

Hyper Scalable Software Systems

Ian Gorton

Professor, Director of Computer Science,
Northeastern University, Seattle Campus

Email: i.gorton@neu.edu

Scalability drives

Everything









Location!
Location!
Location!



Where the tech jobs are: Seattle, S.F., D.C.

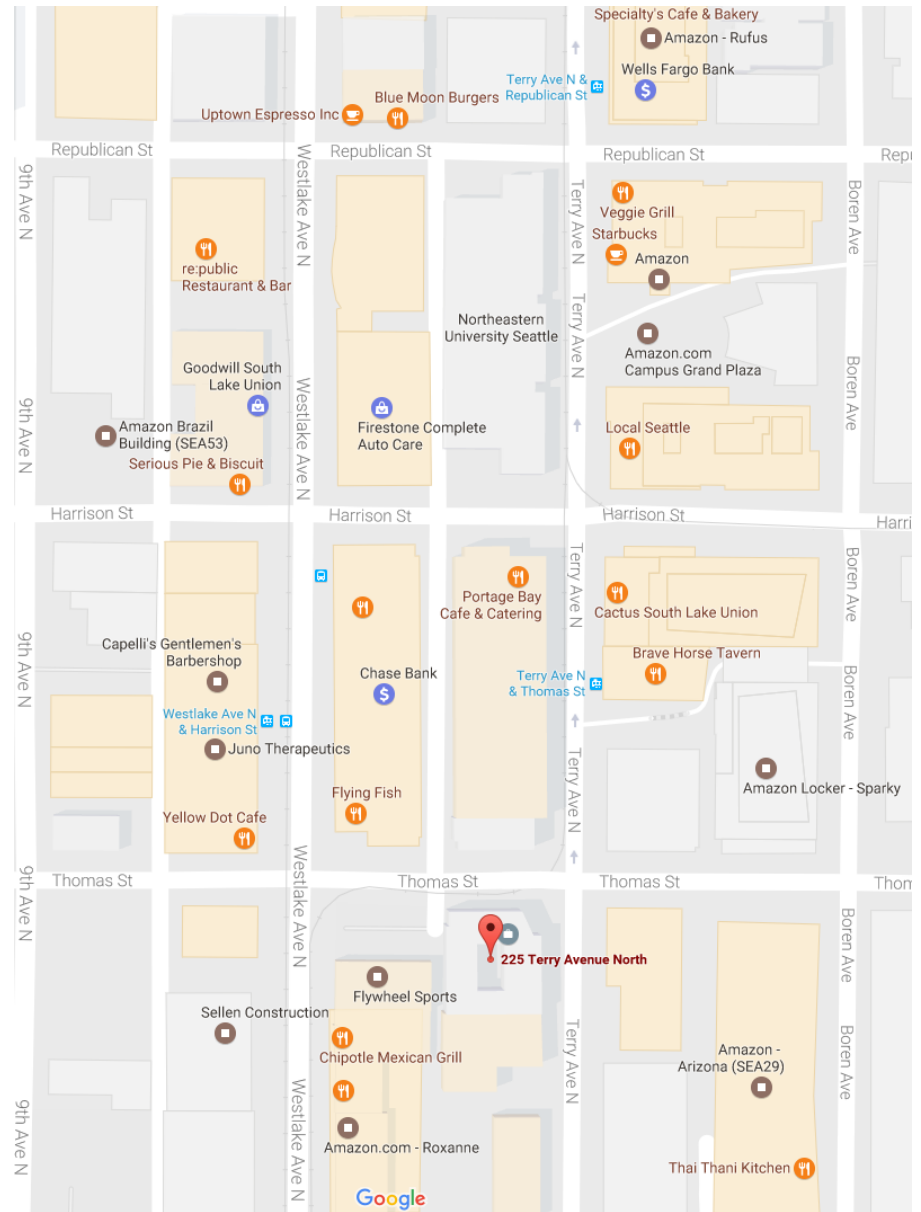
[Jon Swartz](#), USA TODAY Published 5:21 p.m. ET Aug. 7, 2017 | Updated 12:55 p.m. ET Aug. 8, 2017

