UBER RUSH

AND REBUILDING UBER'S DISPATCHING PLATFORM

QCon NEW YORK

UBER

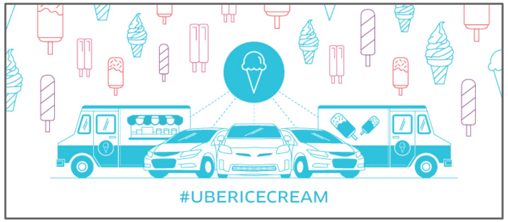
motivation

CHAPTER 1 OF 8

UBER

MOTIVATION TOWARDS MICROSERVICES

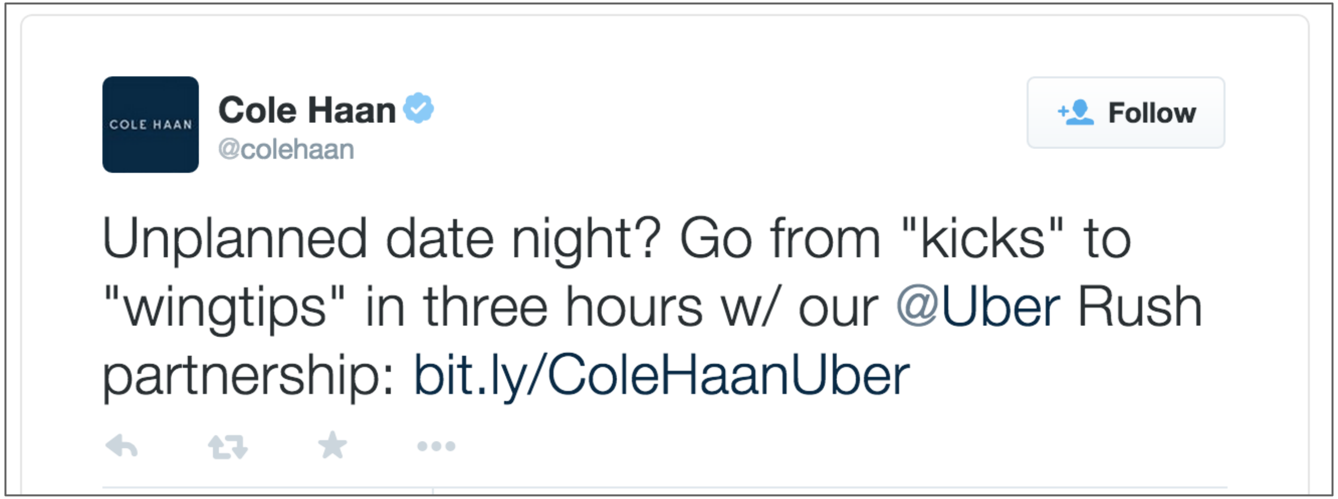
STUNTS AND EXPERIMENTS



UBER

UBER RUSH, DELIVERY SERVICE

MOTIVATION TOWARDS MICROSERVICES



Cole Haan
@colehaan

Unplanned date night? Go from "kicks" to "wingtips" in three hours w/ our @Uber Rush partnership: bit.ly/ColeHaanUber

UBER

UBER RUSH REQUIREMENTS

MOTIVATION TOWARDS MICROSERVICES

- MULTI-PICKUP
- BULK DELIVERIES
- MULTI-DISPATCH
- SOPHISTICATED MATCHING
- CAPACITY MANAGEMENT

NEW YORK

A RELIABLE RIDE FOR YOUR DELIVERIES

APRIL 7, 2014

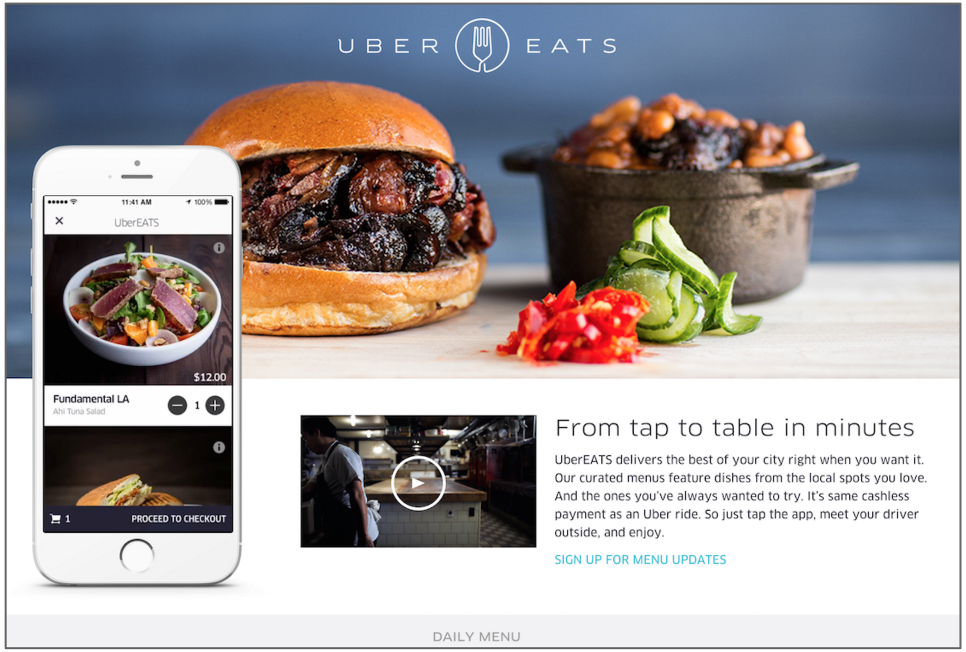
POSTED BY KIMIKO

With UberRUSH, your packages travel like a VIP. You get fast messenger pickups and immediate deliveries of the things you need to send.

UBER

UBER EATS REQUIREMENTS

MOTIVATION TOWARDS MICROSERVICES



From tap to table in minutes

UberEATS delivers the best of your city right when you want it. Our curated menus feature dishes from the local spots you love. And the ones you've always wanted to try. It's same cashless payment as an Uber ride. So just tap the app, meet your driver outside, and enjoy.

