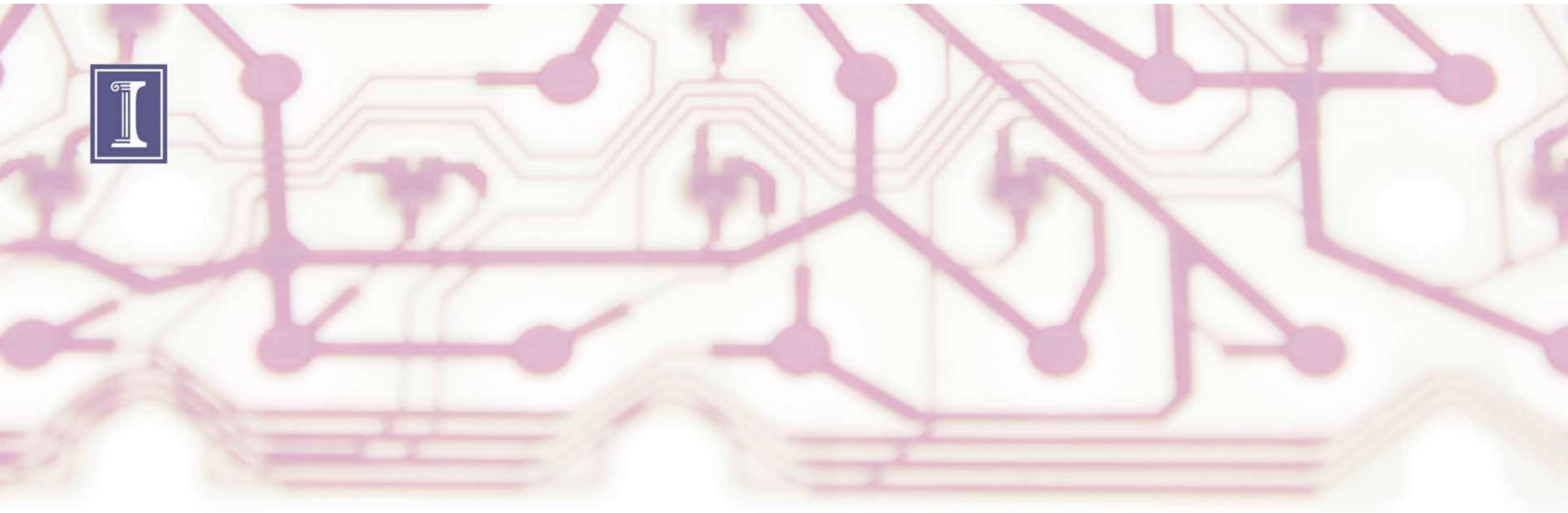


CLOUD COMPUTING APPLICATIONS

CLOUD COMPUTING INTRODUCTION

Roy Campbell & Reza Farivar

Tremendous Buzz

"Not only is it faster and more flexible, it is cheaper. [...] the emergence of cloud models radically alters the cost-benefit decision"

(FT)

"Cloud computing achieves a quicker return on investment"

(Lindsay Armstrong of salesforce.com)

"In an economic downturn, the appeal of that cost advantage will be greatly magnified"

(IDC)

"Revolution, the biggest upheaval since the invention of the PC in the 1970s [...] IT departments will have little left to do once the bulk of business computing shifts [...] into the cloud"

(Nicholas Carr)

"No less influential than e-business"

(Gartner)

"The economics are compelling, with business applications made three to five times cheaper and consumer applications five to 10 times cheaper"

(Merrill Lynch)

"Domestic cloud computing estimated to grow at 53%"

(moneycontrol.com)

Perils of Corporate Computing

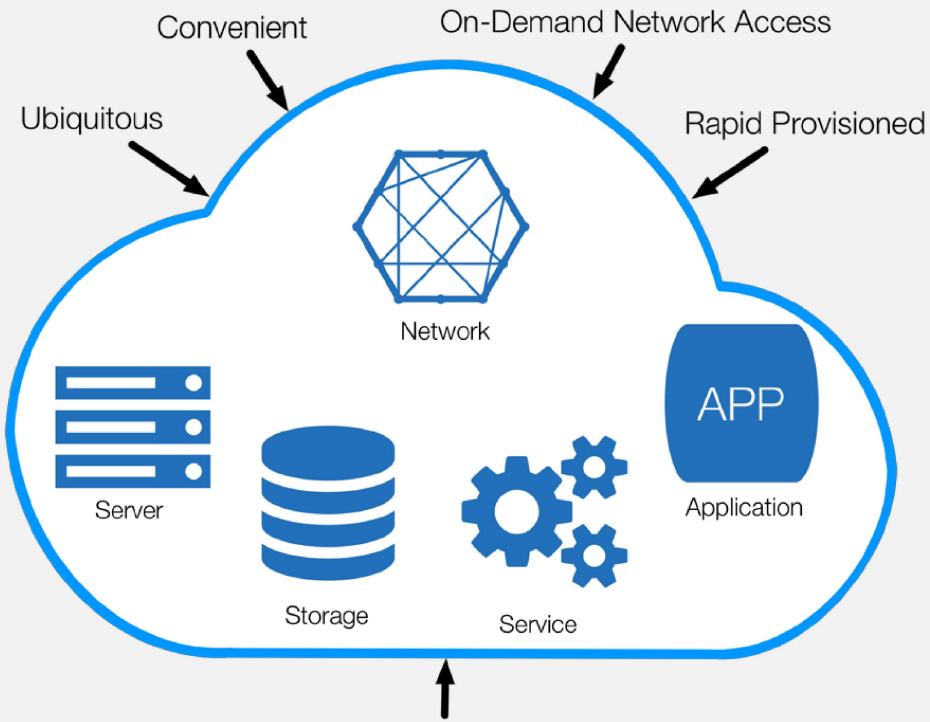
- Own information systems ☺
- However
  - Capital investment ☹
  - Heavy fixed costs ☹
  - Redundant expenditures ☹
  - High energy cost, low CPU utilization ☹
  - Dealing with unreliable hardware ☹
  - High levels of overcapacity (technology and labor) ☹
- NOT SUSTAINABLE

Back to the Future

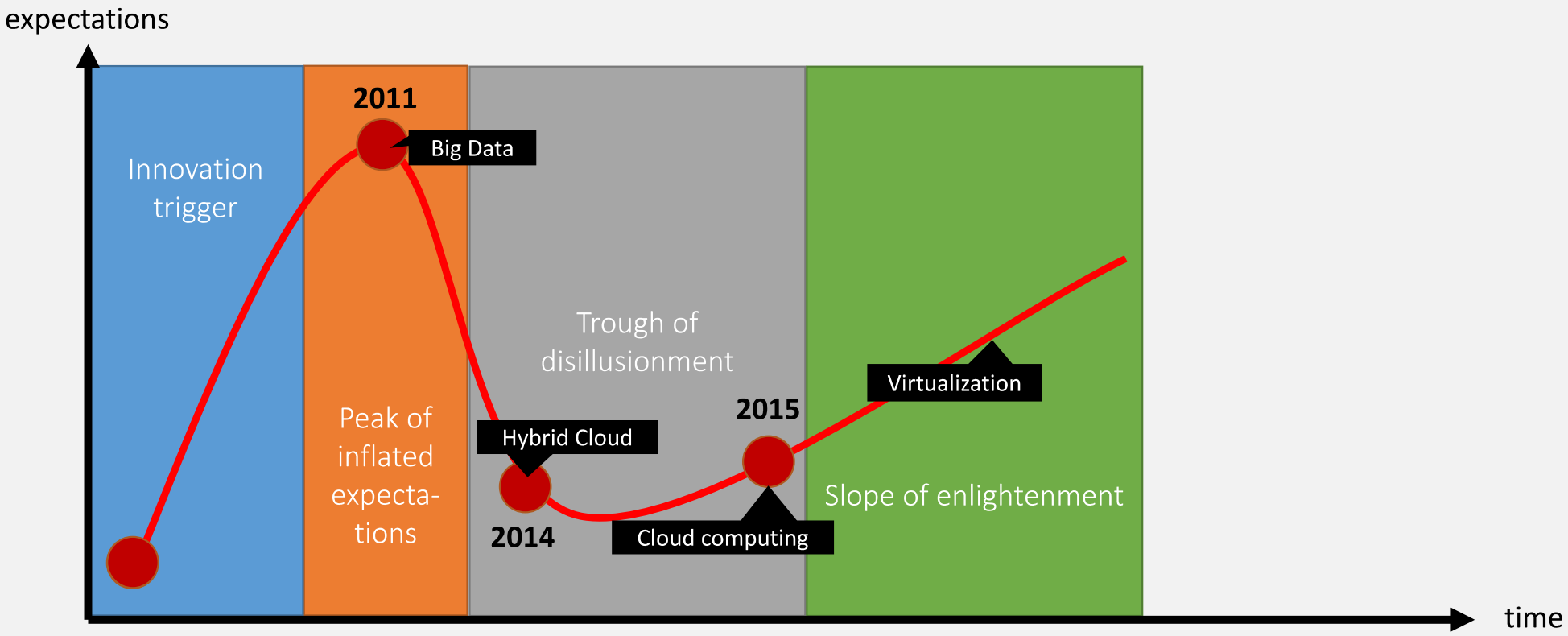
“Computing may someday be organized as a public utility, just as the telephone system is organized as a public utility”

(John McCarthy, 1961)

Cloud Computing



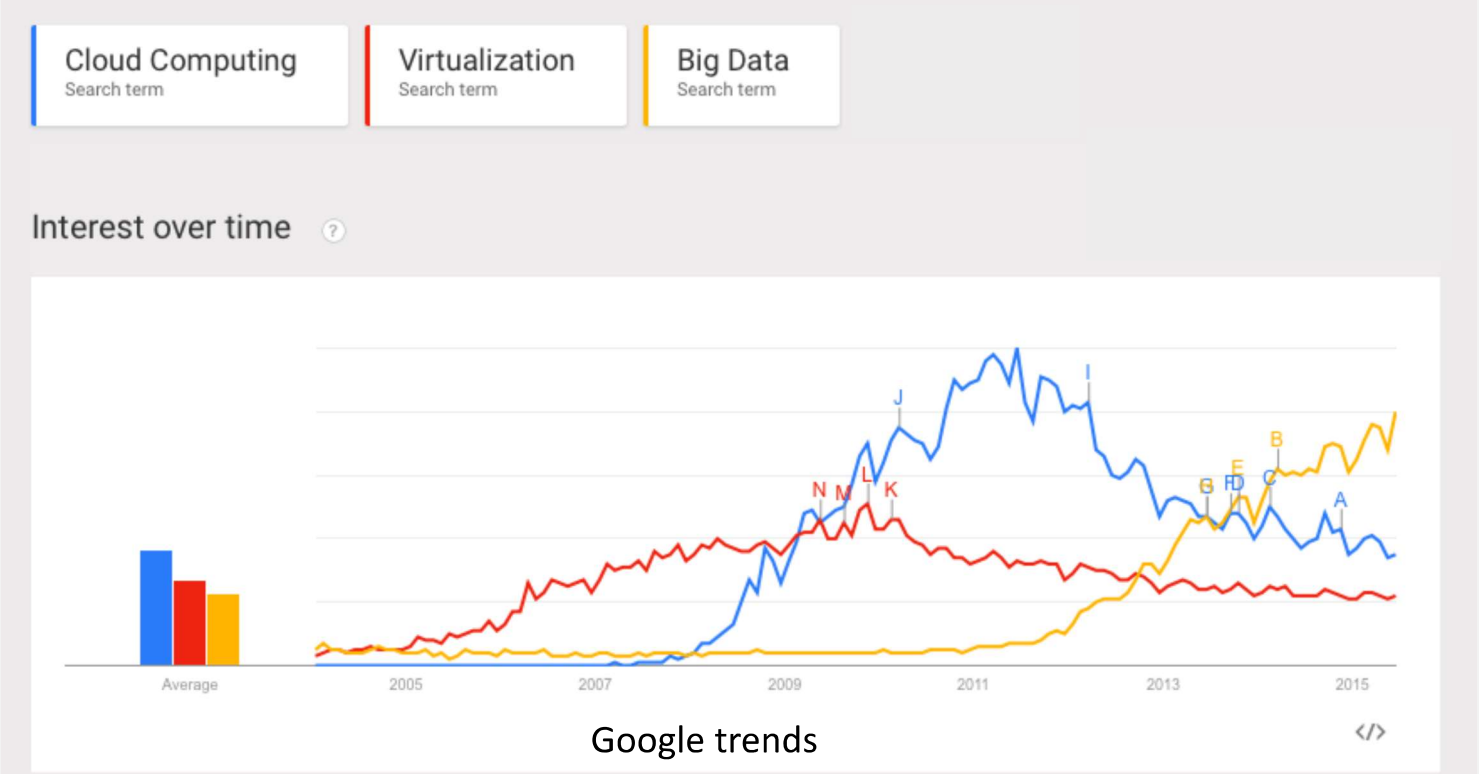
Cloud Adoption: Gartner's Hype Cycle



Delivery Models

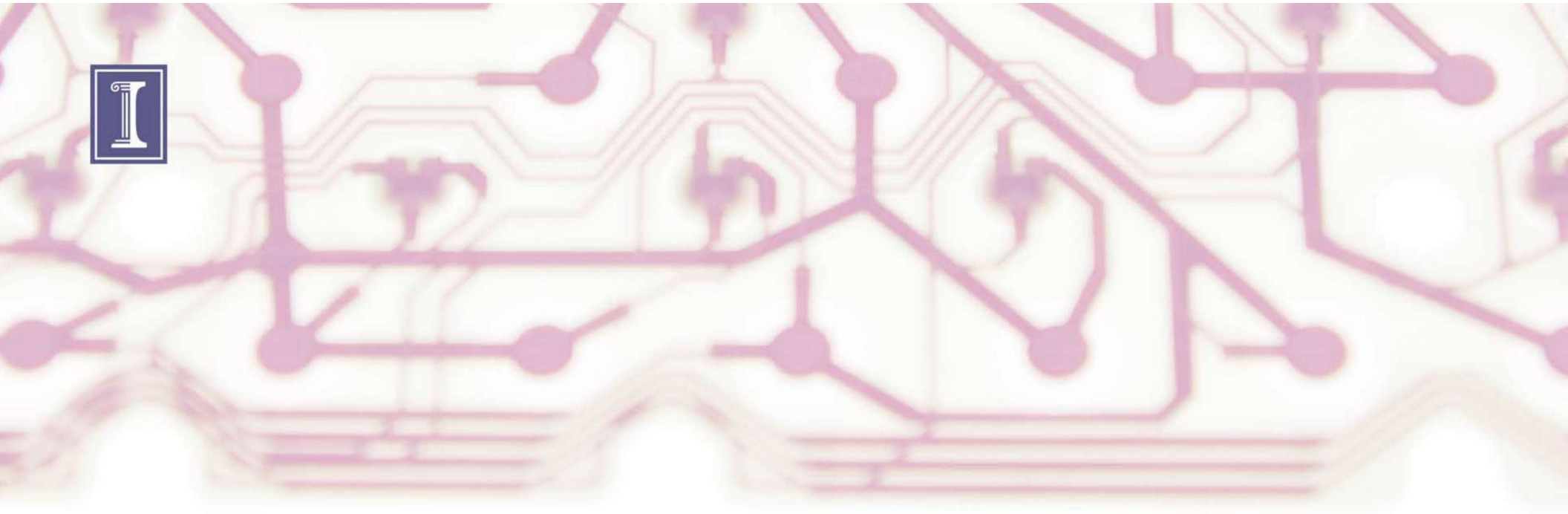
- Software as a Service (SaaS)
  - Use provider's applications over a network
  - SalesForce.com
- Platform as a Service (PaaS)
  - Deploy customer-created applications to a cloud
  - AppEng
- Infrastructure as a Service (IaaS)
  - Rent processing, storage, network
  - Capacity and other fundamental computing resources
  - EC2, S3

Synergy: Cloud Computing, Virtualization, and Big Data



Big Data Data Revolution and Clouds

- Data collection too large to transmit economically over Internet -- Petabyte data collections
- Computation is data intensive
  - Lots of disks, networks and CPUs
  - Overhead of maintaining cyber infrastructure is expensive
  - Users buy Big Data services from Clouds to share overhead
- Easy-to-write programs, fast turnaround
- MapReduce – Hadoop, PIG, HDFS, HBase



CLOUD COMPUTING APPLICATIONS

CLOUDONOMICS: PART 1

Roy Campbell & Reza Farivar

Cloudonomics: Part 1

Economics necessitates Cloud computing:

- Part 1: Utility Pricing
- Part 2: Benefits Common Infrastructure

See other details and benefits in

"Cloudonomics: A Rigorous Approach to Cloud Benefit Quantification," Joe Weinman

https://www.csiac.org/sites/default/files/journal_files/stn14_4.pdf

Value of Utility Pricing

- Cloud services don’t need to be cheaper to be economic!
- Consider a car
  - Buy or lease for $10 per day
  - Rent a car for $45 a day
  - If you need a car for 2 days in a trip, buying would be much more costly than renting
  - **It depends on the demand**

Utility Pricing in Detail

D(t): demand for resources, 0<t<T

P = max(D(t)): Peak Demand; A = Avg (D(t)): Average Demand

B = Baseline (owned) unit cost; B_T = Total Baseline Cost

C = Cloud unit cost; C_T = Total Cloud Cost

U = C / B : Utility Premium (for the rental car example, U = 4.5)

C_T

(because the Baseline should handle Peak Demand)

When is the Cloud cheaper than owning?

