SKIPJAQ

Load Test Like a Pro

Rob Harrop

Agenda

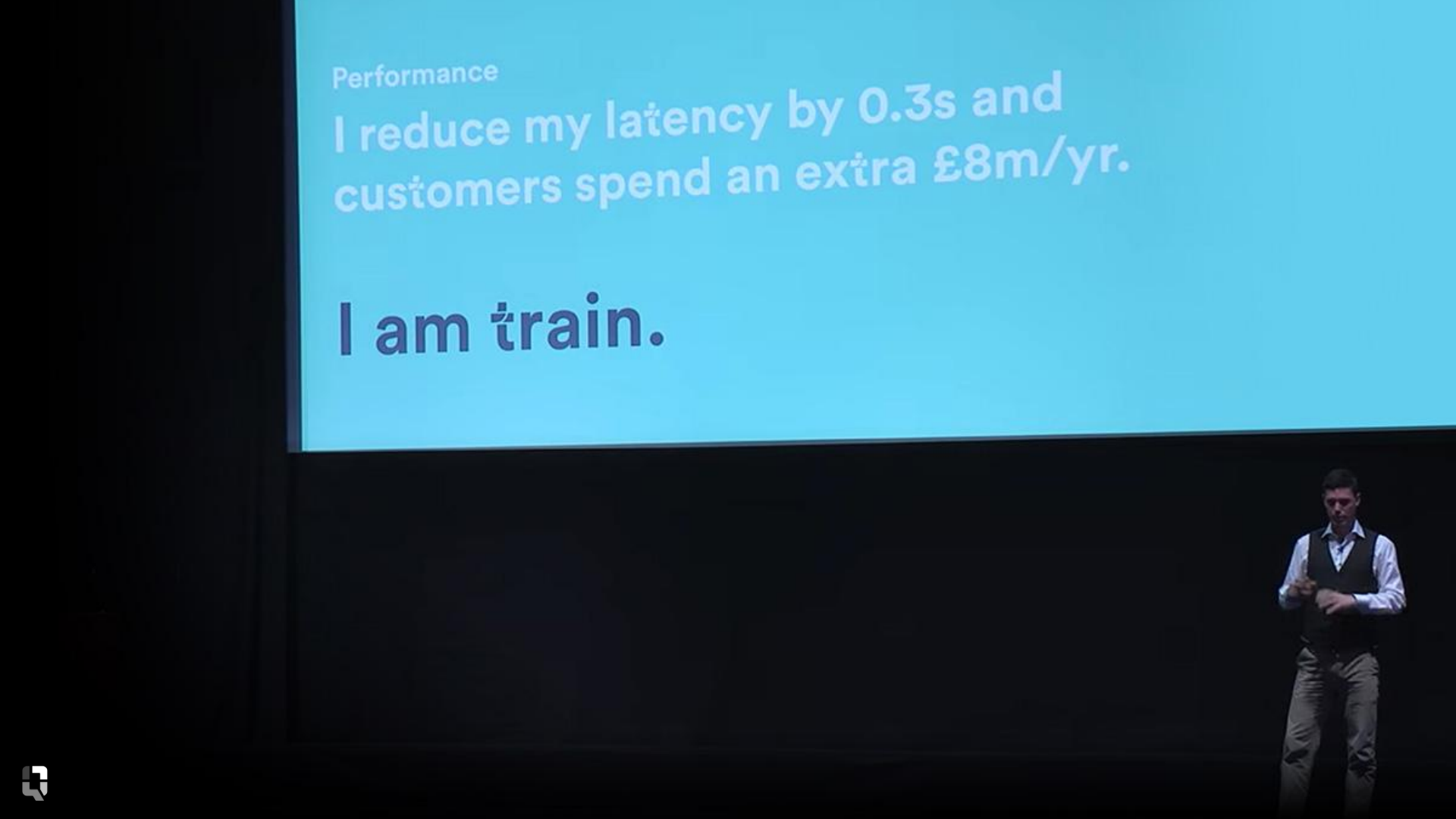
- Who am I?
- Why bother load testing?
- Load Testing Process
  - **Designing** load tests like a pro
  - **Running** load tests like a pro
  - **Analysing** load tests like a pro

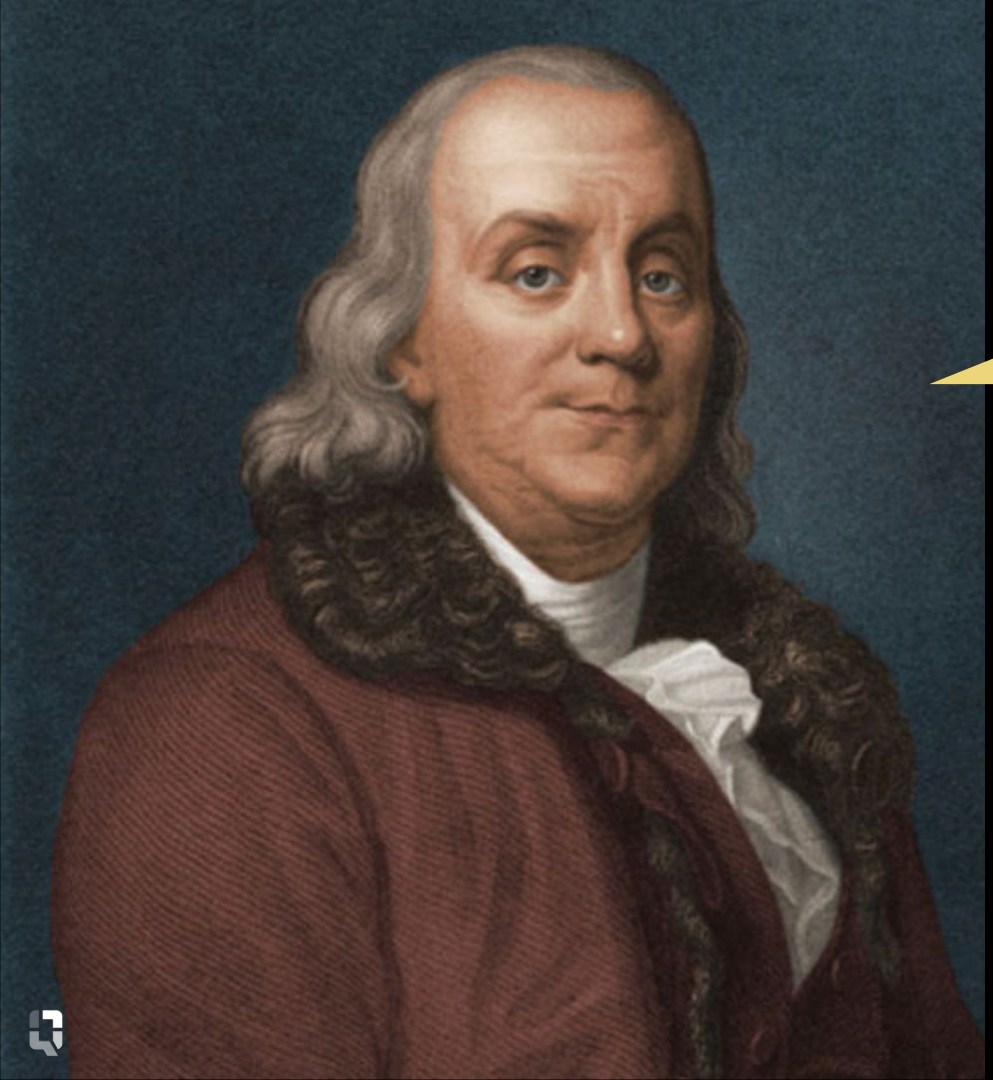
Who Am I?

CEO @SKIPJAQ

Co-Founder @SpringSource

Why Bother?


Performance
I reduce my latency by 0.3s and customers spend an extra £8m/yr.
I am train.



Time is Money

shopzilla

5s latency reduction

⌄⌄

7-12% increased conversion

Walmart

Every 1s latency reduction

⌄⌄

~2% increase in conversion

moz://a

2.2s reduction in average latency

⌄⌄

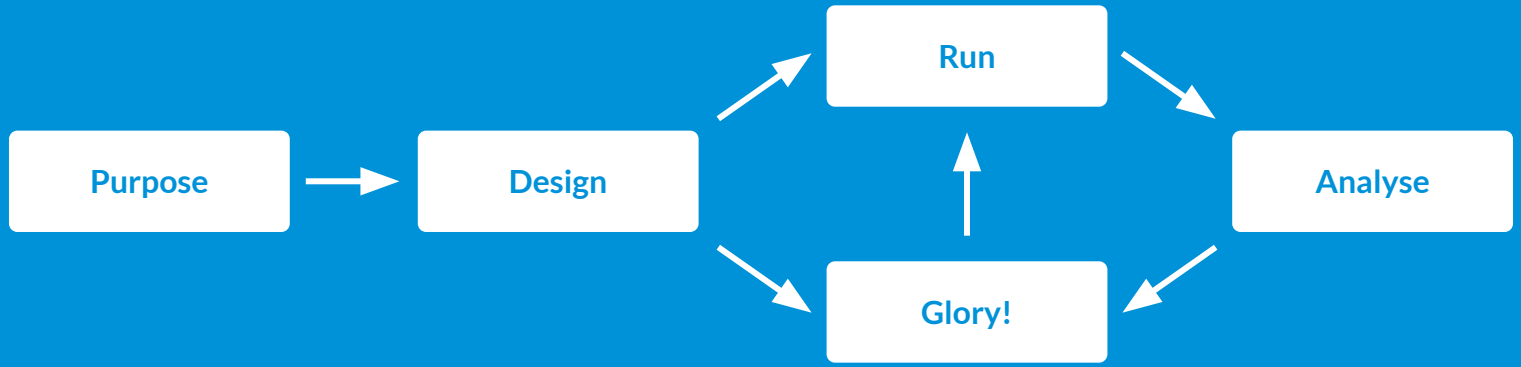
15.4% increase in download conversion

Load Testing is a Process

In Fact ...

Load Testing is a

Continuous Process


Purpose
Design
Run
Glory!
Analyse

- Once designed, a load test is good for multiple runs
- How many runs depends on:
  - How often you run
  - How often your code changes
  - How often your audience changes
- You **must** revisit the design as the landscape changes

Designing a Load Test

Load Test?

Step One: Define the purpose for testing

Step Two: Construct a high-quality test plan

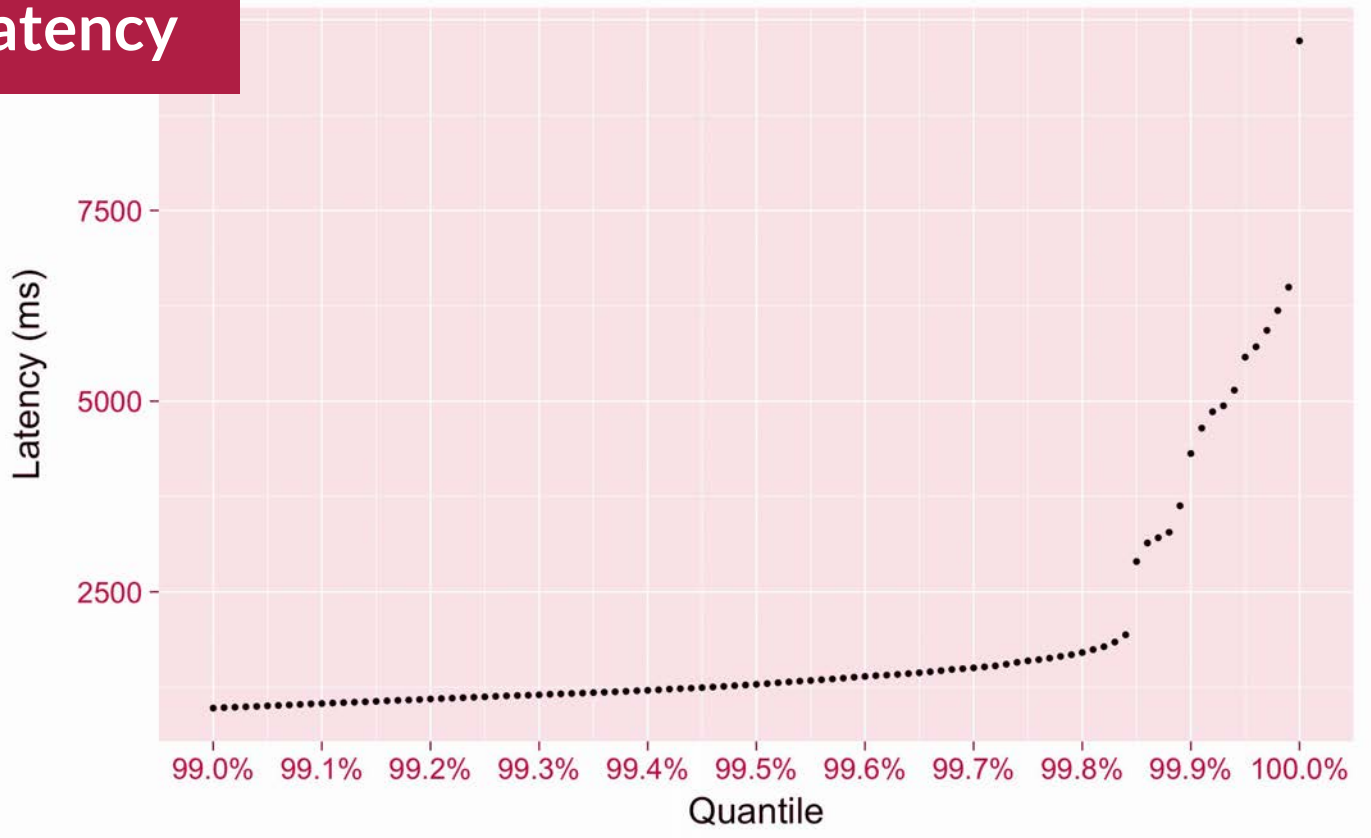
Purpose

- What **latency** do I see with **X concurrent users**?
- How **many concurrent users** can I handle before **latency exceeds Xms**?
- How many **concurrent users** can I handle before my **system is saturated**?

Measuring Latency

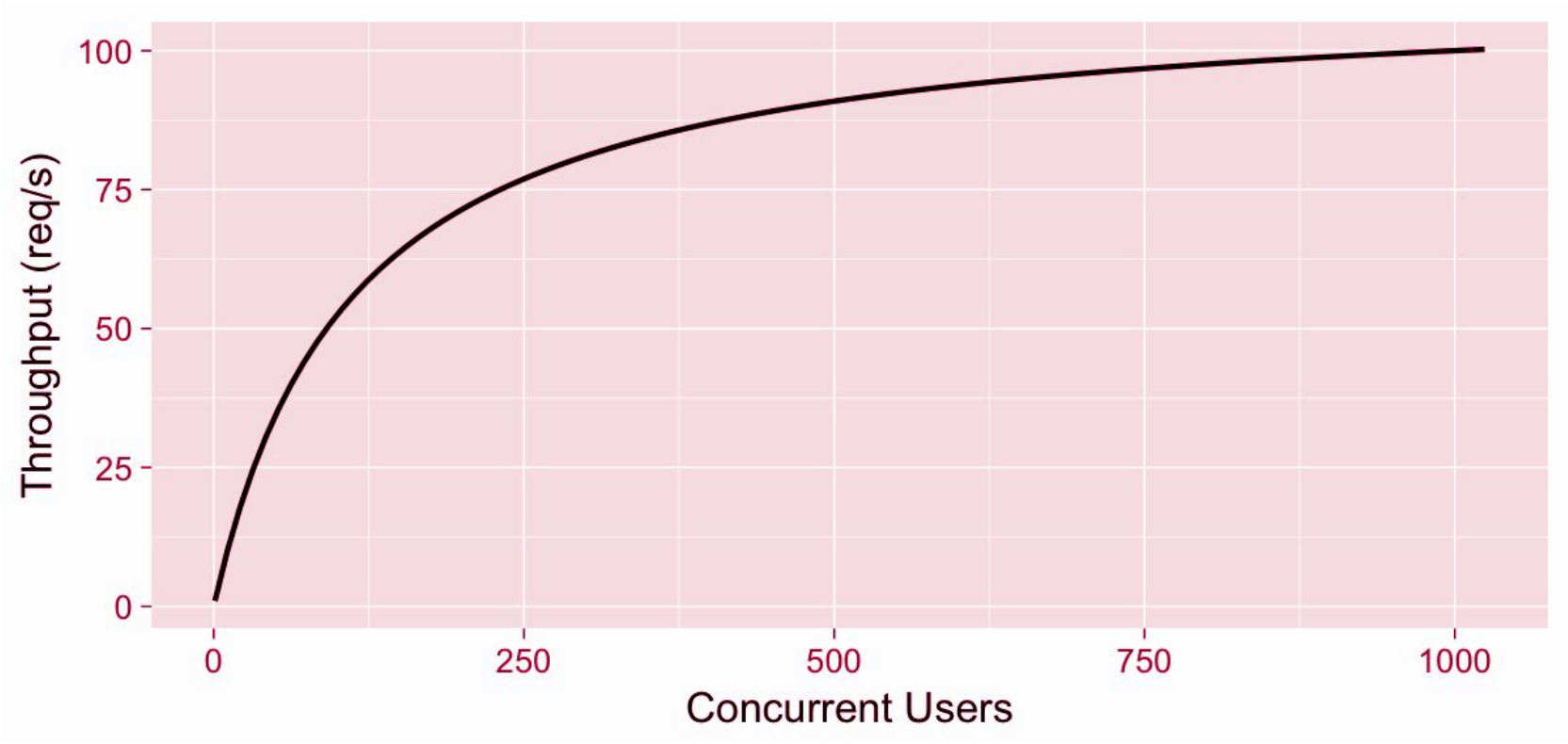
- Average latency is mostly useless as a metric
- Later percentiles are better: 90th, 95th, 99th, 99.9th
- Max is a useful measure but is highly susceptible to measurement error
- **For best results, consider whole distribution analysis**