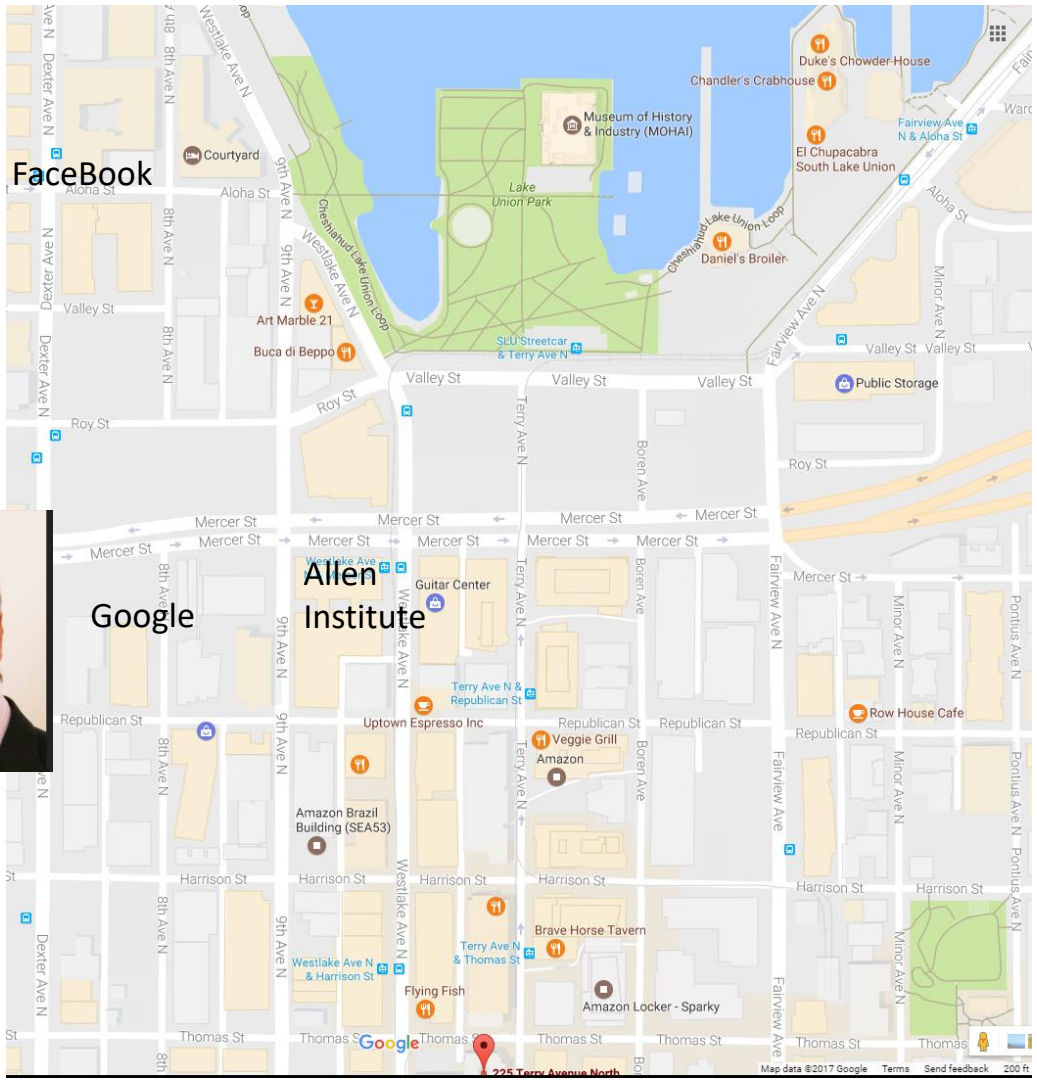
Drilling deeper, in a separate study, career assistance firm Paysa analyzed more than 8,200 job postings and more than 70,000 resumes from the world's leading tech companies to figure out who's hiring who, and which skills are the most in demand.

The Top Talent of Tech Disruptors and Titans

What it discovered is the skill most in demand was computer science, followed by infrastructure, management and analysis.

South Lake Union





What is Scalability?

- Let's start with some examples
- Remember
 - Youtube started in November 2005
 - Facebook went available to the public in September 2006

Example: YouTube

- 4 billion video views per day
- 6 billion hours of video watched per month
- 300 hours of video uploaded per minute

<http://expandedramblings.com/index.php/youtube-statistics/>

Viewed October 2015

Facebook Photo Storage

- 2009
 - 15 billion photos, 4 replicas each
 - 1.5PB
 - 30 million new photos per day
- 2013
 - 240 billion photos total
 - 350 million photos a day
- 2015
 - 2 billion photos *per day*,
 - 40 PB of new disk capacity per day

Wordpress.com

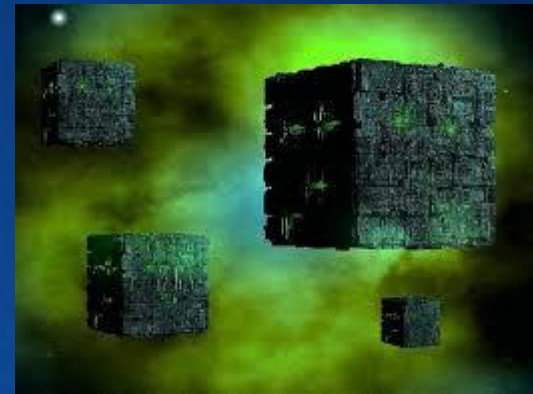
- Runs 28% of the entire internet
- 17 posts per second are published globally
- Around 15,886,000 websites including:
 - 2,645 of the top 10k websites on the web
- WordPress gets more unique visitors than Amazon (126 million per month vs. 96 million per month)
- 409+ million people view more than 19.6 billion pages on WordPress.com monthly
- 54.2 million new posts and 49.9 million new comments are added monthly
- Has only 564 employees. Wordpress is open source!

Scale is being driven by ...



So where are we heading?

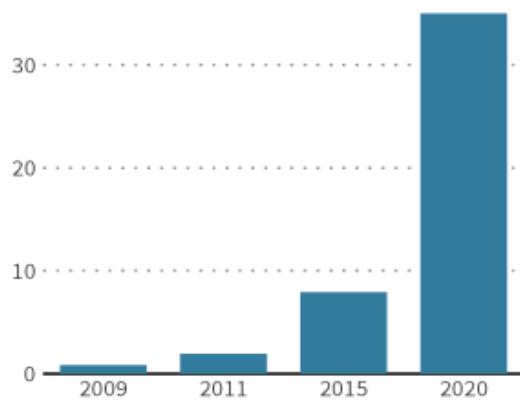
- We're software engineering research-y folks
- So we should be tackling problems that will manifest themselves in the future?
 - Maybe 10 years?