SIGN UP FOR MENU UPDATES

- NO PICKUP LOCATION
- TEMPERATURE REGULATION
- INVENTORY MANAGEMENT
- RE-SUPPLY STATIONS
- CHECKOUT FLOW

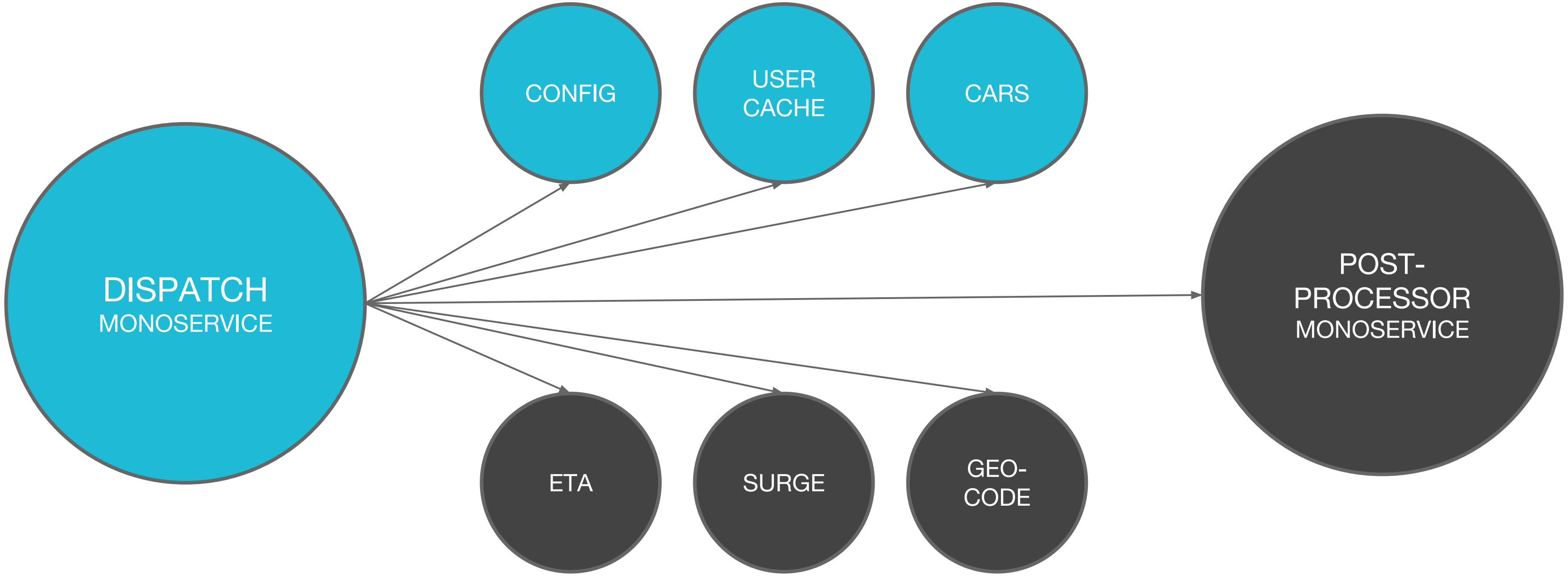
evolution

CHAPTER 2 OF 8

UBER

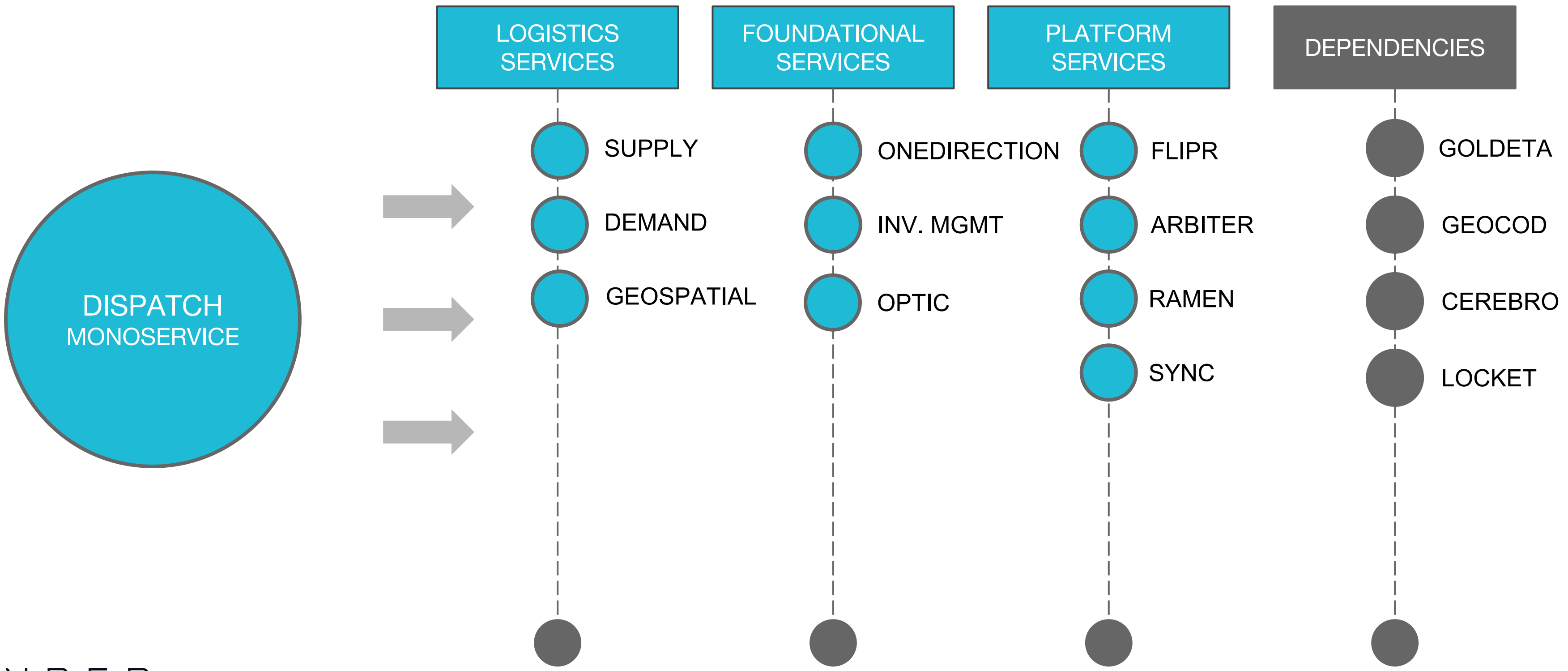
MONOSERVICE TO MICROSERVICES

MONOLITHIC ARCHITECTURE



1ST GENERATION MICROSERVICES

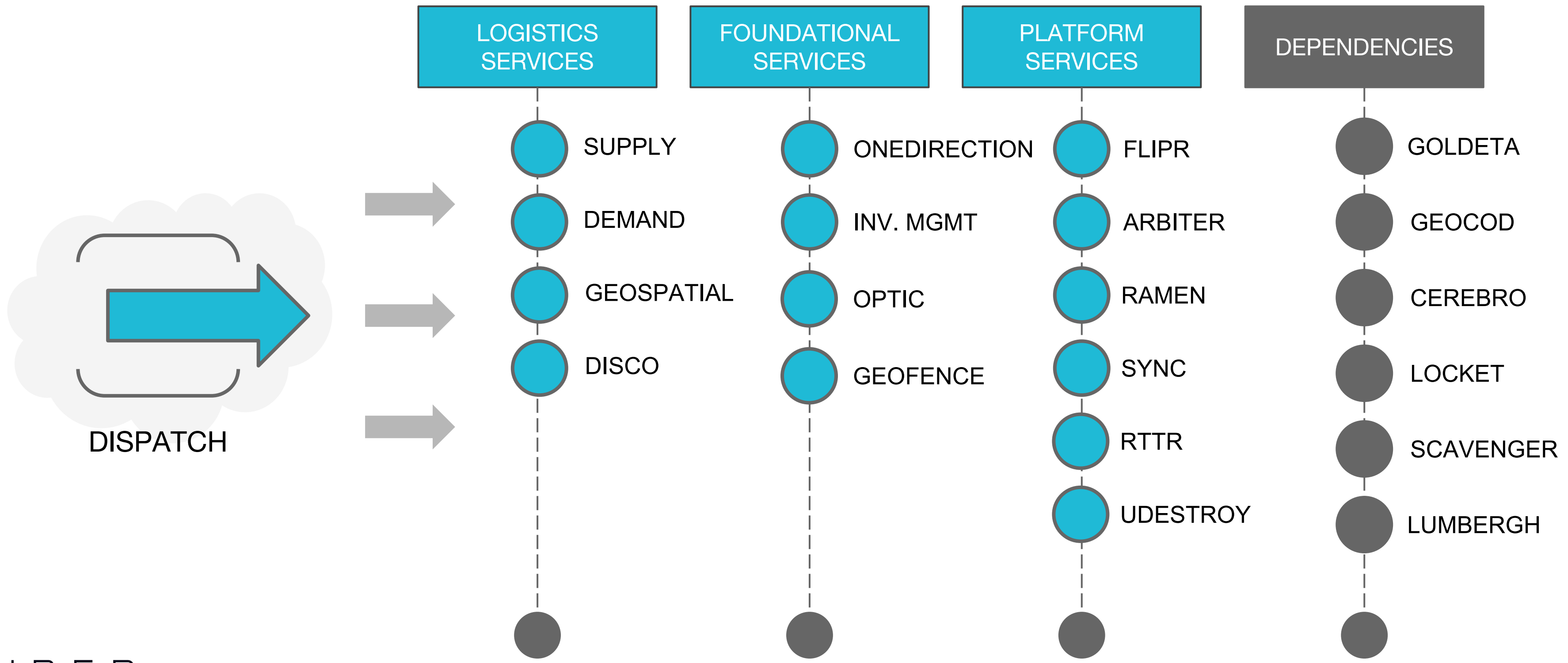
MONOSERVICE TO MICROSERVICES



UBER

2ND GENERATION MICROSERVICES

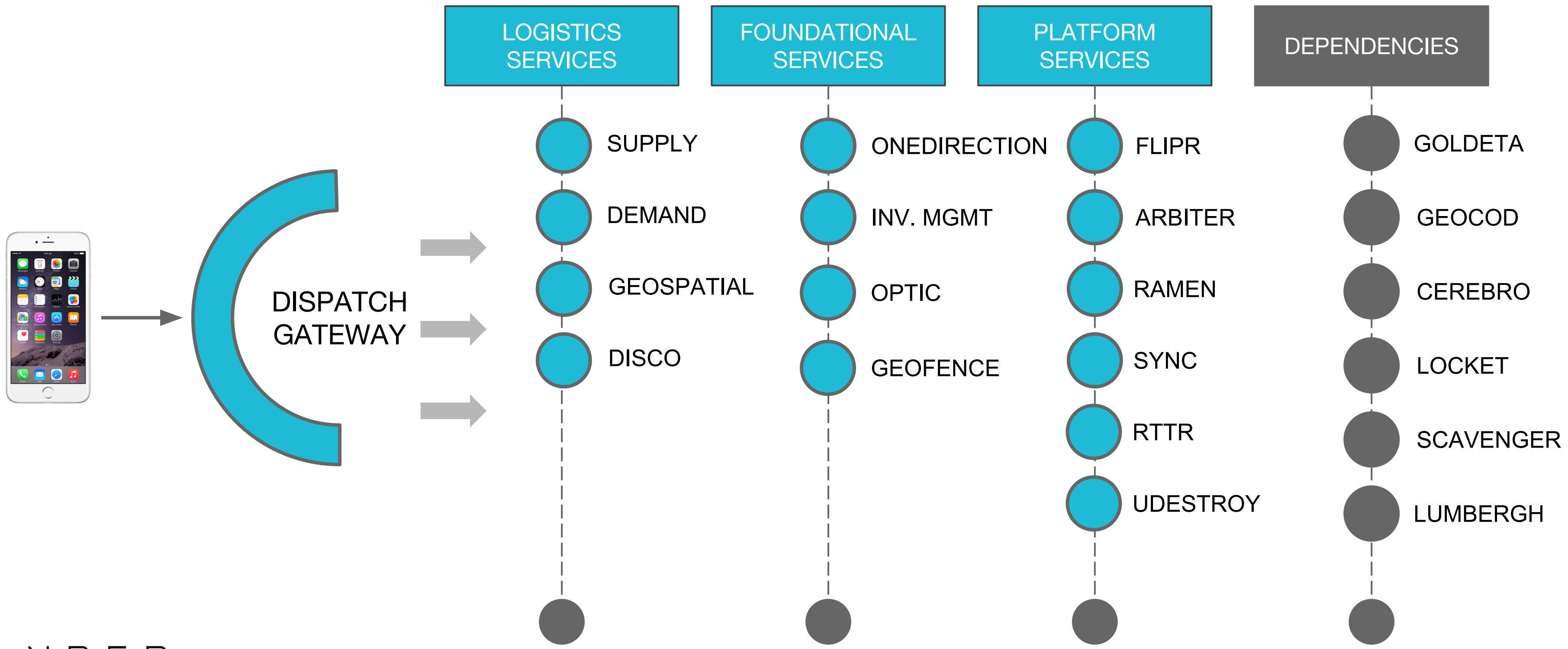
MONOSERVICE TO MICROSERVICES



UBER

A MICROSERVICE GATEWAY

MONOSERVICE TO MICROSERVICES