Substituting for C_T, B_T :

which implies

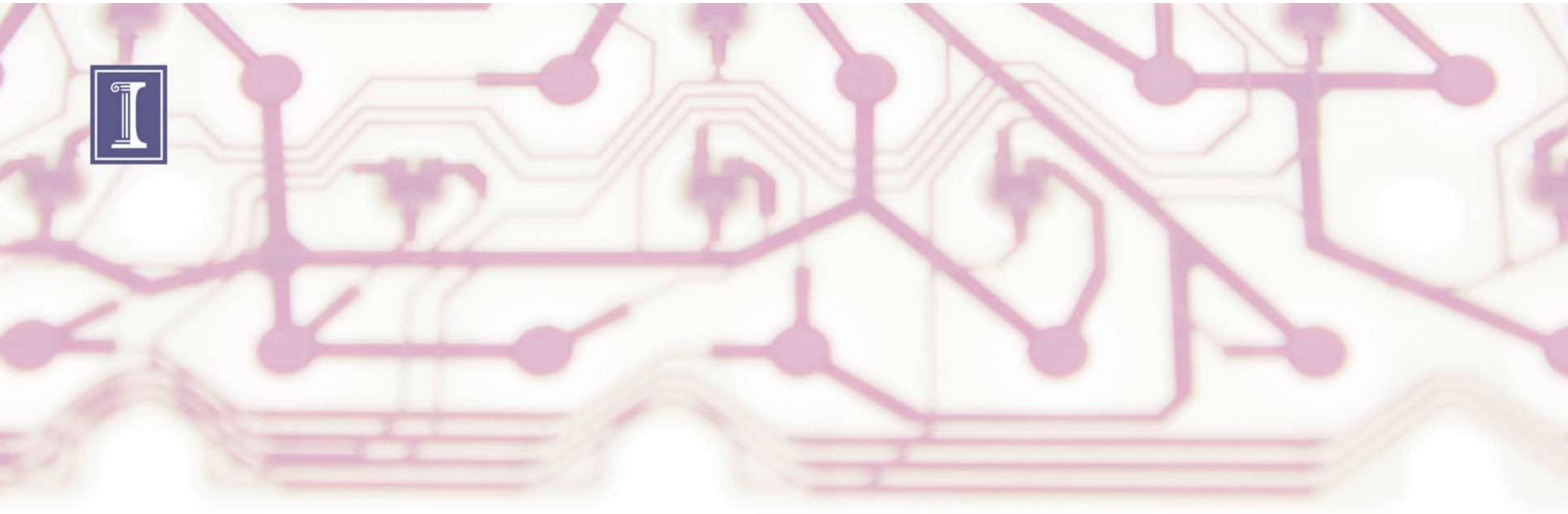
i.e., when Utility Premium is less than ratio of Peak Demand to Average Demand

Utility Pricing in Real World

- In practice, demands are often highly spiky
  - News stories, marketing promotions, product launches, Internet flash floods (Slashdot effect), tax season, Christmas shopping, etc.
- Often a hybrid model is the best
  - You own a car for daily commute, and rent a car when traveling or when you need a van to move
  - Key factor is again the ratio of Peak Demand to Average Demand
  - But we should also consider other costs
    - Network cost (both fixed costs and usage costs)
    - Interoperability overhead
    - Consider reliability, accessibility

Summary

- Utility Pricing is good when demand varies over time, as is the case of a start-up or a seasonal business
- When Utility Premium is less than ratio of Peak Demand to Average Demand, Cloud computing is beneficial
- Next, we look at the possible savings that Cloud providers can create using statistical multiplexing



CLOUD COMPUTING APPLICATIONS

CLOUDONOMICS: PART 2

Roy Campbell & Reza Farivar

Cloudonomics: Part 2

Economics necessitates Cloud Computing:

- Part 1: Utility Pricing
- Part 2: Benefits Common Infrastructure

See other details and benefits in

"Cloudonomics: A Rigorous Approach to Cloud Benefit Quantification," Joe Weinman

https://www.csiac.org/sites/default/files/journal_files/stn14_4.pdf

The Value of Common Infrastructure

- For infrastructure built to peak requirements: Multiplexing demand → higher utilization
  - Lower cost per delivered resource than unconsolidated workloads
- For infrastructure built to less than peak: Multiplexing demand → reduce the unserved demand
  - Lower loss of revenue or a Service-Level agreement violation payout

A Useful Measure of “Smoothness”

The coefficient of variation:

$$C_v = \frac{\text{standard deviation } \sigma}{\text{mean } |\mu|}$$

C_v is a measure of smoothness

- small is smooth!
- large mean and/or smaller standard deviation

Implications of “Smoothness”

- A fixed-asset facility servicing highly variable jobs yields low utilization
- Same facility servicing smooth jobs yields high utilization
- **Multiplexing jobs with different distributions may reduce the coefficient of variation C_V**

Case Study of C_V for Independent Jobs

- X_1, X_n ,…, X_n independent jobs with standard variation σ and mean μ
- Aggregated jobs
  - Mean → sum of means: $n.\ \mu$
  - Variance → sum of variances: $n.\sigma^2$
  - Aggregate C_v → $\frac{\sqrt{n}.\sigma}{n.\mu} = \frac{\sigma}{\sqrt{n}.\mu} = \frac{1}{\sqrt{n}} C_v$

Case Study of C_V for Independent Jobs

Adding *n* independent jobs reduces C_v by $1/\sqrt{n}$

- Penalty of insufficient/excess resources grows smaller
- Aggregating 100 workloads brings the penalty to 10%

Case Study of C_V for Correlated Jobs

- Best Case: Negative correlation
  - Optimal packing of customer jobs
  - X and 1-X → Sum is 1, $C_v = 0$
  - Optimally smooth, best CPU utilization

- Worst Case: Positive correlation
  - Mean: n.μ(X), standard deviation: n.σ(X)
  - Aggregate $C_v = C_v(X) = \frac{\sigma(X)}{\mu(X)}$
  - Which isn't smoother!

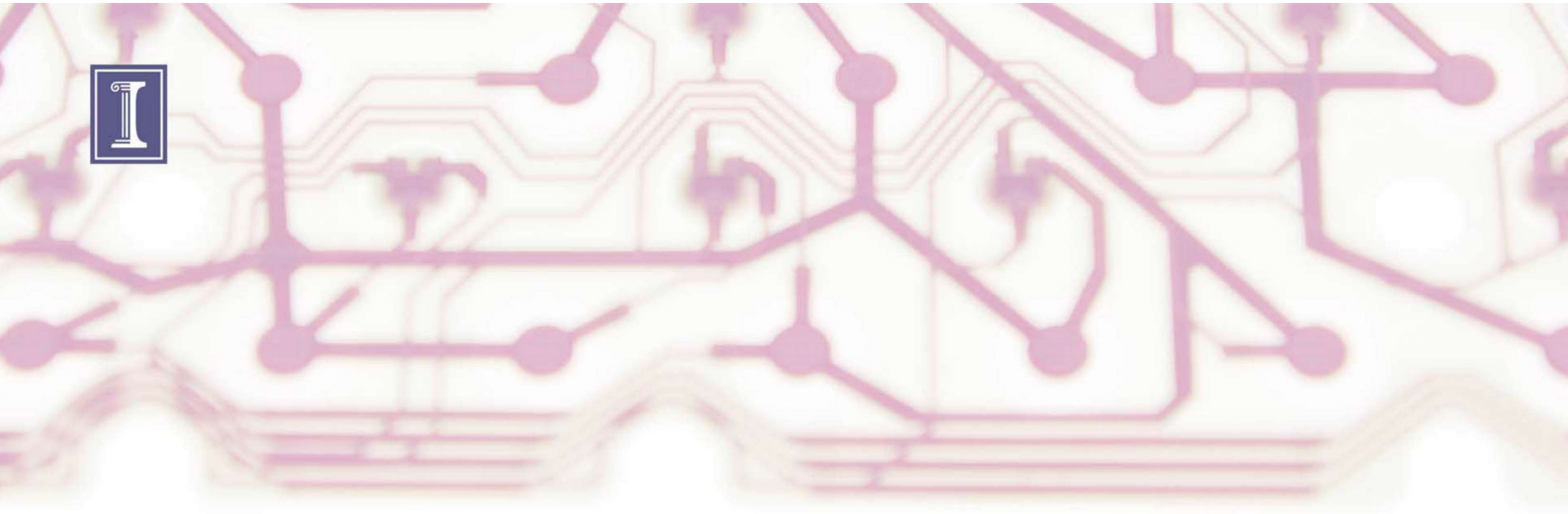
Results from Theory

- Negative-correlated jobs
  - Private, mid-size, and large-size providers can experience similar statistics of scale
- Independent jobs
  - Mid-size providers can achieve similar statistical economies to an infinitely large provider

Common Infrastructure in Real World

- Available data on economy of scale for large providers is mixed
  - Use the same COTS computers and components
  - Locate near cheap power supplies → everyone can do that
  - Early entrant automation tools → 3rd parties take care of it

- Takeaway lesson: you don't need to be as large as Amazon.com to compete! ☺
  - At least according to "Value of Common Infrastructure"



CLOUD COMPUTING APPLICATIONS

BIG DATA

Roy Campbell & Reza Farivar

Big Data (a Singular Phrase)!

- A collection of data sets so large and complex, it's impossible to process it on one computer with the usual databases and tools

- Because of its size and complexity, Big Data is hard to capture, store, copy, delete (privacy), search, share, analyze, and visualize

Big Data

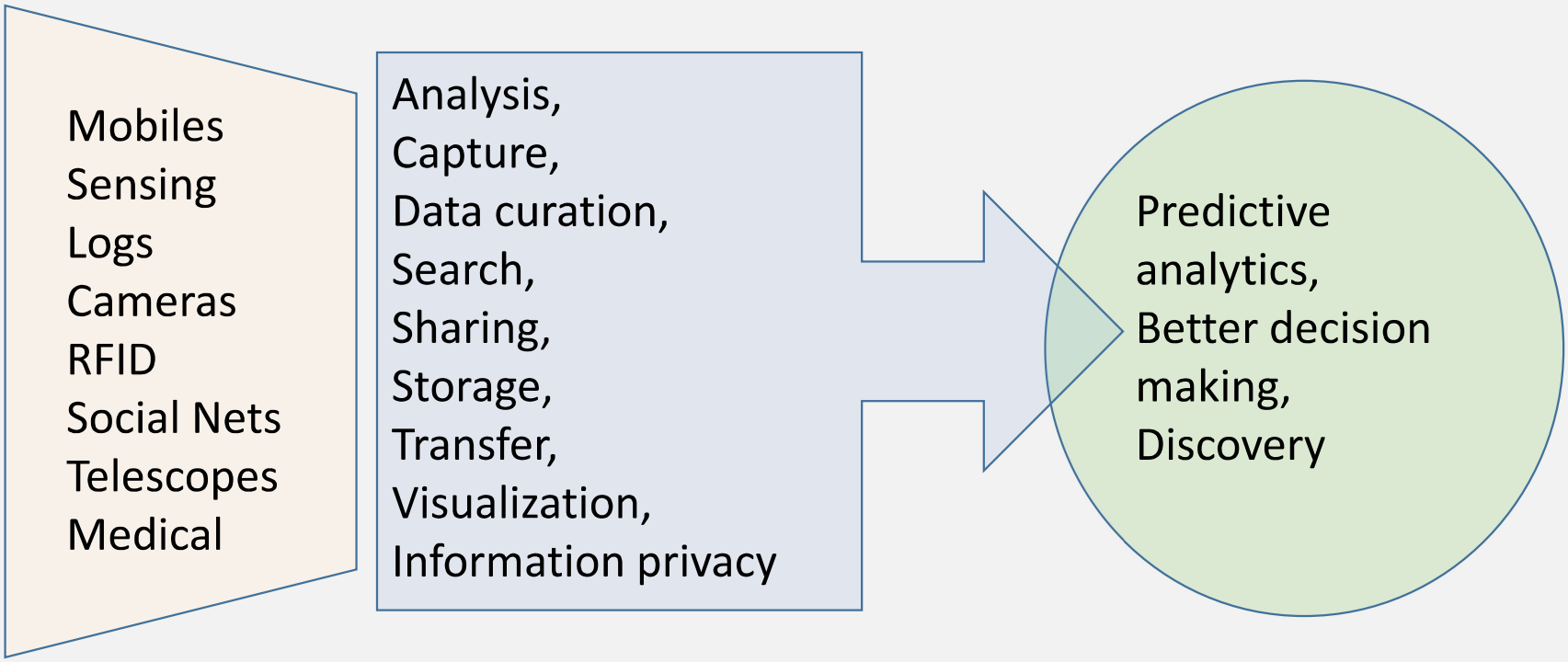
Big Data represents the information assets characterized by such high

