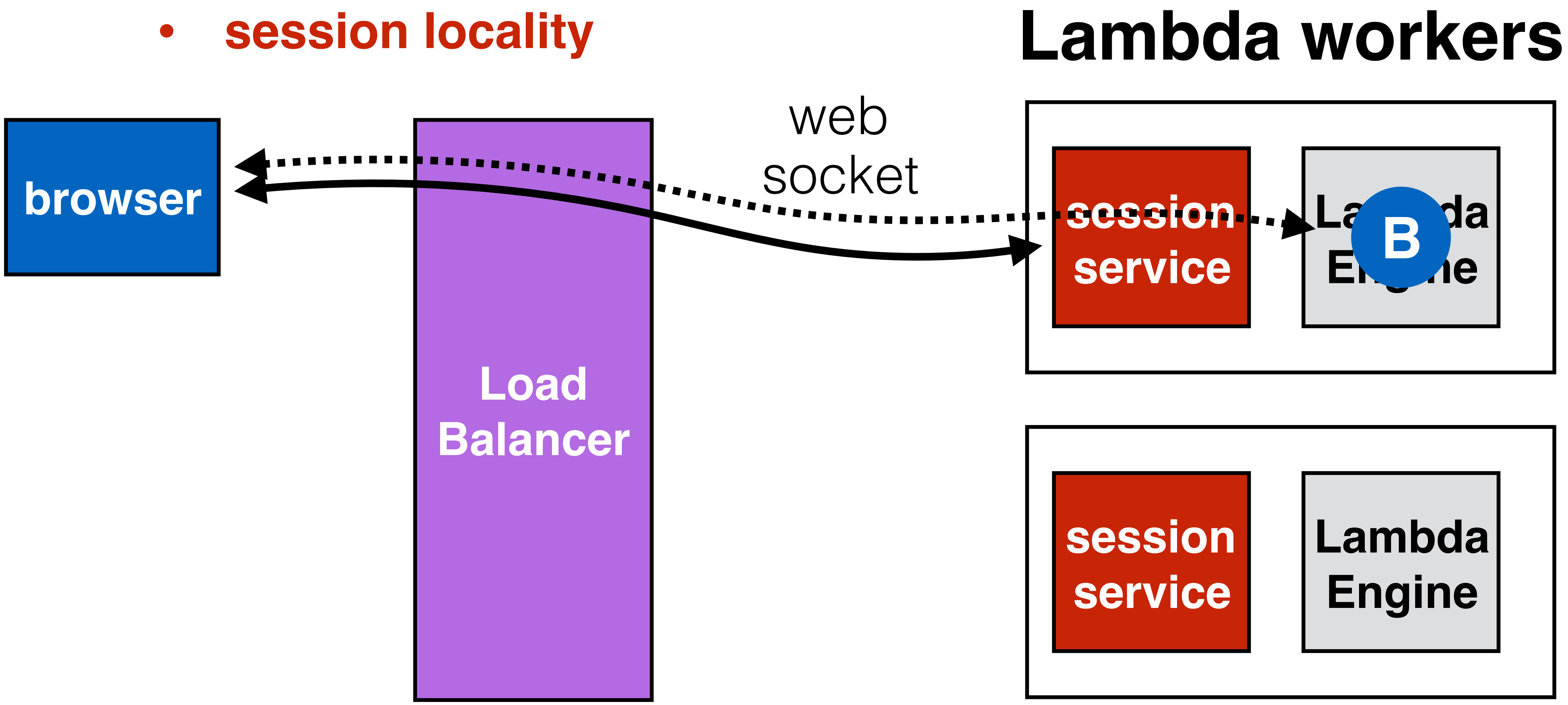
Building locality-aware Lambdas

Use deep inspection of RPCs for routing