UBER

MOTIVATION TOWARDS MICROSERVICES

THE TRADE-OFFS

MONOSERVICE vs. MICROSERVICE

- UPGRADES ARE PAINFUL
- TEST SUITE IS SLOW
- FAILURE IS CATASTROPHIC
- CODE IS BRITTLE
- DEPLOYS ARE SLOW

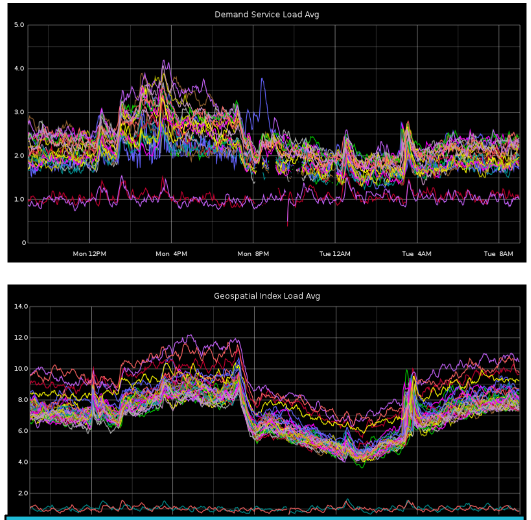
UBER

topologies

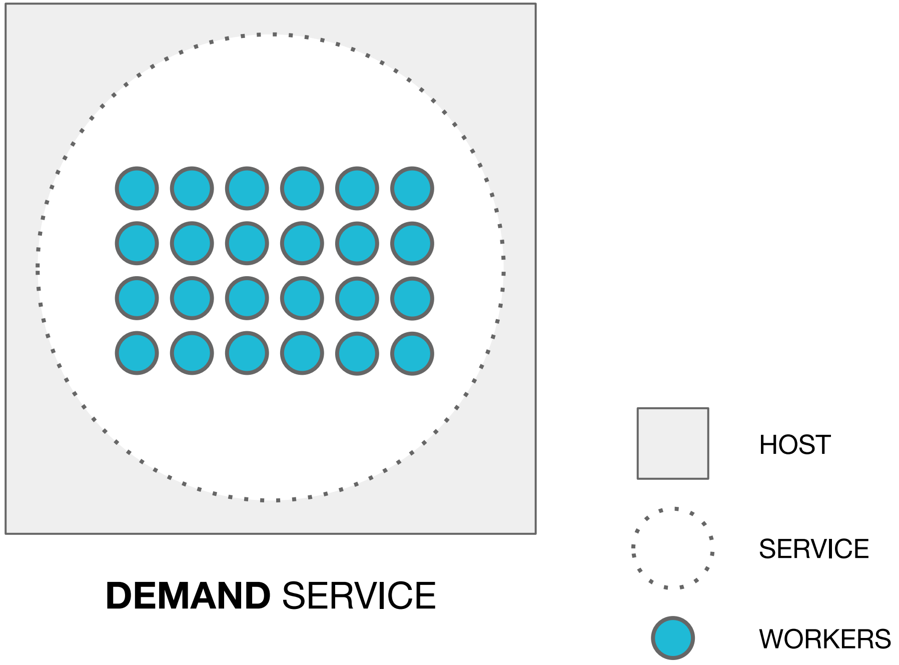
CHAPTER 3 OF 8

MICROSERVICE LAYOUT

INDEPENDENT, INDIVIDUALLY ADDRESSABLE SERVERS



MICROSERVICE LOAD AVERAGE

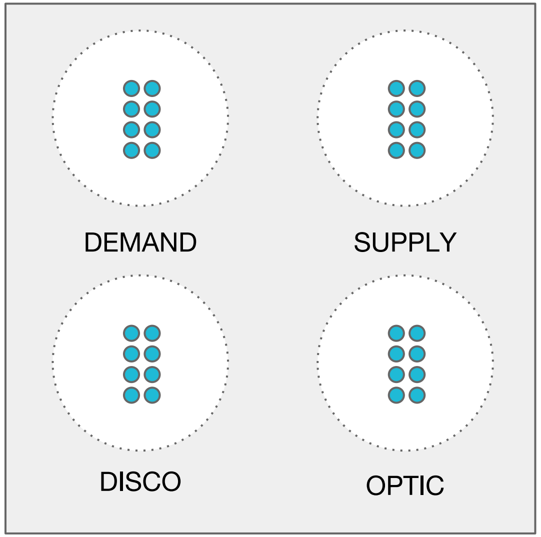


DEMAND SERVICE

UBER

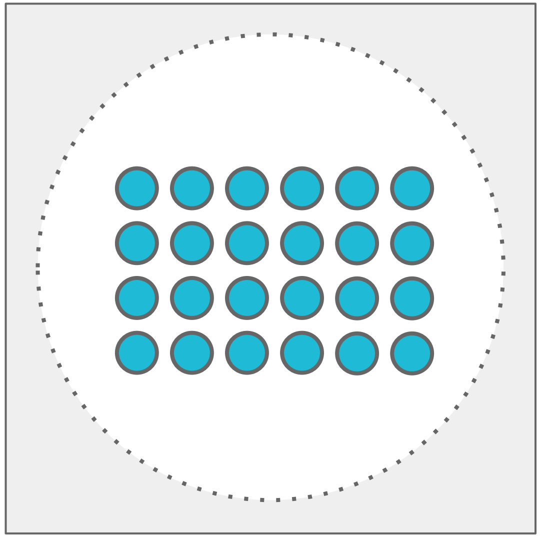
ARRANGEMENT OF MICROSERVICES

MULTI-TENANT OR DEDICATED HOSTS?