- Volume,
- Velocity, and
- Variety

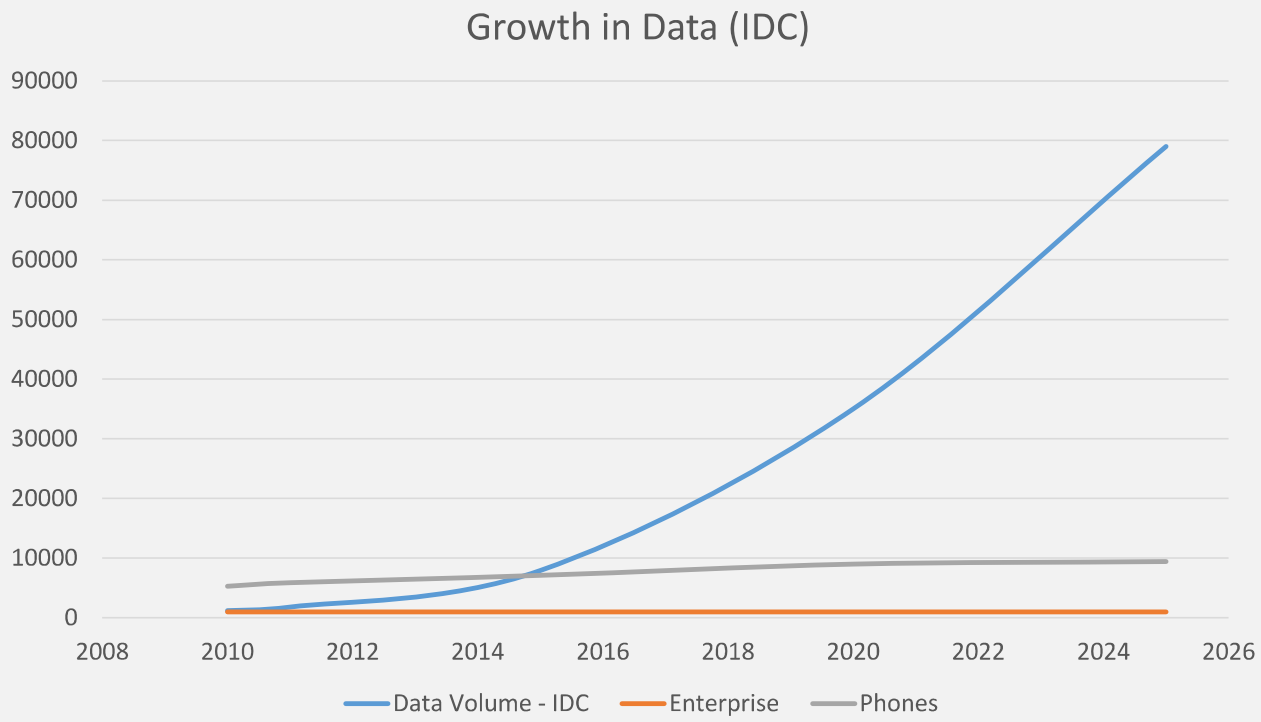
as to require specific technology and analytical methods for its transformation into

- Value

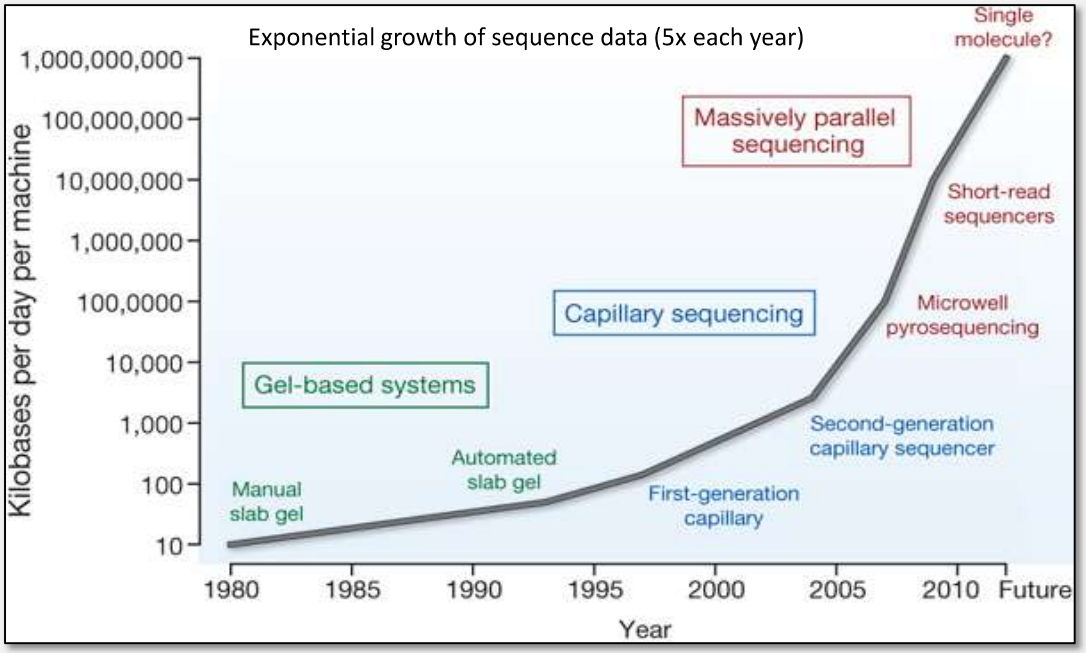
Challenges


Mobiles
Sensing
Logs
Cameras
RFID
Social Nets
Telescopes
Medical
Analysis,
Capture,
Data curation,
Search,
Sharing,
Storage,
Transfer,
Visualization,
Information privacy
Predictive analytics,
Better decision making,
Discovery

Timeline

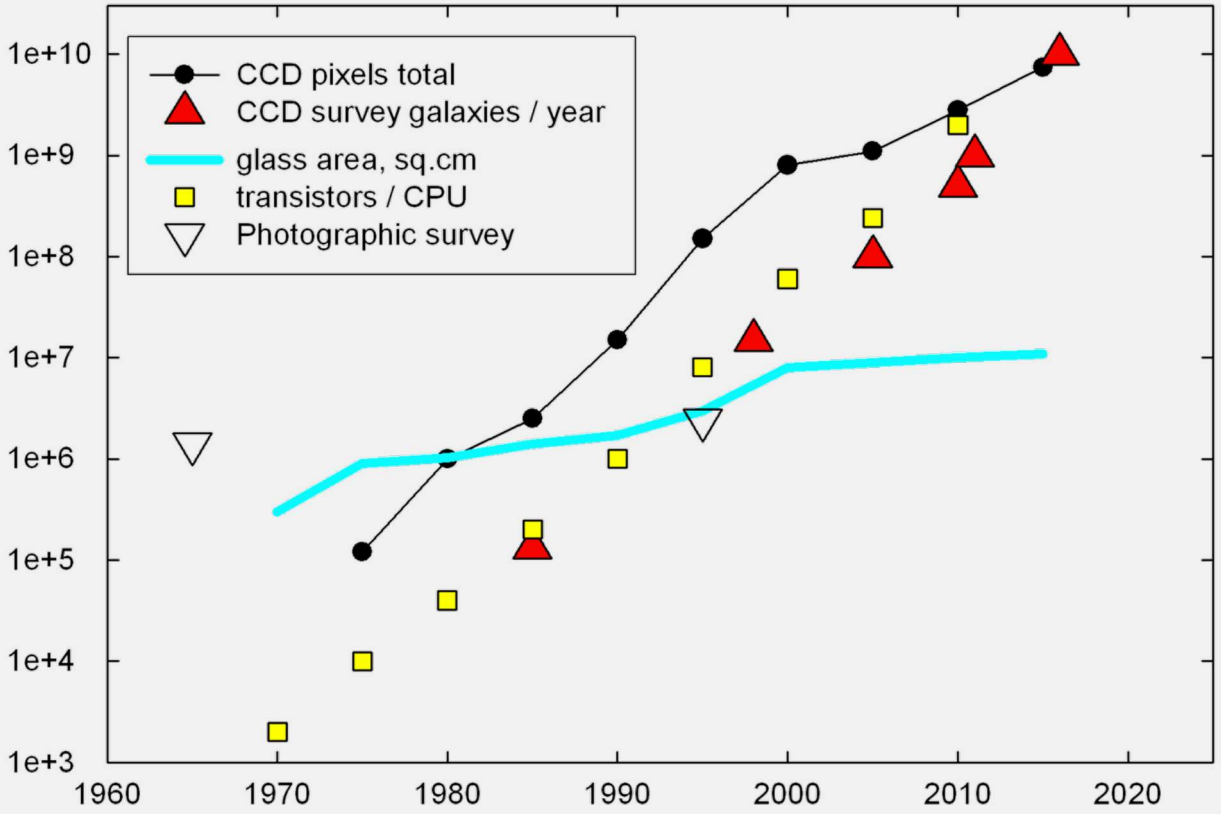

Growth in Data (IDC)
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
2008
2010
2012
2014
2016
2018
2020
2022
2024
2026
Data Volume - IDC
Enterprise
Phones

Example: Bioinformatics

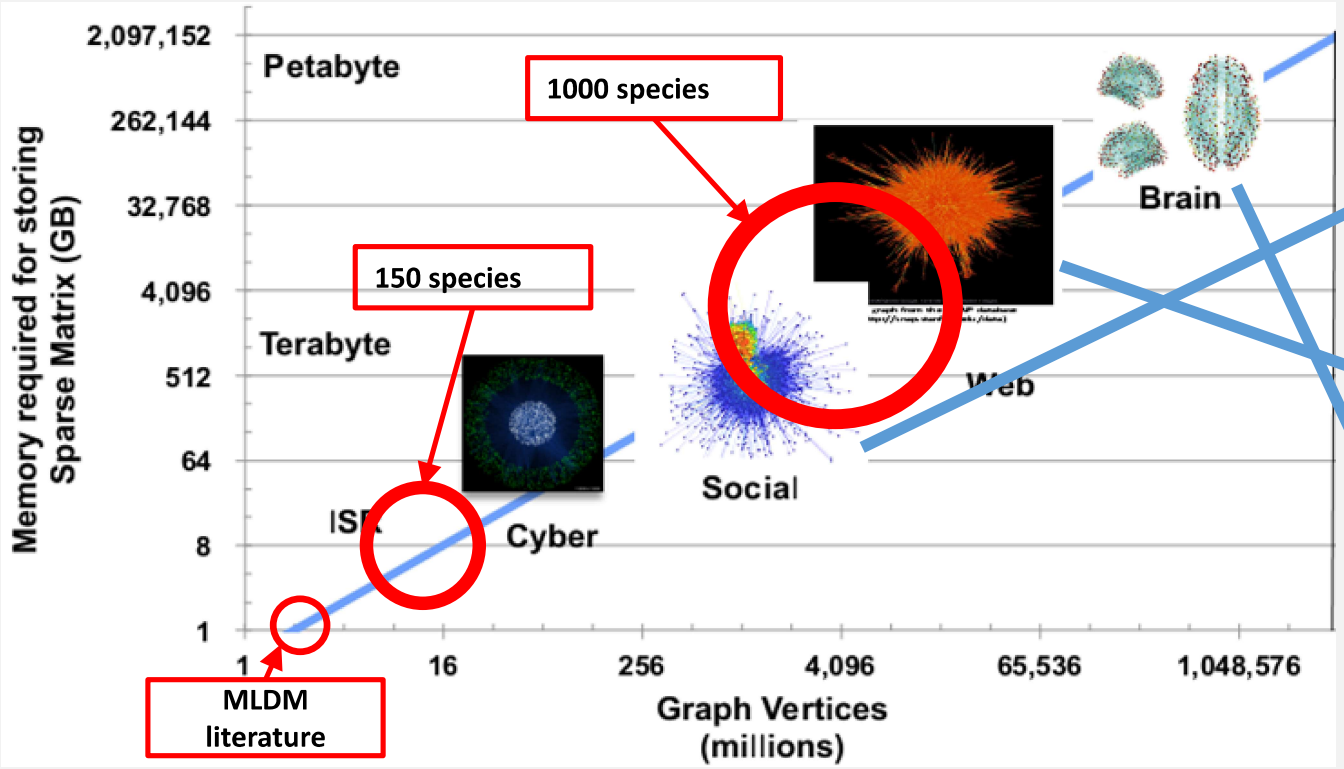


Michael R. Stratton, Peter J. Campbell & P. Andrew Futreal
Nature ***458, 719-724(9 April 2009)***