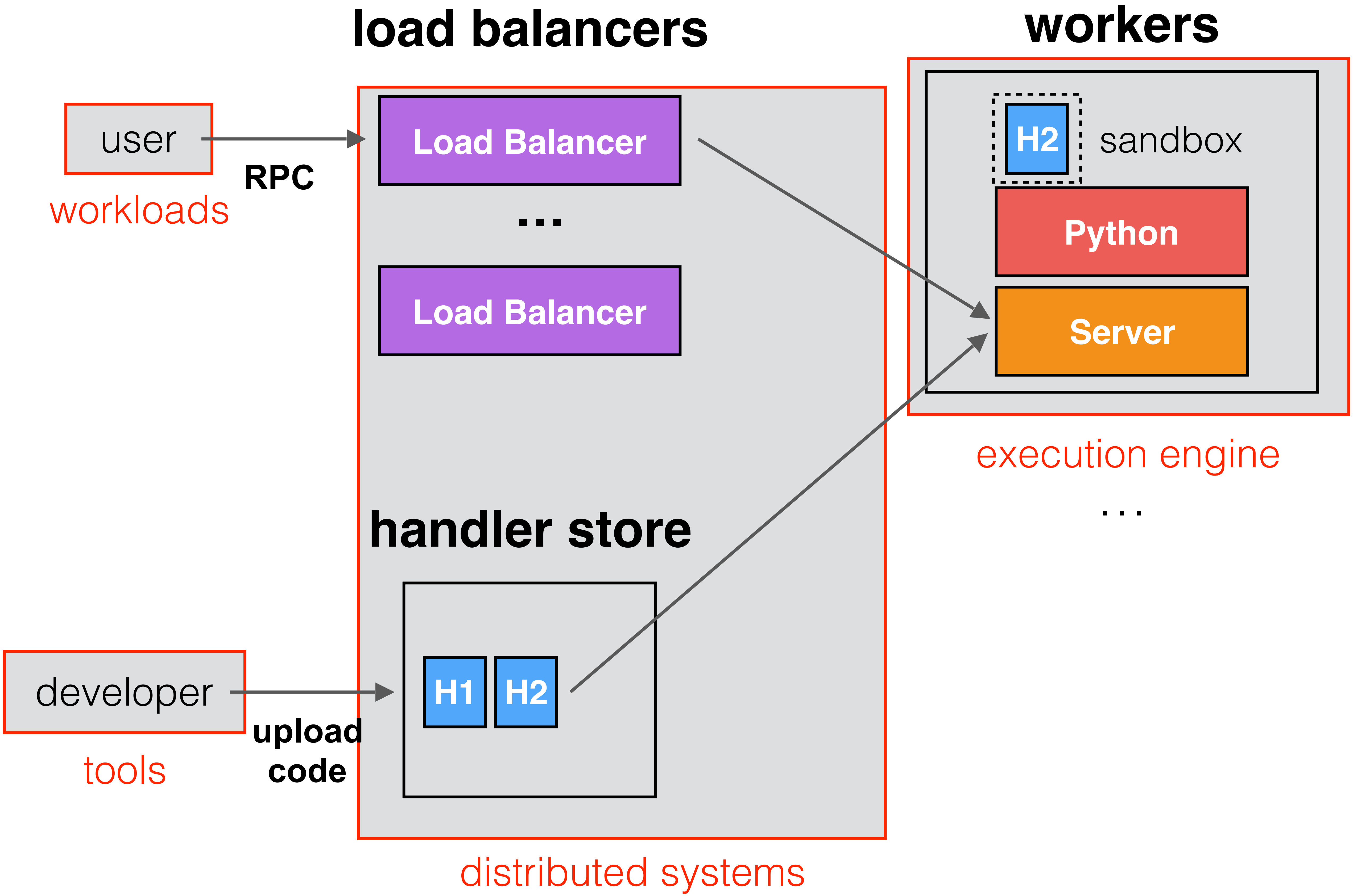
- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors

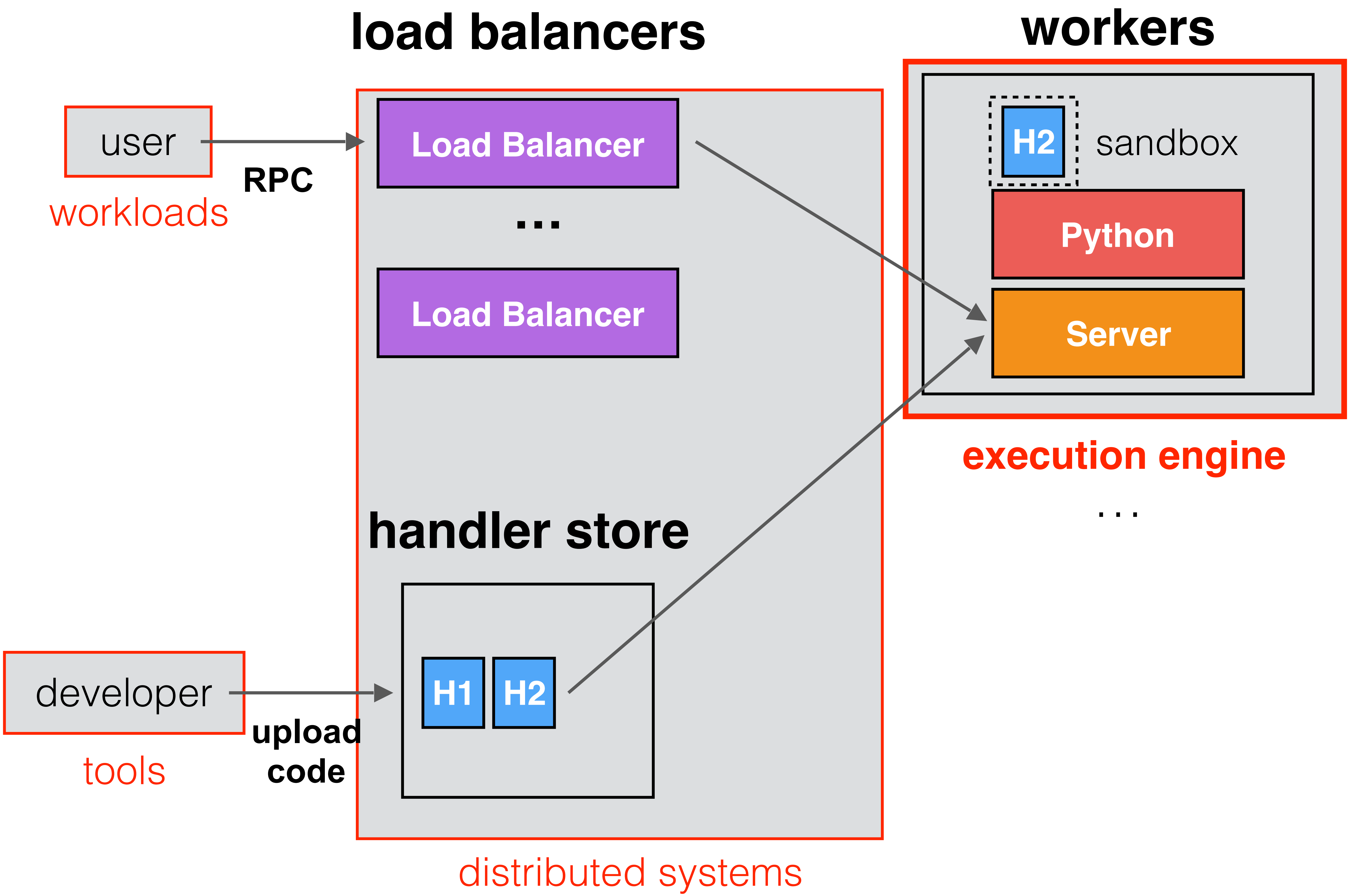
- code locality
- data locality
- **session locality**