MULTI-TENANT HOSTS

OR



DEDICATED **DEMAND** HOST

UBER

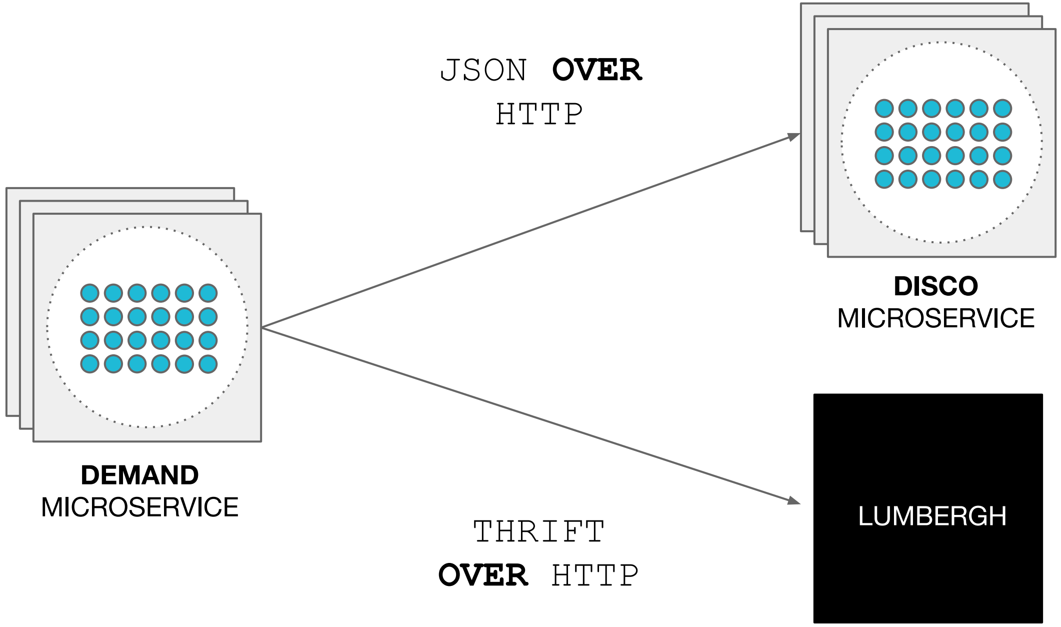
communications and fault tolerance

CHAPTER 4 OF 8

UBER

MANAGING MICROSERVICE DEPENDENCIES

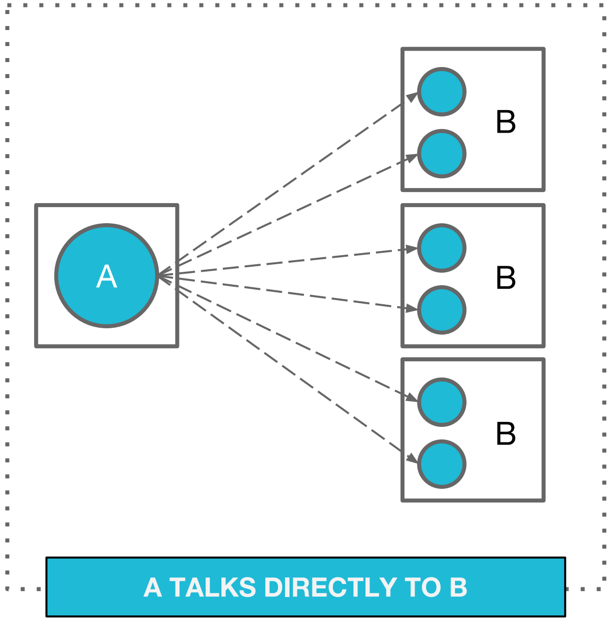
AUTO-GENERATED CLIENTS



UBER

LOAD-BALANCING MICROSERVICES

WITH CLIENT-SIDE LOAD-BALANCING

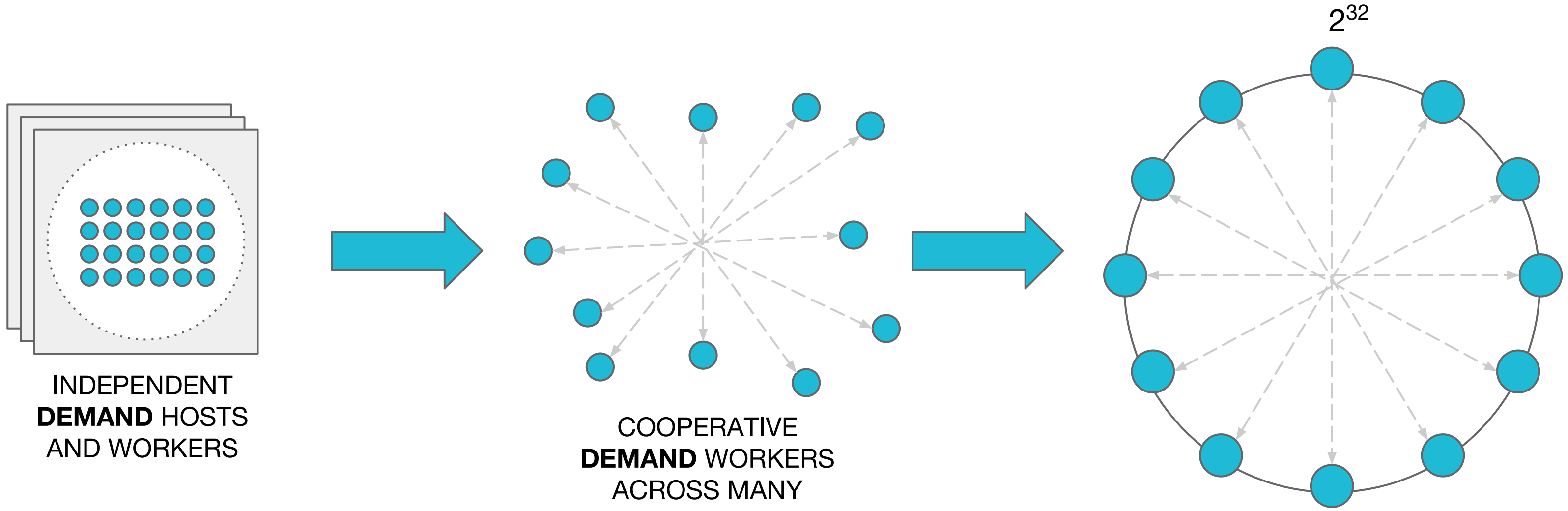


- HOT HOST FILE RELOADING
- HEALTH CHECKING
- PERSISTENT CONNECTIONS
- CONNECTION POOLING
- RETRIES

UBER

COOPERATIVE MICROSERVICE INSTANCES

FROM INDEPENDENT WORKERS TO COOPERATIVE



INDEPENDENT **DEMAND** HOSTS AND WORKERS

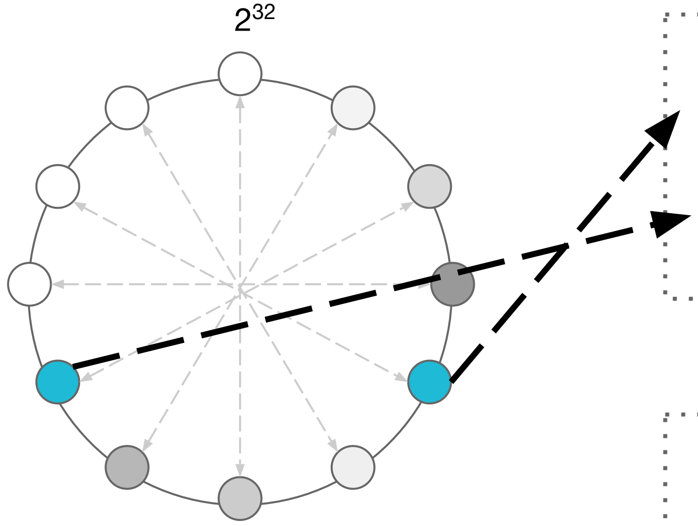
COOPERATIVE **DEMAND** WORKERS ACROSS MANY HOSTS

GOSSIP WITH ONE ANOTHER AND MAINTAIN A **HASH RING** OF EACH WORKER

UBER

COOPERATIVE MICROSERVICE INSTANCES

WITH RINGPOP @ GITHUB.COM/UBER/RINGPOP



EACH **DEMAND** WORKER OWNS A PORTION OF THE KEYSPACE

HASH WORKER ADDRESSES

```
> hash('10.31.1.2:9000')
53554892

> hash('10.31.8.9:9000')
1325776234
```