Example: Astronomy



Example: Graphs



Social scale …
1 billion vertices, 100 billion edges

2.92 TB adjacency list

Web scale …
50 billion vertices, 1 trillion edges

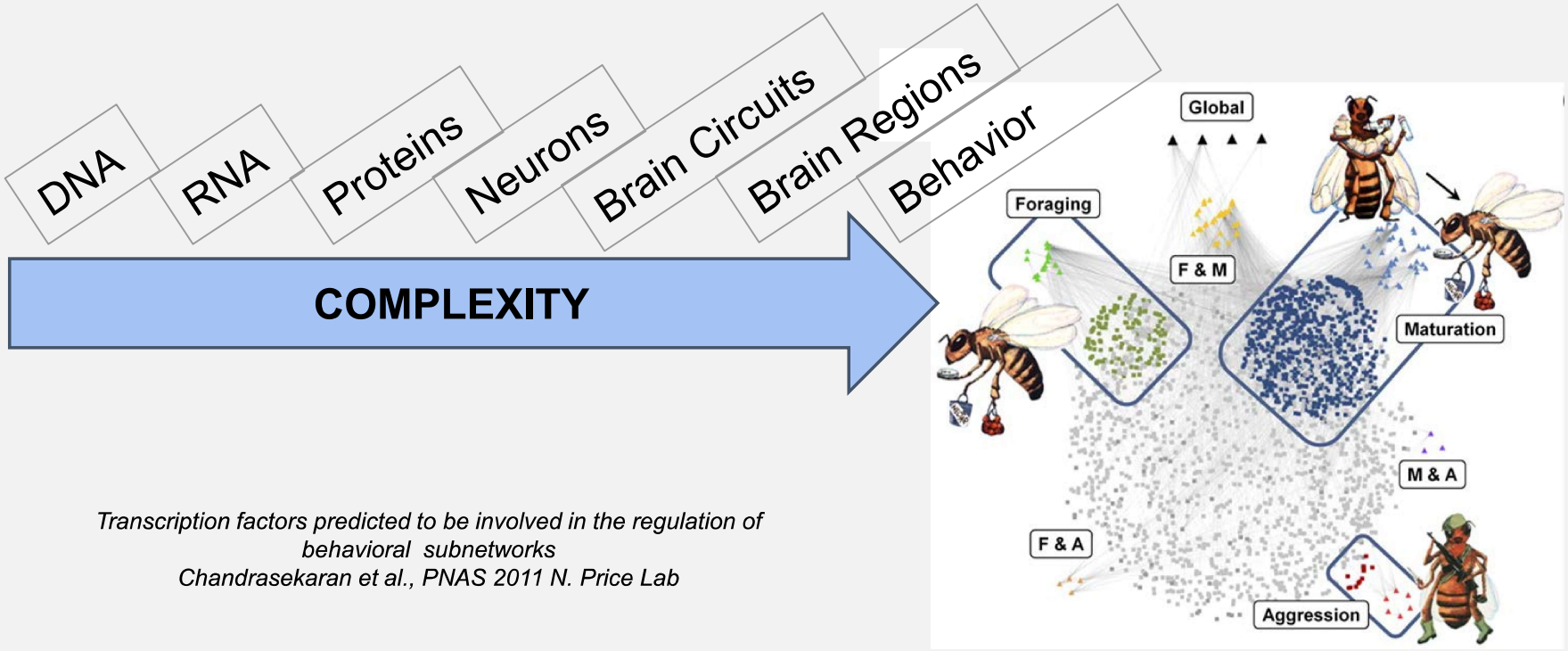
29.5 TB adjacency list

Brain scale …
100 billion vertices, 100 trillion edges

2.84 PB adjacency list

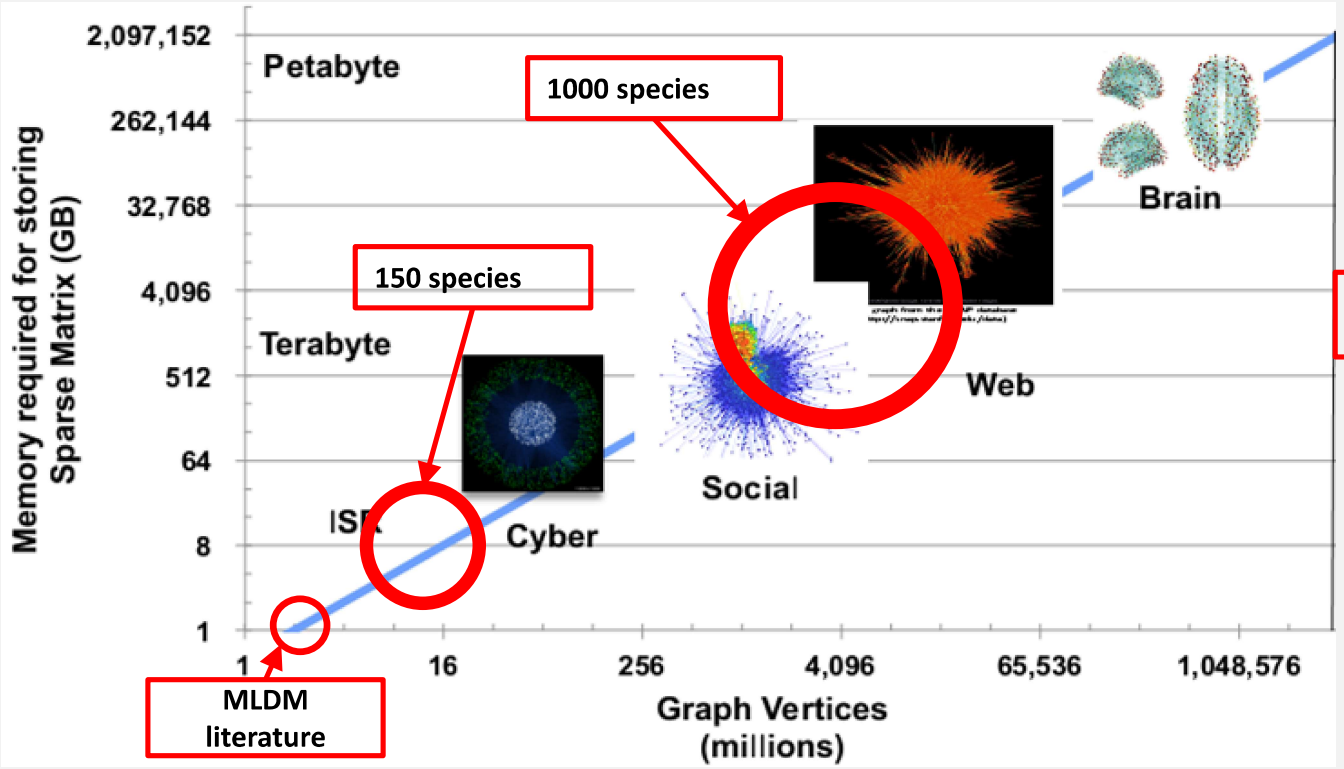
Future: Scale of Real-World Graphs

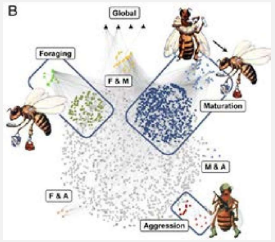
From genes to brains and social behavior…



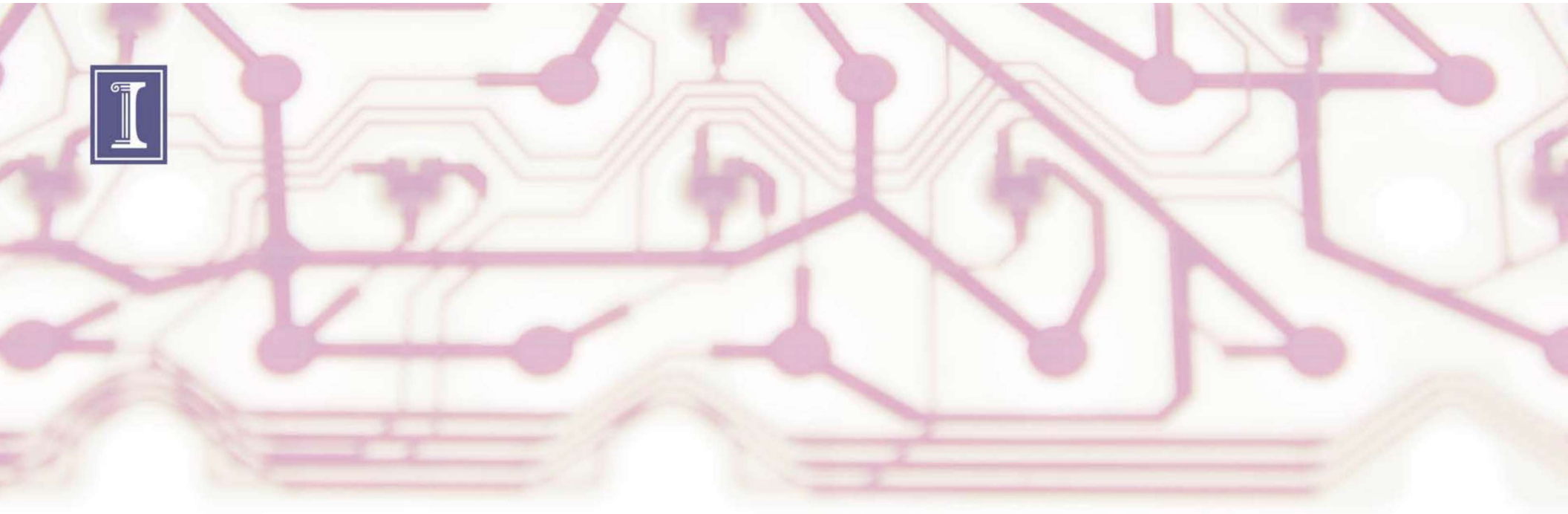
Transcription factors predicted to be involved in the regulation of behavioral subnetworks
Chandrasekaran et al., PNAS 2011 N. Price Lab

Scale of Real-World Graphs: Today





Genomic and brain to social behavior



CLOUD COMPUTING APPLICATIONS

SUMMARY OF CLOUD INTRODUCTION

Roy Campbell & Reza Farivar

Takeaways

- Multiple reasons for Cloud adoption
- Clouds allow economies
- Sharing for the Big Data revolution?
- Another round in innovation: everyone benefits from Clouds?

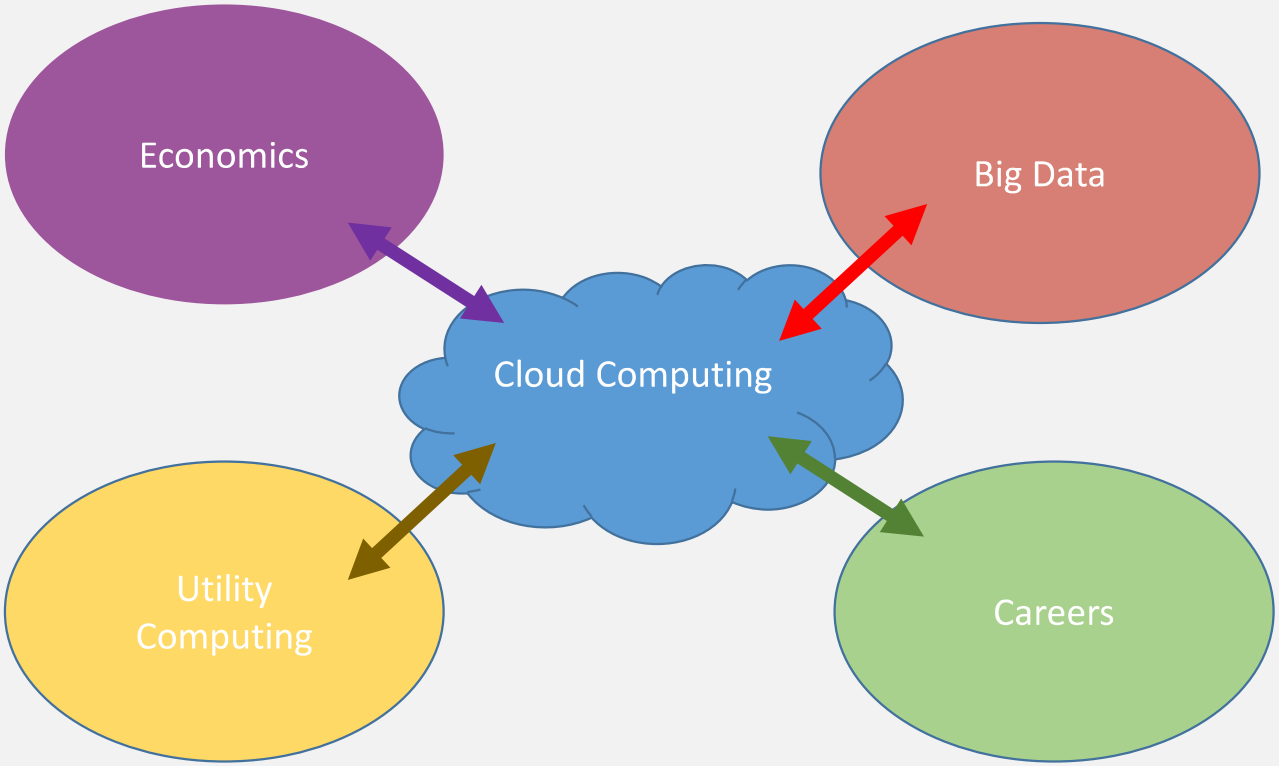
Careers

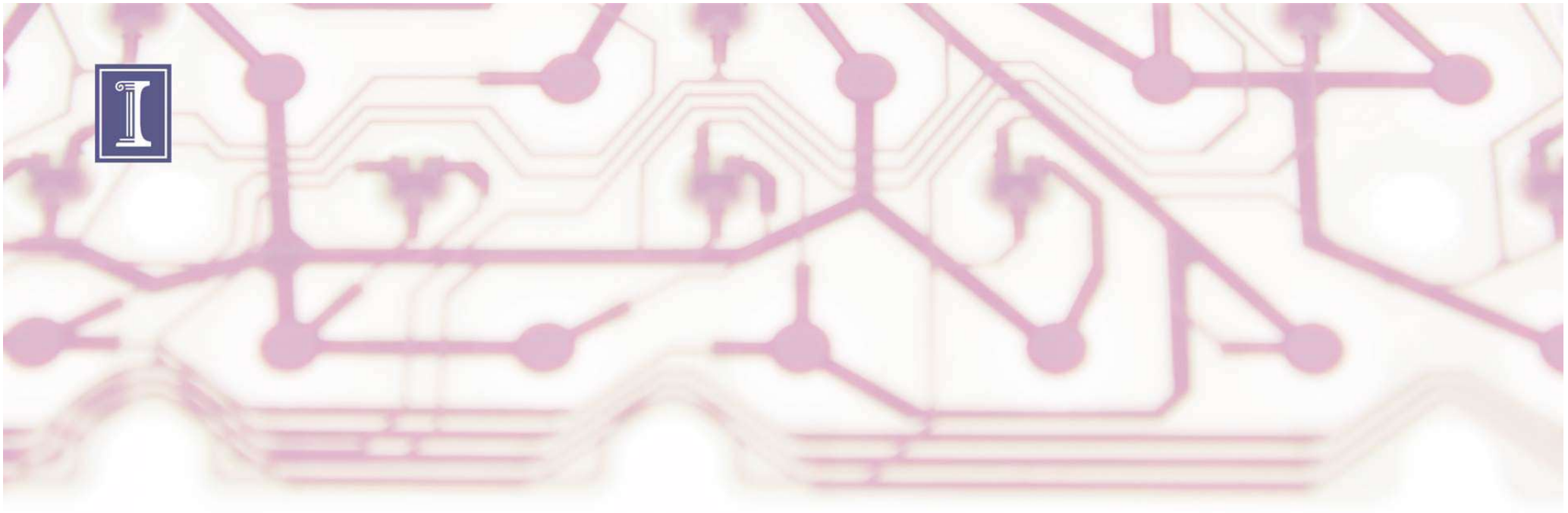
- Almost all people will use Cloud computing: search, video streaming, social networking
- IT specialists need to know Cloud
  - When to outsource, applications, uses
  - Architectures, sharing, privacy
  - Programming, efficiency, parallelism, scale
- General audience
  - Huge impact on society, business, government
  - New capabilities, policies, ways to communicate
  - Privacy and security

First Exercise

- Show scalability of web service
- Demonstrate user application
- Software-defined architecture
  - Management controls
  - Billing
  - Software modules available
- Distributed computing
  - Reliability, parallelism
- Provides insight to the power of sharing data

Summary


Economics
Big Data
Cloud Computing
Utility Computing
Careers



CLOUD COMPUTING APPLICATIONS

Infrastructure as a Service

Reza Farivar

Infrastructure as a Service

- The most fundamental of Cloud Computing Models
- Allows the user to “rent” computing resources
- The product is a “virtual” computer, that you can access remotely and do whatever you want
  - From the choice of the I/O specs for attached “hard drives” to the Operating System, middleware and applications
  - Network Connection
  - You are now responsible for managing everything running on the machine, including security of your server
- These resources are usually virtualized

Virtualized Resources

- Different customers have different needs
  - Ephemeral needs
- The Cloud Provider cannot operate a pool of many different sized computers
- Solution: Cloud provider operates a fleet of similar, powerful, hardware
- Carve out chunks of resources through virtualization
  - CPU
  - Memory
  - Storage
  - Network
  - Accelerators

Virtualized Resources

- Different customers have different needs
  - Ephemeral needs
- The Cloud Provider cannot operate a pool of many different sized computers
- Solution: Cloud provider operates a fleet of similar, powerful, hardware
- Carve out chunks of resources through virtualization: VM Instance
  - CPU
  - Memory
  - Storage
  - Network
  - Accelerators
- Metal as a Service (MaaS)

Dedicated Host SKUs (VM series and Host Type)	Available vCPUs	Available RAM	CPU
Dasv4_Type1	96	768 GiB	2.35 GHz AMD EPYC™ 7452
Ddsv4_Type1	80	504 GiB	Intel® Xeon® Platinum 8272CL (Cascade Lake)
Dsv4_Type1	80	504 GiB	Intel® Xeon® Platinum 8272CL (Cascade Lake)
Dsv3_Type1	64	256 GiB	2.3 GHz Intel® Xeon® E5-2673 v4 (Broadwell)
Dsv3_Type2	76	504 GiB	Intel® Xeon® Platinum 8171M (Skylake)
Esv3_Type2	76	504 GiB	Intel® Xeon® Platinum 8171M (Skylake)
Esv3_Type3	80	504 GiB	Intel® Xeon® Platinum 8272CL (Cascade Lake)
Fsv2_Type2	72	144 GiB	Intel® Xeon® Platinum 8168 (Skylake)
Fsv2_Type3	86	504 GiB	Intel® Xeon® Platinum 8272CL (Cascade Lake)
Lsv2_Type1	80	640 GiB	2.55 GHz AMD EPYC™ 7551
Ms_Type1	128	2,048 GiB	Intel® Xeon® Platinum 8280 (Cascade Lake)
Msm_Type1	128	3,892 GiB	Intel® Xeon® Platinum 8280 (Cascade Lake)
Msmv2_Type1	416	11,400 GiB	Intel® Xeon® Platinum 8180M (Skylake)

Virtualized Resources

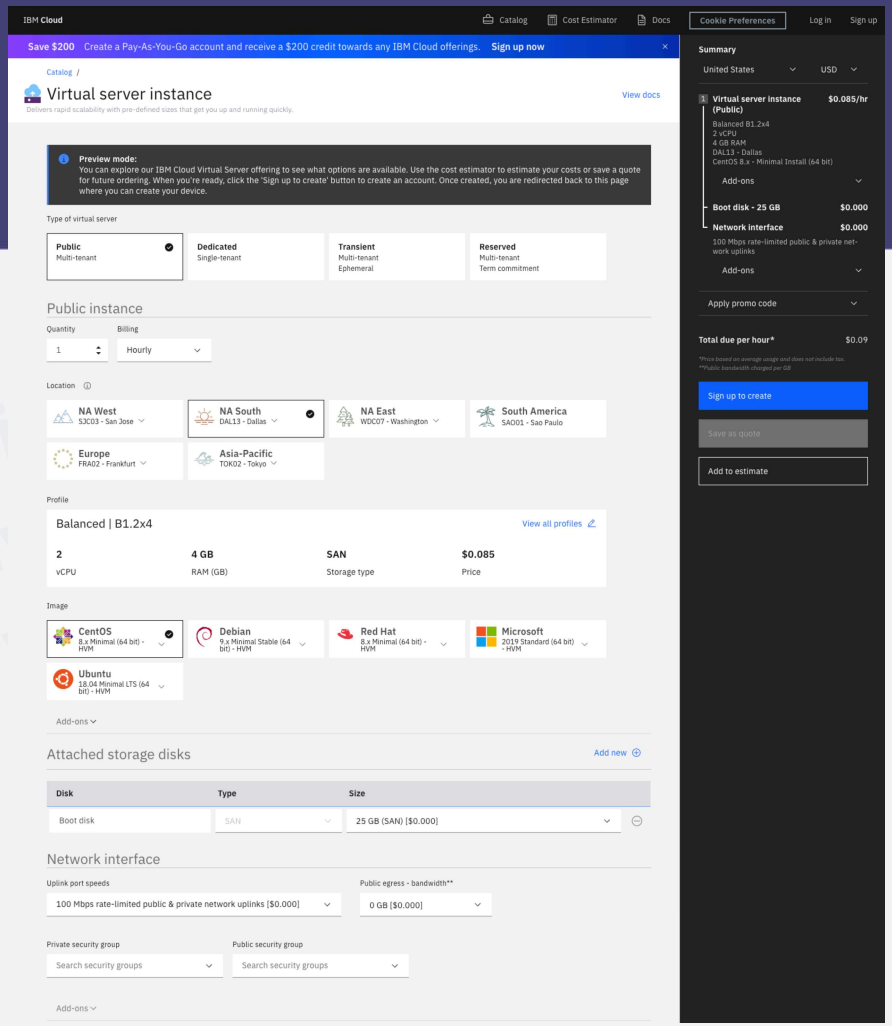
- Different customers have different needs
  - Ephemeral needs
- The Cloud Provider cannot operate a pool of many different sized computers
- Solution: Cloud provider operates a fleet of similar, powerful, hardware
- Carve out chunks of resources through virtualization: VM Instance
  - CPU
  - Memory
  - Storage
  - Network
  - Accelerators
- Metal as a Service (MaaS)

Advantages of IaaS vs. On-Prem

- No need to run a data center
  - No worries about space, power supplies, physical building security, network, failing components, …
- OpEx vs. CapEx
- Use different instances when needed
  - Rapid innovation
  - Quick response to shifting business conditions

IaaS Examples

- Microsoft Azure
- Amazon EC2 (Elastic Compute Cloud)
- Google Cloud Platform Compute Engine
- Oracle Cloud
- IBM Cloud
- Alibaba Cloud
- Rackspace
- Vultr
- …



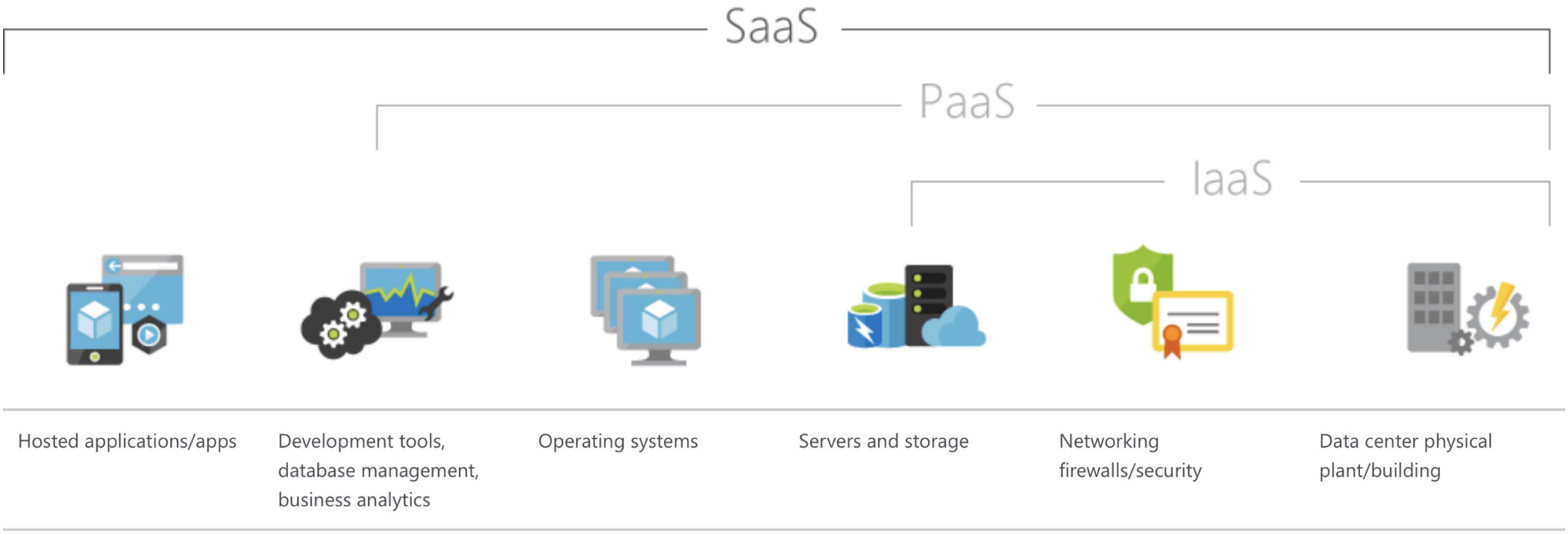
Instance Pricing

- On-Demand
- Reserved
- Spot Pricing

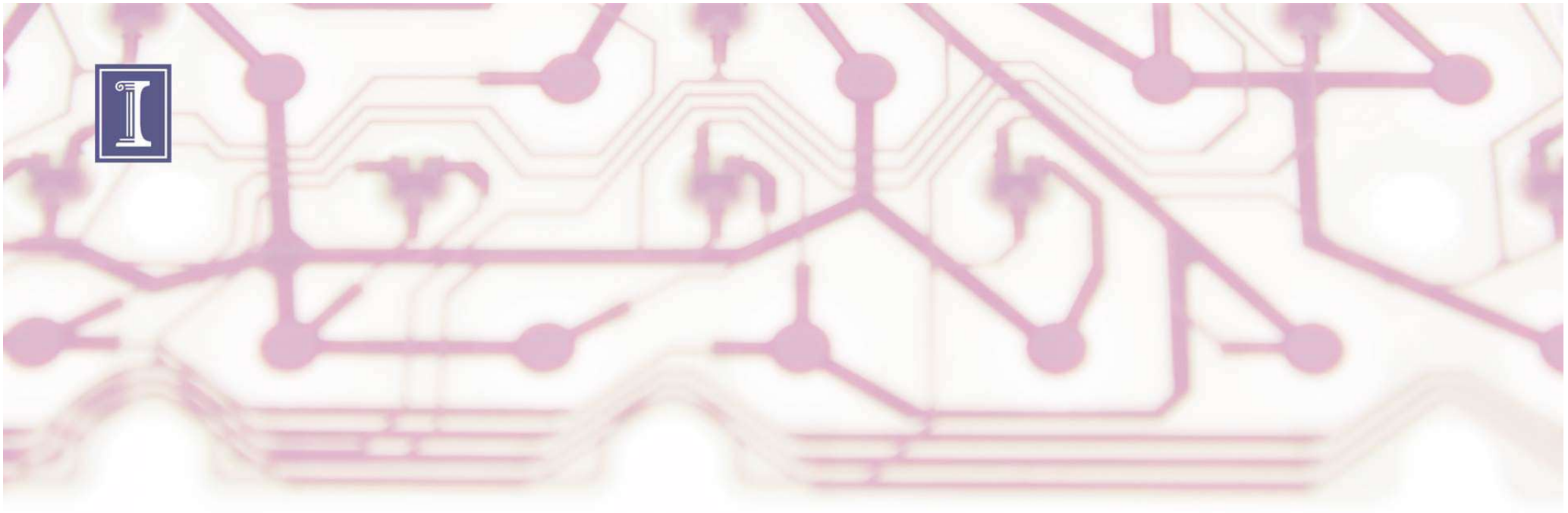
IaaS Sub-Category: Containers and Orchestration

- A subcategory of IaaS, or a place half-way between IaaS and PaaS (more towards the IaaS)
- You may think of a container as a light-weight Virtual Machine
  - Time to spin up a VM is tens of seconds to a few minutes
  - Time to start a container is fraction of a second to a few seconds
- Linux-Only

SaaS in Perspective



** Image courtesy of Microsoft Azure*



CLOUD COMPUTING APPLICATIONS

Infrastructure as a Service: Regions and Zones

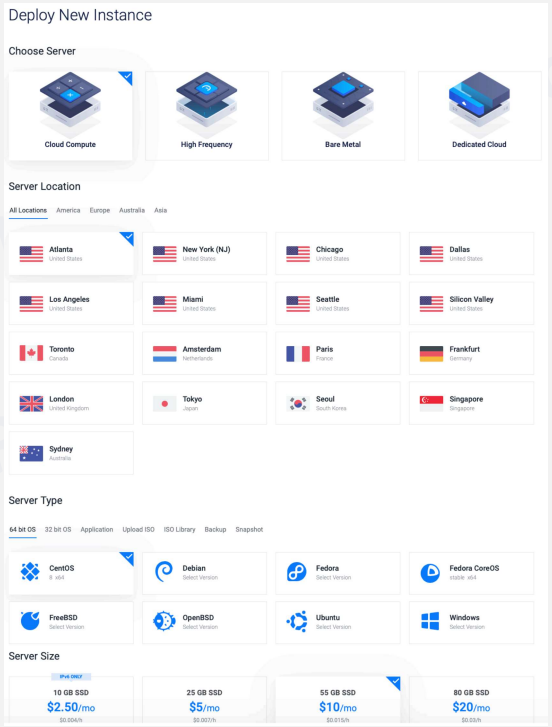
Reza Farivar

Virtual Machine Instance Location

- The cloud provider has multiple physical data centers, all over the globe
- Where does your virtual machine reside?
- Regions
- Availability Zones / Zones

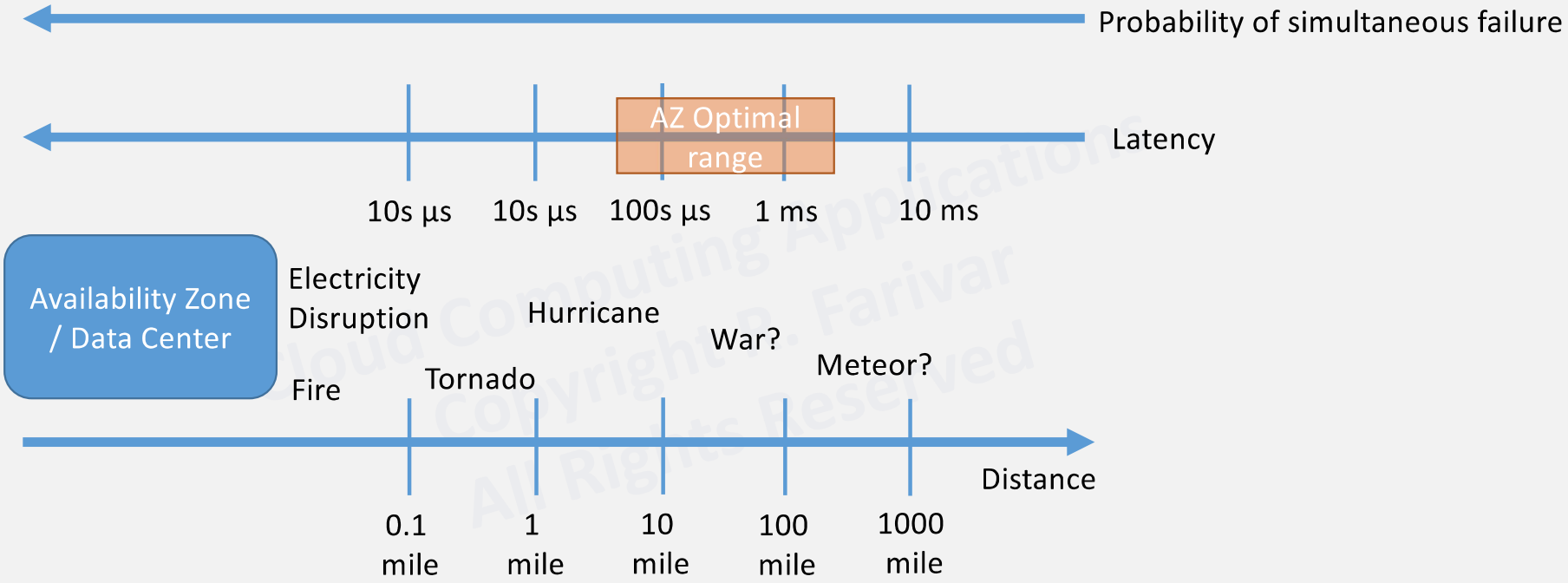
Data Center Location

- Some Cloud Providers simply let you choose the Data Center
- E.g. Vultr:



Availability Zones and Regions

Availability Zones Locations



Availability Zones and Regions

- Typically each user ID gets access to a handful of availability zones to launch VM instances per each region
  - AWS: us-east-1 ➔ us-east-1a, us-east-1b, …, us-east-1f
  - Azure: US East ➔ 1, 2, 3
  - GCP: us-east1 ➔ us-east1-b, us-east1-c, us-east1-c
- The availability zone “a” for user1 is NOT the same as availability zone “a” for user2

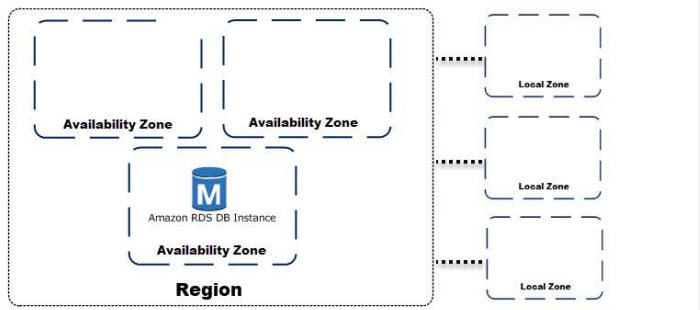
SLA for Virtual Machines

Last updated: July 2020

- For all Virtual Machines that have two or more instances deployed across two or more Availability Zones in the same Azure region, we guarantee you will have Virtual Machine Connectivity to at least one instance at least 99.99% of the time.
- For all Virtual Machines that have two or more instances deployed in the same Availability Set or in the same Dedicated Host Group, we guarantee you will have Virtual Machine Connectivity to at least one instance at least 99.95% of the time.
- For any Single Instance Virtual Machine using Premium SSD or Ultra Disk for all Operating System Disks and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 99.9%.
- For any Single Instance Virtual Machine using Standard SSD Managed Disks for Operating System Disk and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 99.5%.
- For any Single Instance Virtual Machine using Standard HDD Managed Disks for Operating System Disks and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 95%.

Availability Zones and Regions

- Each zone is made up of **one or more datacenters** equipped with independent power, cooling, and networking.
  - Chance of simultaneous failure in all the separate AZs in a region is extremely small
- *Local zones: A *Local Zone* is an extension of a Region that is geographically close to your users



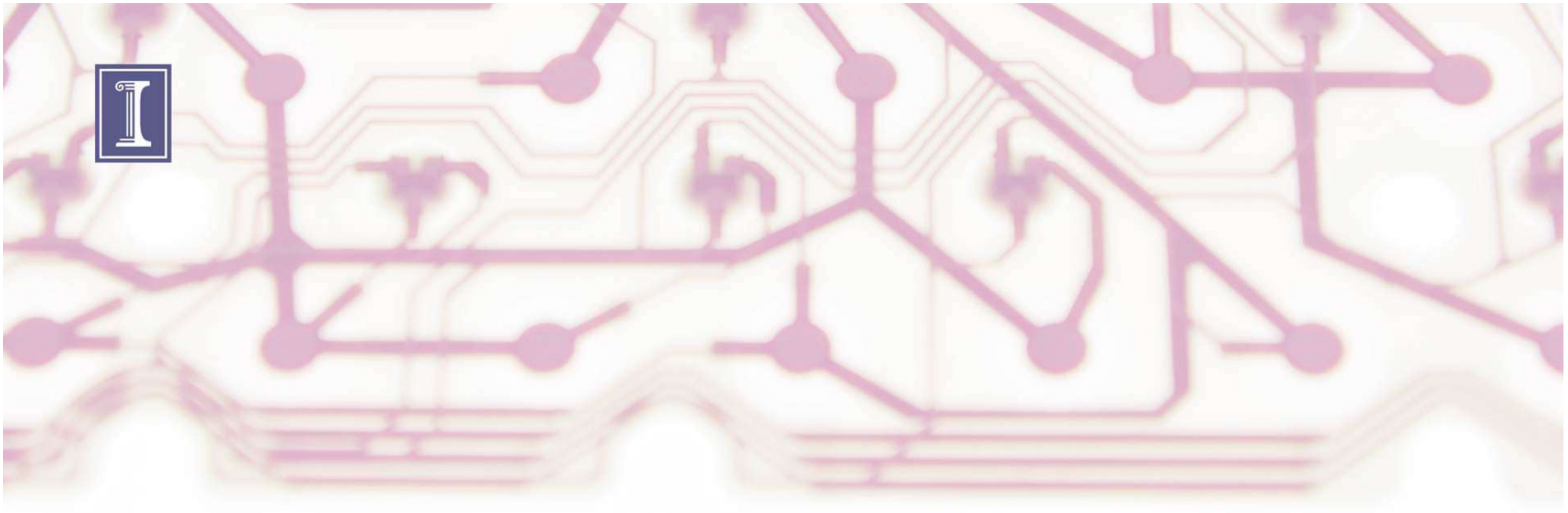
Availability Zones and Regions

- For high availability, design your system to have instances running in multiple AZ in a region, and maybe even have more than one region
- Data traffic costs most from one region to another, then from one AZ to another in the same region, and it is cheapest (or free) in the same AZ

SLA for Virtual Machines

Last updated: July 2020

- For all Virtual Machines that have two or more instances deployed across two or more Availability Zones in the same Azure region, we guarantee you will have Virtual Machine Connectivity to at least one instance at least 99.99% of the time.
- For all Virtual Machines that have two or more instances deployed in the same Availability Set or in the same Dedicated Host Group, we guarantee you will have Virtual Machine Connectivity to at least one instance at least 99.95% of the time.
- For any Single Instance Virtual Machine using Premium SSD or Ultra Disk for all Operating System Disks and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 99.9%.
- For any Single Instance Virtual Machine using Standard SSD Managed Disks for Operating System Disk and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 99.5%.
- For any Single Instance Virtual Machine using Standard HDD Managed Disks for Operating System Disks and Data Disks, we guarantee you will have Virtual Machine Connectivity of at least 95%.



CLOUD COMPUTING APPLICATIONS

Platform as a Service

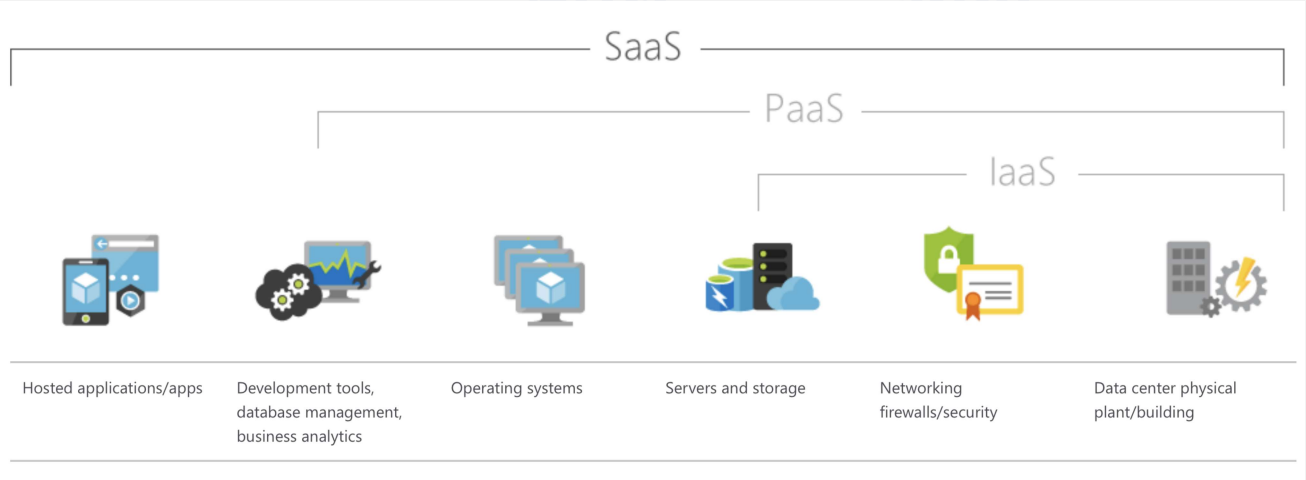
Reza Farivar

Platform as a Service

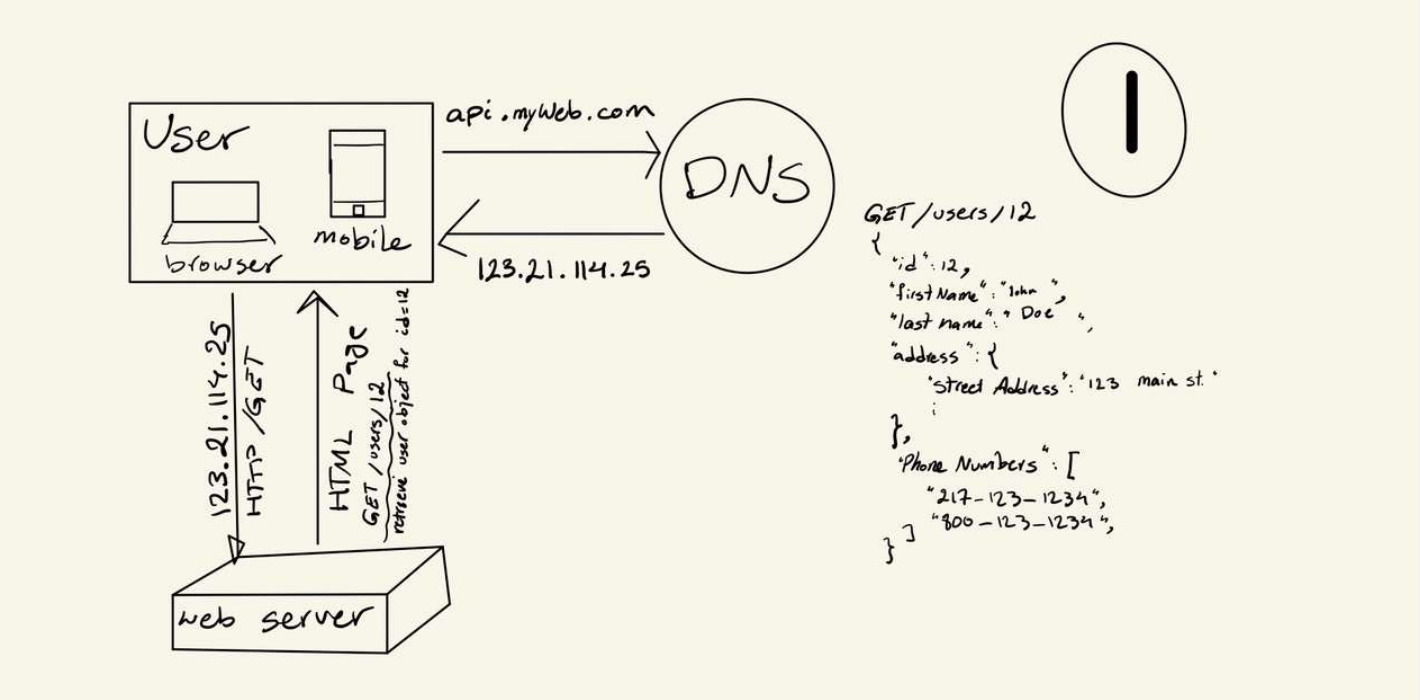
- The main goal is to run the user's distributed web service in a managed environment
  - Much more opinionated than IaaS

Platform as a Service

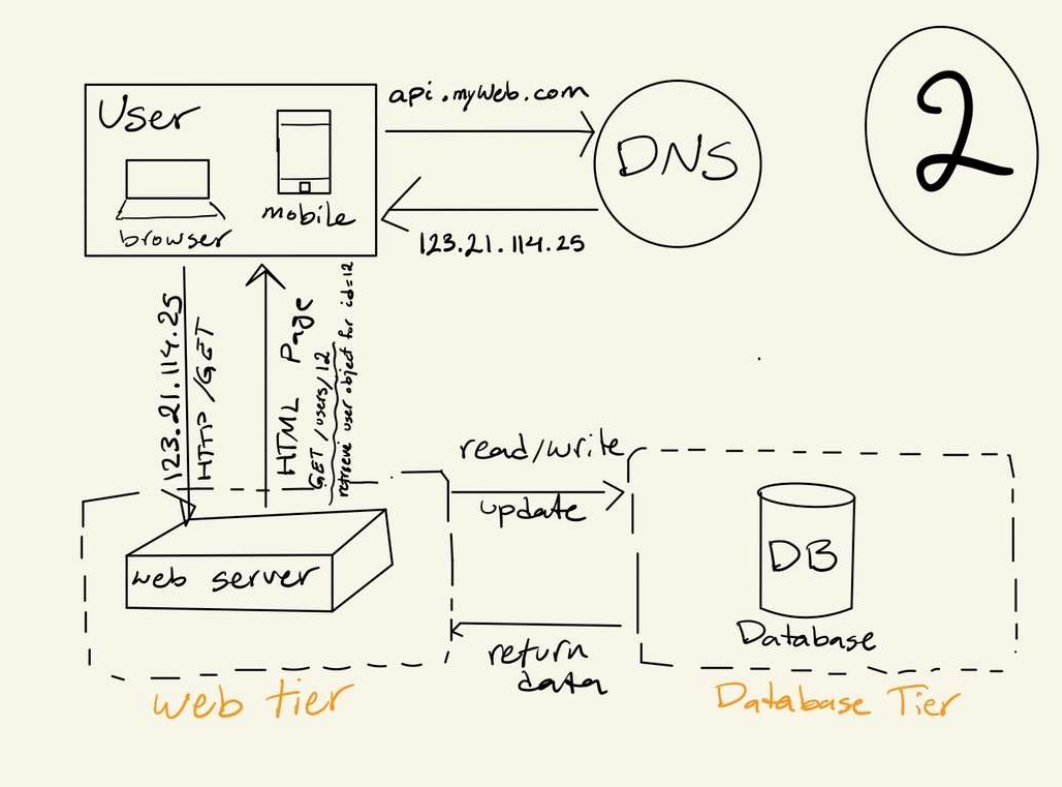
- The main goal is to run the user's distributed web service in a managed environment
  - Much more opinionated than IaaS



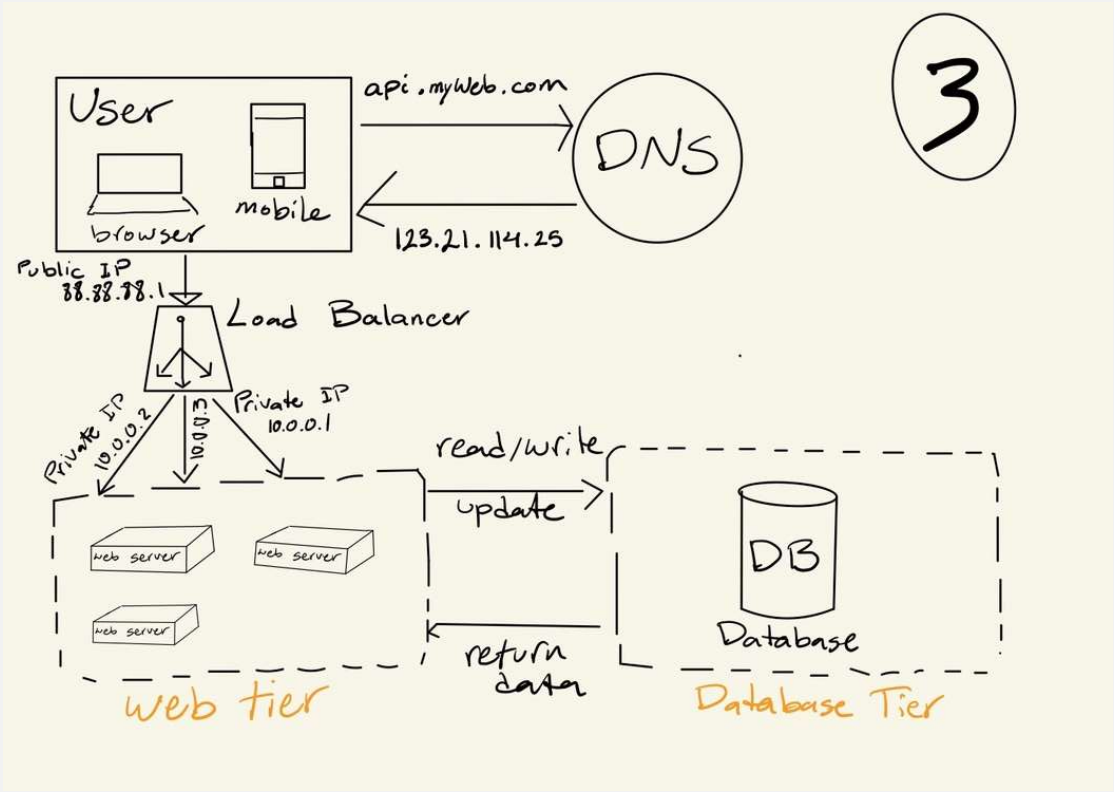
Anatomy of a Web Service Application



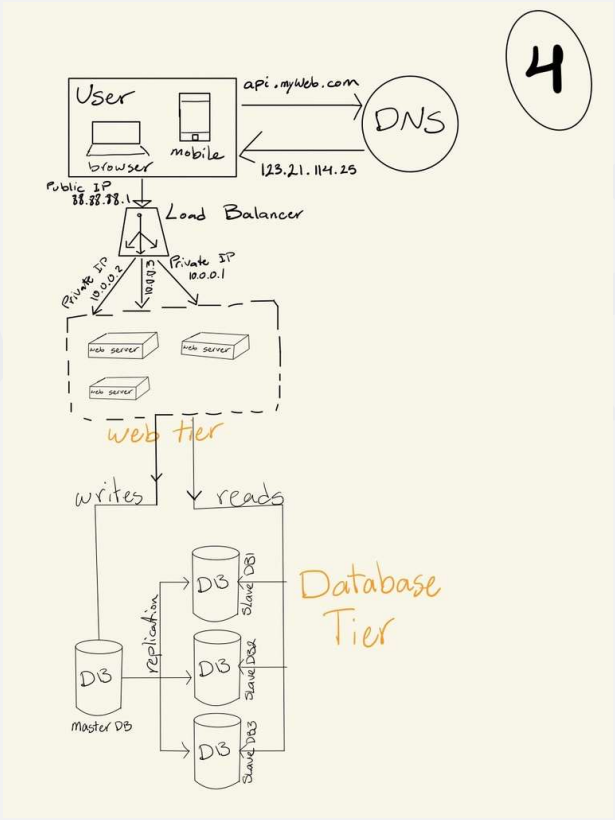
Anatomy of a Web Service Application



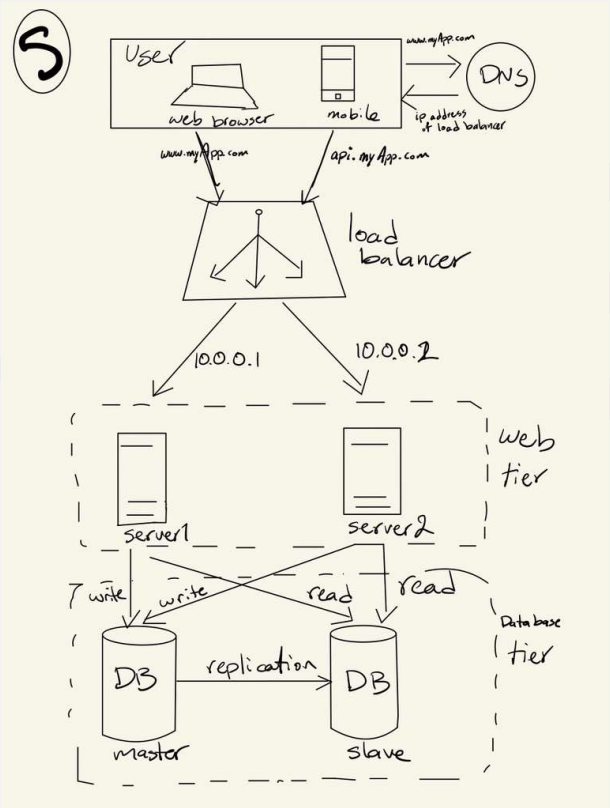
Anatomy of a Web Service Application



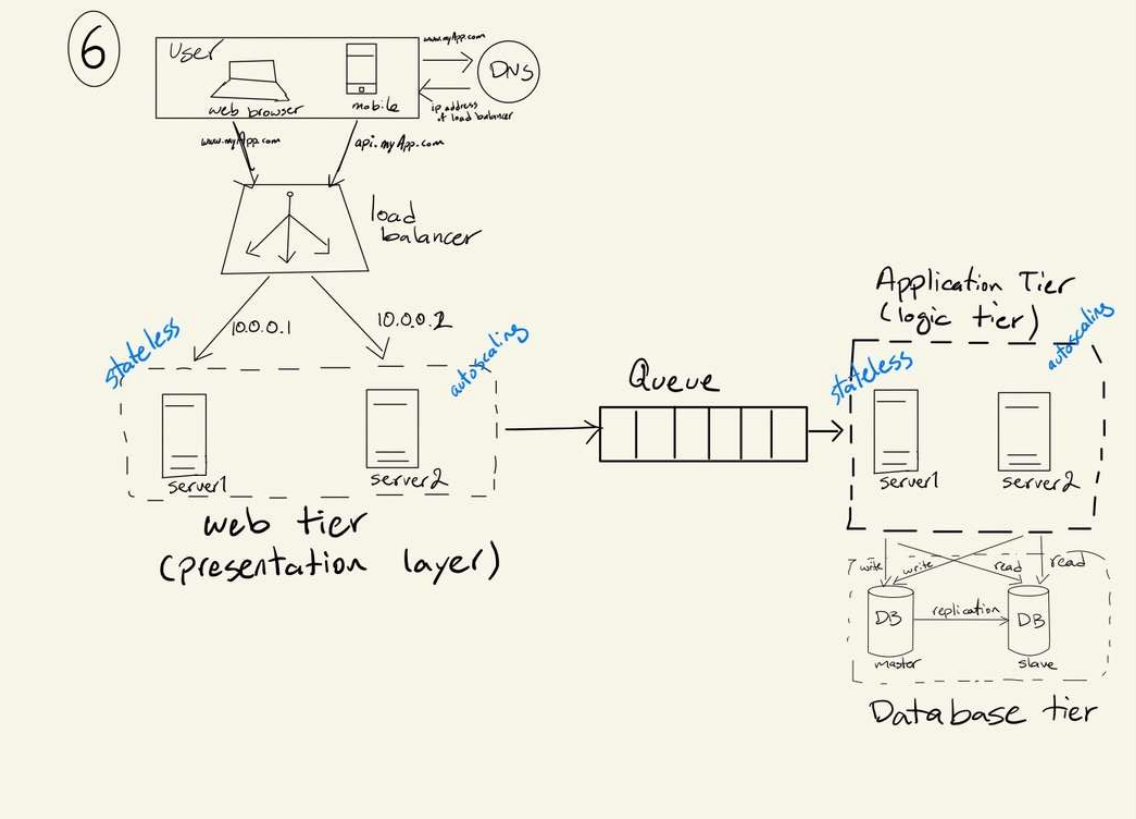
Anatomy of a Web Service Application



Anatomy of a Web Service Application



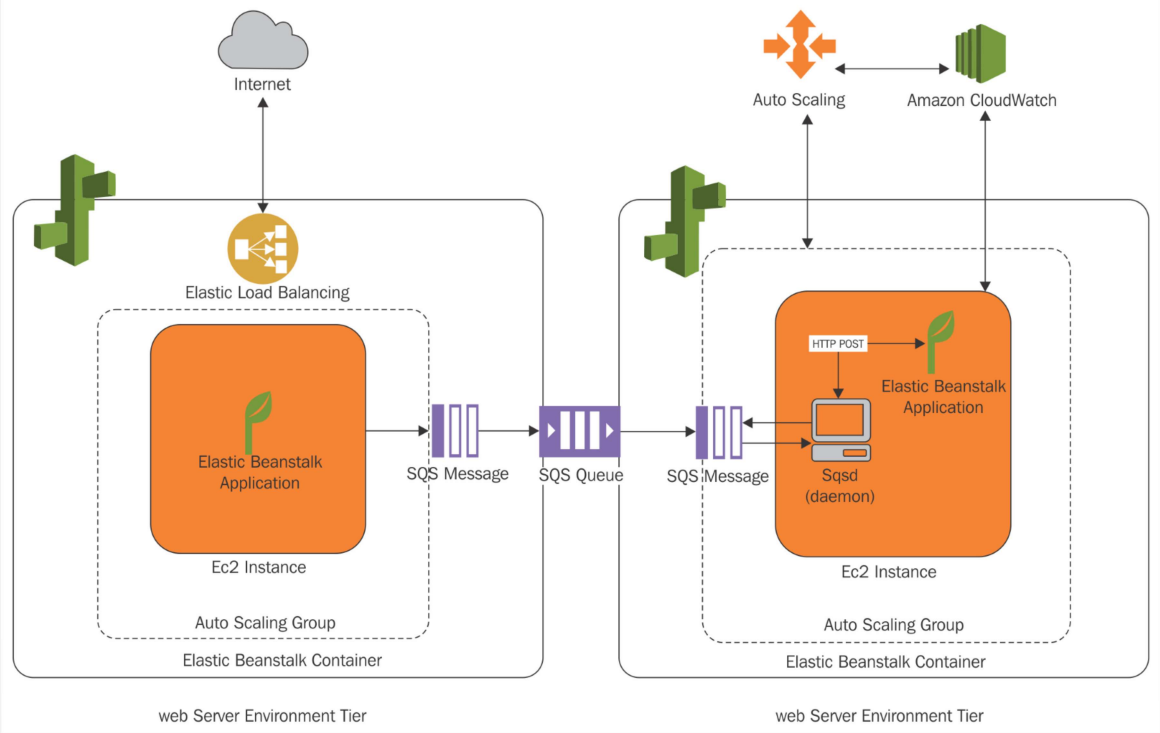
Anatomy of a Web Service Application



Platform as a Service

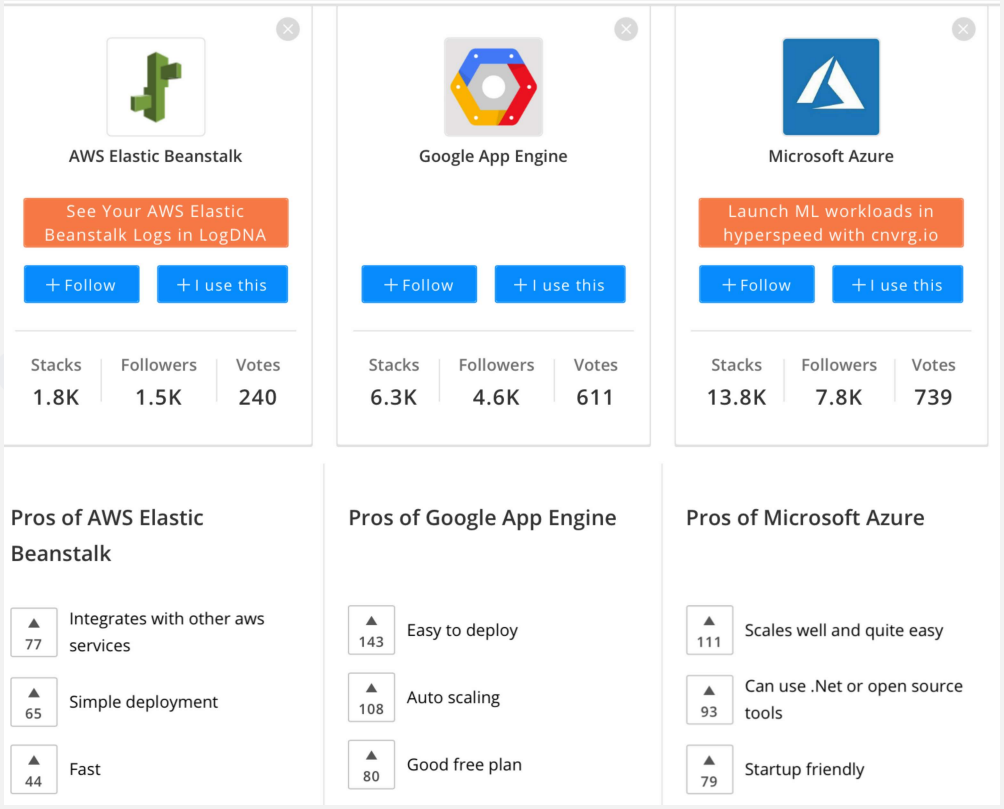
- Platform as a Service provides environments where all the machinery needed for the distributed application is provided by the cloud provider and managed by them
  - Autoscaling groups
  - Load balancers
  - Queues
  - Daemons managing the queues
  - Consistent data Storage solutions
    - SQL Databases
    - NoSQL solutions

Platform as a Service

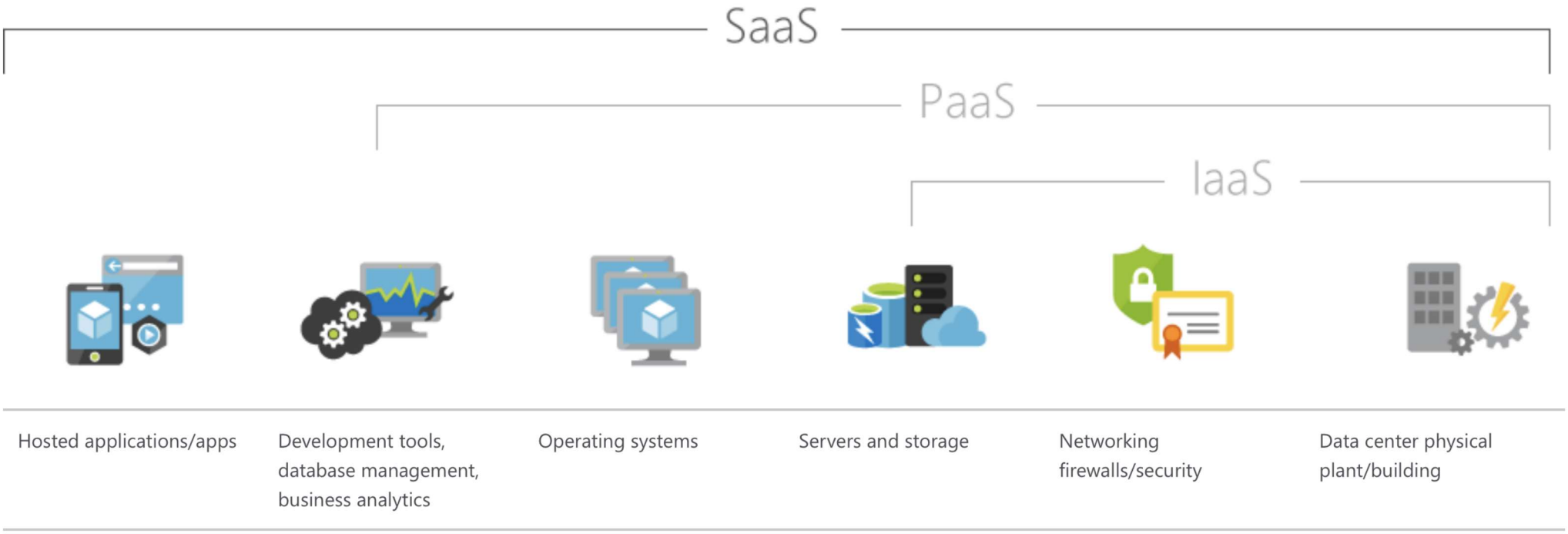


Examples

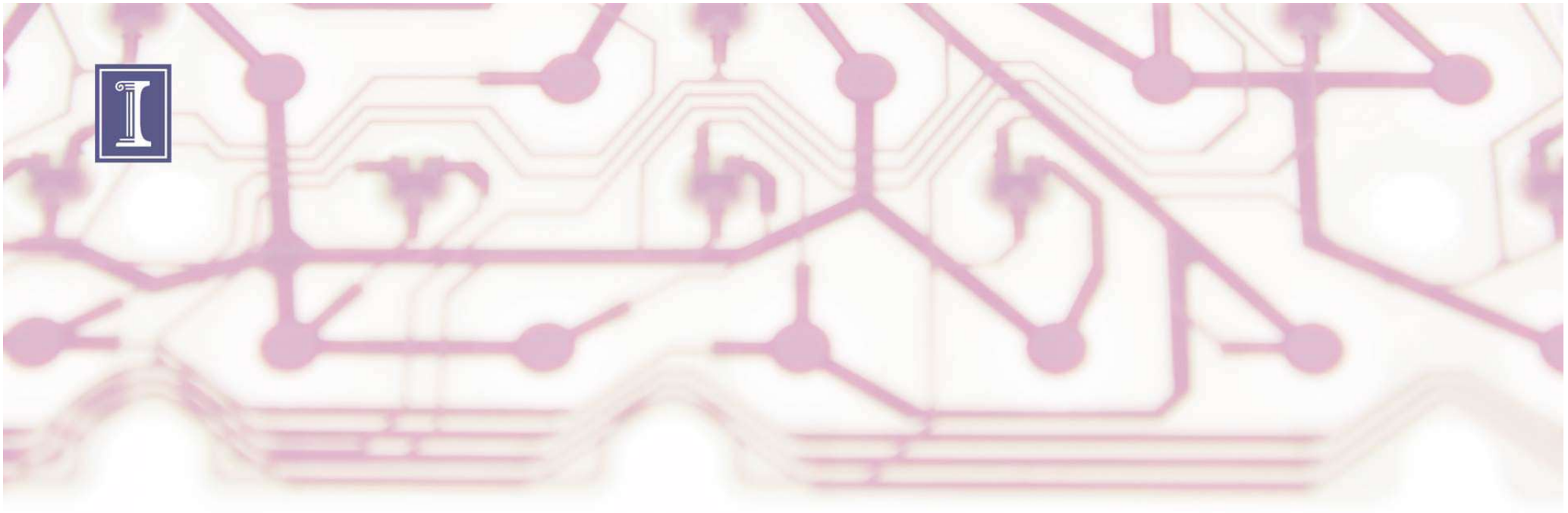
- Microsoft Azure App Service
- Google App Engine
- Amazon Elastic BeanStalk
- Heroku
- IBM Cloud Foundry