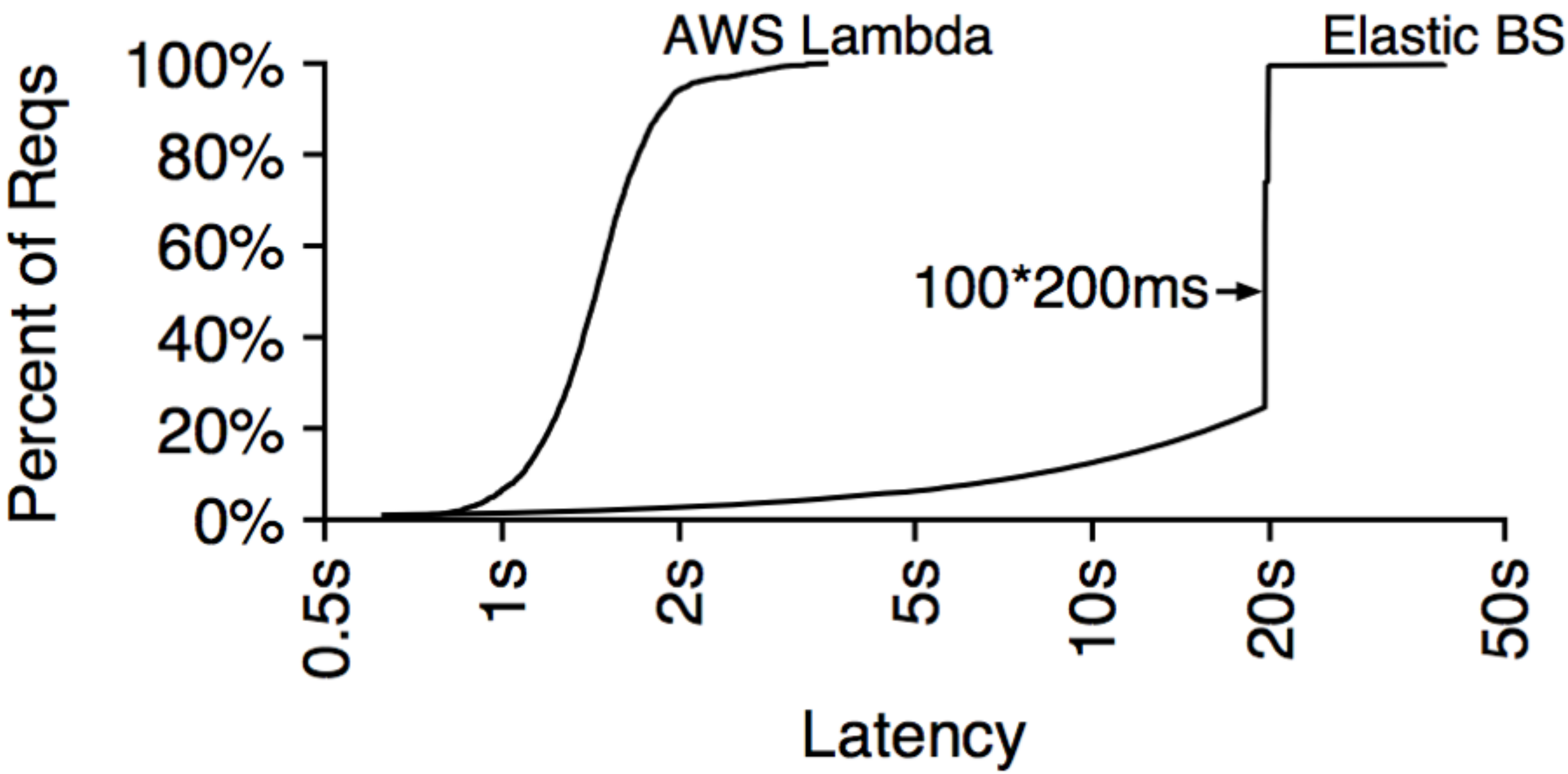
OpenLambda research topics



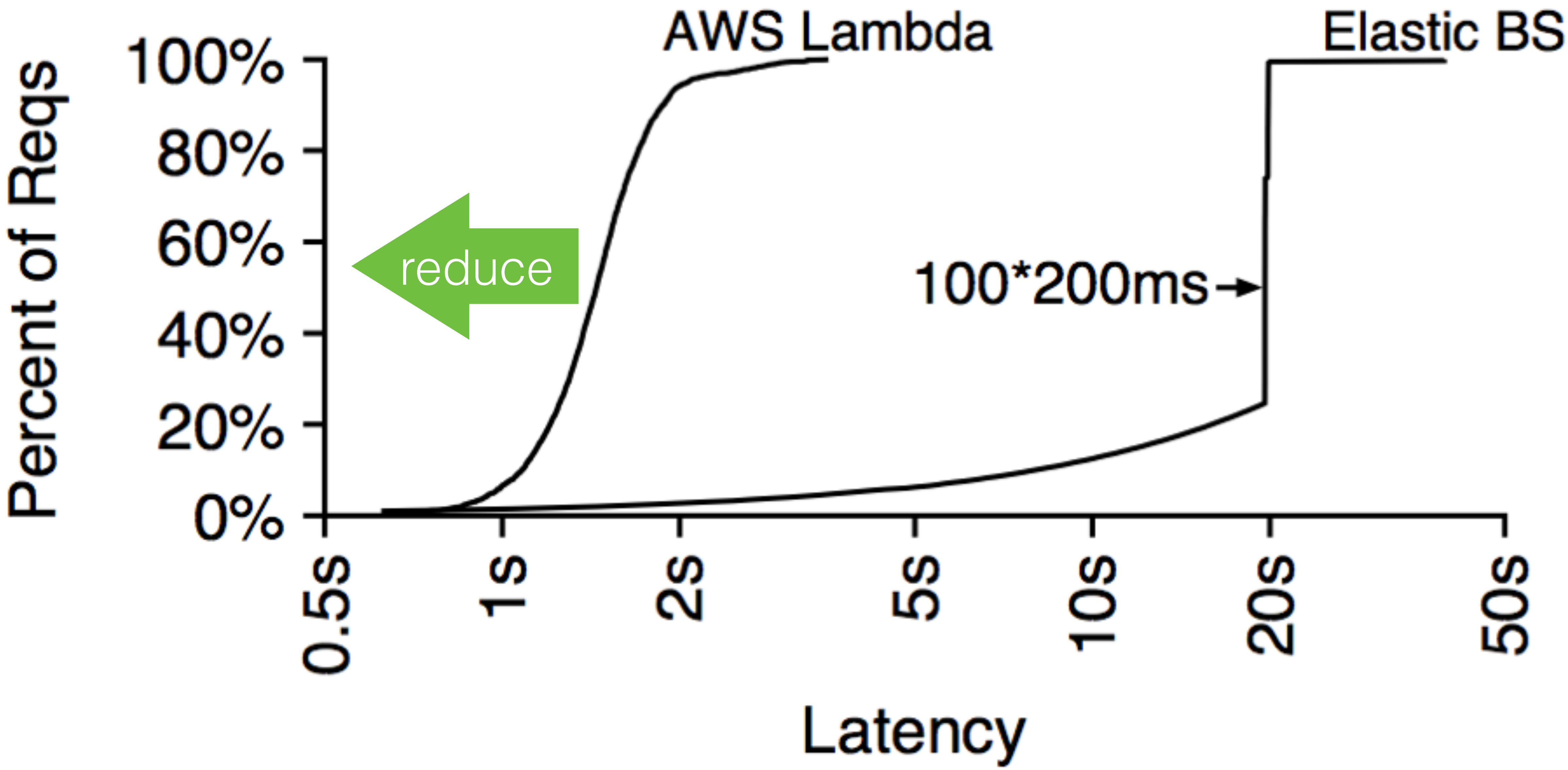
OpenLambda research topics



Minimizing latency



Minimizing latency



How can we reduce base latency?