HASH APPLICATION IDS

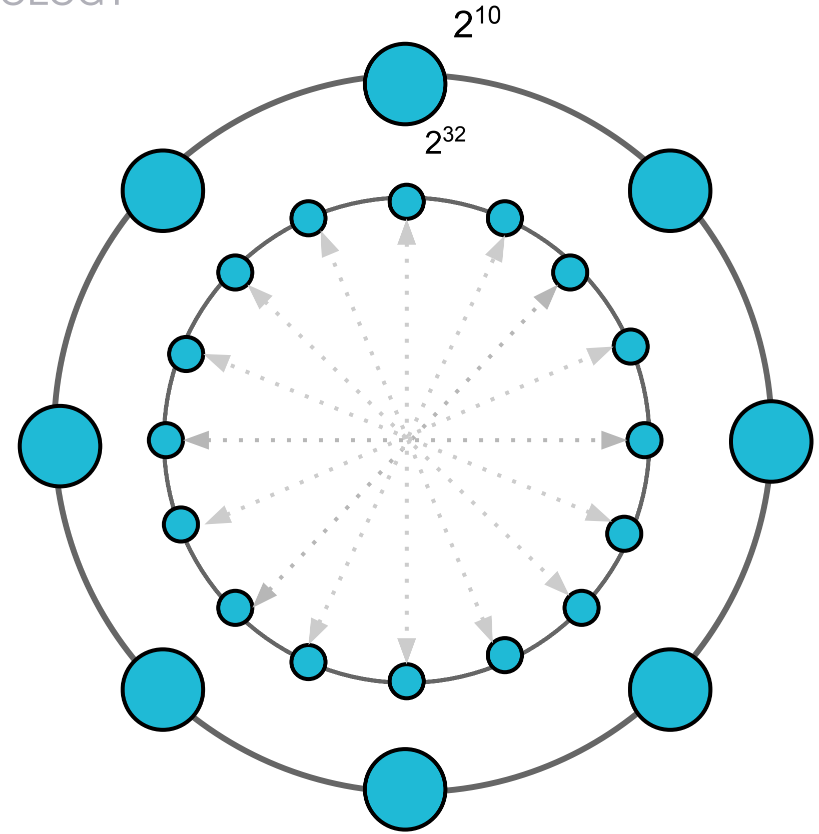
```
> hash('33e2dc8c-16fd-4a19-9fad-4ebfc76c66c9')
2312992577

> hash('8828169c-69c5-4b79-ae5e-6204c5f615ff')
2640491360
```

UBER

RELIABLE BACKGROUND OPERATIONS

WITH HASH RING TECHNOLOGY

VNODE KEYSPACE (OUTER RING)
FIXED
AND SMALLER

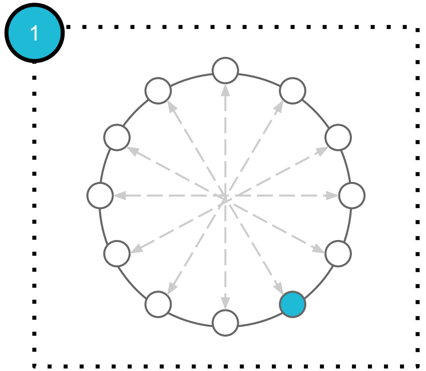


ENTITY KEYSPACE (INNER RING)
DYNAMIC
AND LARGER

UBER

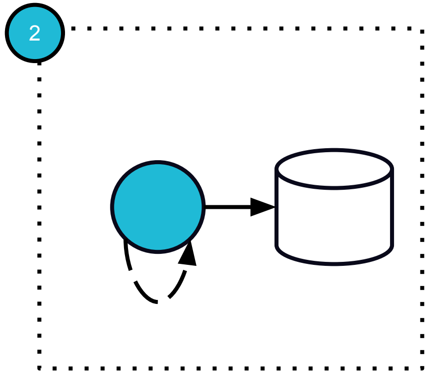
RELIABLE BACKGROUND OPERATIONS

WITH HASH RING TECHNOLOGY



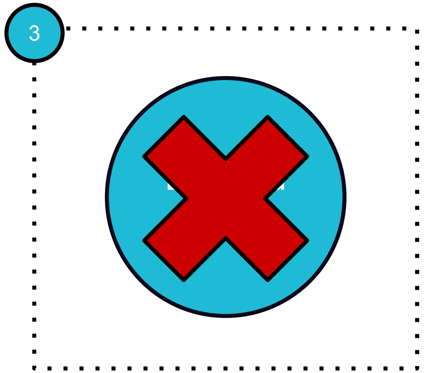
DEMAND A WORKER
RECEIVES DELIVERY &
INITIATES DISPATCH

`POST /jobs`



DEMAND A WORKER
WRITES UUID TO VNODE
SET IN THE DB AND
STARTS TIMER

`hash(uuid) % 1024`

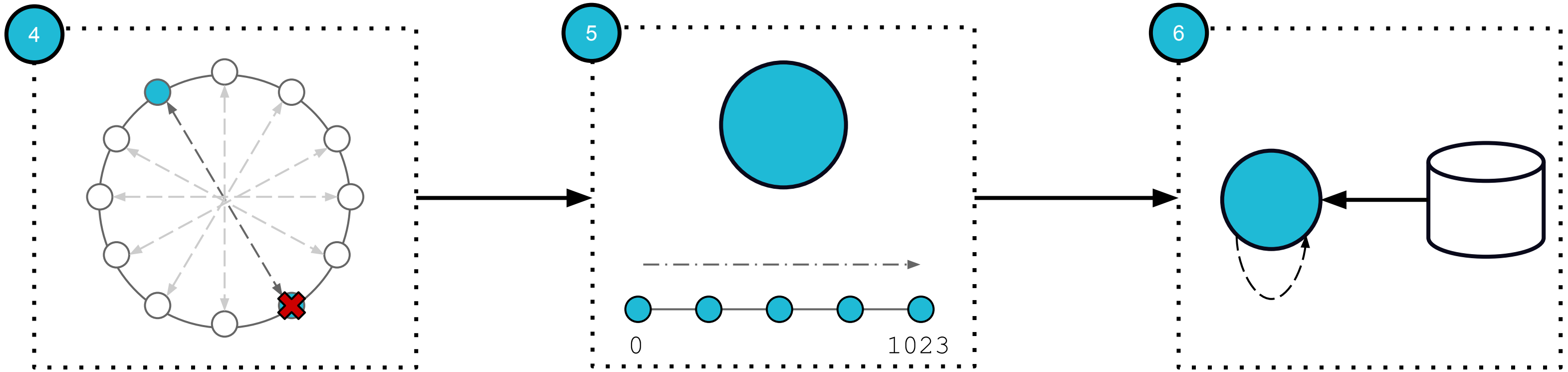


DEMAND A WORKER
CRASHES BEFORE IT
EXPIRES DISPATCH

UBER

RELIABLE BACKGROUND OPERATIONS

WITH HASH RING TECHNOLOGY



DEMAND B DETECTS MEMBERSHIP CHANGE IN RING

DEMAND B SCANS ENTIRE VNODE KEYSPACE

DEMAND B LOADS VNODE SET FROM DB AND RESTORES BACKGROUND TIMERS

```
for vnode in range(0, 1023)
  if lookup(vnode) == whoami()
    restore(load_uuids(vnode))
```

UBER

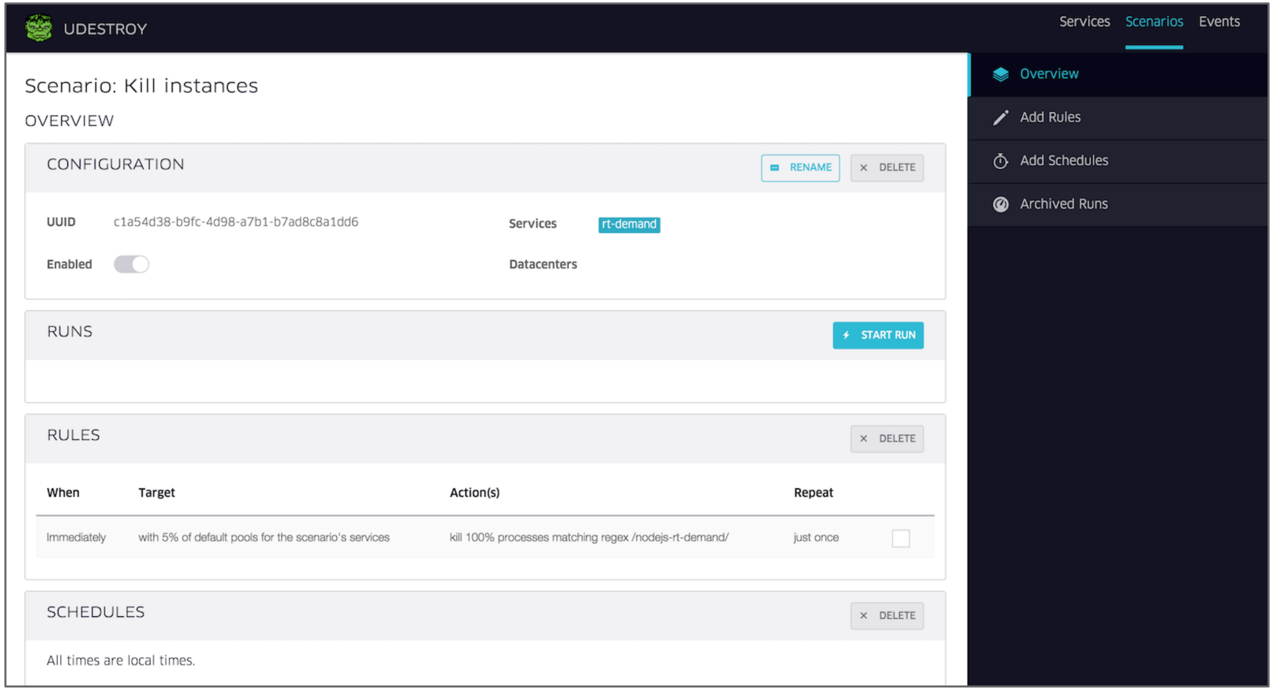
failure, monitoring and alerting

CHAPTER 5 OF 8

UBER

FAILURE TESTING MICROSERVICES

WITH REPEATABLE FAILURE SCENARIOS



Scenario: Kill instances

OVERVIEW

CONFIGURATION

UUID c1a54d38-b9fc-4d98-a7b1-b7ad8c8a1dd6

Services rt-demand

Enabled

Datacenters

RUNS

RULES

When	Target	Action(s)	Repeat
Immediately	with 5% of default pools for the scenario's services	kill 100% processes matching regex /nodejs-rt-demand/	just once

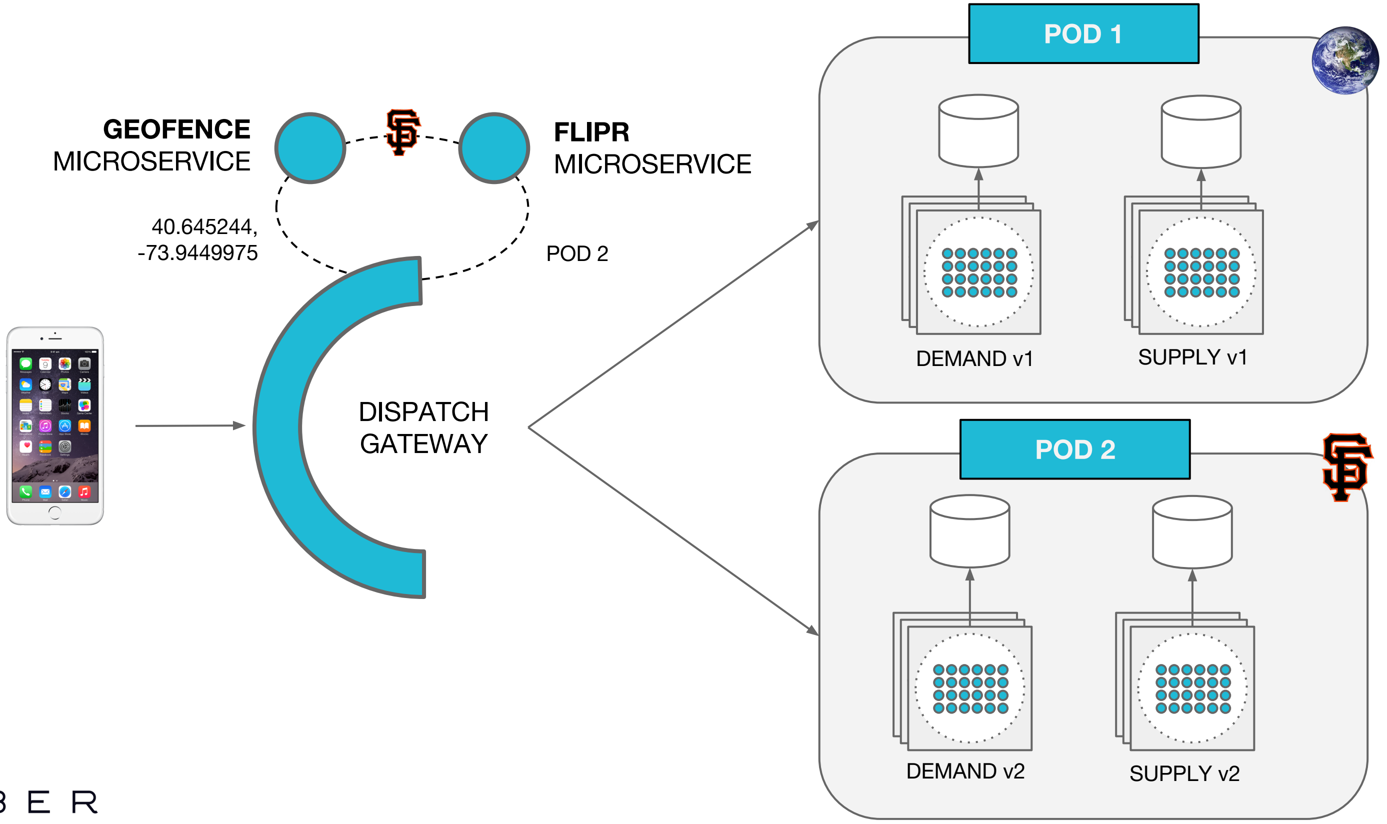
SCHEDULES

All times are local times.

UBER

FAULT ISOLATION IN MICROSERVICES

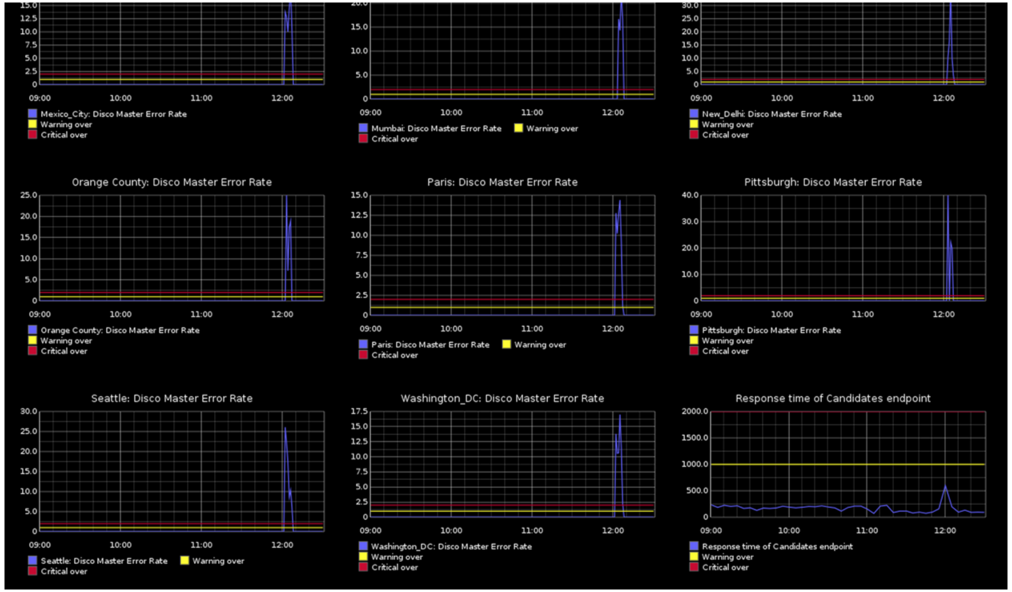
WITH DEPLOYMENT PODS



UBER

MICROSERVICE ALERTING

WITH GRAPHITE/NAGIOS INTEGRATED THRESHOLD CHECKS



- PER REPO THRESHOLDS
- IMPORTED PYTHON
- BUILT AGAINST GRAPHITE
- ALERTS THROUGH NAGIOS

UBER

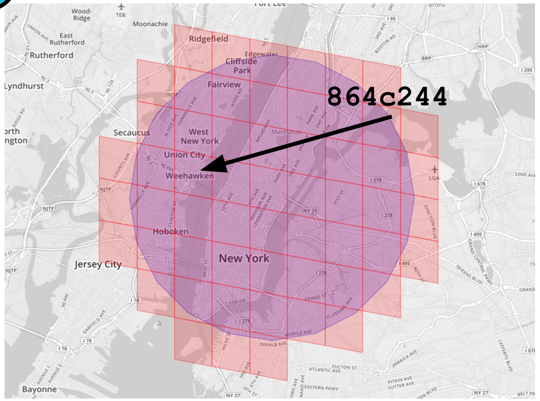
scalability and sharding

CHAPTER 6 OF 8

PARTITIONING A MICROSERVICE

A SCALABLE GEOSPATIAL INDEX

1



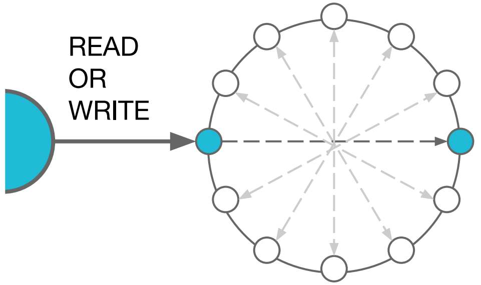
EARTH IS BROKEN UP INTO **CELLS**. EACH CELL HAS AN ID.