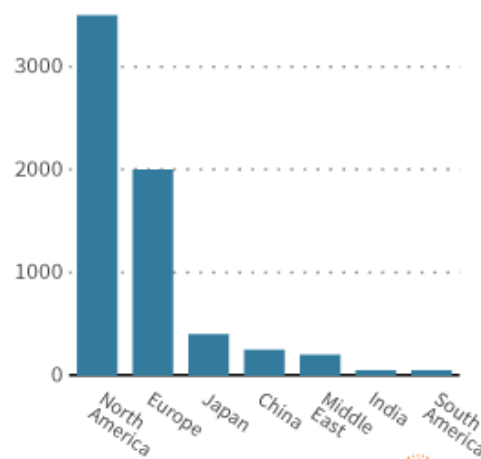
Big data growth

Big data market is estimated to grow 45% annually to reach \$25 billion by 2015

Growth of Global data - Zettabytes
Zettabyte = one million petabytes



2010 Stored data* - Petabytes
Petabyte = one quadrillion (short scale) bytes



*greater than

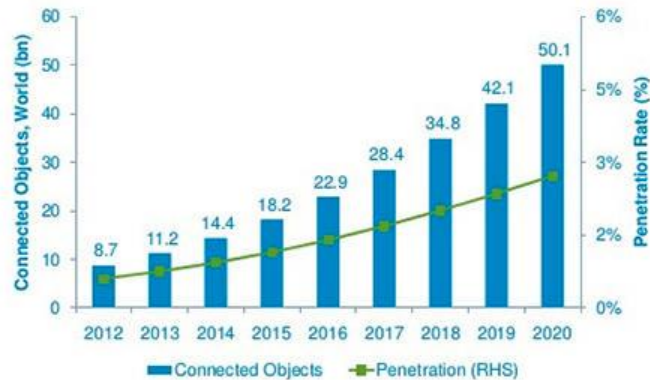
Sources: Nasscom -CRISIL GR&A analysis



Reuters graphic/Catherine Traveilhan 05/10/13

2025???

Number of Connected Objects Expected to Reach 50bn by 2020



Penetration of connected objects in total 'things' expected to reach 2.7% in 2020 from 0.6% in 2012

Source: CCS, 2013

© 2013 Cisco and/or its affiliates. All rights reserved.

Cisco/PAIS 3

2025???

<http://www.zdnet.com/article/the-internet-of-things-and-big-data-unlocking-the-power/>

The future is about
Big Data
Big Problems

Just need more machines and disks?



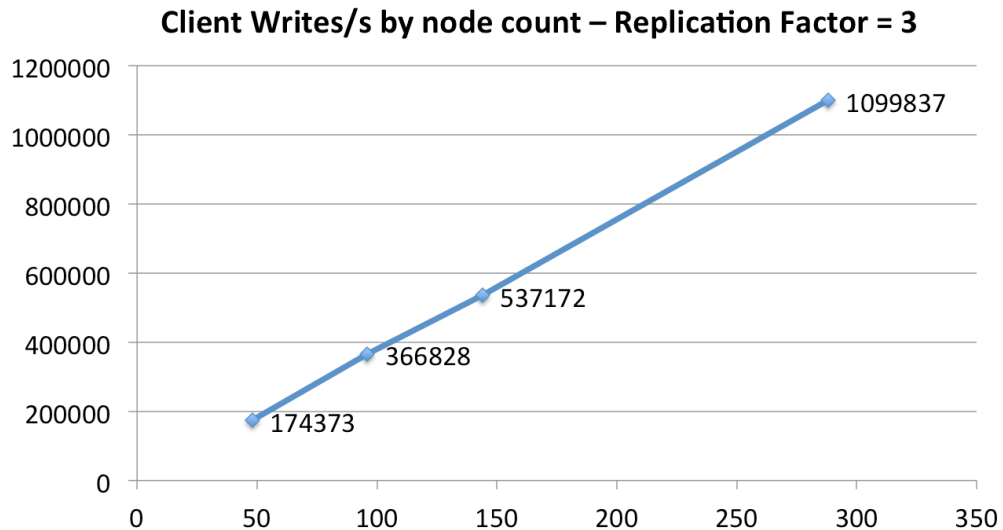
2025???



Major Internet Companies
have 1m+ servers in 2013

And Scale Linearly

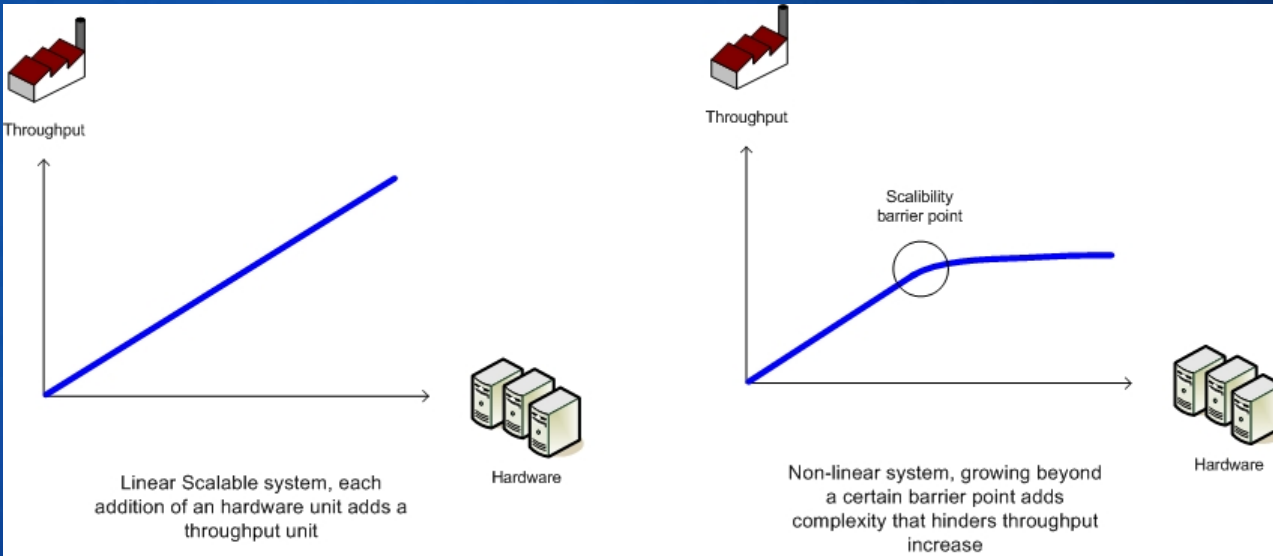
Scale-Up Linearity



<http://www.datastax.com/2012/01/choosing-the-right-architecture-for-big-data-scale>



Anyone
remember
Amhdahl's
law?

