

Northeastern University - Seattle



CS6650 Building Scalable Distributed Systems
Professor Ian Gorton

Building Scalable Distributed Systems

Week 1 – Introduction to Scalable Systems

Introductions

- Before we begin let's quickly introduce ourselves
- Can you briefly indicate:
 - Write your name and
 - a hint on how to pronounce it (be creative!!)
 - Something about you that no one else in class knows

Outline

- A brief history of how we got here
 - Internet Scale Systems
- Modern Web Sites and Scale
- What is Scalability?
- Course Topics
- Course Structure

Learning objectives

1

Describe the evolution of software systems to achieve web scale

2

Explain the difficulties inherent in achieving linear scalability

3

Explain performance, availability and scalability

4

Lab: Get up and running on AWS

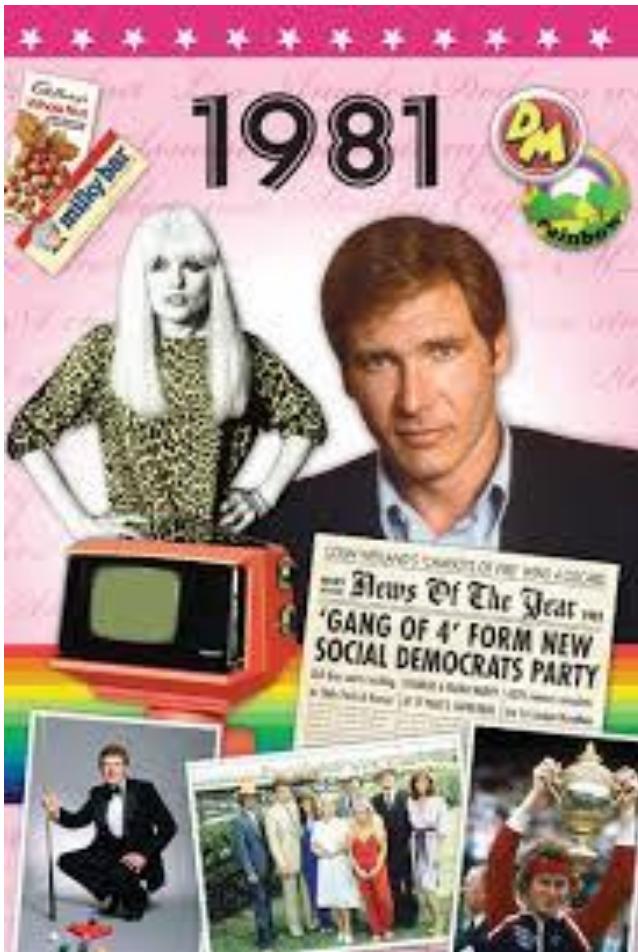
Internet Scale Systems – Some History

The first
computer I
used

(similar anyway)











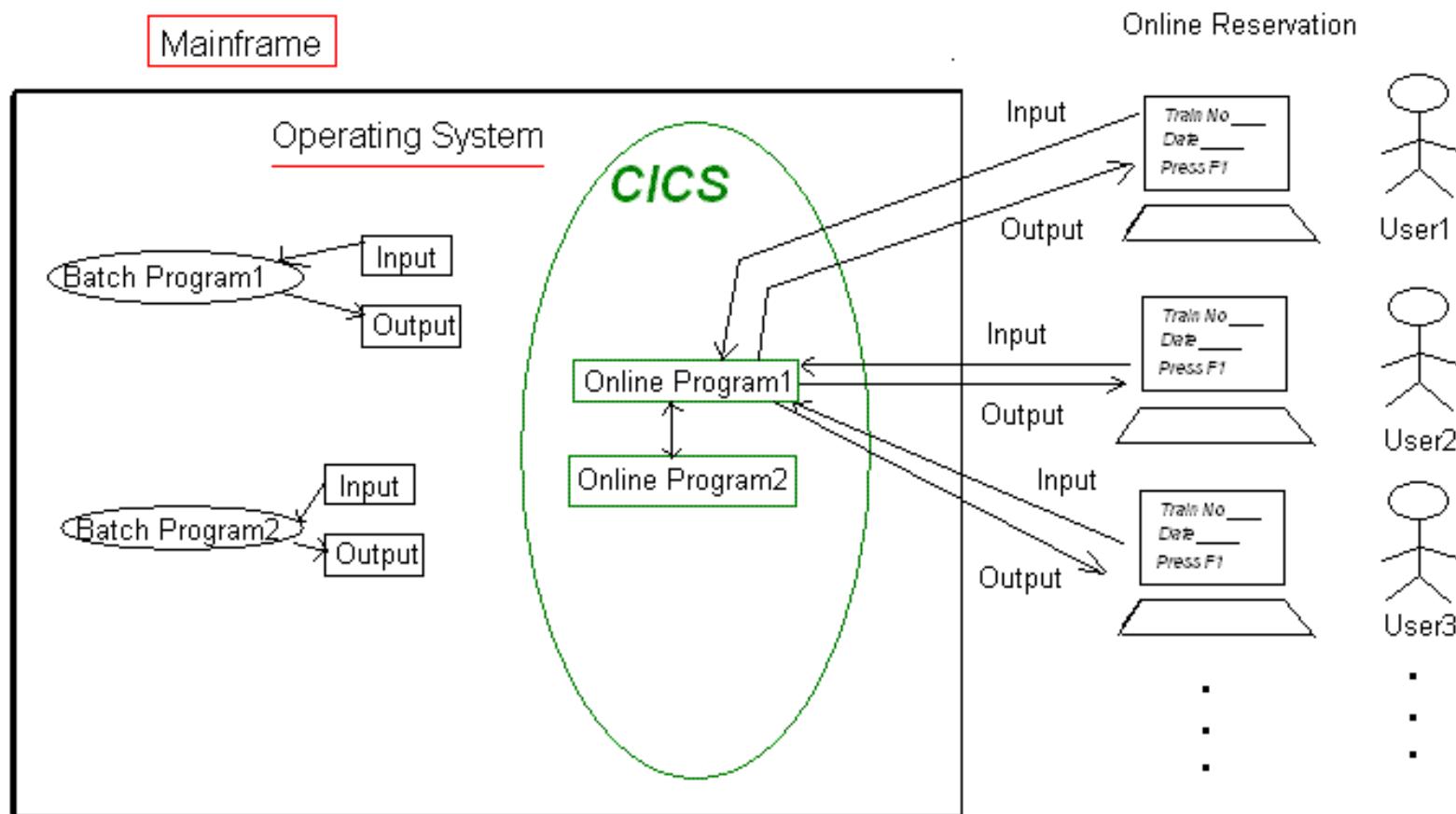
Cheshire County Council IT Department 1984



```
SV          MSS-21 Management of Sales and Service      Production Files
Clear Communications

Service Management
1. Job Ticket Entry/Update      16. Recurring Scheduling History
2. Job Ticket Parts/Labor       17. Weekly Calendar
3. Job Ticket Ready to Price    18. Quick Parts Scan
4. Job Ticket Ready to Invoice   19. Invoice Number Inquiry
5. Job Ticket Inquiry          20. Customer Service Inquiry
6. Job Ticket Profit Analysis   21. Item Inquiry
7. Scheduling                   22. Serial Number Inquiry
8. Job Ticket Release from Crd Hold 23. Serial Num Inq By Sys/Dec/Hex Id
9. Tech Hours Worked Inquiry   25. Summary Sales Analysis
10. Hours Worked Inquiry        26. Customer Maintenance
11. Contract Management Menu    27. Ship To Maintenance
12. Tech Scheduling              28. Item Maintenance
13. Job Ticket Invoicing        29. Technical Maintenance
14. Job Ticket Invoicing         30. Sales Order Management Menu
15. Recurring Scheduling        More...
Selection or command
==>_
F3=Exit F4=Prompt F6=Messages F9=Retrieve F10=Goto Menu F12=Previous
F18=Spool Files F21=Search Menu F22=Initial Menu
(C) copyright systems implementation, inc., 2009
```





```

DEF PROG(PGM1) GROUP(GRP12)
OVERTYPE TO MODIFY
CEDA DEFine PROGram( PGM1      )
  PROGram      : PGM1
  Group        : GRP12
  Description   ==> -
Language    ==> Cobol | Assembler | Le370 | C | Pli
RELoad       ==> No | Yes
RESident     ==> No | Yes
USAge        ==> Normal | Transient
USElpacopy   ==> No | Yes
Status       ==> Enabled | Disabled
RSl          : 00 | 0-24 | Public
CEdf         ==> Yes | No
DAtalocation ==> Below | Any
EXECKey      ==> User | Cics
Concurrency  ==> Quasirent | Threadsafe
REMOTE ATTRIBUTES
  Dynamic     ==> No | Yes
+ REMOTESystem ==>
I New group GRP12 created.

DEFINE SUCCESSFUL
PF 1 HELP 2 COM 3 END

```

CICS RELEASE = 0630

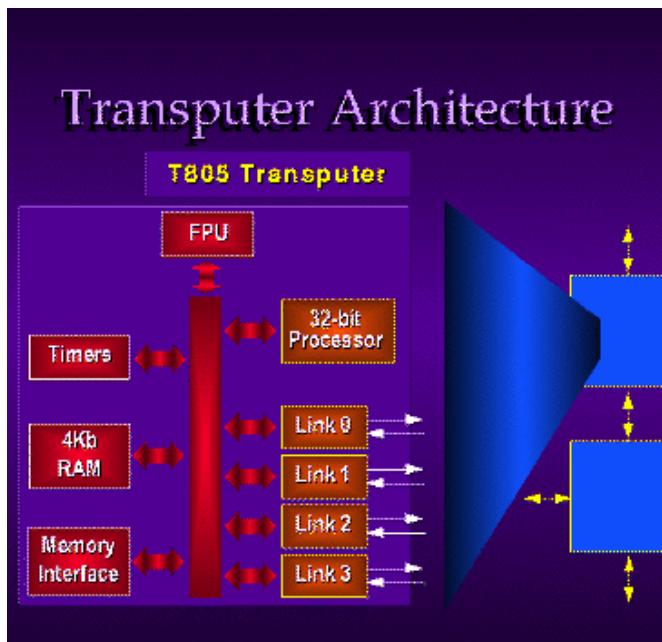
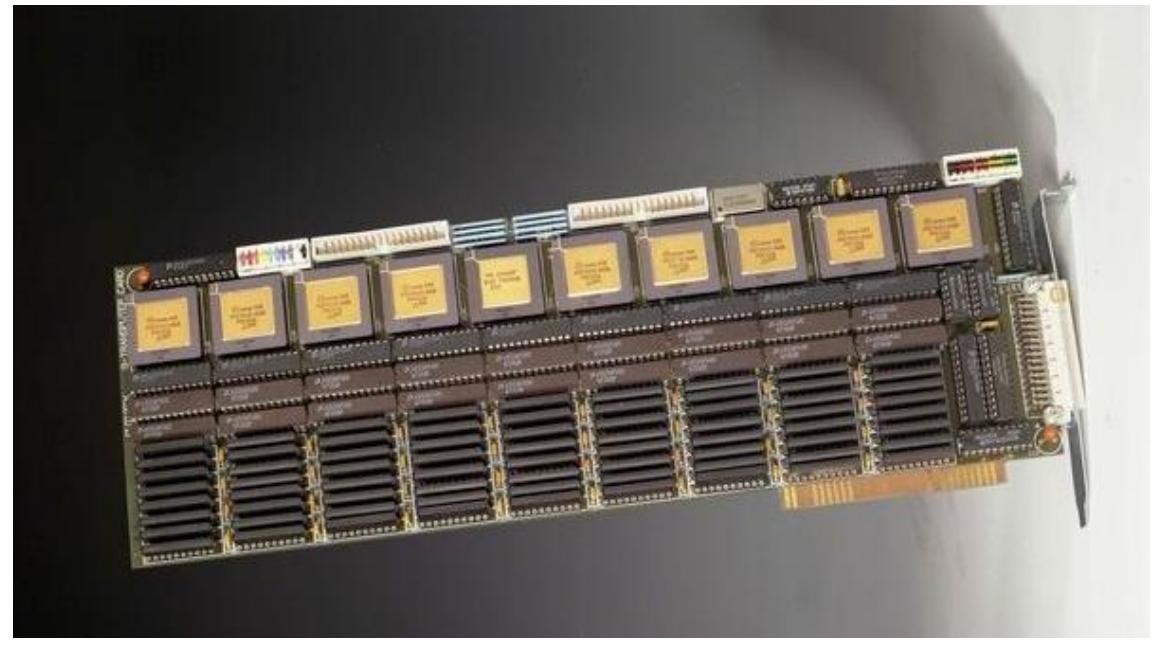
SYSID=CICS APPLID=CICS
TIME: 05.57.10 DATE: 11.280
6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL





A photograph of a man with dark hair and a beard, wearing a light blue button-down shirt. He is sitting at a desk, looking directly at the camera with a neutral to slightly bored expression. His right hand is resting against his forehead, with his fingers partially hidden in his hair. In front of him on the desk is a white mug, a blue folder or clipboard, and a silver laptop. The background is a plain, light-colored wall.

Boredom!

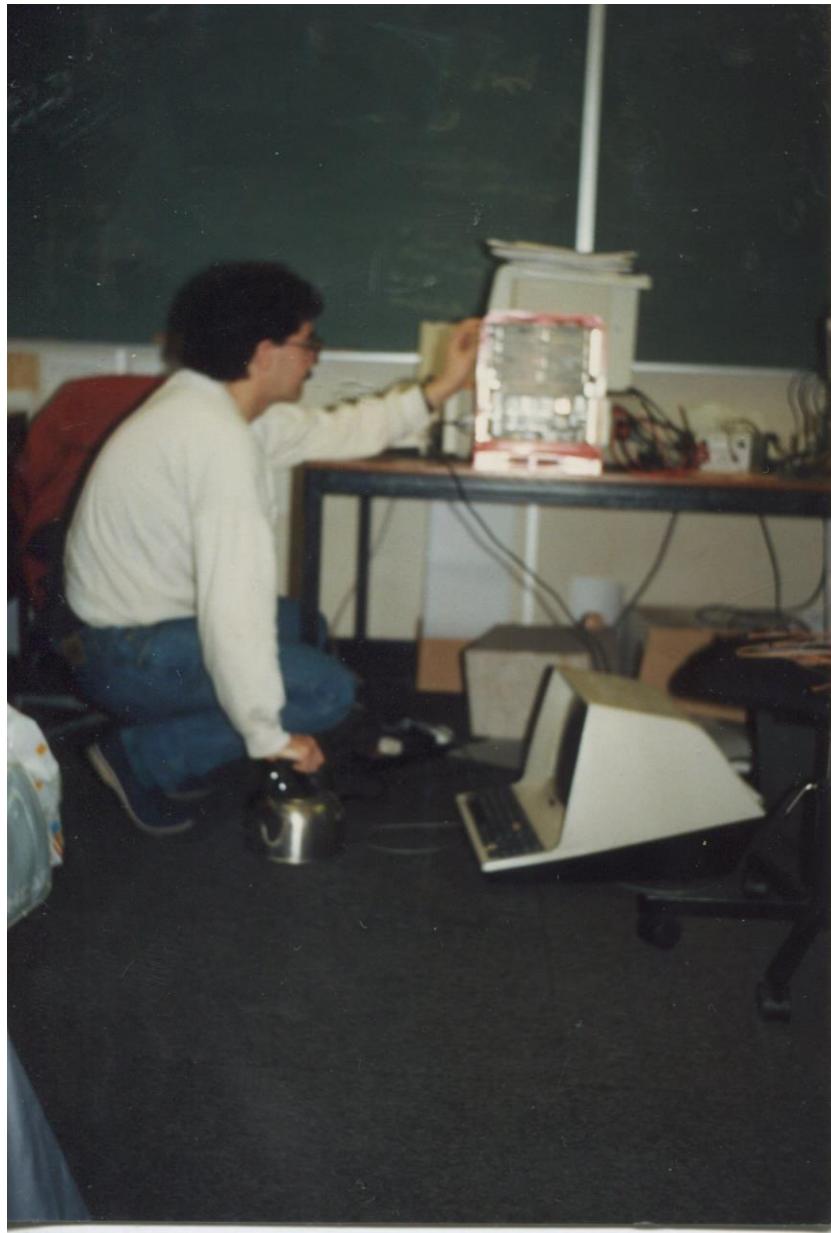


Grad School!!

Multiprocessor Transputer Systems

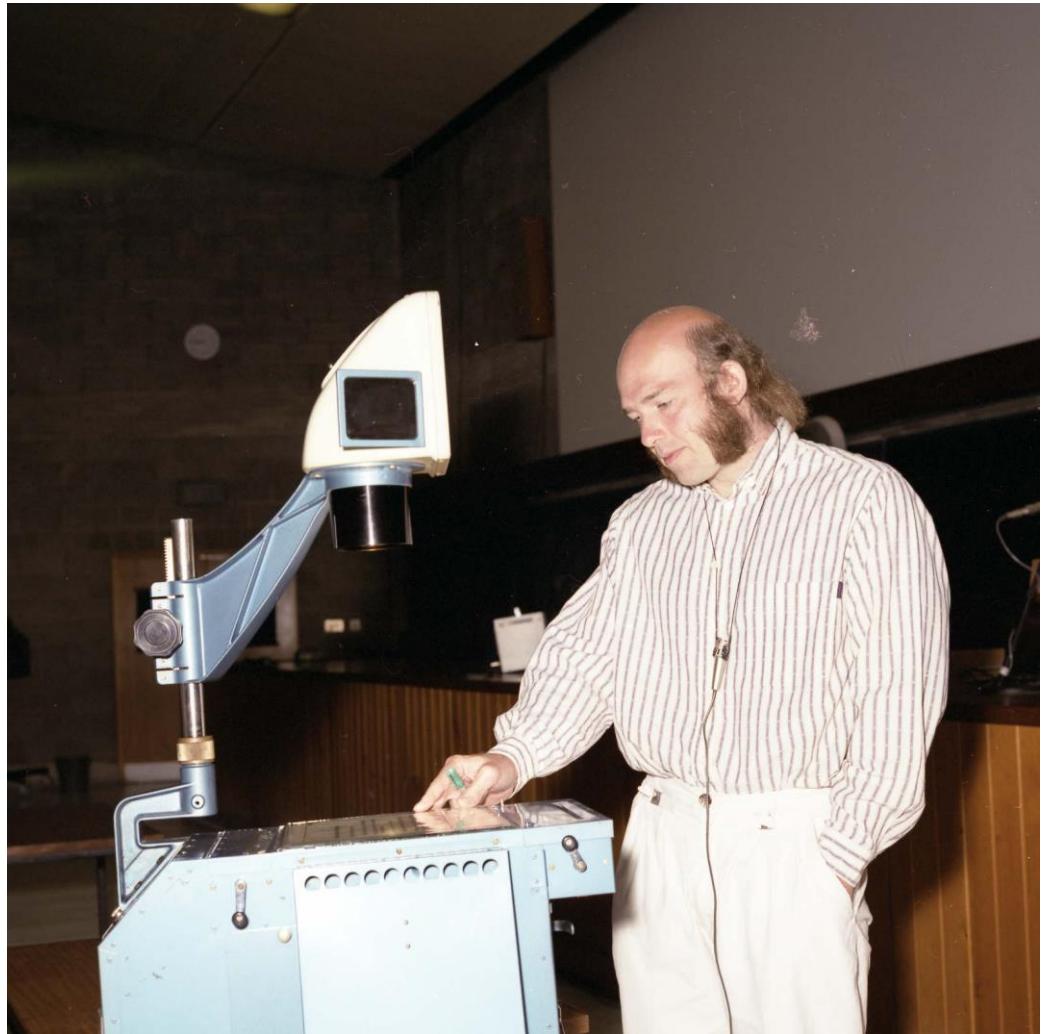


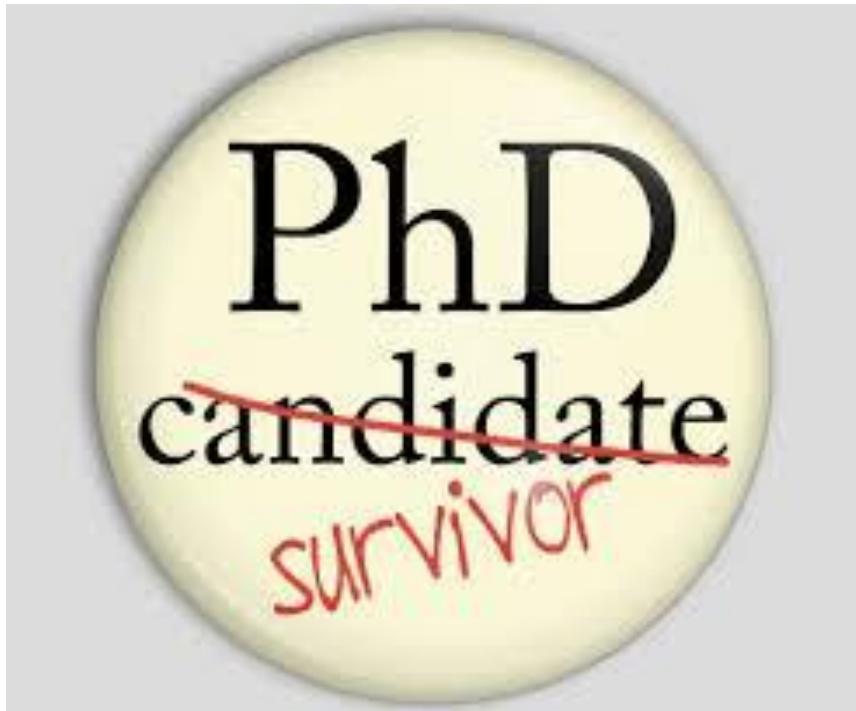
It really is
me!





David May,
Inmos
Chief
Scientist
and Occam
guru





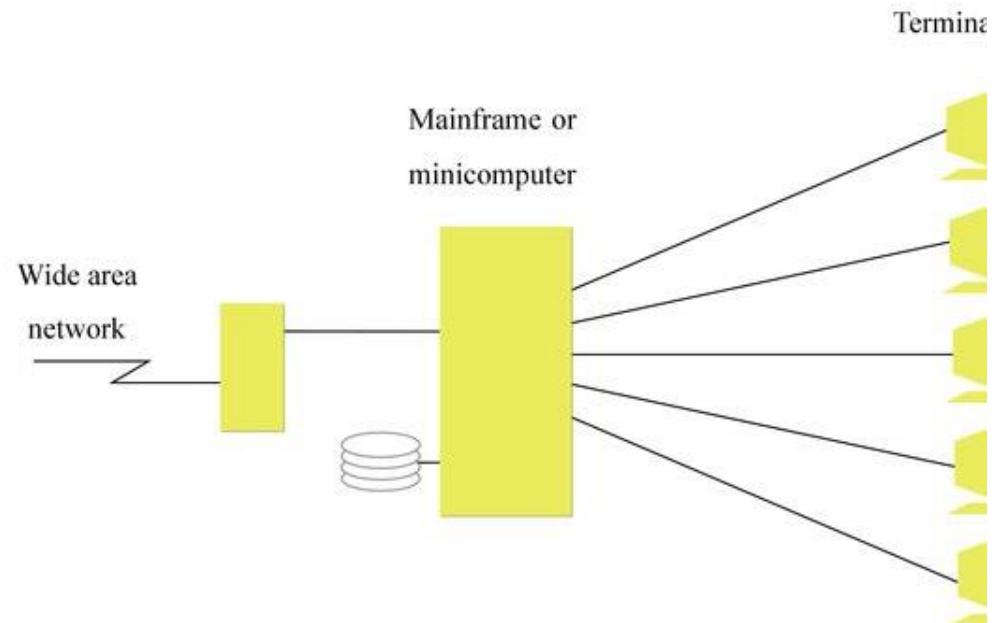
Next Stop – Australia!

1990

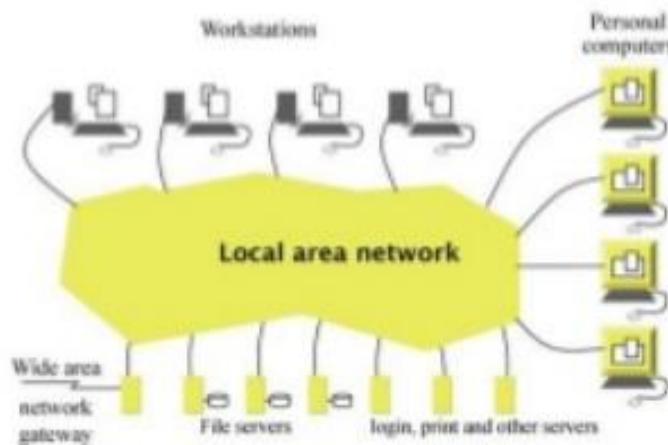
What about distributed systems?

- Rare until basically early 1990s ...
- Networks slow ...
- Protocols slow
- Computers slow 😊

A Centralized Multi-user System

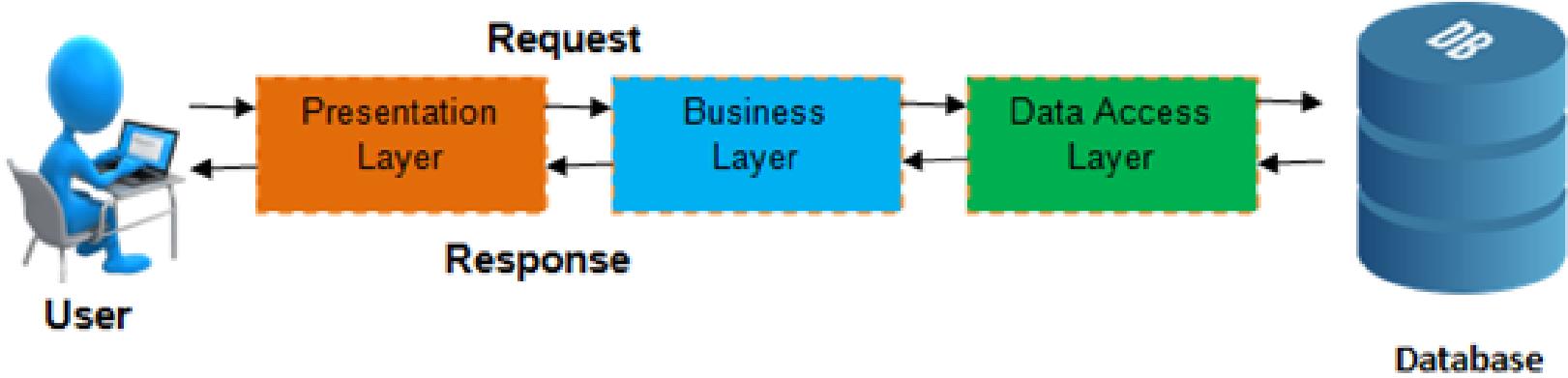


A Distributed System



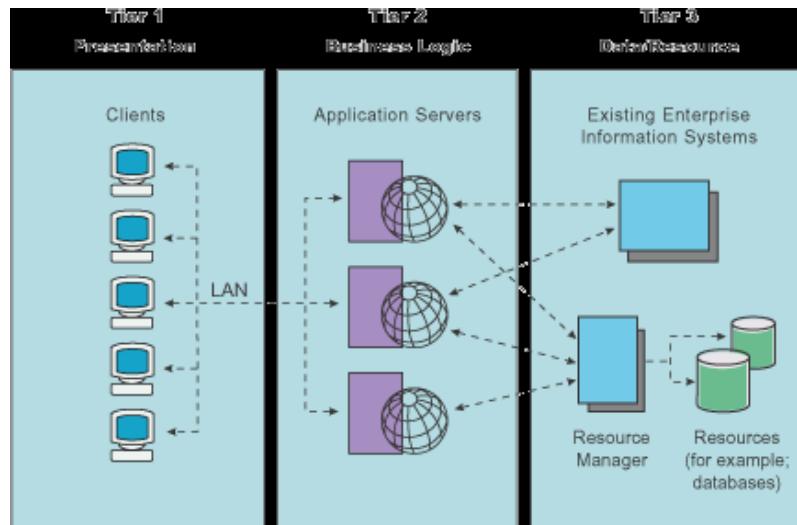
Early 1990s - the internet went mainstream-ish

- PCs and work stations become connected
 - LANs
 - WANs
- Global internet backbone
 - Corporate networks
 - Dial up at home
- Client software
 - Program
 - Browsers very simple static content serving



Three tier system becomes common

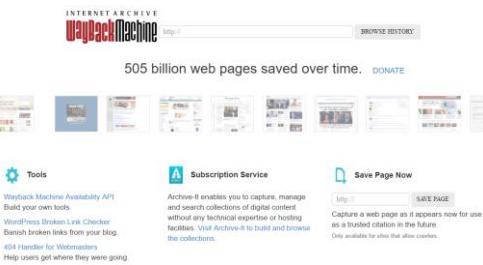
As the internet grew (mid 1990s onwards)



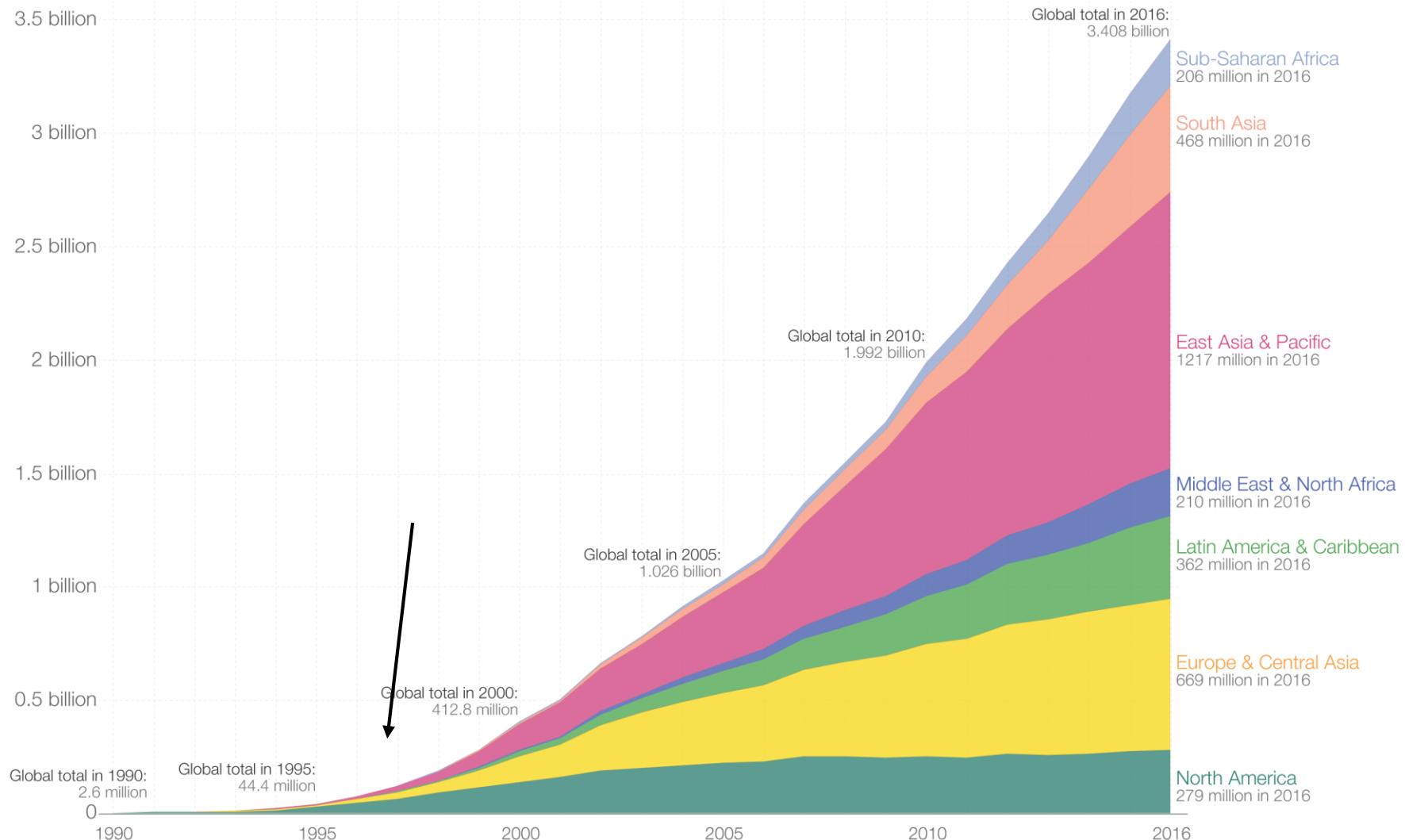
- Step change – business systems opened up to the internet
 - Eg internet banking
- Scalability achieved by
 - replication at middle tier (scale out)
 - Stateless services
 - Scaling up at database tier

What Led to Growth?

- The advent of the WWW was the backbone for the growth
- Obviously, it allowed for access by an increasing number of users
 - Potentially every person in the world can see a Web app
- It also allowed for a new service model for:
 - Enterprises
 - Consumer products
 - eGovernment
- Poke around the Wayback Machine to see the Web-past
 - <https://archive.org/web/>



Internet users by world region since 1990



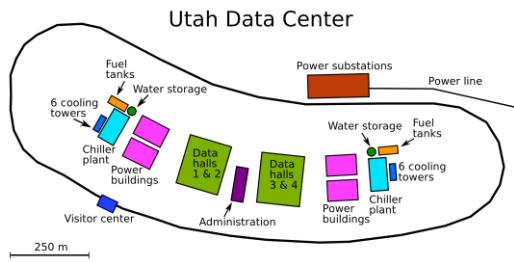
Data source: Based on data from the World Bank and data from the International Telecommunications Union. Internet users are people with access to the worldwide network.

The interactive data visualization is available at OurWorldInData.org. There you find the raw data and more visualizations on this topic.

Licensed under CC-BY-SA by the author Max Roser.



Unprecedented Data Collection and Distribution



- New service models allow for collection of data that wasn't previously available
- This knowledge can provide competitive advantage
 - If they can manage and analyze the data effectively
 - The realm of data sciences

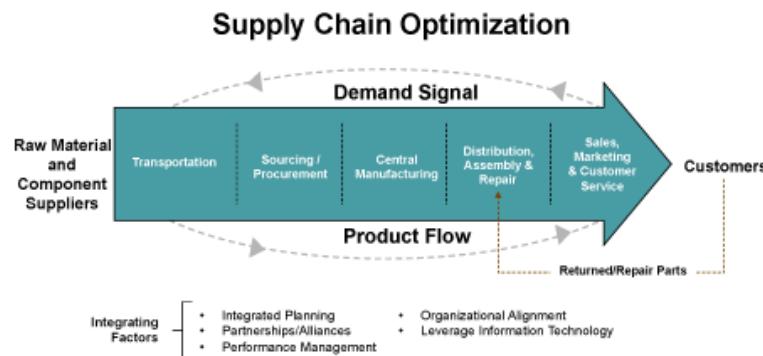
....



Wal-Mart

- Wal-Mart knows what will happen when we have inclement weather
- What items do you think are impacted by an impending hurricane?
 - Certainly water and flashlights
 - The sales of Strawberry Pop Tarts increases seven fold
 - The largest selling item, however, is beer ...
- What does knowing this allow Wal-Mart to do?

Knowledge is Profit



- Predict sales
- This allows them to have sufficient stock on hand
 - Otherwise they could sell out of items
- It also allows them to minimize the surplus stock that they need
 - Thus reducing the overhead by adopting a “leaner” ***just in time*** approach

Significant Competitive Advantage



- Wal-Mart uses this data to it's advantage
- 20 years ago Wal-Mart trailed K-Mart
 - K-Mart had more collective bargaining power and was able to negotiate a lower wholesale price
- Wal-Mart used data to streamline operations
 - They were able to reduce the percentage of the store that was used to stock surplus from the normal 25% to 10%
 - They are able to better coordinate with suppliers (resulting in more efficient production runs)
- Wal-Mart is now the largest retailer in the world

Distributed frameworks

- Sockets – Berkeley, 1983 (earlier implementations existed), became POSIX implementation
- ONC/Sun RPC – 1984-onwards
- OSF DCE – 1989-onwards
- CORBA – early 1990s-onwards
- Java RMI mid 1990-s onwards
- XML Web Services – SOAP 2000-ish
- Http – REST – Fielding thesis 2000
- WebSockets - 2001

Modern web sites and scale



21ST CENTURY APPLICATIONS

Some numbers

Dropbox – exascale storage system, millions of new files per hour, from 2012-16 seen over 12x growth.

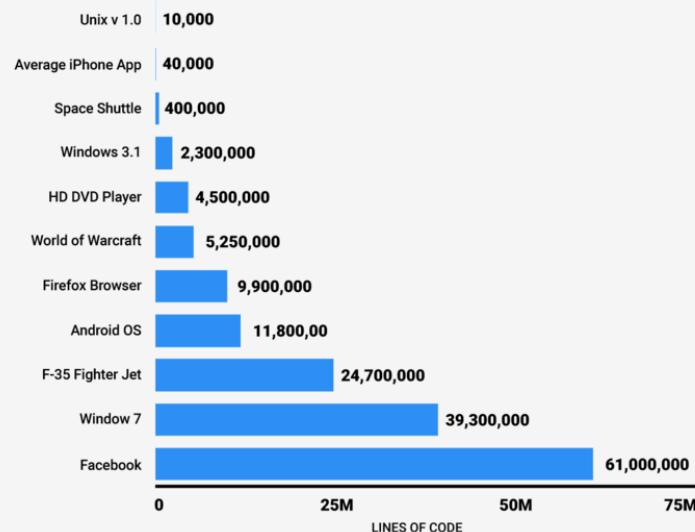
Gmail – 1.2 billion users, blocks 10M spam messages every minute

Netflix – runs on AWS, took 7 years to migrate, uses 15% of global downstream internet traffic

Facebook - More than 300 million photos uploaded per day and 510,000 comments posted and 293,000 statuses updated per minute

Youtube - Users watch 4,146,600 YouTube videos every minute

HOW MANY LINES OF CODE MAKE UP THESE POPULAR TECHNOLOGIES



SOURCE: NASA, Quora, Ohloh, Wired

BUSINESS INSIDER

Google repository statistics, January 2015.

Total number of files	1 billion
Number of source files	9 million
Lines of source code	2 billion
Depth of history	35 million commits
Size of content	86TB
Commits per workday	40,000

- **Anatomy**
- **Of a**
- **21st Century**
- **Software Systems**

Transactions

User interactions

Sensors

Internet of Things



Massive Cloud Scale Processing



Distributed
Globally

CAP Theorem



GLOBAL USER BASE

- COMPLEX BIG DATA ANALYTICS
 - BUSINESS INSIGHTS
 - SYSTEMS OPERATIONS

Building Massive Scale Systems is difficult



Software Architecture Quality Attributes





- Only one large internet site publish detailed usage statistics every year



- So let's look at some usage numbers

Pornhub

=2018=

**33.5
BILLION**
VISITS TO PORNHUB

92 MILLION
DAILY AVERAGE VISITS

=

THE POPULATIONS OF
CANADA, POLAND AND AUSTRALIA
EVERY DAY





4403 PETABYTES OF DATA TRANSFERRED

That's 574 MB of data
for every person on
earth, or about 283
photos worth

147GB
PER SECOND

529,200 GB
PER HOUR

12,700,800 GB
PER DAY

That's more
bandwidth than
the *entire internet*
consumed in 2002

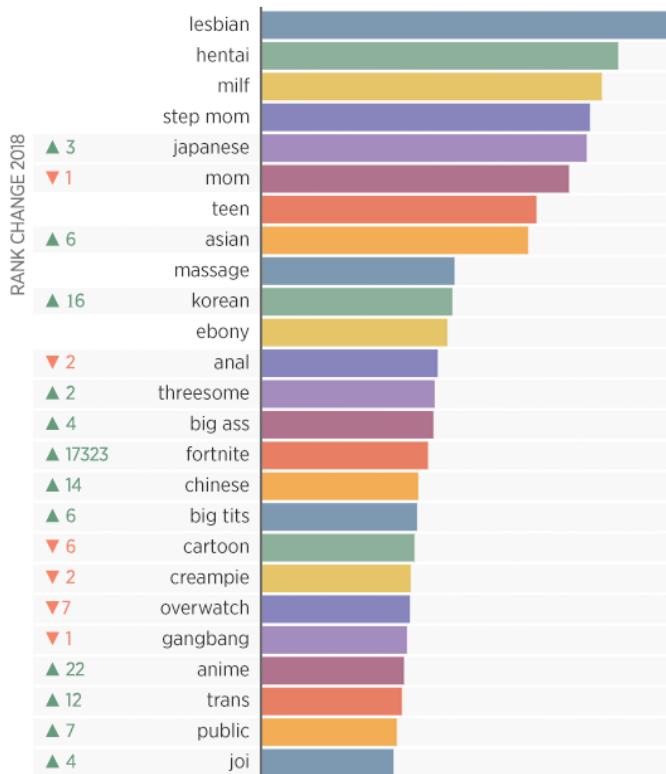




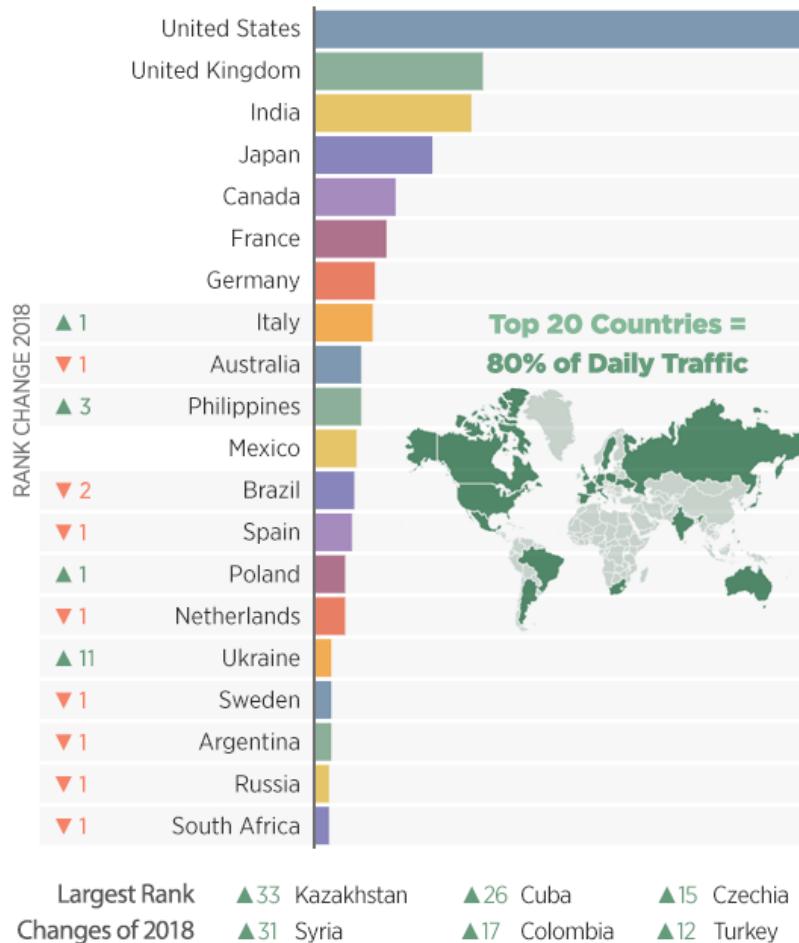
2018
YEAR IN REVIEW

Pornhub

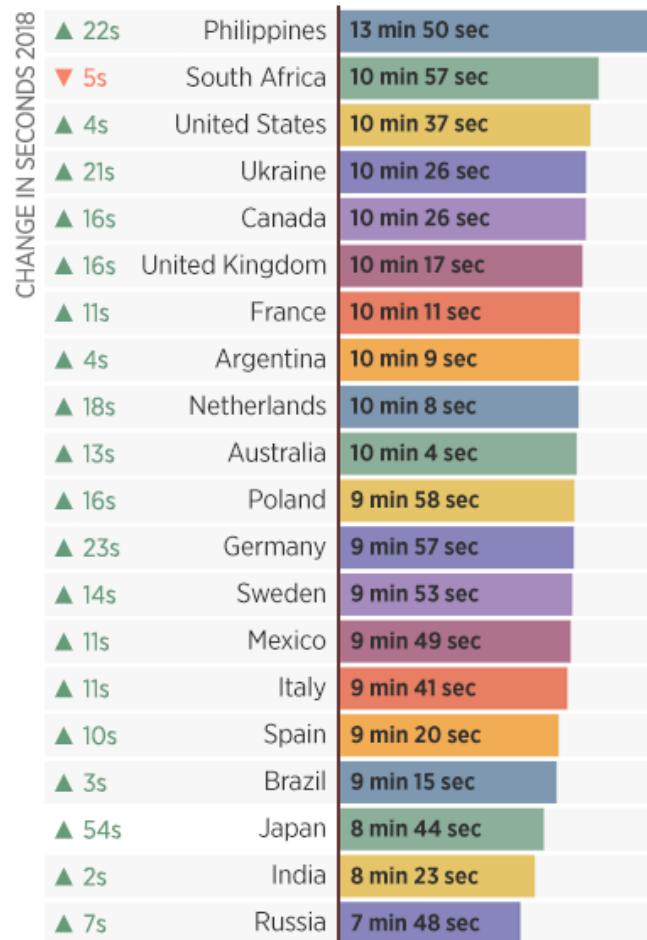
Most Searched for Terms of 2018



Top 20 Countries by Traffic



Time Spent Per Visit



10 min 13 sec

Average Visit Duration
Worldwide

▲ 14 sec

Increase in 2018

Largest Duration
Increases of 2018

Czech Republic ▲ 112s

Algeria ▲ 80s

Morocco ▲ 56s

Sri Lanka ▲ 55s

Japan ▲ 54s

Latvia ▲ 50s

Fiji ▲ 47s

Ethiopia ▲ 52s

Moldova ▲ 40s

Hong Kong ▲ 37s

2018 YEAR IN REVIEW

Porn hub

Time Spent Per Visit - United States

States That Last the Longest



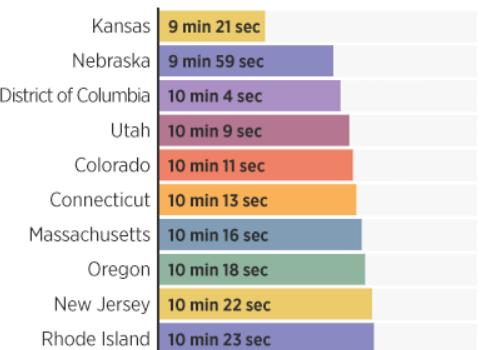
10 min 37 sec

Average Visit Duration
in the United States

▲ 4 sec

Increase in 2018

States That Last the Shortest



Largest Duration Changes of 2018

Idaho	▲ 54s
Oregon	▲ 23s
District Columbia	▼ 20s
Nebraska	▼ 19s
Kansas	▲ 16s
New Hampshire	▲ 14s
Maine	▲ 13s
Montana	▲ 12s
South Dakota	▲ 11s
Vermont	▲ 11s
Mississippi	▼ 11s



What is Scalability?

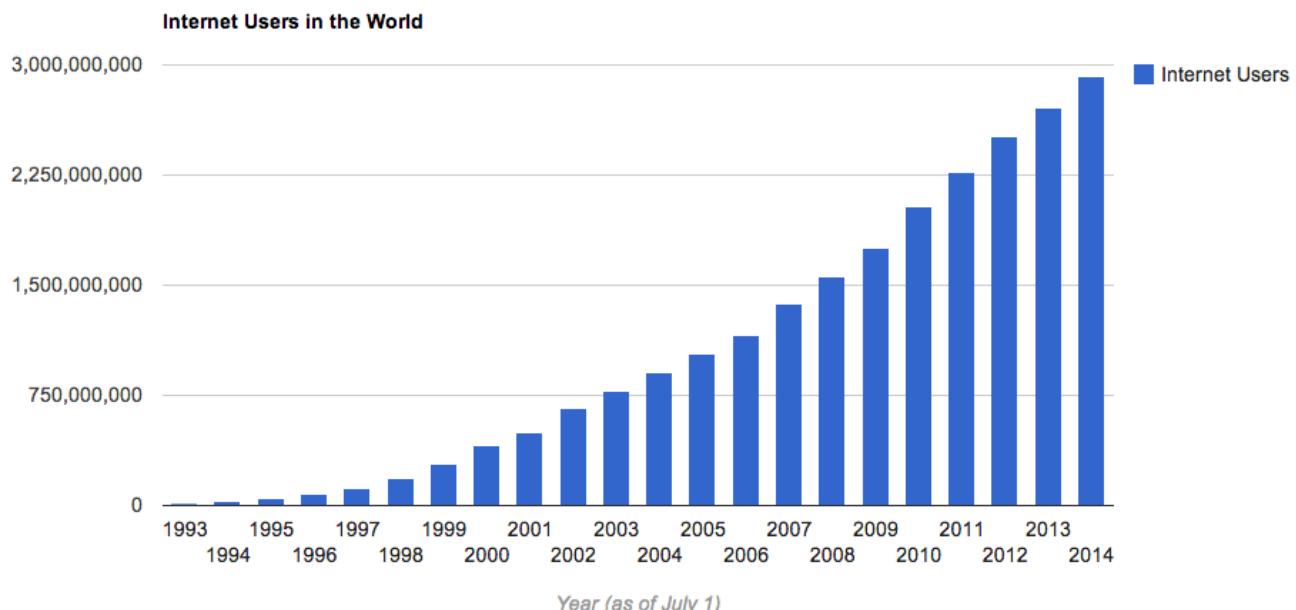
Scale is being driven by ...

Around 40% of the world population has an internet connection today ([view all on a page](#)). In 1995, it was less than 1%.

The number of internet users has increased tenfold from 1999 to 2013.

The **first billion** was reached in 2005. The **second billion** in 2010. The **third billion** in 2014.

The chart and table below show the number of global internet users per year since 1993:



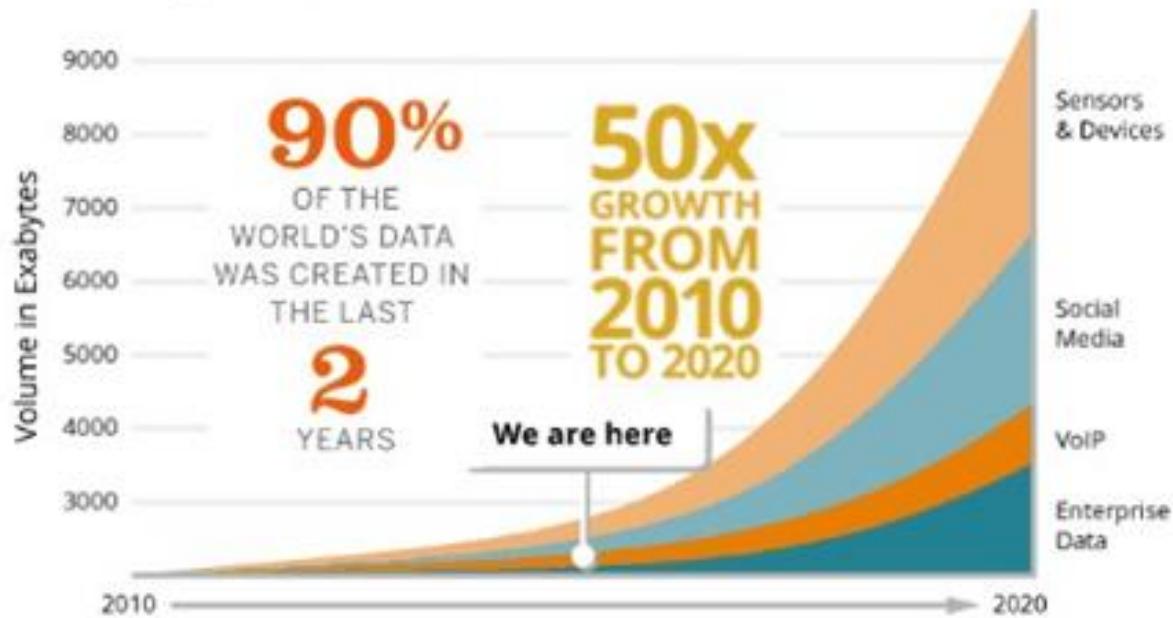
As well as



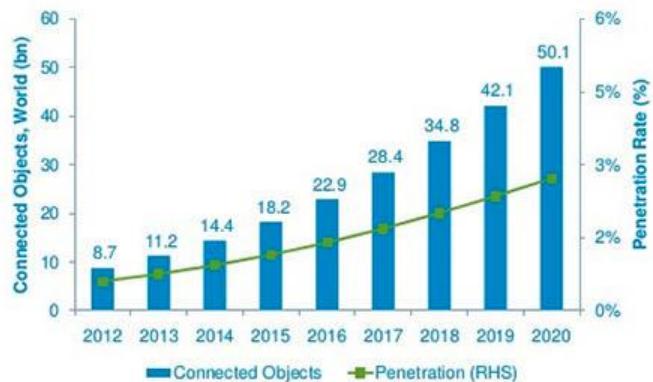
Data Volumes

BIG IN GROWTH, TOO.

1 exabyte (EB) = 1,000,000,000,000,000 bytes



Number of Connected Objects Expected to Reach 50bn by 2020



2025???

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

Question

- How are usage statistics derived in a scalable system?
- Information must be gathered about every single request
 - Session length
 - Session origin
 - Session behavior
- In pornhub.com's case, billions of visits a year
- Lot of data
- A big data system?



The future is about
Big Data
Big Problems
@Web Scale

Web scale



Globally accessible



(Practically) infinite user base



Constant usage (24x7)



Ability to handle surges and spikes
in requests



Controllable costs

Scalability

- The ability of a system to increase or decrease capacity in response to changing demands, while continuing to satisfy service level agreements for latency and availability
- Hyperscalability – as above, while supporting exponential growth with linear costs

Just need more machines and disks?



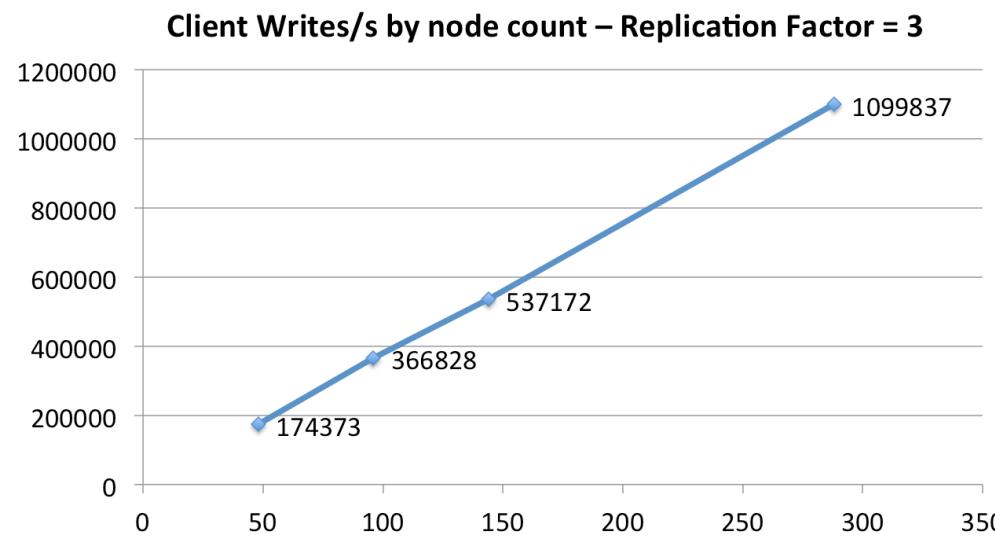
2025???



**Major Internet
Companies
have 1m+ servers in
2013**

And Scale Linearly

Scale-Up Linearity

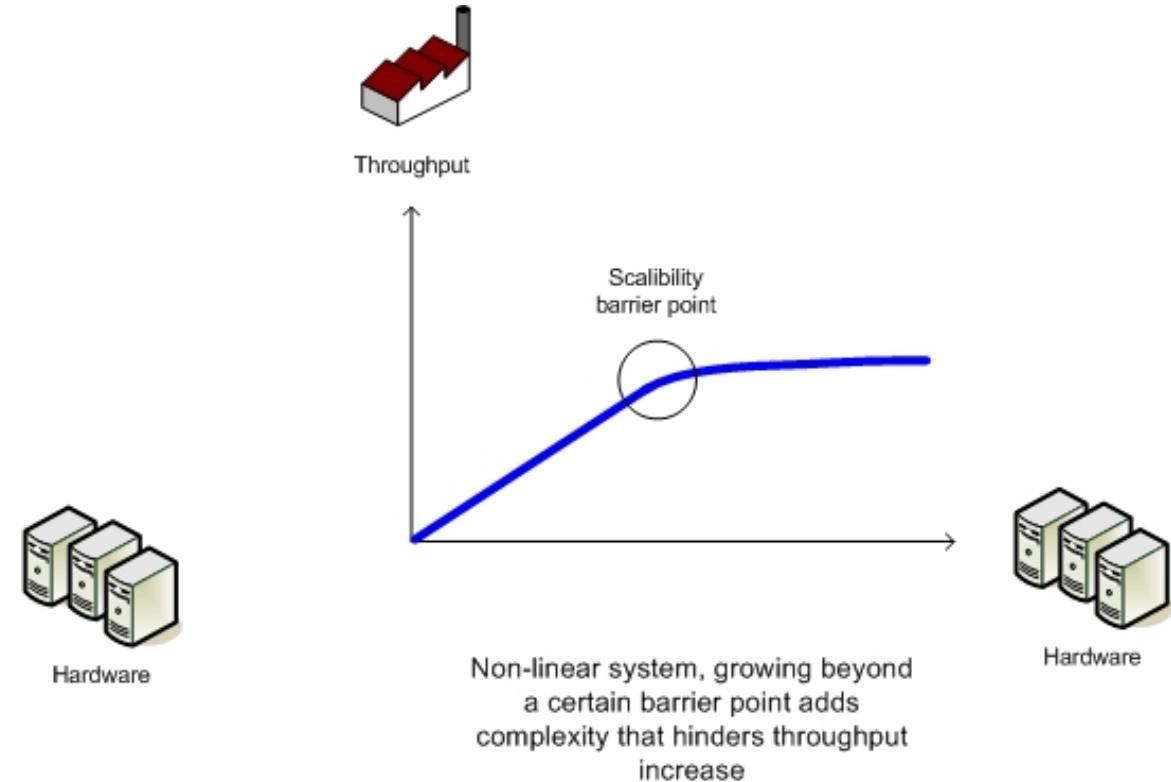
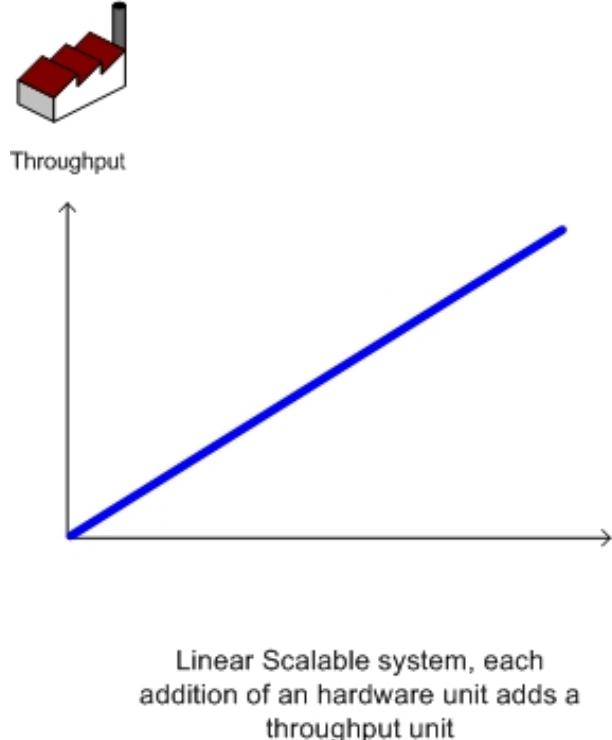


NETFLIX

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

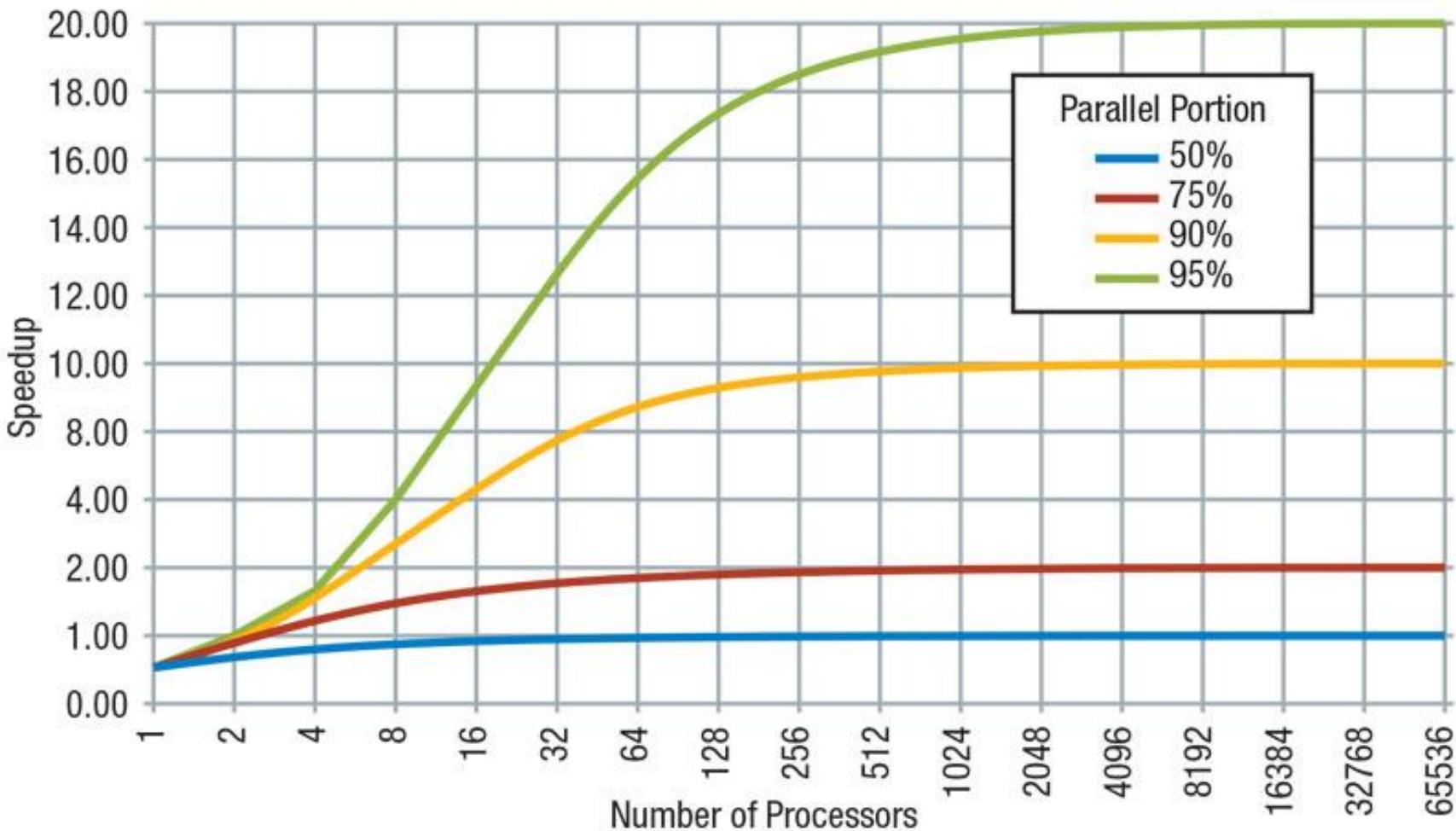


Anyone
heard of
Amhdahl's
law?



Amhdahl's Law

Amdahl's Law





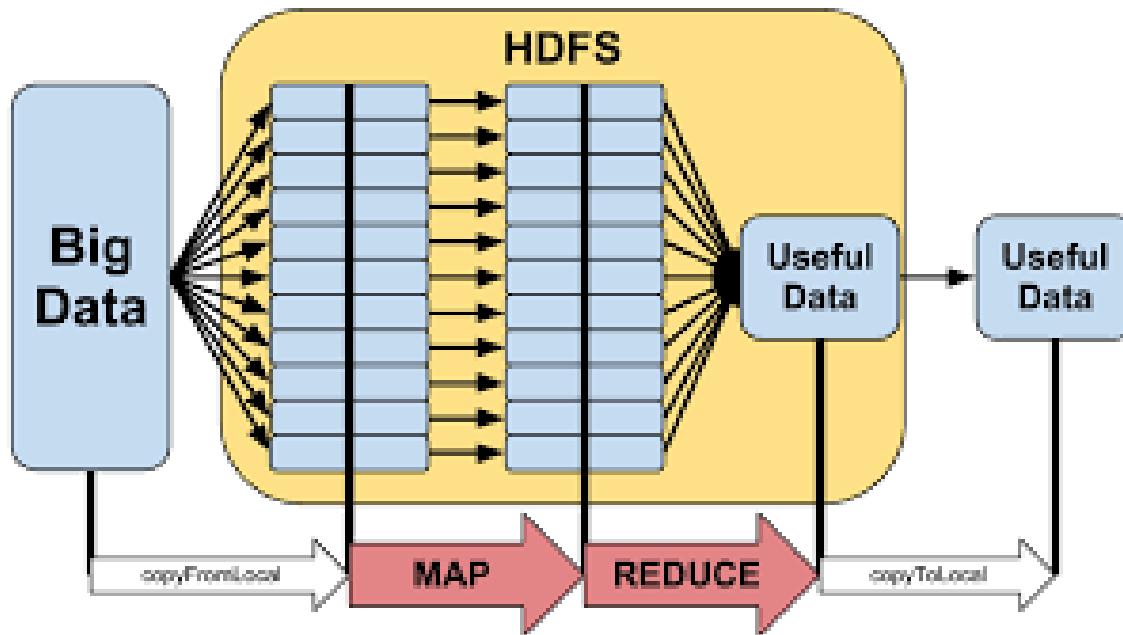
A large blue circle is positioned in the upper left. A smaller yellow circle is located below and to the right of it. A very small gray dot is positioned between the yellow circle and a large dark gray shape on the right side.

Amhdahl's Law in 2025???

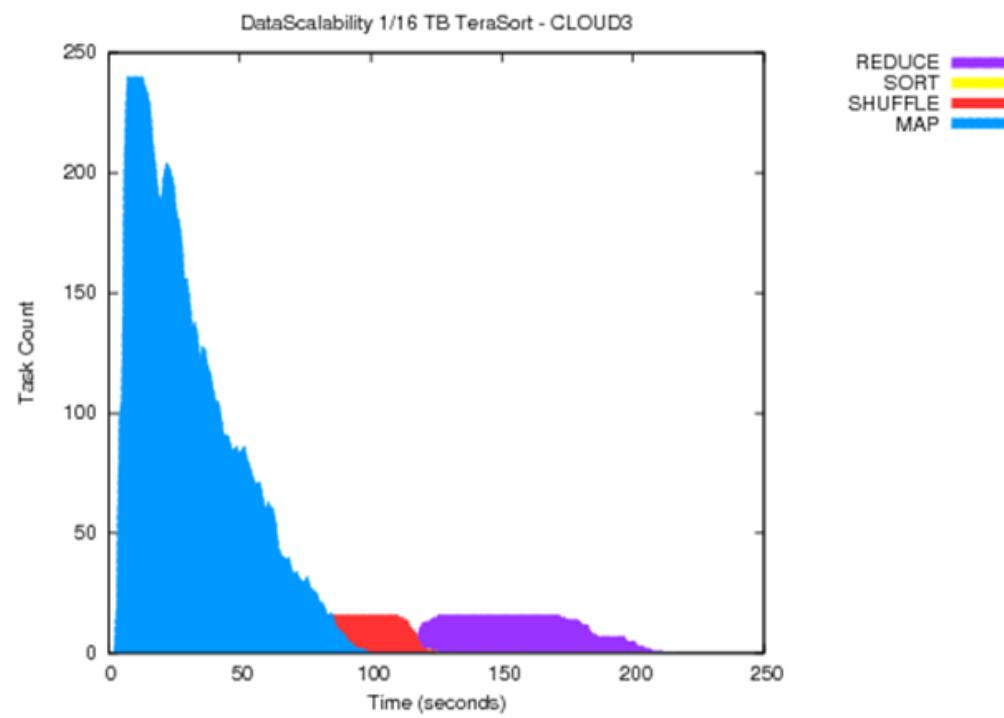


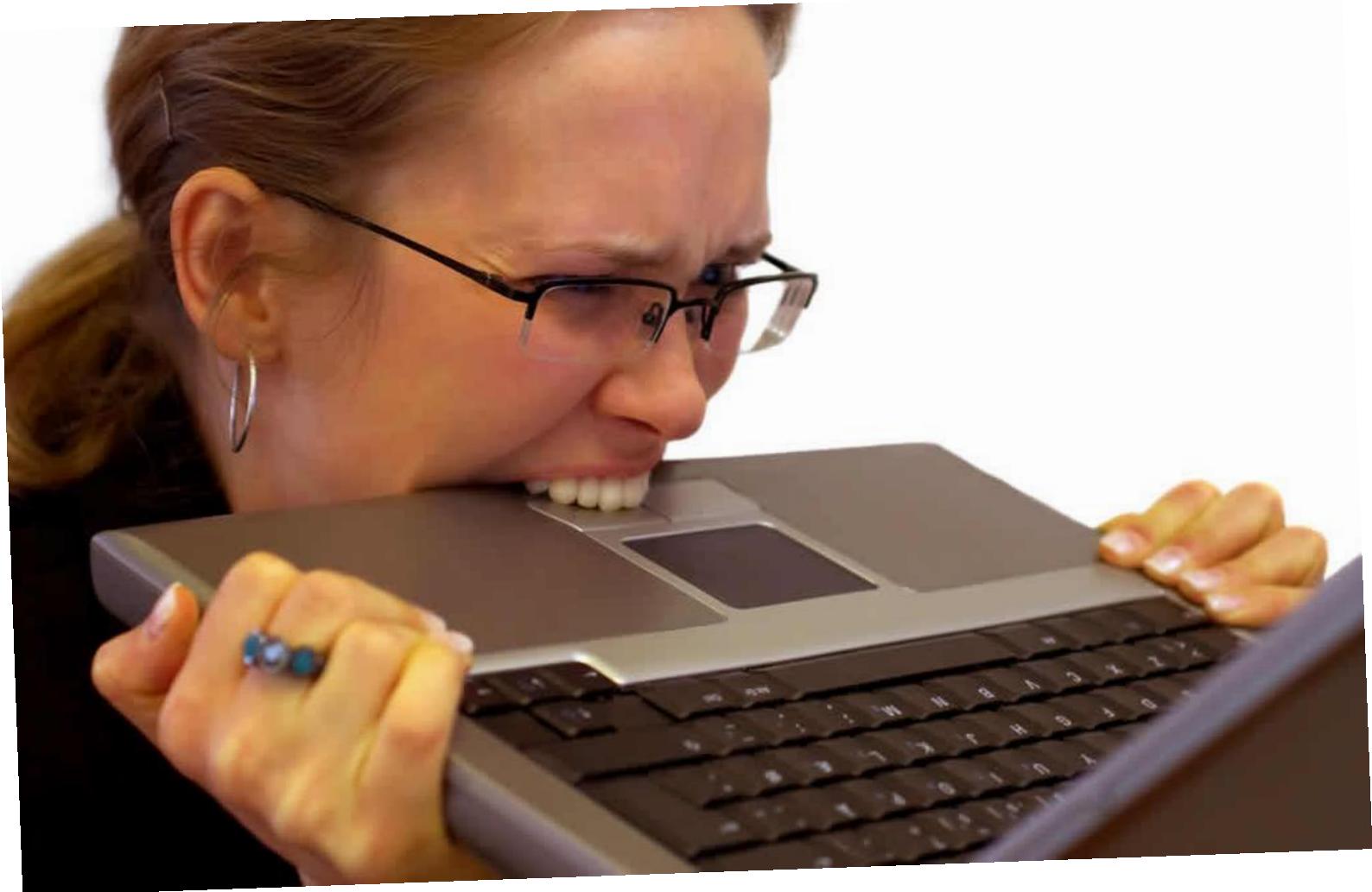
**NO
CHANGE**

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



Engineering Challenges

- High traffic:
 - Consider ad exchanges that are responding to 1,000,000 requests/second
- Huge volumes of data:
 - Often terabytes or petabytes of data
- Strict requirements for systemic properties:
 - Response time, throughput, security, availability, compute cost, ...

Response time



What Is Latency

- Also known as latency
- Round trip time from when a request sent until a response is received

Response Time



Response time has an impact on user behavior

Many studies show a decrease in sales and customer retention as the response time increases

<http://www.martinlugton.com/page-load-speed-important-impact-site-speed-conversions-revenue/>

“Google: increasing page load time from 0.4 seconds to 0.9 seconds decreased traffic and ad revenues by 20%”



In some cases it's required by contract

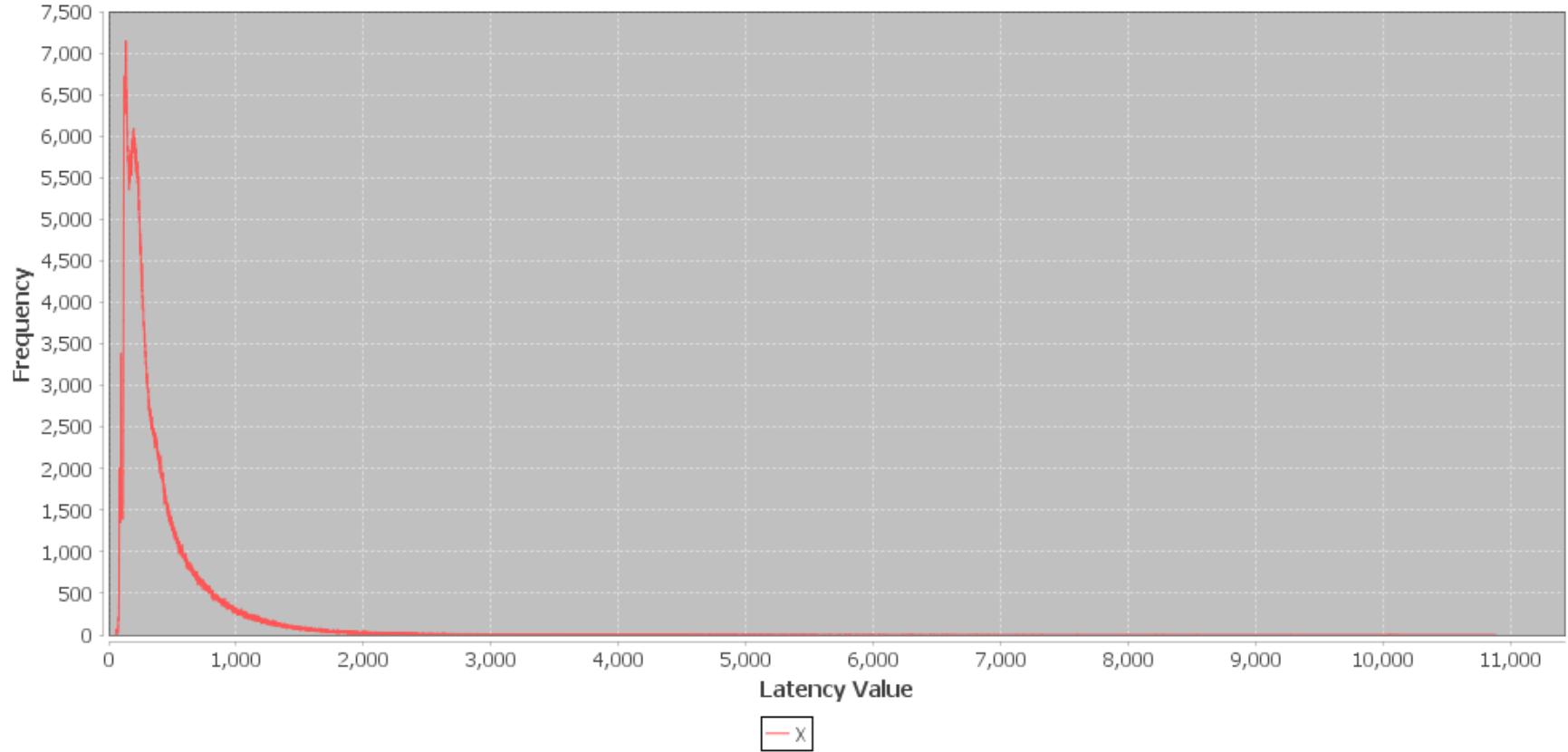
Responses to ad-exchanges for example need to be within 40 milliseconds



Website response time impacts google rankings

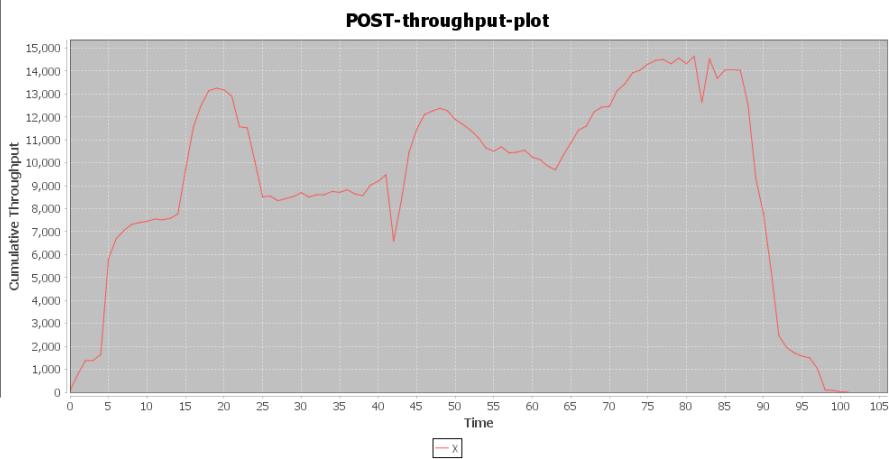
One factor google uses when determining ad listings

POST-latency-plot



Typical Response Time – Long Tail

Throughput



- The capacity of a system to process requests
- Typically measured as number of requests/second
- Cf latency:
 - Latency is a measure for a single request
 - Throughput is a bulk measure of many requests

Throughput



Massive throughput is needed at scale

Millions of requests per second



Before web scale systems, we just used bigger more powerful systems

Particularly for the database server



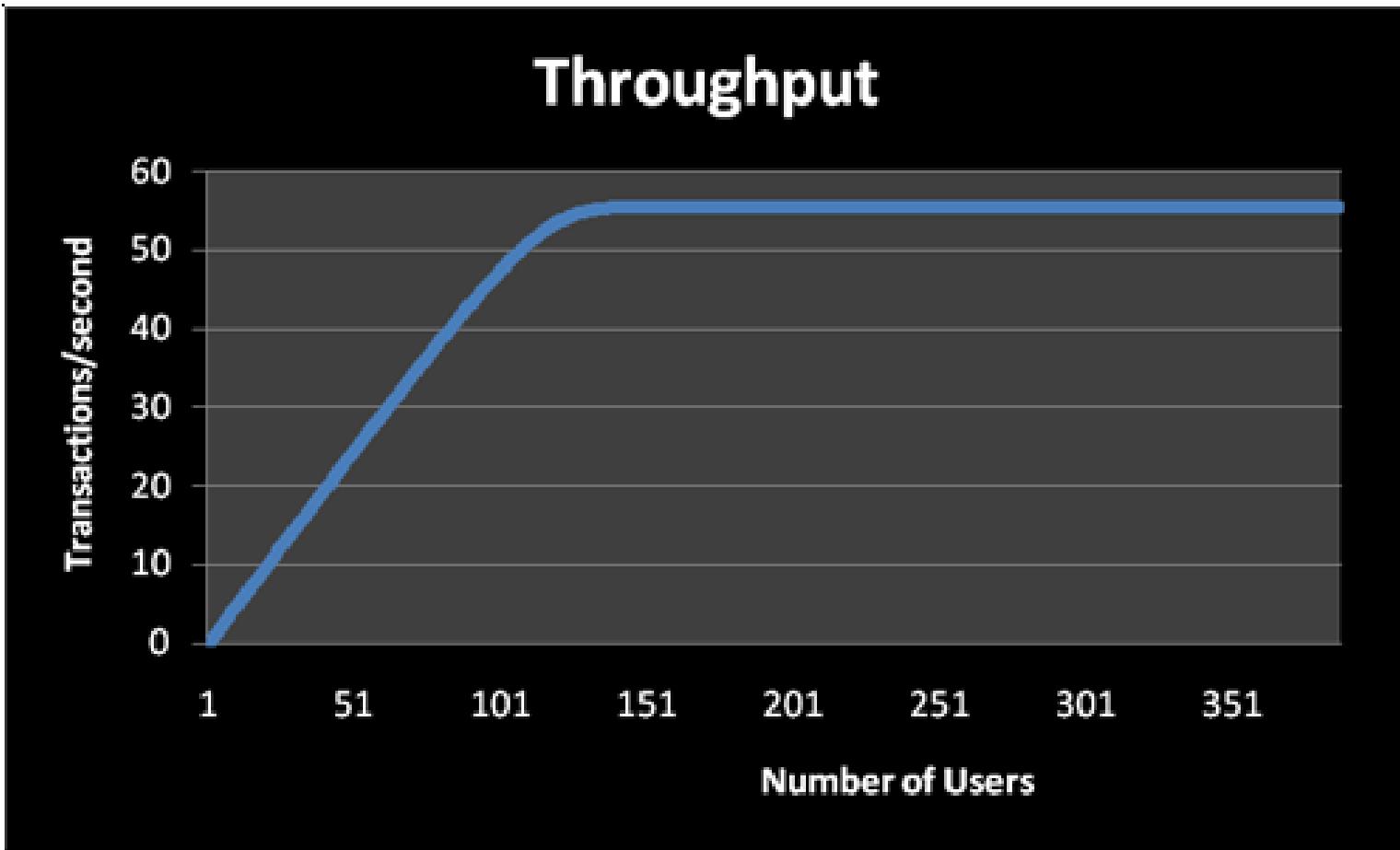
That's no longer an option at web scale



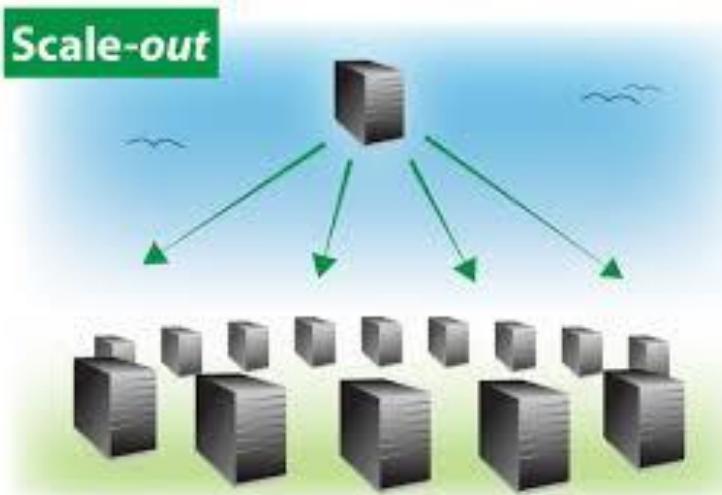
We need to be able to efficiently scale out

In other words be able to split the load across multiple nodes

Throughput with fixed capacity



Scale Out



- **Scale** horizontally (or **scale out**) means adding more nodes to a system
- Increases capacity and hence throughput
- 100s to 1000s of nodes needed at scale
- Scale in/down equally important
 - Why???

Availability



**The percentage of time a system
is accessible to user**



At web scale

Global customer base
High availability required



How high?

99%

99.999%

99.999999%

Availability

- If a system has 1 component
 - It is not available if the element (software/hardware) fails
 - Single Point of Failure (SPoF)
 - So need very highly reliable component
- To achieve scalability, we need to scale out
 - 100s and 1000s of components
 - Do they all need to be always available?
- At scale *everything* breaks ...
- Systems need to be able to deal with regular component failures
 - In other words, *fault tolerant*
 - ***Built to handle problems as failure is expected***

Question

- If my system runs on 10,000 nodes (machines), each with its own disk, how many disk crashes might my system experience every year?
- What information do we need to estimate this?

Annualize Failure Rate

- $\text{AFR (\%)} = (8760/\text{MTBF}) * 100$
- a common drive may have a 300,000 hours MTBF,
- gives a theoretical 2.92% annualized failure rate
 - i.e. a 2.92% chance that a given drive will fail during a year of use.
- A CMU 2007 study showed an estimated 3% mean AFR over 1–5 years
 - <https://www.usenix.org/legacy/events/fast07/tech/schroeder.html>

Managing Tradeoffs

- There is not one way to build such systems
- Requirements and context are going to differ from one system to another
- Optimizing on one system property is inevitably going to compromise another concern, eg
 - Throughput versus cost
 - Performance versus availability
- As an engineer you need to be able to understand:
 - the concerns and context of a given system
 - the nature of the tradeoff that results from a set of decisions
 - And use metrics to guide your designs

Course Outline

Course Overview

- Web site: <https://gortonator.github.io/bsds-6650/>
- In this course you'll learn both theory and practical knowledge
 - To be able to engineer robust, scalable, efficient internet scale systems
- The theory will be learned by:
 - Reading relevant materials/seminar
 - Lectures
 - Quizzes
- The practical knowledge will be gained by:
 - In class exercises
 - Projects

Course Outline

Course Content

Modules

Week	Topic	Date
1	Introduction to Distributed Systems	9/10
2	Concurrency	9/17
3	Distributed Systems Fundamentals	9/24
4	Data Layer: Replication, Partitioning and Consistency	10/1
5	Data Layer: NoSQL Databases	10/8
6	Scalable Request Processing: APIs and State Management	10/15
7	Caching and Load Balancing	10/22
8	Asynchronous Systems	10/29
9	Microservices	11/5
10	Data Processing Architectures - Lambda/etc	11/12
11	Scalable Data Analytics	11/19
12	Stream Processing	11/26
13	Security	12/3
14	Final Project Presentations	12/10

Grading

- There will be three components to the grades
 - 4 programming assessments (72%)
 - 4 in class Quizzes (24%)
 - Short paper (4%)

Final Stuff

- All course materials will be distributed through web site/blackboard
- We have a Piazza site for questions/discussions
- We won't be taking attendance, but you are expected to attend and actively participate in class
 - Discussion is welcome

Assignments overview

Project 1 – Using AWS

Client



Server



Multithreading
Measurement
Presentation

Project 2

Client



Server



Server thread models
State management
Data storage schemas



Project 3

Client



Server



Availability
Scalability
Caching
Monitoring

A dark, horned, winged creature with glowing yellow eyes and sharp claws, standing on a white surface.

Project 4

How to succeed in this course

- Do the reading
- Study for the quizzes
- Start the assignments early
 - Really #notkidding
- Work consistently
- Experiment
- Be inquisitive and have fun
- Expect to work hard!!!