2

```
> convert(40.645, -73.944)
"864c244"

> hash("864c244")
3747631425

> lookup(3747631425)
"10.31.1.2:9000"
```

GEOSPATIAL READS/WRITES **CONVERTS** LAT/LNG TO **CELL ID**. CELL ID IS THEN **HASHED** ALONG **RING**.

3

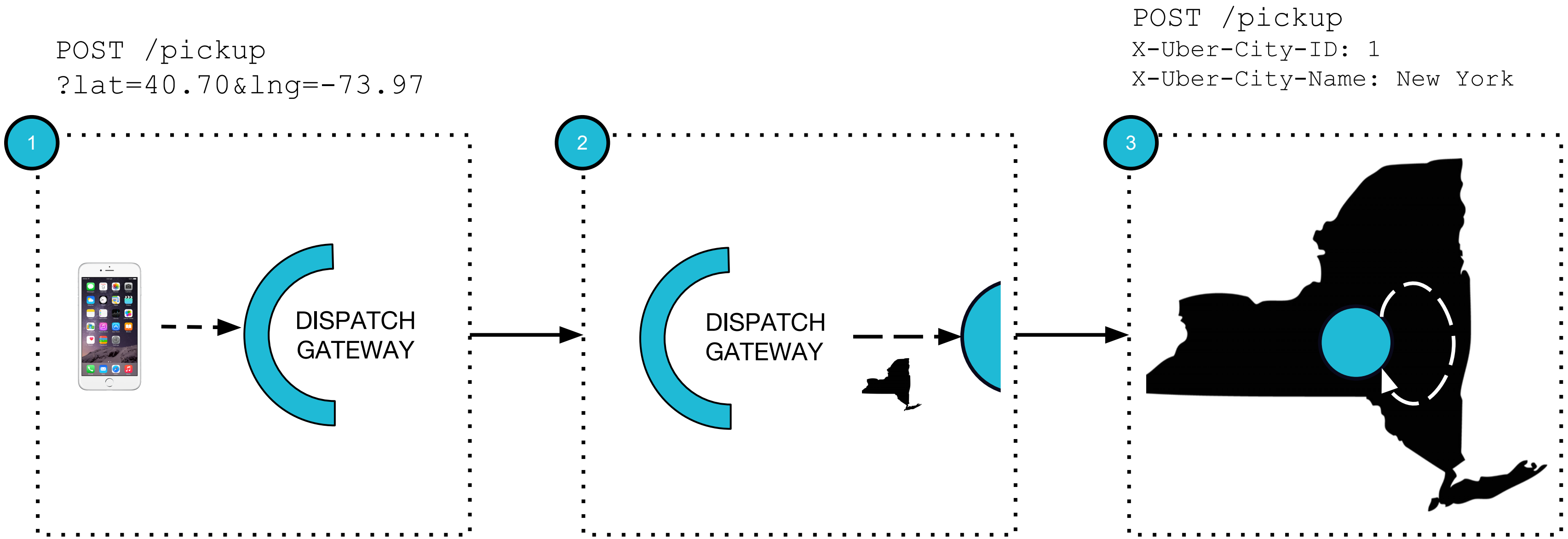


REQUEST IS EITHER **HANDLED** OR **FORWARDED** BY ONE OF THE 1300 GEOSPATIAL INDEX WORKERS.

UBER

LOCATION-AWARE MICROSERVICES

WITH CONTEXT-SPECIFIC METADATA



DEVICE SENDS PICKUP REQUEST TO DISPATCH GATEWAY

DISPATCH GATEWAY **RESOLVES** LAT/LNG AGAINST GEOFENCE SERVICE

SERVICE PROCESSES REQUEST WITHIN **CONTEXT** OF CITY

UBER

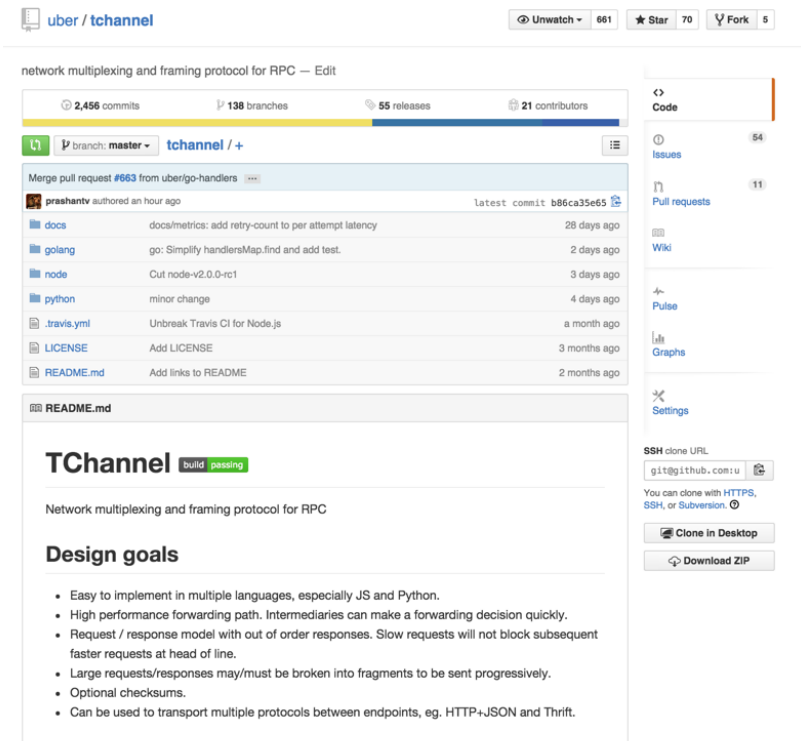
performance and diagnostics

CHAPTER 7 OF 8

HIGH-PERFORMANCE MICROSERVICES

WITH TCHANNEL @ GITHUB.COM/UBER/TCHANNEL

- PERFORMANT
- MULTIPLEXING
- STREAMING
- RETRIES + CIRCUIT BREAKING
- POWERS RINGPOP



uber / tchannel

Unwatch 661 | Star 70 | Fork 5

network multiplexing and framing protocol for RPC — Edit

2,456 commits | 138 branches | 55 releases | 21 contributors

branch: master | tchannel / +

Merge pull request #663 from uber/go-handlers ...

prashantv authored an hour ago | latest commit b86ca35e65

docs	docs/metrics: add retry-count to per attempt latency	28 days ago
golang	go: Simplify handlersMap.find and add test.	2 days ago
node	Cut node-v2.0.0-rc1	3 days ago
python	minor change	4 days ago
.travis.yml	Unbreak Travis CI for Node.js	a month ago
LICENSE	Add LICENSE	3 months ago
README.md	Add links to README	2 months ago

README.md

TChannel build passing

Network multiplexing and framing protocol for RPC

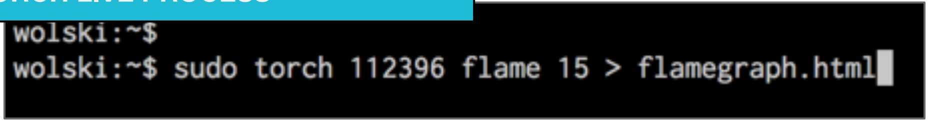
Design goals

- Easy to implement in multiple languages, especially JS and Python.
- High performance forwarding path. Intermediaries can make a forwarding decision quickly.
- Request / response model with out of order responses. Slow requests will not block subsequent faster requests at head of line.
- Large requests/responses may/must be broken into fragments to be sent progressively.
- Optional checksums.
- Can be used to transport multiple protocols between endpoints, eg. HTTP+JSON and Thrift.

HIGH-PERFORMANCE MICROSERVICES

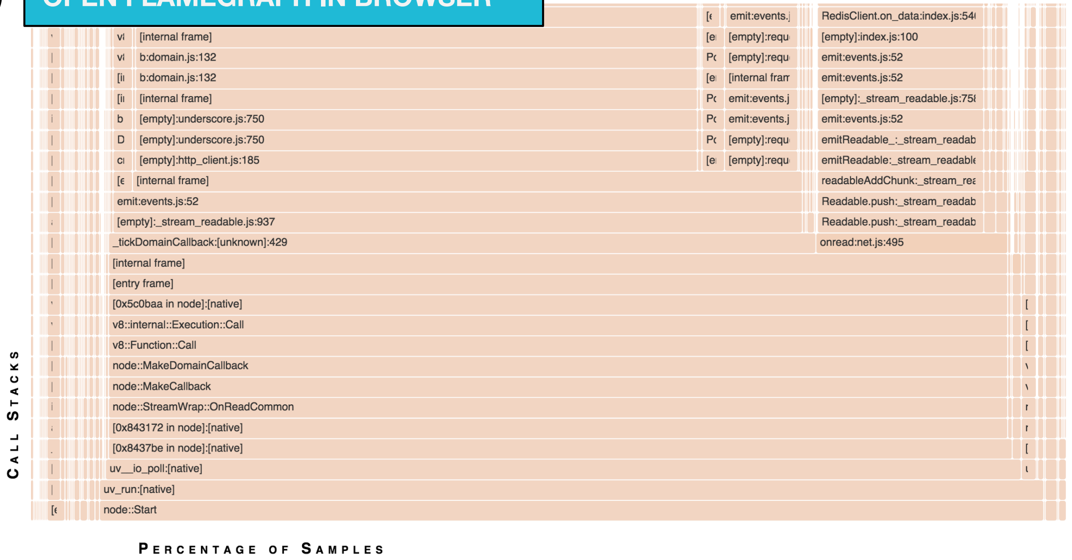
WITH NODESTAP @ GITHUB.COM/UBER/NODE-STAP

1 **TORCH LIVE PROCESS**



```
wolski:~$
wolski:~$ sudo torch 112396 flame 15 > flamegraph.html
```

2 **OPEN FLAMEGRAPH IN BROWSER**

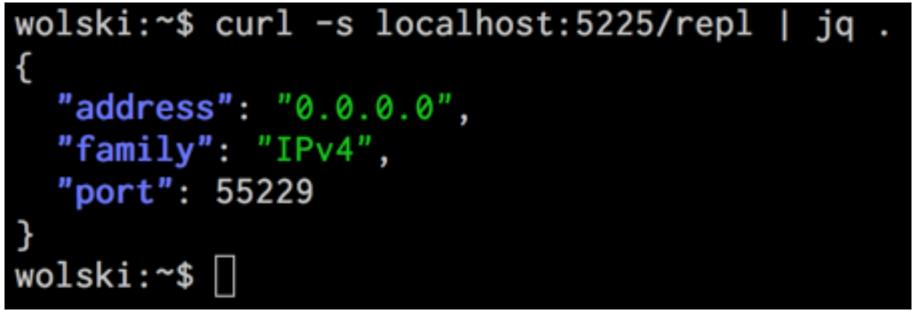


UBER

DEBUGGING MICROSERVICES

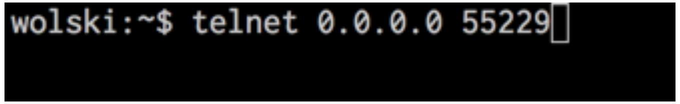
INSPECT INTERNALS WITH NODE REPL

1 **CURL REPL ENDPOINT FOR REPL PORT**



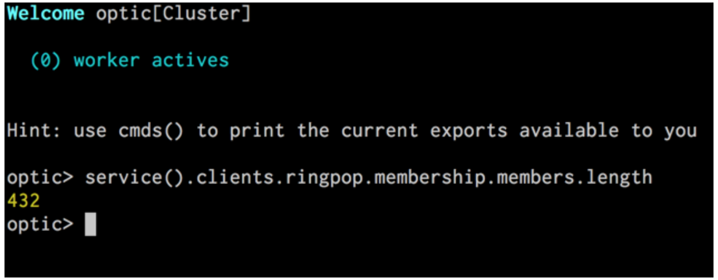
```
wolski:~$ curl -s localhost:5225/repl | jq .
{
  "address": "0.0.0.0",
  "family": "IPv4",
  "port": 55229
}
wolski:~$ 
```

2 **TELNET INTO REPL**



```
wolski:~$ telnet 0.0.0.0 55229
```

3 **INSPECT THE STATE OF YOUR WORKER**



```
Welcome optic[Cluster]

  (0) worker actives


Hint: use cmds() to print the current exports available to you

optic> service().clients.ringpop.membership.members.length
432
optic> 
```

UBER

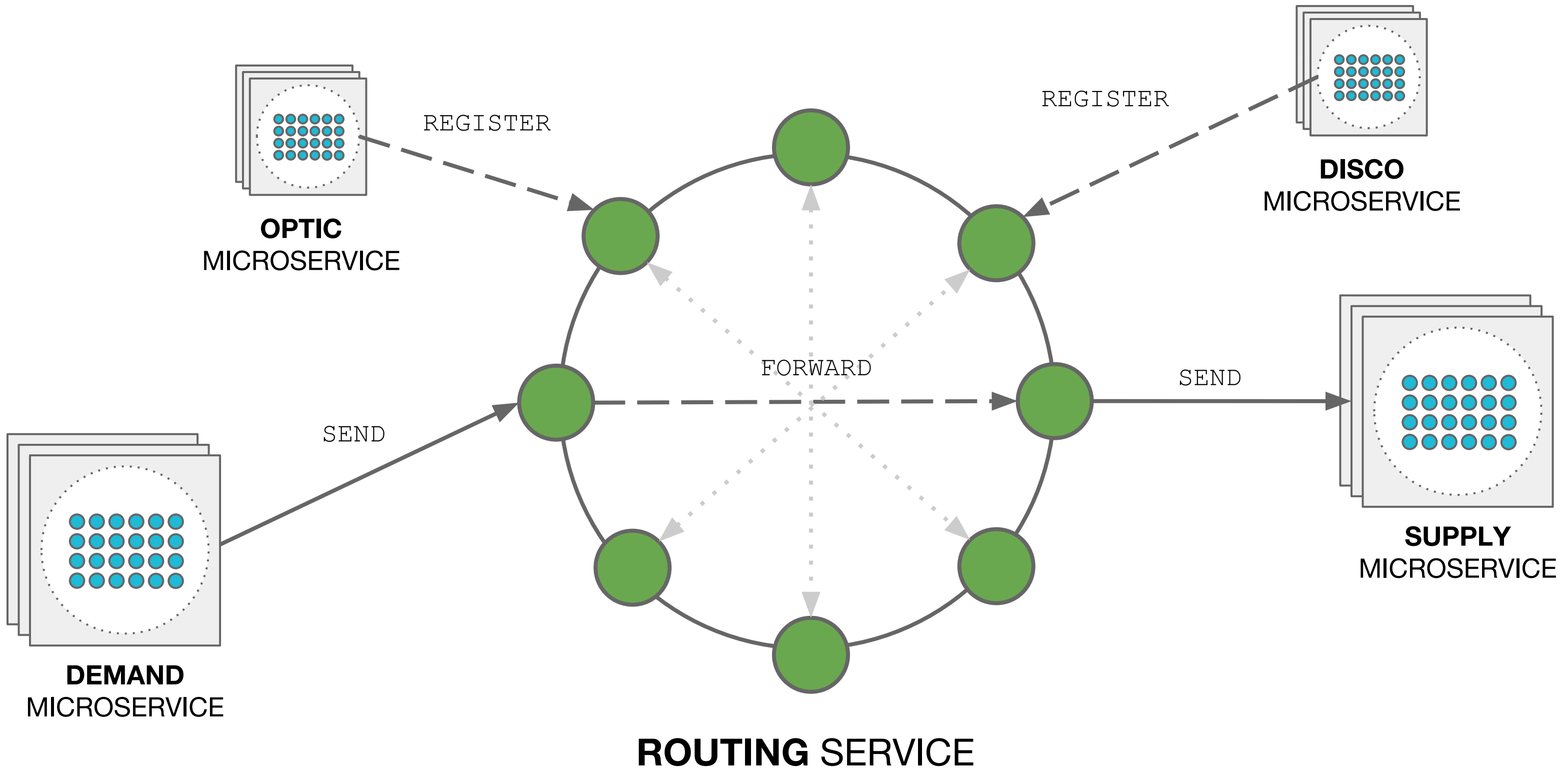
the next generation

CHAPTER 8 OF 8

UBER

NEXT GENERATION MICROSERVICES

A OVERLAY NETWORK FOR MICROSERVICE ROUTING



UBER

THANKS!

Presented by Jeff Wolski <wolski@uber.com>

Uber is hiring!