Execution engine

Sandboxing

- Process VMs (e.g., JVM): how to mostly initialize?
- Containers: how to speed up **restart** and optimize **pausing**?

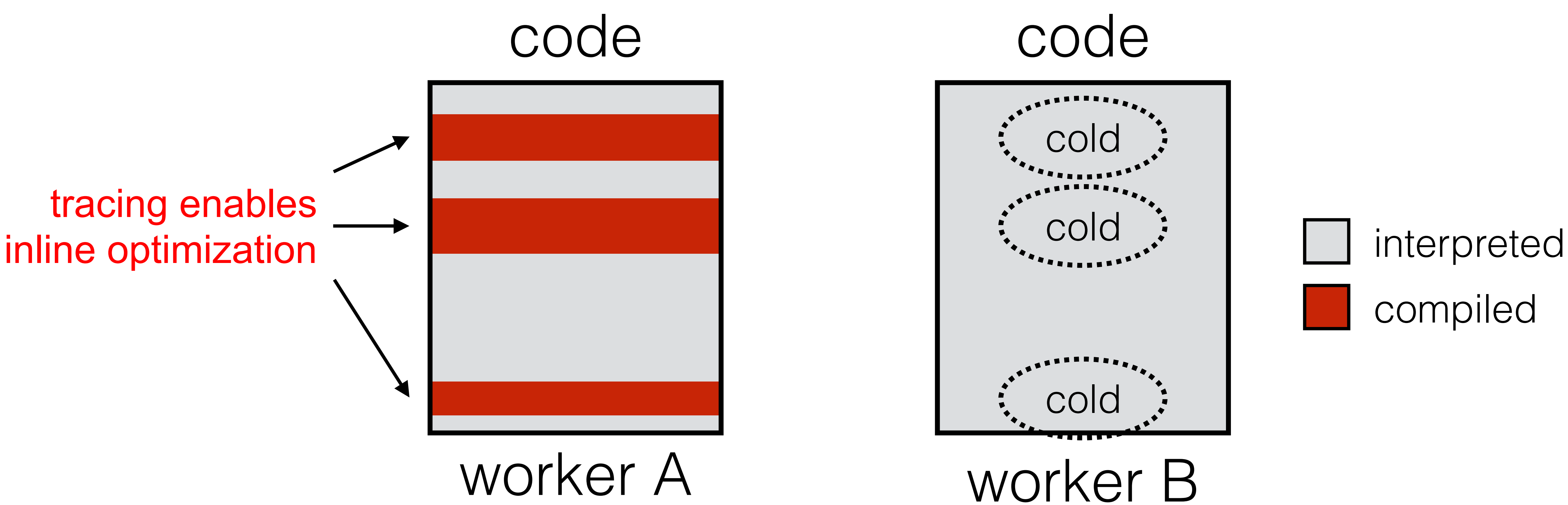
Execution engine

Sandboxing

- Process VMs (e.g., JVM): how to mostly initialize?
- Containers: how to speed up **restart** and optimize **pausing**?

Language runtimes

- Challenge: **code warms up** over time
- How to share dynamic optimizations?



Outline

Evolution of compute

Non-conventional virtualization

Lambda model

Why OpenLambda?

Conclusion

Conclusion

Lambdas finally deliver on promises of the cloud

- finally pay-as-you-go
- finally elastic
- will fundamentally change how people build scalable applications

New challenges in every area of systems

- scheduling, isolation, languages, debugging, tools, storage, …

Getting involved

- contribute at https://github.com/open-lambda
- site: https://open-lambda.org/

OPEN

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON