Future-proofing
Production Systems

@kavya719
kavya
analyzing
of the
performance
of systems
performance

• What’s the additional load the system can support, without degrading response time?

• What’re the system utilization bottlenecks?

• What’s the impact of a change on response time, maximum throughput?

capacity

• How many additional servers to support 10x load?

• Is the system over-provisioned?
...use **prod** to make prod better.

more robust, performant, scalable
...use **prod** to make prod better.

more robust, performant, scalable

- A/B testing, canaries and ramped deploys.
- chaos engineering.
- **stressing** the system.

empirically determine performance characteristics, bottlenecks.
Kraken
a fancy load “simulator”
utilization law, the USL

OrgSim etc.
a standard load simulator
Little’s law

stepping back
the sweet middle ground
Kraken
• Facebook’s load “simulator”. In use in production since ~2013.
kraken

• Facebook’s load “simulator”. In use in production since ~2013.

• Used to determine a system’s capacity.

  maximum throughput, subject to a response time constraint.
kraken

• Facebook’s load “simulator”. In use in production since ~2013.
• Used to determine a system’s **capacity**. maximum throughput, subject to a response time constraint.
• And to identify and resolve **utilization bottlenecks**.
Facebook’s load “simulator”. In use in production since ~2013.

Used to determine a system’s capacity. Maximum throughput, subject to a response time constraint.

And to identify and resolve utilization bottlenecks.

Allowed them to increase Facebook’s capacity by over 20% using the same hardware!
the model

stateless servers
that serve requests without using sticky sessions/server affinity.

load can be controlled by re-routing requests
for example, this does not apply to a global message queue.

downstream services respond to upstream service load shifts
for example, a web server querying a database.
load generation

need a representative workload...use live traffic!

traffic shifting:
increase the fraction of traffic to a region, cluster, server, by adjusting the weights that control load balancing,

monitoring

need reliable metrics that track the health of the system.

user experience

safety

p99 response time
HTTP error rate
CPU utilization
connections, queue length


let’s run it!

...is this good or is there a bottleneck?
Step I: single server capacity

model a web server as a queueing system.

response time = queueing delay + service time

assume no upstream saturation, so service time constant i.e.
response time $\propto$ queueing delay.
utilization = throughput * service time

"busyness"

**throughput** increases

**utilization** increases

**queueing delay** increases (non-linearly);
so, **response time**.
utilization = throughput * service time  

"busyness"

throughput increases

utilization increases

queueing delay increases (non-linearly); so, response time.
Step II: cluster capacity

Iff linear scaling,

cluster of N servers’ capacity = single server capacity * N

theoretical cluster capacity

… but systems don’t scale linearly.

Universal Scalability Law (USL):

• contention penalty
due to queueing for shared resources
• consistency penalty
due to increase in service time

target cluster capacity should account for this.
Facebook sets target cluster capacity = 93% of theoretical.

...is this good or is there a bottleneck?
Facebook sets target cluster capacity = 93% of theoretical.

Cluster capacity is ~90% of theoretical, so there’s a bottleneck to fix!
bottlenecks uncovered

- cache bottleneck
- network saturation
- poor load balancing
- misconfiguration

Also, insufficient capacity
i.e. no bottlenecks per-se, but organic growth.

...so, can we have it too?
OrgSim etc.
load generation

Run a **configurable number of virtual clients**. A virtual client sends/receives in a loop.

Use **synthetic workloads**. OrgSim’s load profile is based on historical data.

monitoring

external to the load simulator system. We use Datadog alerts on metric thresholds.
gotchas

• synthetic workloads may not be representative of actual.
number of virtual clients \((N) = 1, \ldots, 100\)

**wrong shape**
for response time curve!

should be
(from the USL)

... load simulator hit a bottleneck!
gotchas

• synthetic workloads may not be representative of actual.
• load simulator may hit a bottleneck!

Little’s Law:
concurrency = throughput * response time
stepping back
Case for performance testing in production
empiricism is queen.

Case for performance modeling
expectations better than no expectations.

...performance testing or modeling?

yes.
Kraken

Performance modeling
Performance Modeling and Design of Computer Systems, Mor Harchol-Balter
How to Quantify Scalability, Neil Gunther:
http://www.perfdynamics.com/Manifesto/USLscalability.html

@kavya719
speakerdeck.com/kavya719/future-proofing-production-systems

Special thanks to Eben Freeman for reading drafts of this.
non-linear responses to load

non-linear scaling
Throughput vs. Latency

- Non-linear responses to load
- Non-linear scaling

Microservices: systems are complex

Continuous deploys: systems are in flux
load generation

need a **representative workload**.

profile (read, write requests)
arrival pattern including traffic bursts

...use **live traffic**.

capture and replay
**traffic shifting**
traffic shifting

adjust weights that control load balancing, to increase the fraction of traffic to a cluster, region, server.
monitoring

need **reliable metrics** that track the health of the system.

- **user experience**
  - p99 response time
  - HTTP error rate

- **safety**
  - CPU utilization
  - memory utilization
  - connections, queue length
let’s run it!
samsara

industry site

devices

the cloud (AWS)

user’s browser

web dashboard
samsara

industry site

devices

hubs

data processors

storage

frontend servers

the cloud (AWS)

websocket

sticky sessions

user’s browser

web dashboard