Measuring Latency



- You may legitimately want to answer **multiple questions at once**

- Consider a load test as a series of **explorations up your load curve**

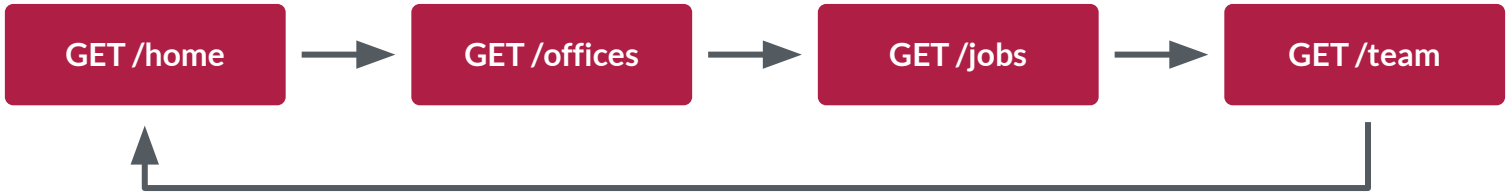

Throughput (req/s)
100
75
50
25
0
0
250
500
750
1000
Concurrent Users

Test Plan

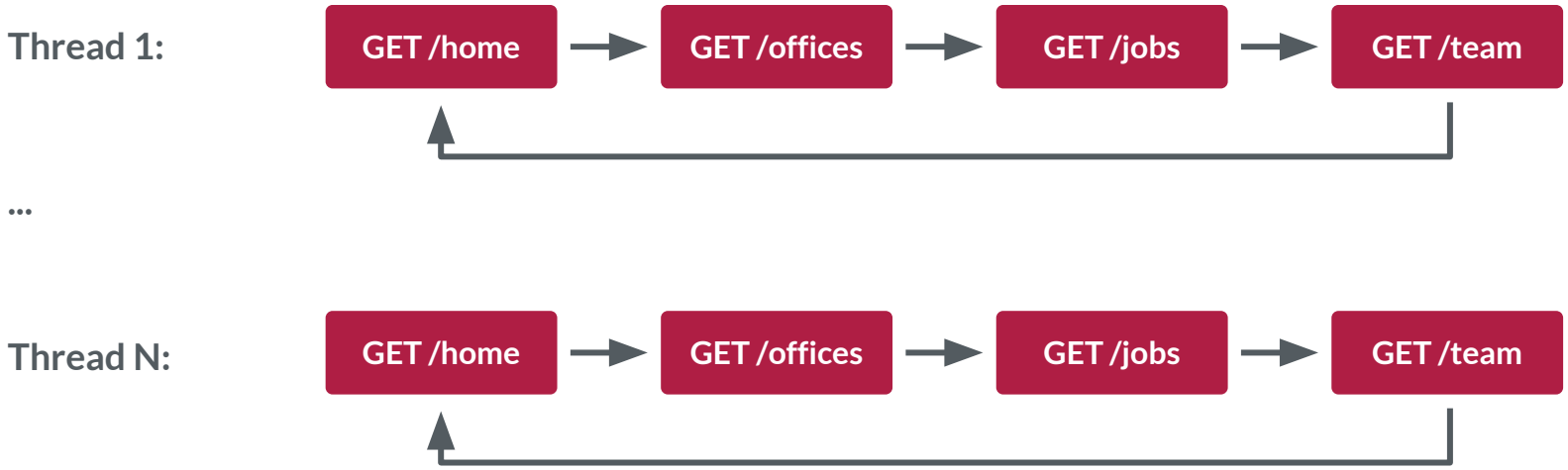
Test ~~Plan~~ Model

- Traditional test plans are **prescriptive** and deterministic
- A great test model is **descriptive** and **probabilistic**