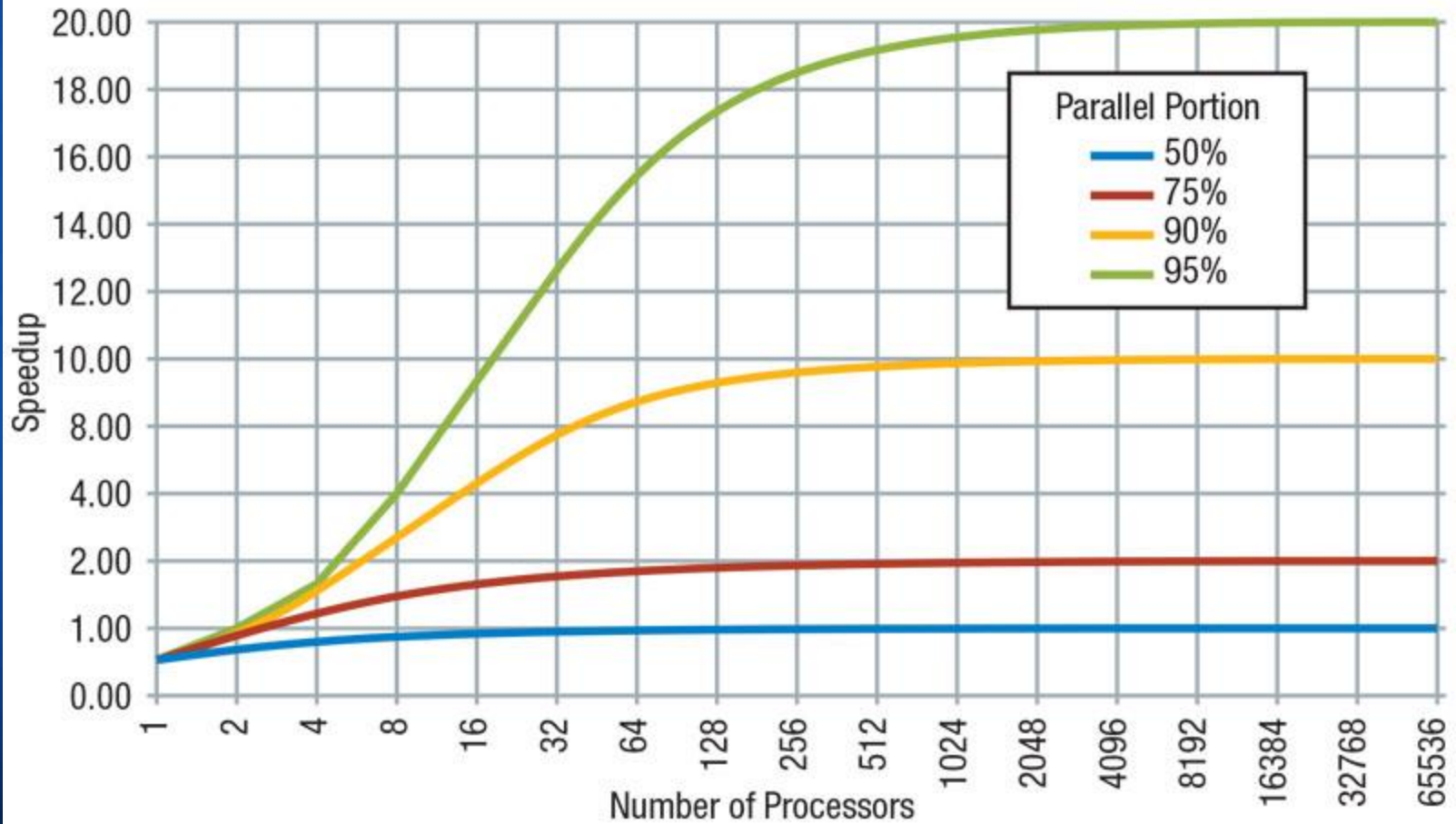


Ahmdahl's Law

2025???

Ahmdahl's Law

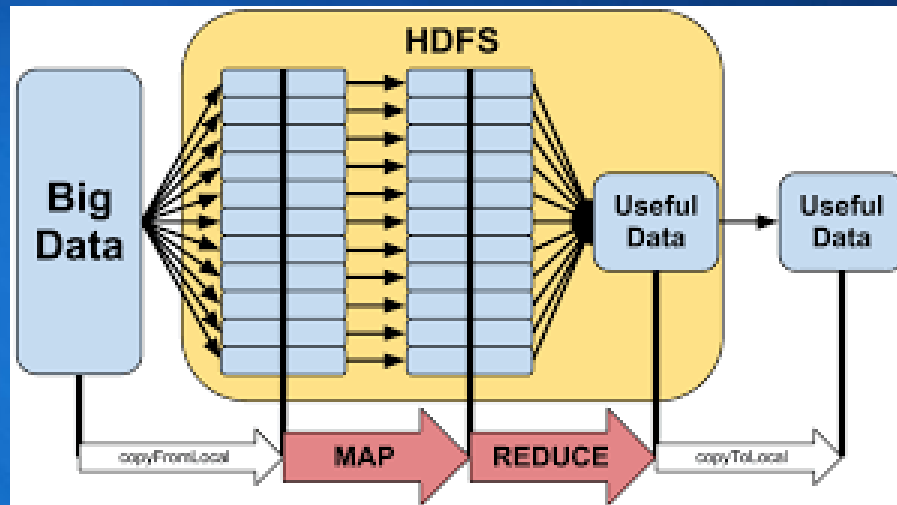
Amdahl's Law

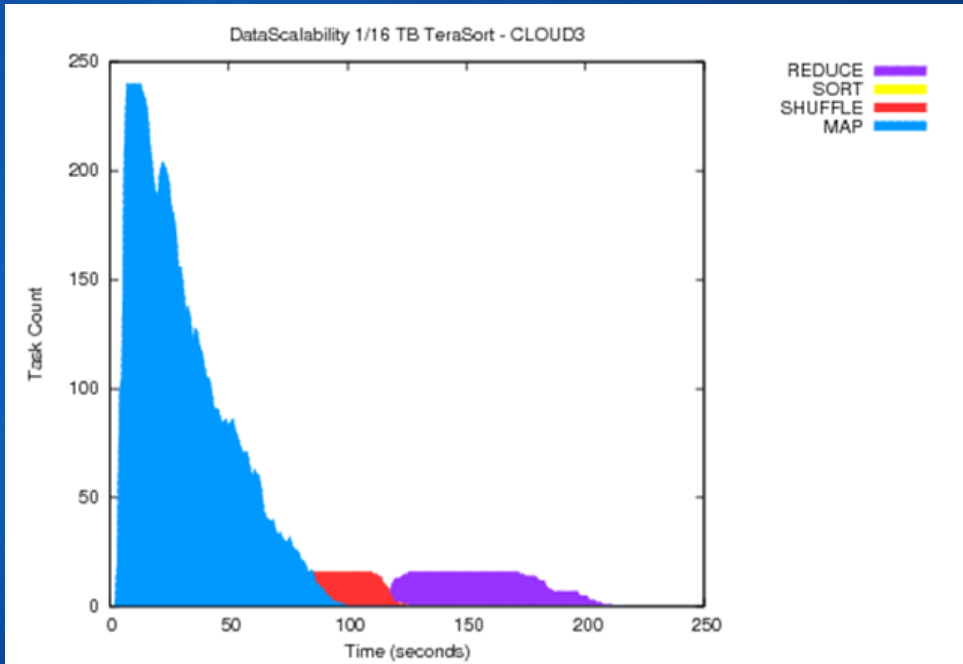


Ahmdahl's Law in 2025???



Many
problems
are
easily
parallelized
though?





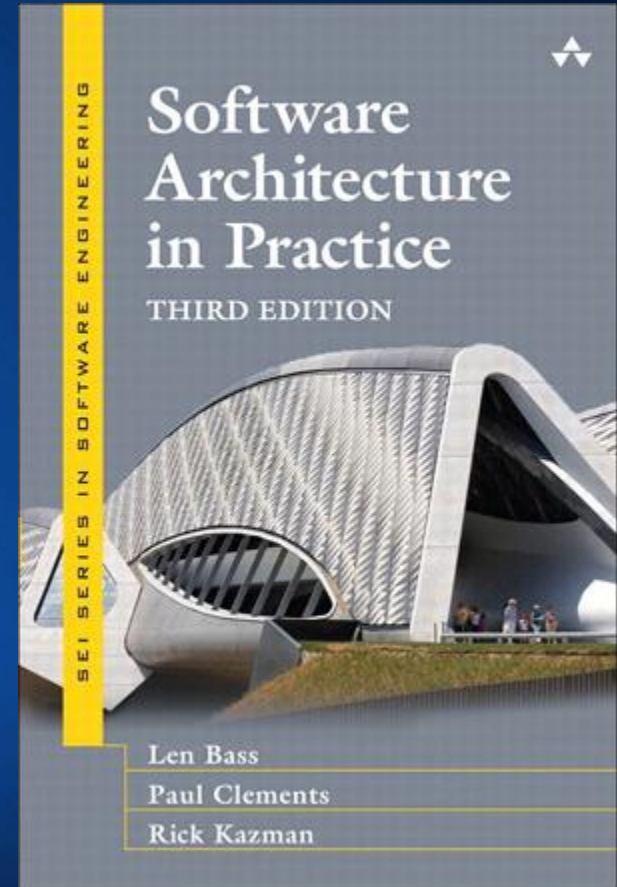
Data
Skew???



The future is Big Problems
That are difficult to design
And difficult to build
And difficult to deploy

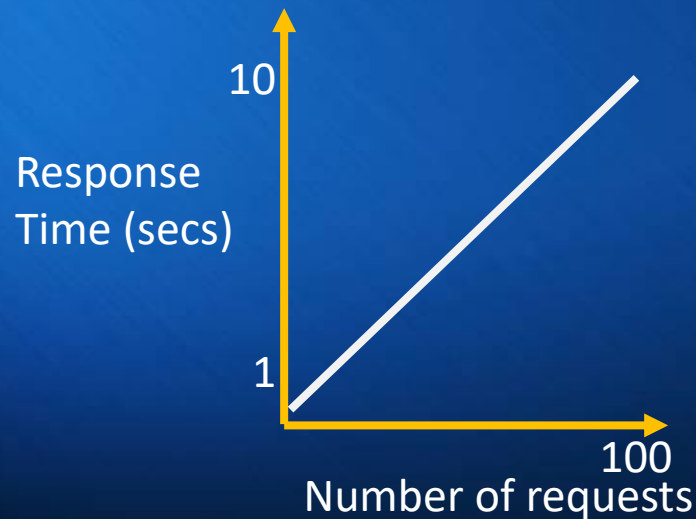
What is scalability?

- Let's ask the experts....
 - First mention is on p187
 - 4th of eight 'other quality attributes'
 - 1/3 of a page ...
 - in a 589 page book
- Weird?
 - No criticism implied, just weird



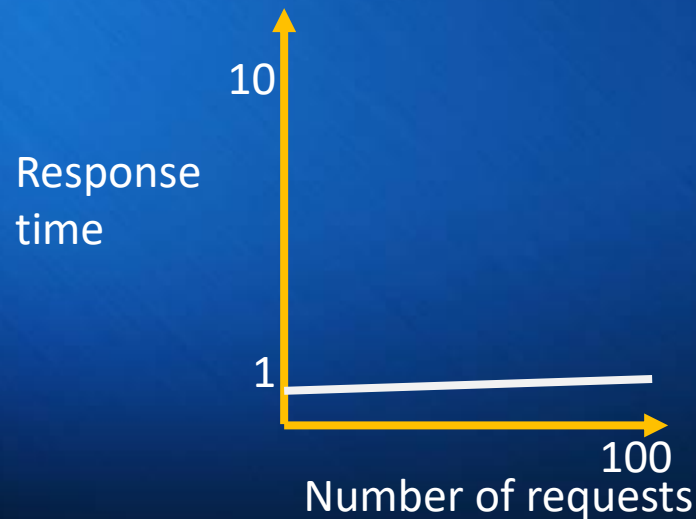
What is scalability?

- Let's try out some scenarios
 - My system supports 10 concurrent requests with 1 second response time on one server
 - I want to support 100 concurrent requests
 - Test with 1 server



What is scalability?

- Let's try out some scenarios
 - My database system supports 10 concurrent requests with 1 second response time on one server
 - I want to support 100 concurrent requests ...
 - Test with 10 servers



What is scalability?

- 10 servers gives us 1 second response time 😊
- if moving from 1 server to 10 servers took N hours of effort, was the original system scalable if N is:
 - 1?
 - 10?
 - 50?
 - 100?
 - 1000?
 - 100000?

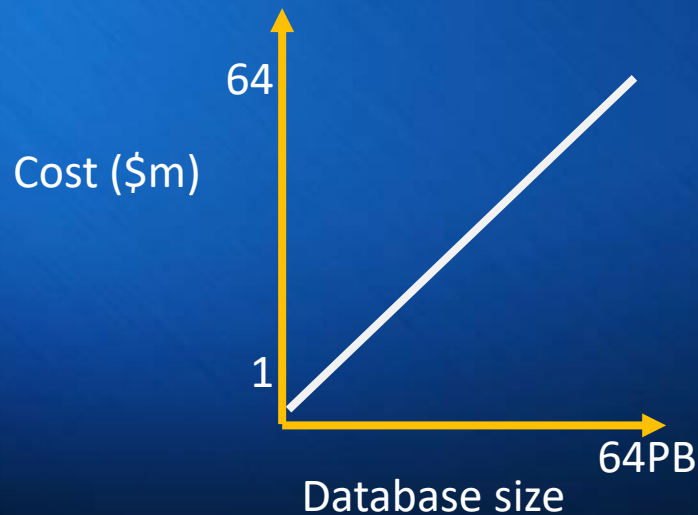
What is Scalability? An Insight

- A system not designed to scale will cost more to transform
 - But it's probably doable
- Scalability = $f(\text{effort, cost, runtime})$



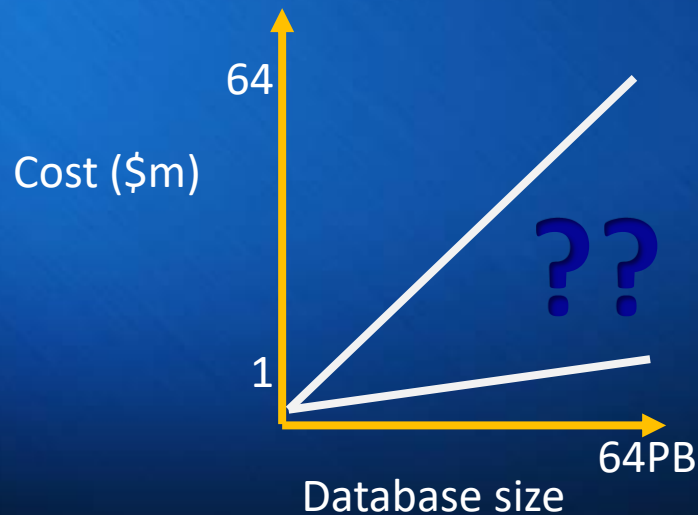
What is Scalability?

- Let's try an effort/cost scenario
 - My database is 1PB
 - Needs to grow from 1PB to 64PB in 1 year
 - Total cost = $f(\text{hardware, effort})$
 - Current cost are \$1million/month for deployment



What is Scalability?

- Let's try an effort/cost scenario
 - My database is 1PB
 - Needs to grow from 1PB to 64PB in 1 year
 - Total cost = $f(\text{hardware, effort})$
 - Current cost are \$1million/month



What is Scalability? Another insight

...

- Costs = fixed + variable
- In our example: Costs = hardware + effort
- Successful scalability strategies require 'minimal' effort

Scalability in the Ideal World



Inability to scale can be fatal

1. Myspace didn't have programming talent capable of scaling the site to compete with Facebook.
2. Choosing the Microsoft stack made it difficult to hire people capable of competing with Facebook. .Net programmers are largely Enterprise programmers who are not constitutionally constructed to create large scalable websites at a startup pace.
3. Their system had " "hundreds of hacks to make it scale that no one wants to touch," which hamstrung their ability to really compete.
4. Because of their infrastructure MySpace can't change their technology to make new features work or make dramatically new experiences.
5. Firing a lot of people nose dived morale and made hiring tough. (duh)
6. Los Angeles doesn't have startup talent capable of producing a scalable social network system.
7. Facebook's choice of the LAMP stack allowed them to hire quicker and find people who knew how to scale.

<http://highscalability.com/blog/2011/3/25/did-the-microsoft-stack-kill-myspace.html>

What is Scalability - Better

the

*capability of a system, network, or process to
handle a growing amount of work,*

and

*its potential to be enlarged
in order to accommodate that growth.*

http://johnewart.net/posts/soa_in_practice/birth_of_the_monolith/



FAXINGEOM

Hyper scalable systems exhibit
exponential growth rates in computational resources
while
exhibiting **linear** growth rates in the operational costs
of resources required to build, operate, support and
evolve
the required software and hardware resources.

Engineering at HyperScale is
Difficult

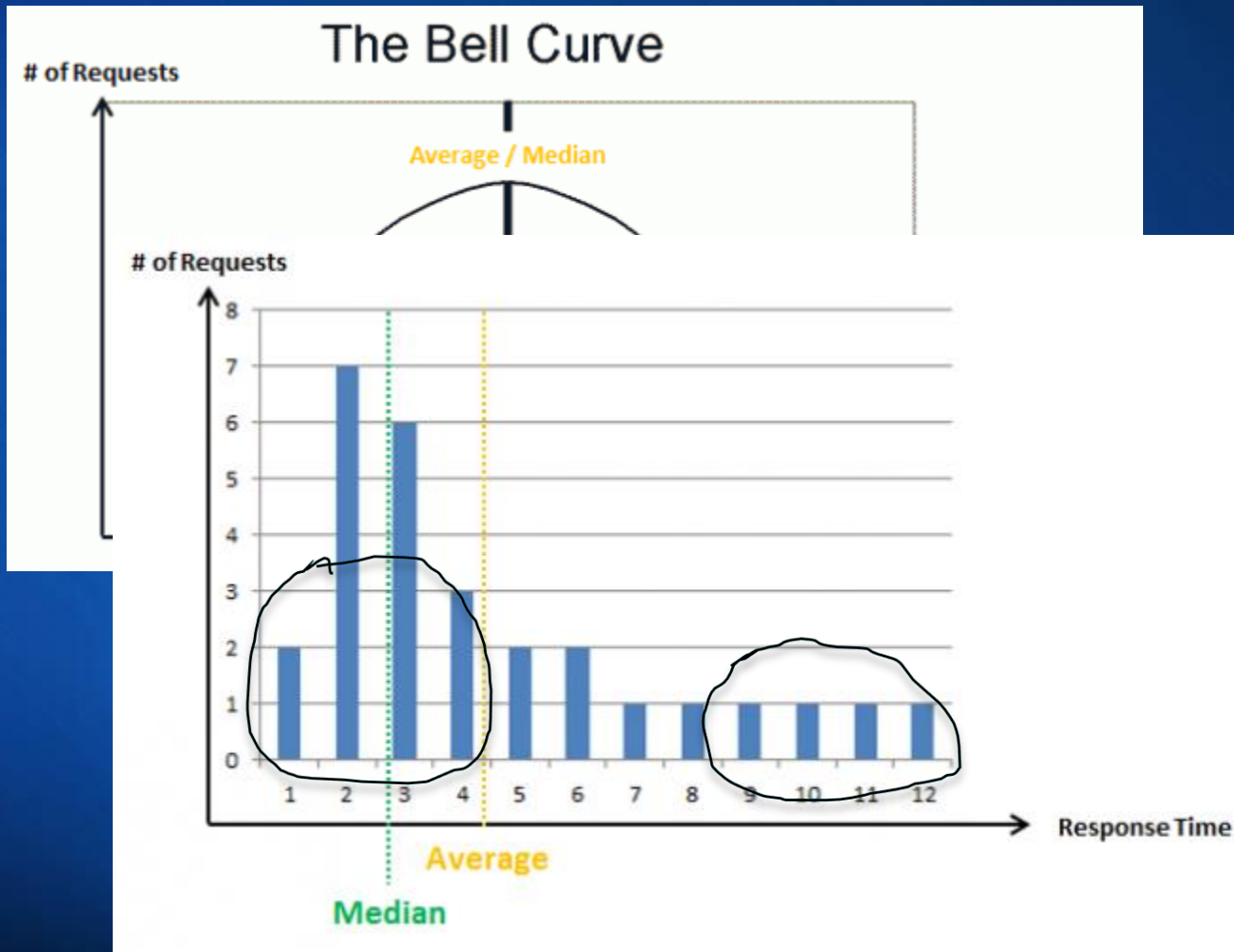
Systems at massive scale are different

- Scale in itself changes everything
 - Well understood principles and practices don't work at scale
- Some examples
 - Architectures/Patterns
 - Team organizations
 - Testing approaches

Architecture Metrics

- Systems typically built to provide an acceptable average response time
 - E.g 2 seconds mean response for GUI/Browser
 - 100 ms mean response time for database query

Mean Response Times



<http://apmblog.dynatrace.com/2012/11/14/why-averages-suck-and-percentiles-are-great/>

Slow responses at scale

- most applications have a few very heavy outliers
 - curve has a *long tail*
 - A few requests that are magnitudes slower than the mean
- This is a BIG problem at scale
 - They consume resources for an unpredictable period of time
 - Slow responses impact business value
 - Can cause cascading failures

Use Percentiles Instead

- Target service response times for e.g. 99th percentile
 - 99% of requests will be satisfied in 100ms
- Service requestors can use aggressive timeouts with this knowledge
 - Timeout after 300ms?
- Prevents blocked resources and risk of cascading failures
- Circuit breakers can be used in service requestor to shed load if the called service is unexpectedly slow

The Pragmatic
Programmers

Release It!