Prescriptive

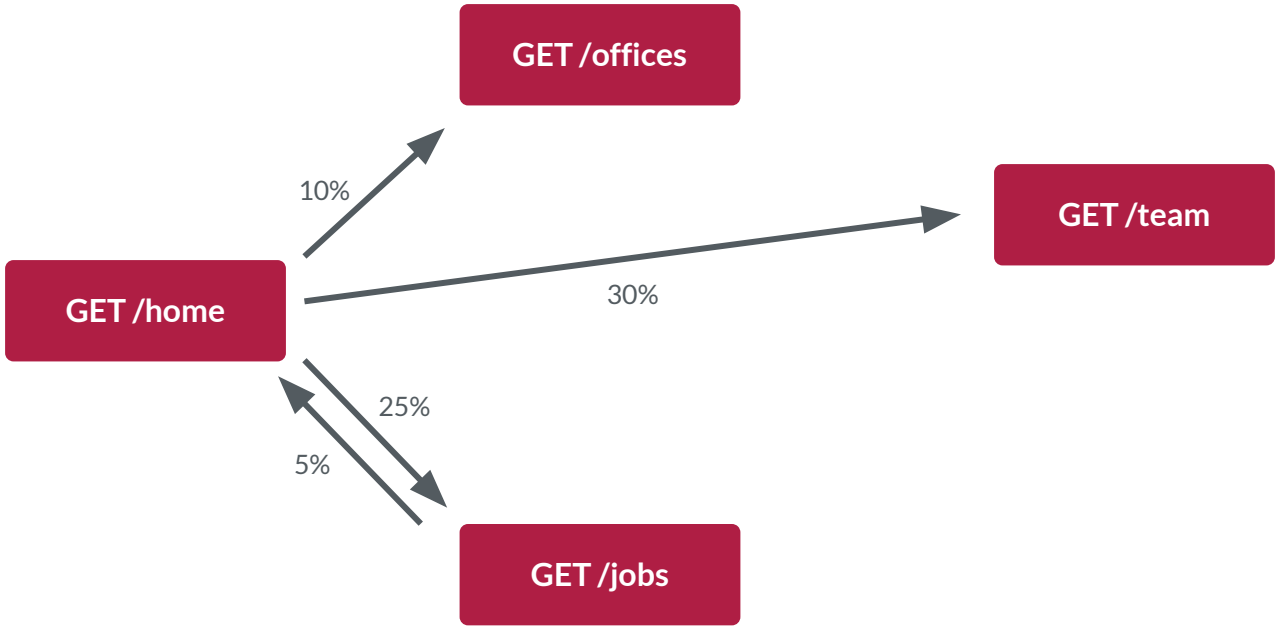


Prescriptive



This is where load tests go to die

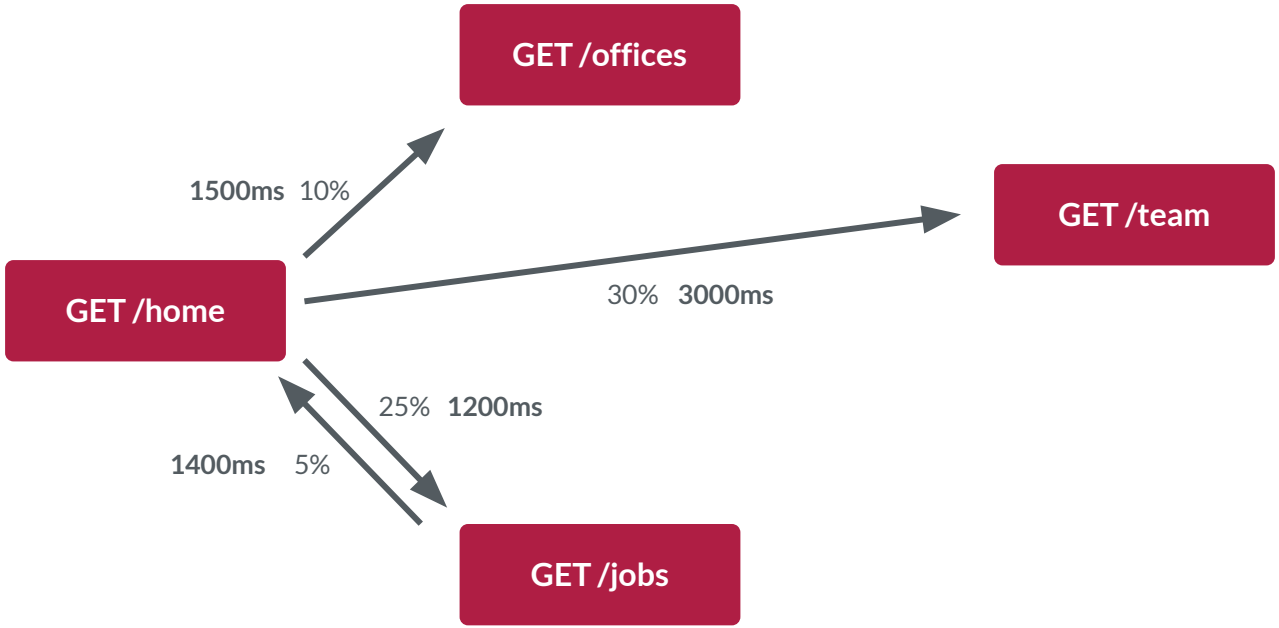
Descriptive



Each user is different

Transition Probabilities Capture This Difference

Descriptive



Wait Times are

A Thing™

- Wait times are typically non-zero
- Wait times are drawn **from a distribution**
- If you don’t know the distribution, **exponential** is a good fallback

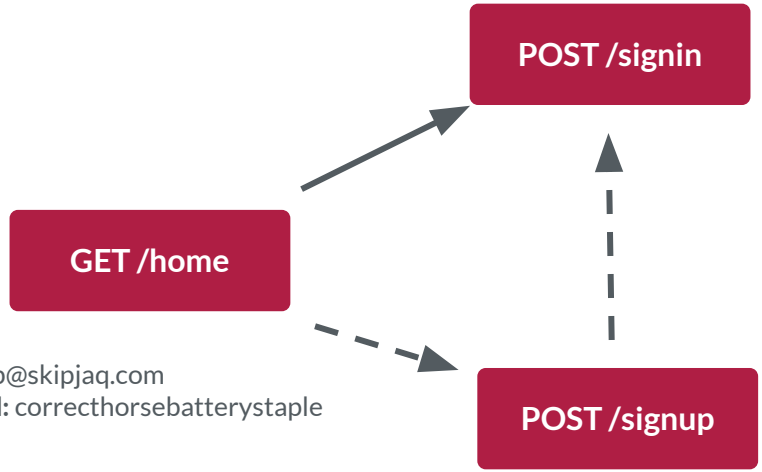
Source Model Parameters From Real Data

- Google Analytics, MixPanel, Kiss Metrics
- Log files
- OpenZipkin, AWS X-Ray
- NewRelic, AppDynamics, Instana

Handling Data Mutation

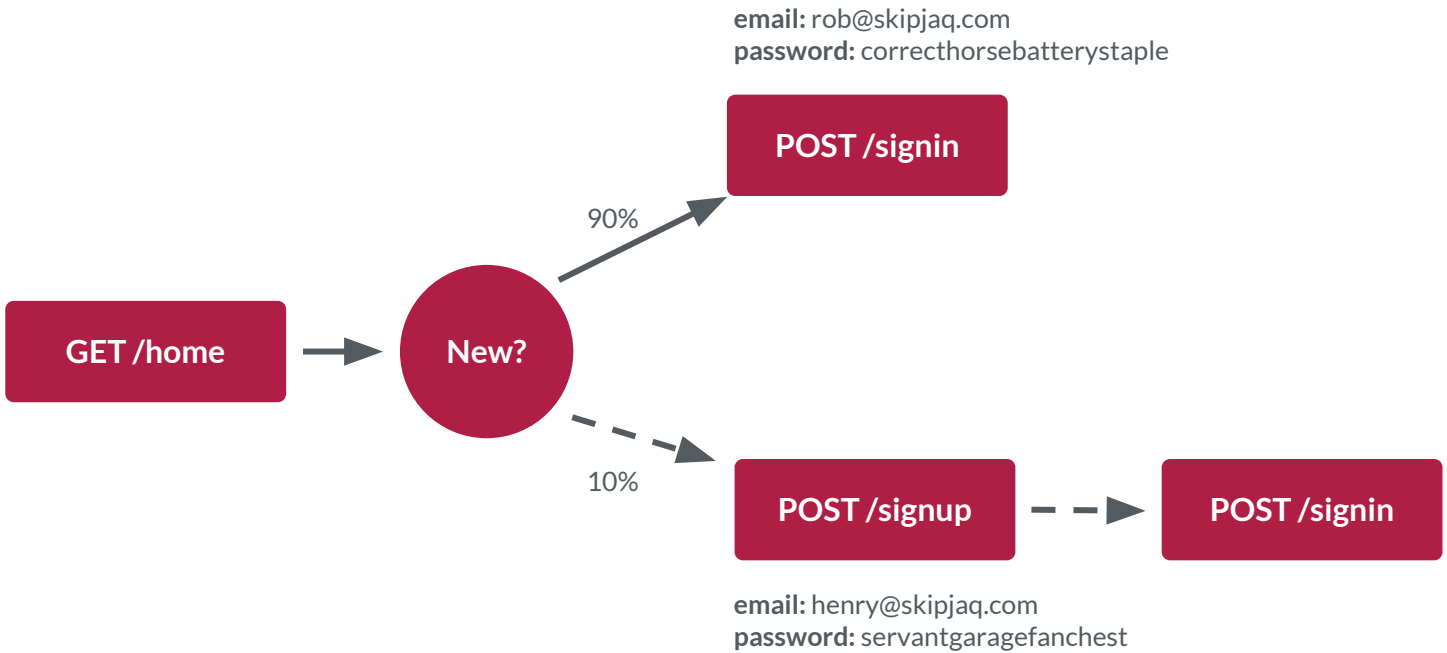
- Snapshot and restore a data starting point for each test
- Assign/generate test data *per virtual user*
- **Mutation workflows modify transition probabilities**
- **Assigned data varies based on workflow**

Naive Mutation



email: rob@skipjaq.com
password: correcthorsebatterystaple

Modulated Mutation



Recap

- Design with a purpose in mind
- Each user is different, capture this difference in your model
- Probabilistic models are better than prescriptive plans
- Production data is the best source of model parameters

Running a Load Test

Step One: Create a high-fidelity load test environment

Step Two: Provision enough load generation capacity

Step Three: Instrument the System Under Test

Step Four: Implement the test model for your load generator of choice

Step Five: Go!

The System Under Test

- The closer to production the better
- Ideally, you're already automating production provisioning
- Stubbing is acceptable, but you must do it well

Hoverfly

- Stubs must exhibit service times like real systems
- Service times and wait times are modelled the same way
- github.com/spectolabs/hoverfly

Load Generators

- Laptops are **not** load generators
- Less is **not** more
- If in doubt, over provision
- Make sure that you're tracking metrics from the load generators

Instrumentation

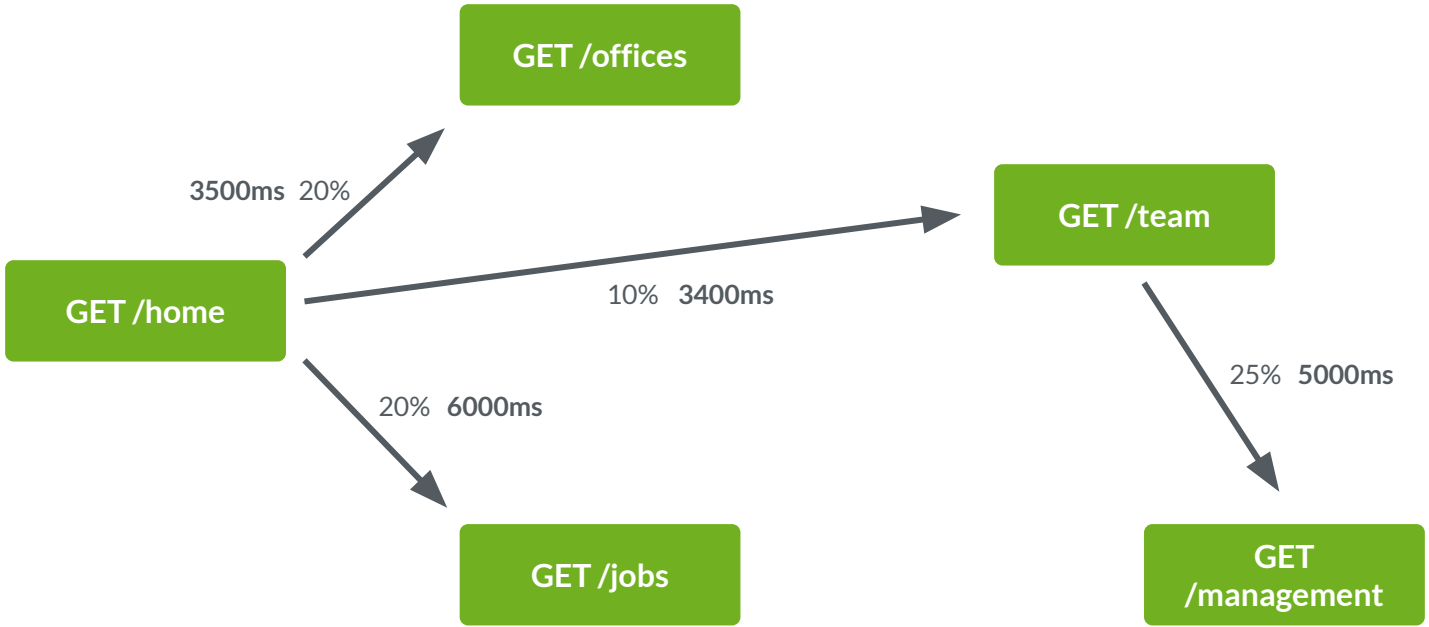
- CPU, memory, I/O
- Usage, Saturation and Errors (USE)
- Background reading: Systems Performance by Brendan Gregg

USE Metrics

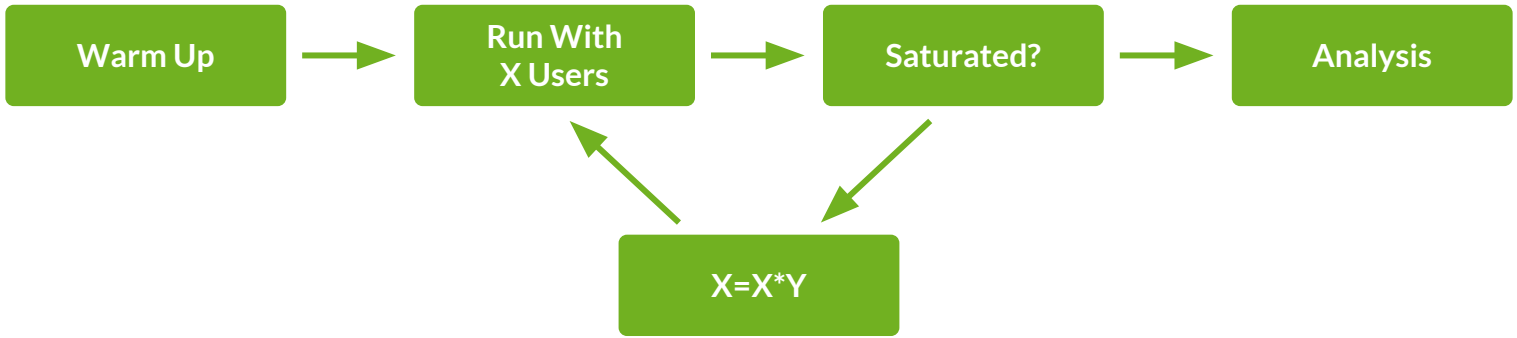
Resource	Utilisation	Saturation	Errors
CPU	CPU %	Run Queue Length	ECC events/Failed CPUs
Memory	Free/Used %	Anon. Paging/Thread Swapping	Failed mallocs?
Network I/O	RX/TX throughput	NIC events (drops/overruns)	
Disk I/O	Busy %	Wait Queue Length	Device Errors

Implementing in JMeter

The Load Model



Let’s Go!



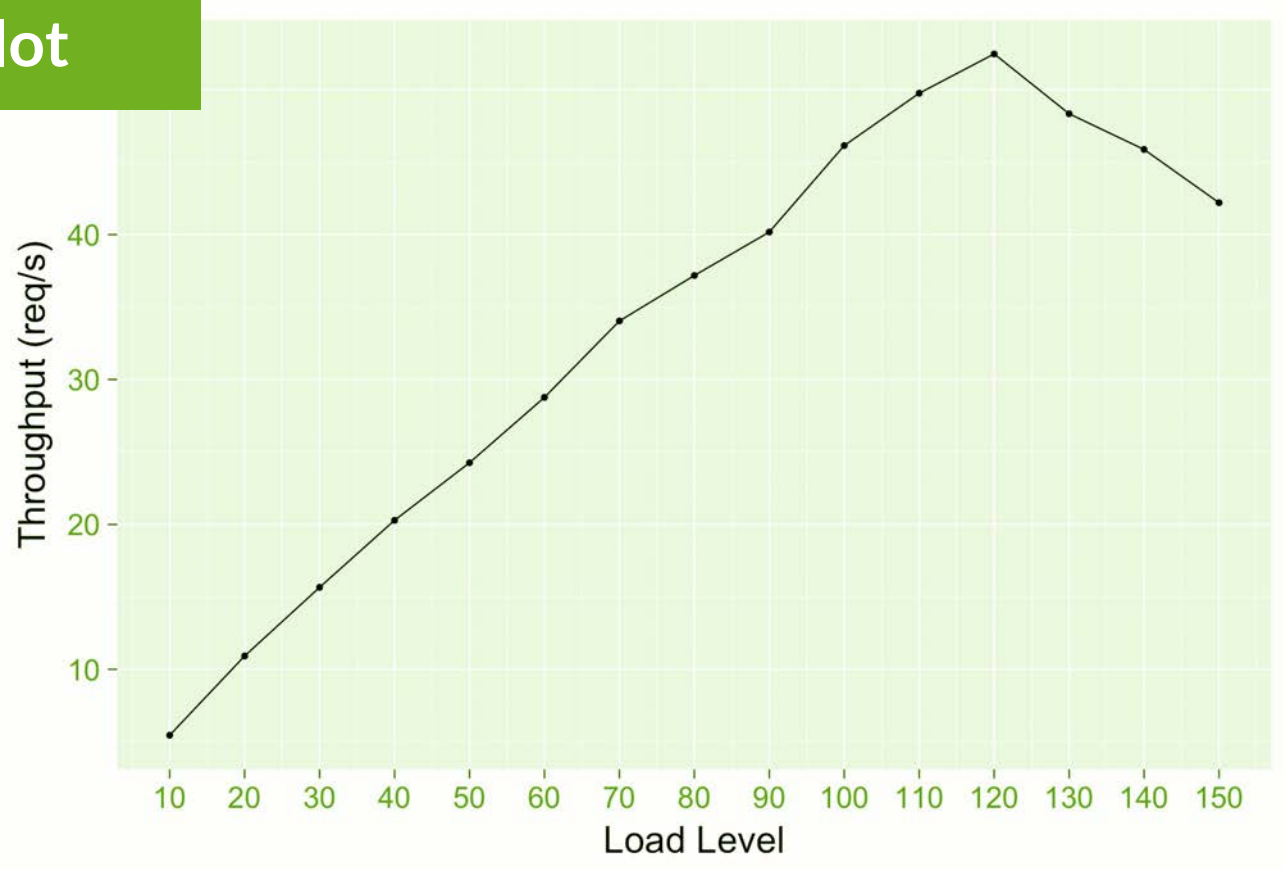
Detecting Saturation

- **Super low tech:** check the results by hand
- **Low tech:** check a *plot* of the results
- **High tech:** fit a curve to the plot and check *that*
- **Super high tech:** take the derivative of the fitted curve