Design and Deploy
Production-Ready Software



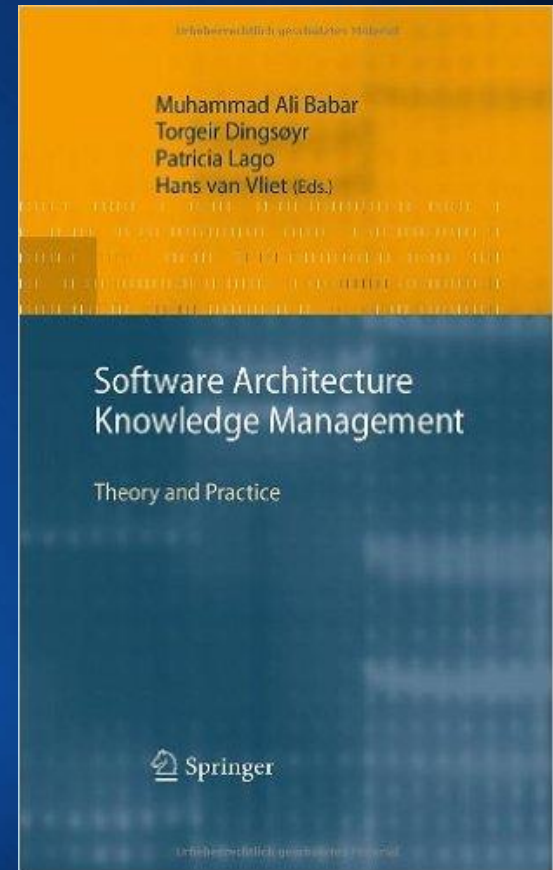
Michael T. Nygard



i

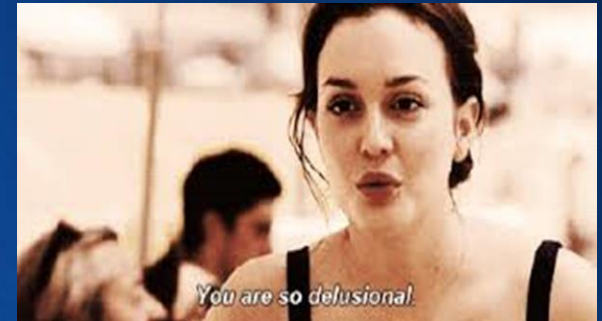
Team Communications

- We all know software architecture documentation is usually limited/non-existent
- Many researchers therefore claim we need more documentation



Team Communications

- If we (ideally) need D docs for a system of size N
- We need $\sim 10D$ for a system of size $10N$?



**But we've
always done
it this way**

GCU 7

Team Communications

- Common approaches are agile and flexible
 - Small teams (e.g. Amazon 2 pizza rule)
- Autonomy of decision making for services a team is responsible for
 - Short sprints, feature driven
- Services provide contracts for performance
 - E.g. 99th percentile response times
- Coordination across teams performed by engineers external to the teams

Microservices

- Same as services
- Only smaller



Microservices

- Single application as a suite of services organized around business responsibilities
- Services run in own process, typically communicate using REST/HTTP
- Independently deployable, scalable services
- Each service makes local decisions on programming languages, database, etc
- Minimal/no centralized control over design/evolution

Remember Conway's Law

- *“Organizations which design systems...are constrained to produce designs which are copies of the communication structures of these organizations.”*

Microservices

- Reflect structure of loosely coupled, independent teams
- A team owns the microservice for its lifetime, as in Amazon's oft-quoted "You build it, you run it."
- Test in production at whole system level

Software Architecture for Big Data and the Cloud

July 10, 2017

foxebook

Computers & Internet



1



0 Reviews

Morgan Kaufmann

2017-06-26

470 pages

Chapter 2: Hyperscalability: The Changing Face of Software Architecture



SCALABILITY

**does it
SCALE?**



- Dashboard
- Course Info
- Modules
- Connect
- Assignments
- My Submissions
- My Profile

Modules

01 | Introduction to Distributed Systems

Start Module

View Module Summary

02 | Concurrent Systems Programming

Experimental

Experiential

Strong theory and
engineering

Team with Qianli Ma & Yao Wang.

In this project, we plan to replace our **original load balancer & scaled Tomcat servers** with **AWS Lambda functions**.


This not only helps us focus more on the business logic and less on infrastructure configurations, but also handles the scaling-up and down automatically. There will be no more pain on the deployment and scaling process.

What we will do in the project:

- Rewrite all server-side (ski data processor) logic with Lambda Function Handlers (Node.js)
- Run client tests and compare performance between two architectures

What we will learn in the project:

- Learn the serverless architecture and apply it into practice with AWS Lambda
- Rewrite server-side code with Node.js and learn the difference between synchronous and asynchronous programming (Java vs. JavaScript)

A nighttime photograph of the Seattle skyline. The Space Needle is the central focus, illuminated in white. Other skyscrapers are lit up with various colors. In the background, Mount Rainier is visible under a dark blue sky. The foreground shows some city lights and structures.

ICSA 2018 Seattle, USA
April 30th-May 4th 2018