Checking The Results

Load Level	Run Time (ms)	Total Requests	Throughput	Errors	Error %
30	729400	11420	15.65670	0	0.00000000
40	752098	15256	20.28459	0	0.00000000
50	786117	19069	24.25720	0	0.00000000
60	798399	22971	28.77133	2	0.00870663
70	784018	26691	34.04386	0	0.00000000
80	819177	30459	37.18244	9	0.02954792
90	852584	34259	40.18255	0	0.00000000
100	825598	38103	46.15200	0	0.00000000
110	843654	41984	49.76448	0	0.00000000
120	871827	45751	52.47715	32	0.06994383
130	1024274	49520	48.34644	0	0.00000000
140	1165331	53468	45.88224	6	0.01122167
150	1354668	57177	42.20739	32	0.05596656

Checking a Plot



Fitting a Curve

Relative capacity

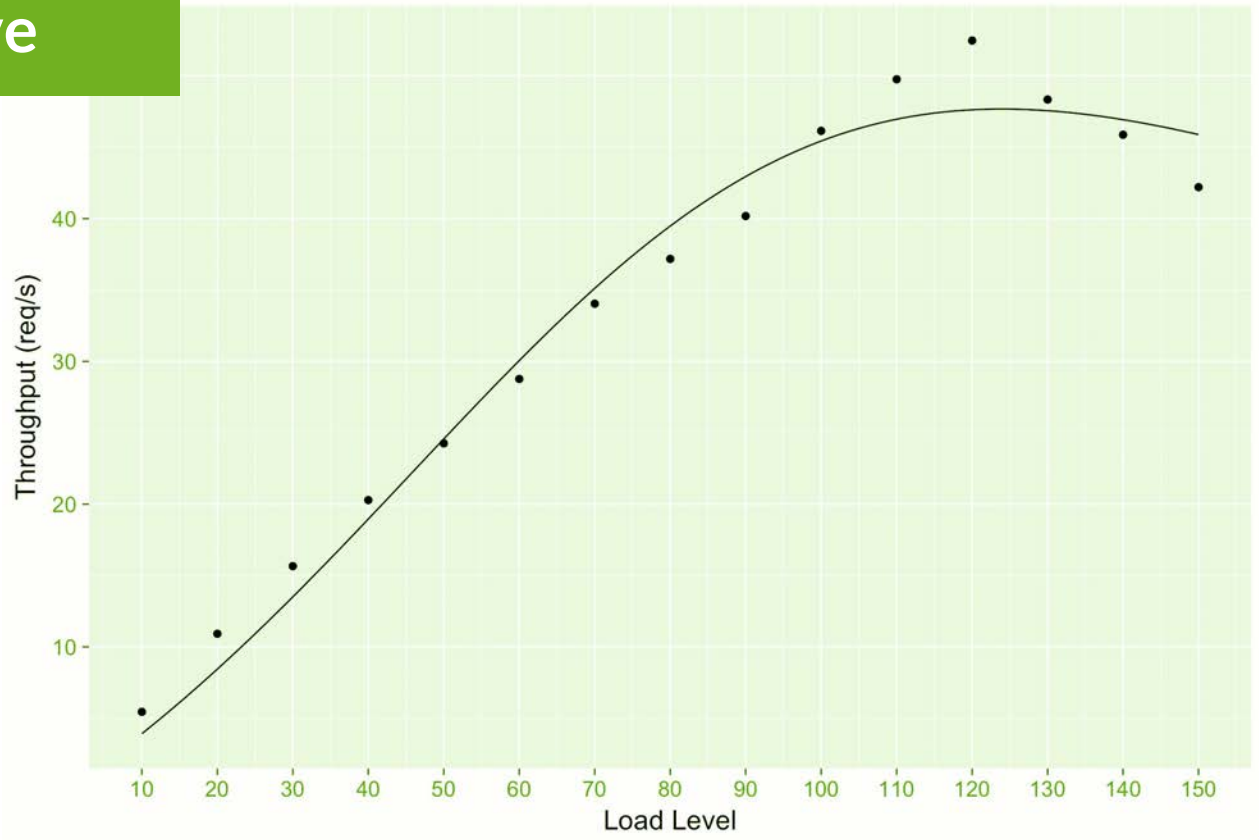
Number of users

$$C(N) = \frac{N}{1 + \alpha(N-1) + \beta N(N-1)}$$

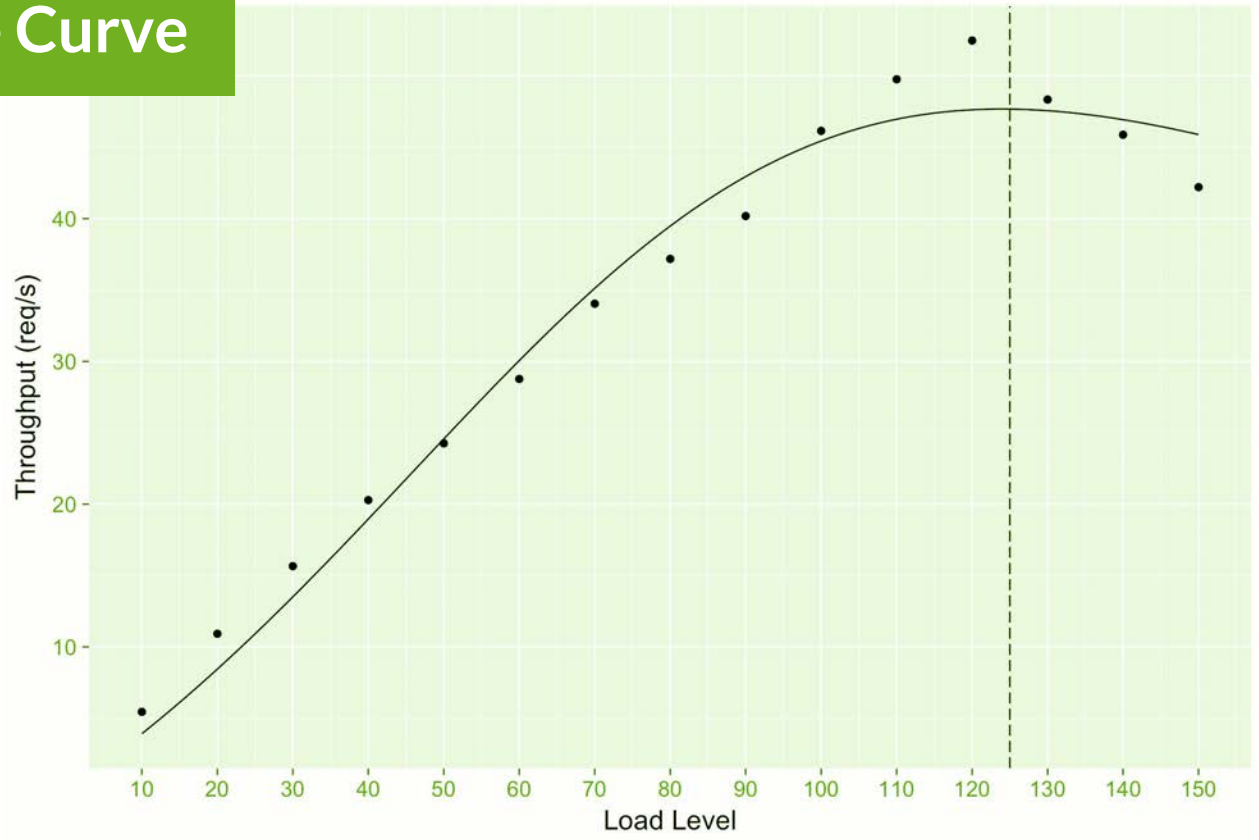
Contention overhead

Crosstalk overhead

Fitting a Curve



Derivative of the Curve



Retrograde Scaling

Load Level	Run Time (ms)	Total Requests	Throughput	Errors	Error %
30	729400	11420	15.65670	0	0.00000000
40	752098	15256	20.28459	0	0.00000000
50	786117	19069	24.25720	0	0.00000000
60	798399	22971	28.77133	2	0.00870663
70	784018	26691	34.04386	0	0.00000000
80	819177	30459	37.18244	9	0.02954792
90	852584	34259	40.18255	0	0.00000000
100	825598	38103	46.15200	0	0.00000000
110	843654	41984	49.76448	0	0.00000000
120	871827	45751	52.47715	32	0.06994383
130	1024274	49520	48.34644	0	0.00000000
140	1165331	53468	45.88224	6	0.01122167
150	1354668	57177	42.20739	32	0.05596656

Recap

- Testing to saturation provides you with plenty of data
- Detecting saturation is relatively simple from the outside
- Tracking internal performance metrics helps to track saturation

Analysing a Load Test

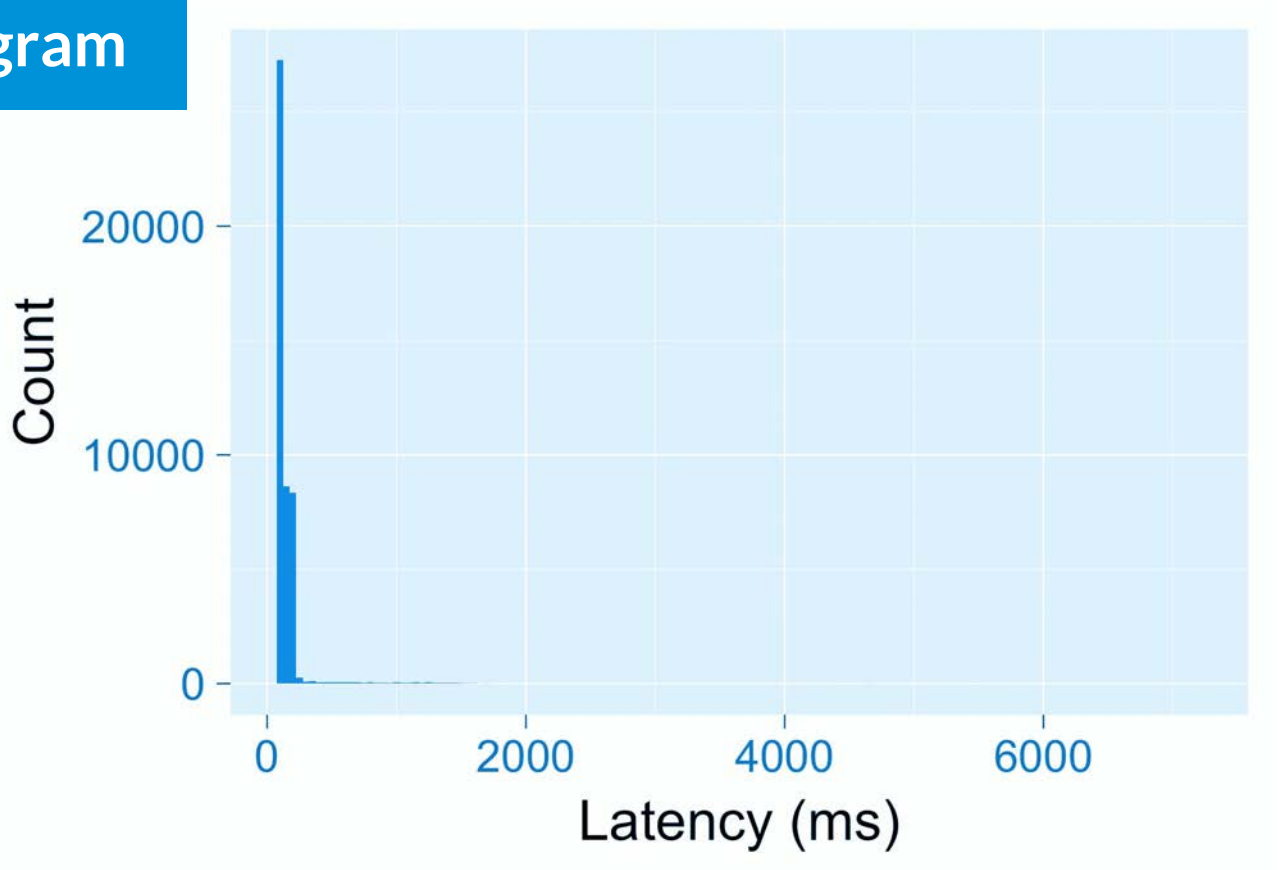
Step Zero: Learn R

Step One: Exploratory analysis

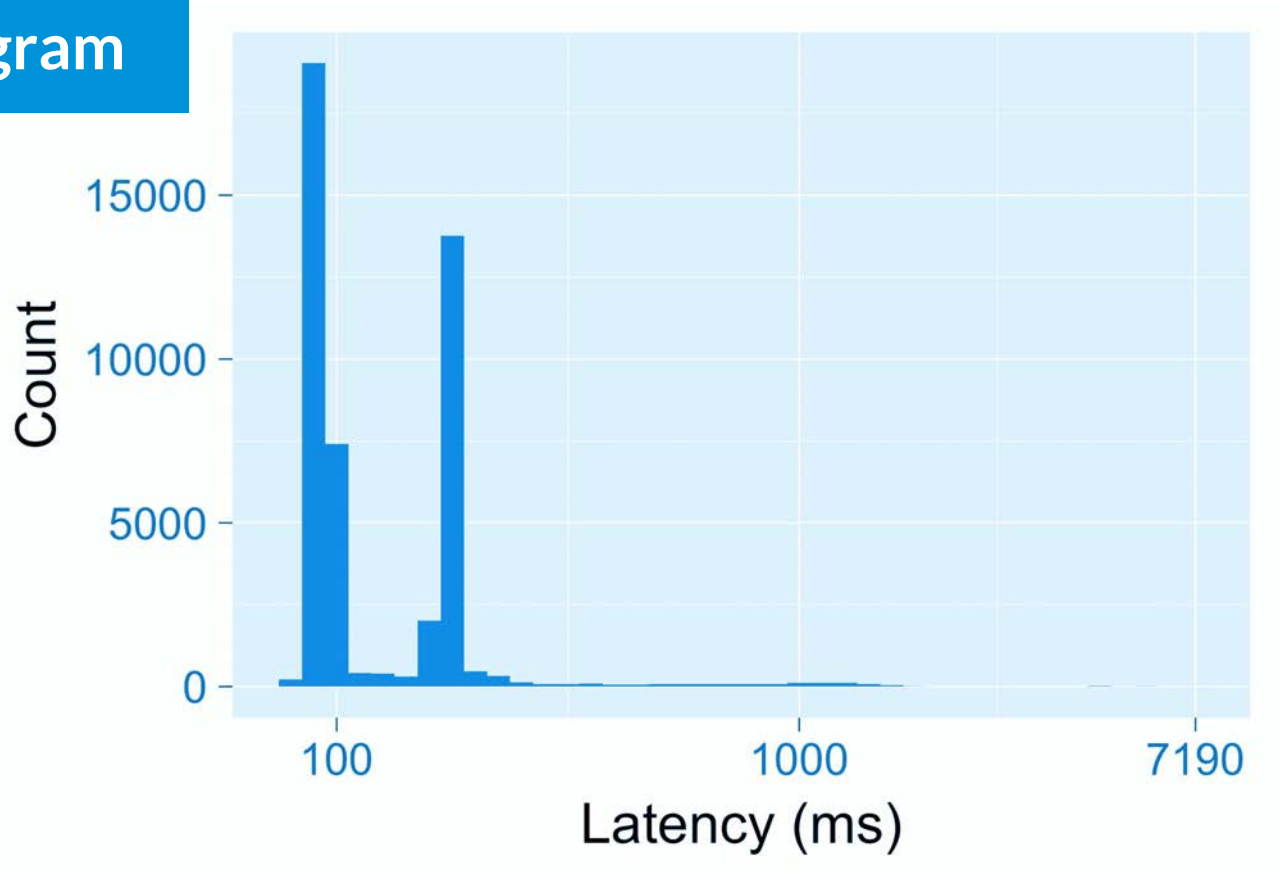
Step Two: Answer your questions

Step Three: Link to production data

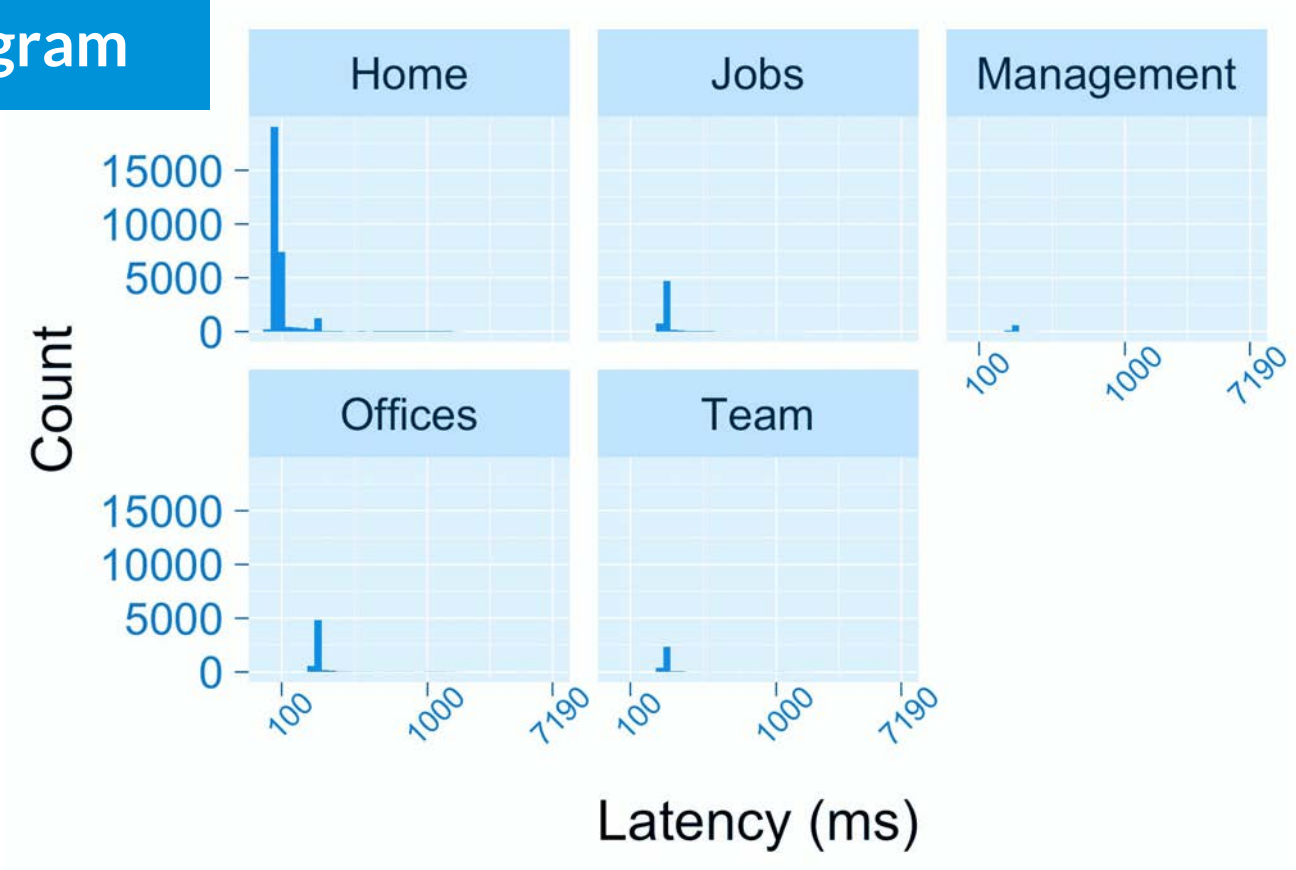
The Basic Histogram



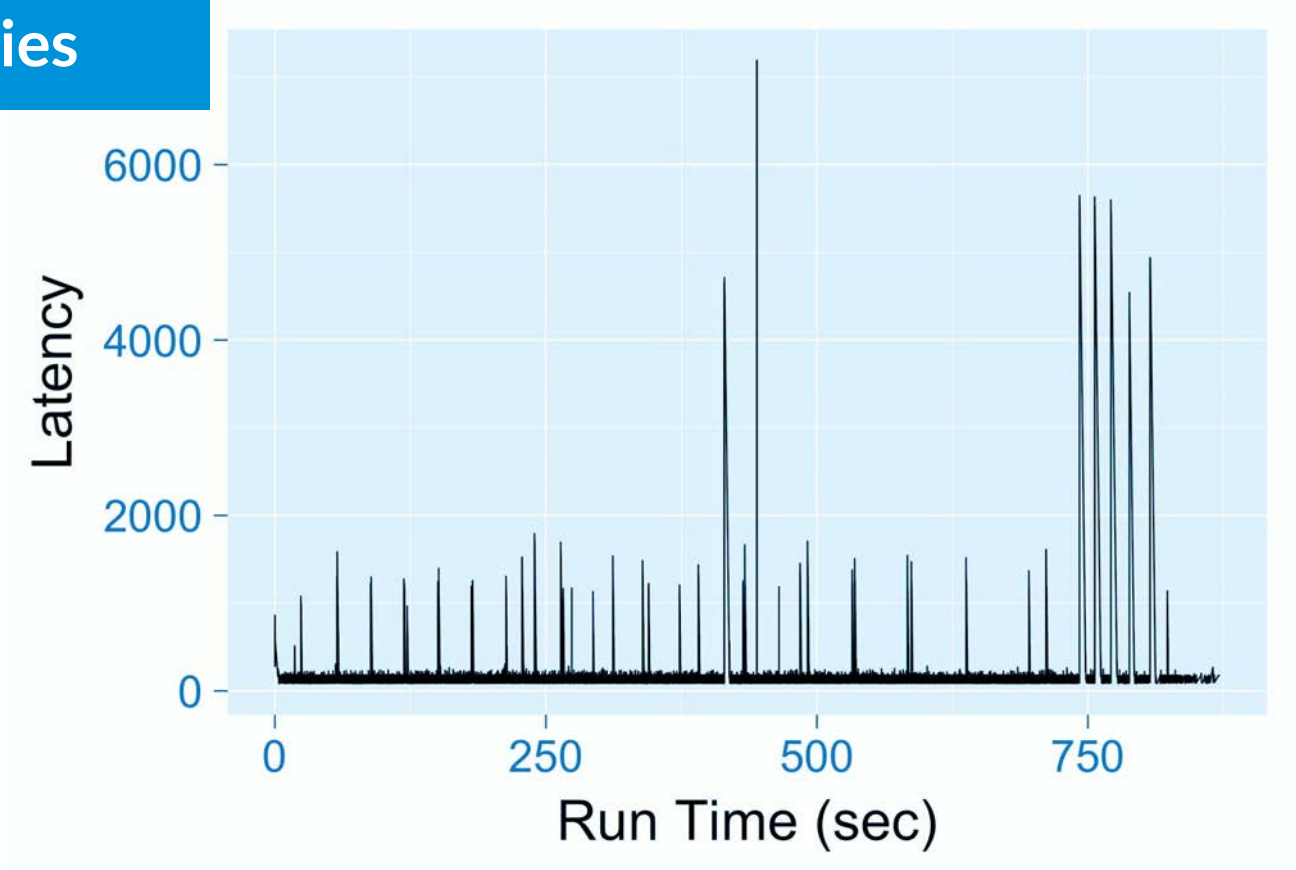
Log Scale Histogram



Faceted Histogram



As a Timeseries



The Humble Table

Label	50%ile	90%ile	95%ile	99%ile	99.9%ile	100%ile
Home	94	116	176	852.01	1474	5599
Jobs	175	185	239.65	960.64	4719.171	7178
Management	176	181	220.2	1041.56	1490.656	1516
Offices	176	186.4	229.7	1129.76	1683.586	7190
Team	175	182	223.7	1082.86	1676.934	4647

QoS per Load Level

Load Level	50%ile	90%ile	95%ile	99%ile	99.9%ile	100%ile
60	95	177	179	409.60	1269.120	2137
70	95	177	179	541.10	1430.170	1817
80	95	177	179	619.00	1423.084	4906
90	95	178	180	843.00	1652.226	4529
100	95	178	181	870.96	1650.898	5248
110	95	178	182	934.17	1711.034	5543
120	96	178	186	952.50	1568.000	7190
130	97	180	286	1152.00	6118.215	9268
140	158	189	331	1241.33	5876.665	7608
150	169	198	318	1386.48	5150.648	9724

Labelled QoS

Label	Load Level	50%ile	90%ile	95%ile	99%ile	99.9%ile	100%ile
Home	60	93	96.0	99.00	374.00	1187.021	1772
Home	70	93	96.0	101.00	463.03	1359.006	1817
Home	80	93	97.0	107.00	515.04	1422.002	4793
Home	90	93	97.0	122.00	705.04	1642.509	4478
Home	100	93	99.0	129.00	715.05	1671.005	5158
Home	110	93	100.0	140.00	864.00	1675.503	5485
Home	120	94	116.0	176.00	852.01	1474.000	5599
Home	130	94	176.0	181.00	1135.00	6176.000	9204
Home	140	94	178.0	187.00	1192.03	5879.001	7523
Home	150	96	180.0	201.00	1384.00	5658.000	9713

Recap

- Experiment with different visualisations
- A simple table might be the best visualisation you have
- Segment analysis by endpoint
- Understand your load in the context of your production metrics

In Summary

Load Test
With Purpose

Load Test With
Real User Models

Explore Your Load Curve

Analyse in the Context of **Real Data**

Performance Matters