SaaS in Perspective



** Image courtesy of Microsoft Azure*



CLOUD COMPUTING APPLICATIONS

Platform as a Service

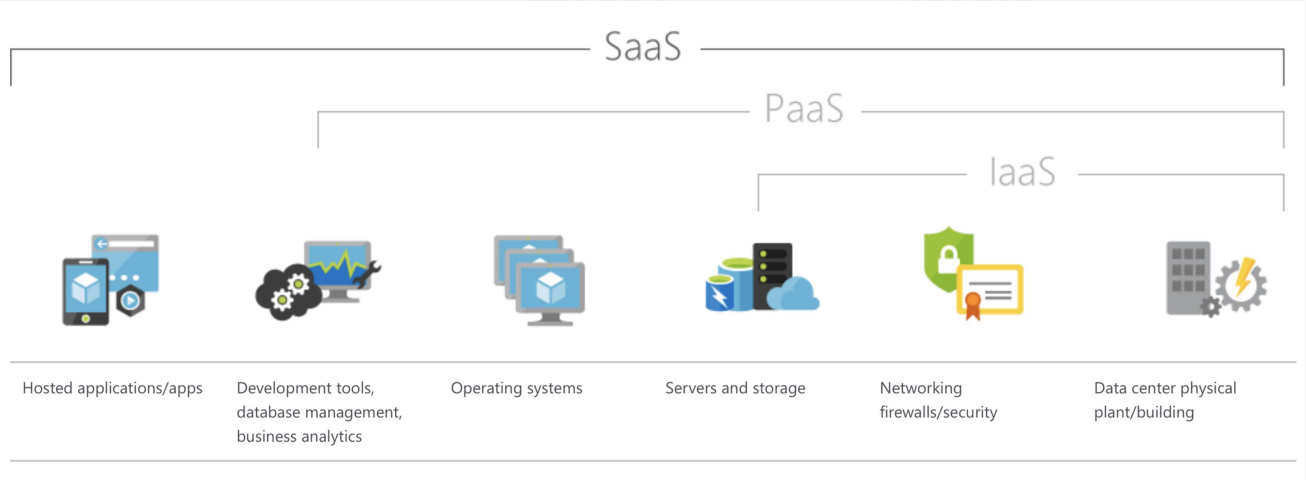
Reza Farivar

Platform as a Service

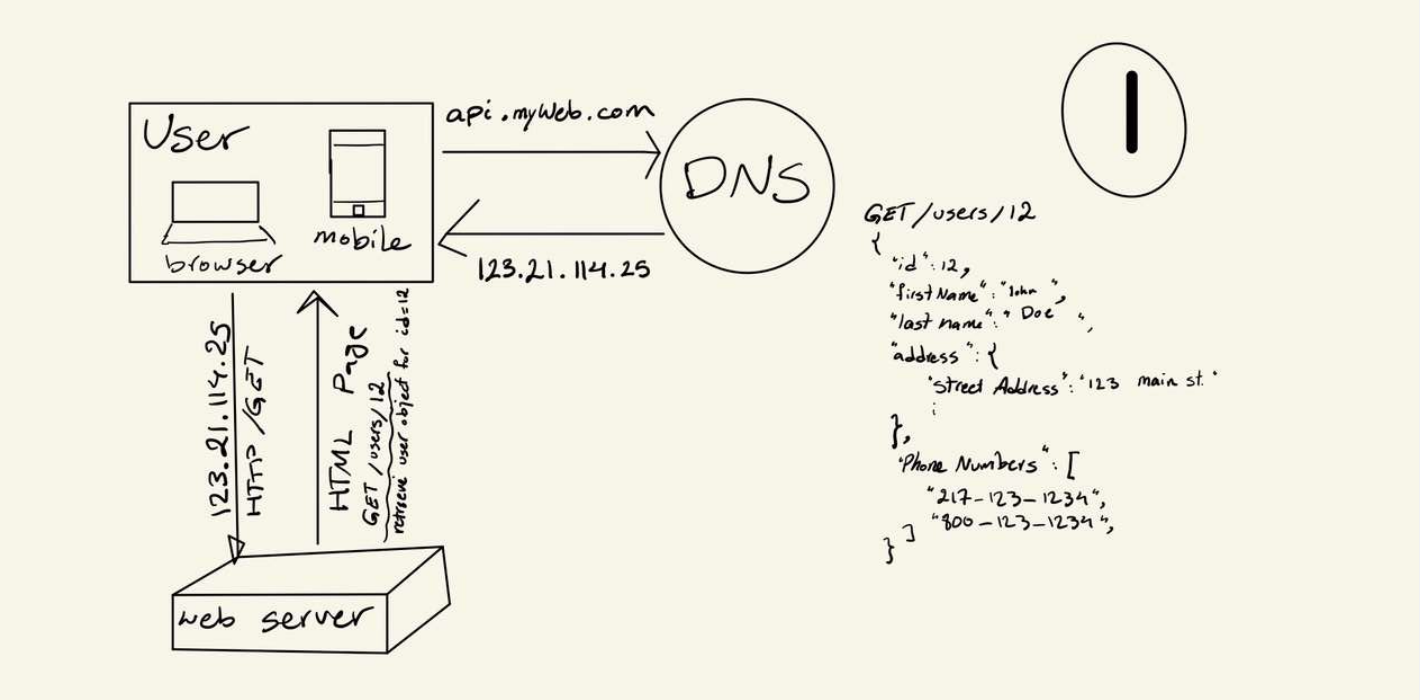
- The main goal is to run the user's distributed web service in a managed environment
  - Much more opinionated than IaaS

Platform as a Service

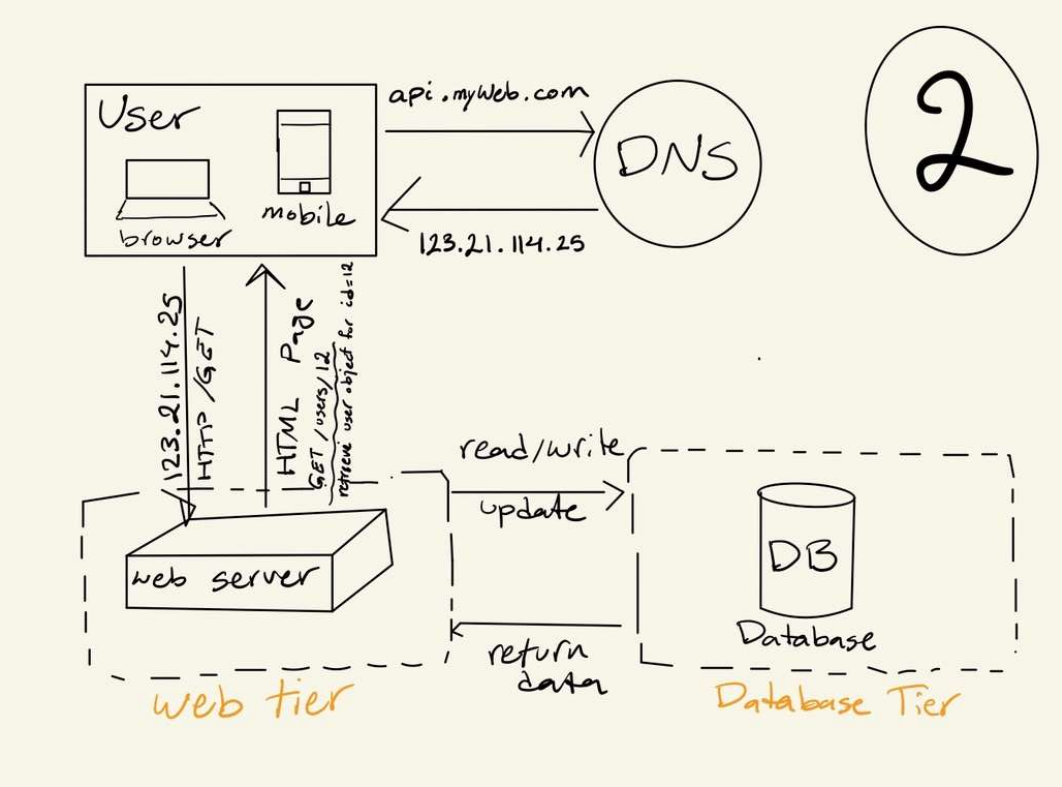
- The main goal is to run the user's distributed web service in a managed environment
  - Much more opinionated than IaaS



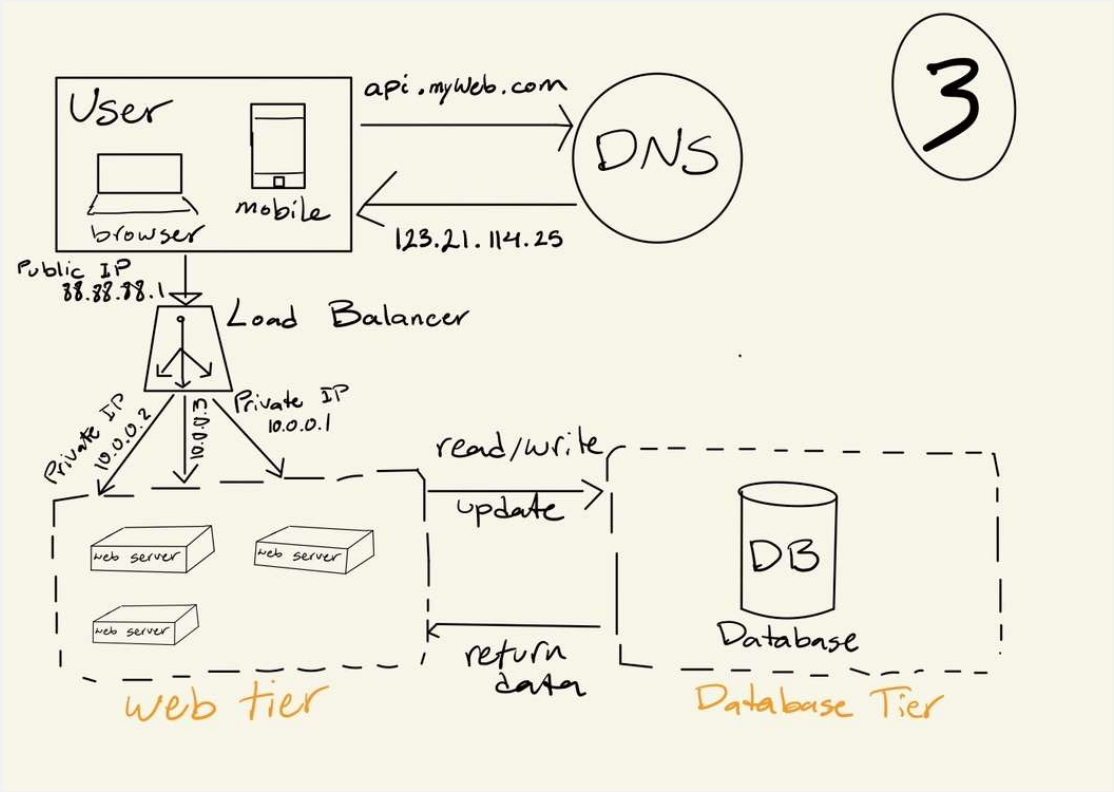
Anatomy of a Web Service Application



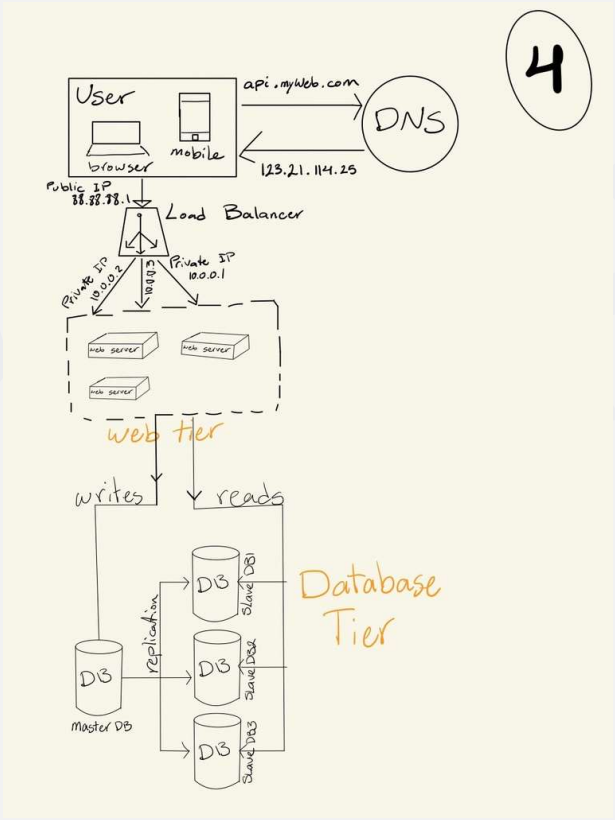
Anatomy of a Web Service Application



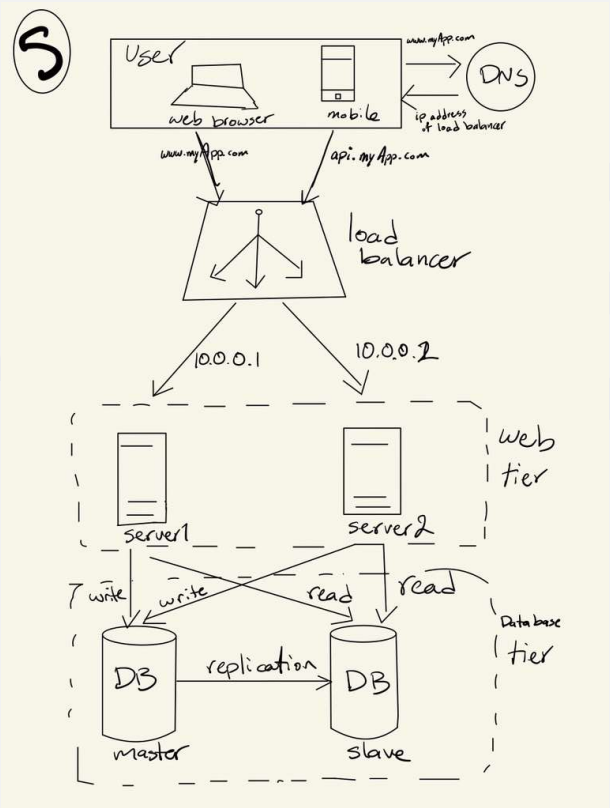
Anatomy of a Web Service Application



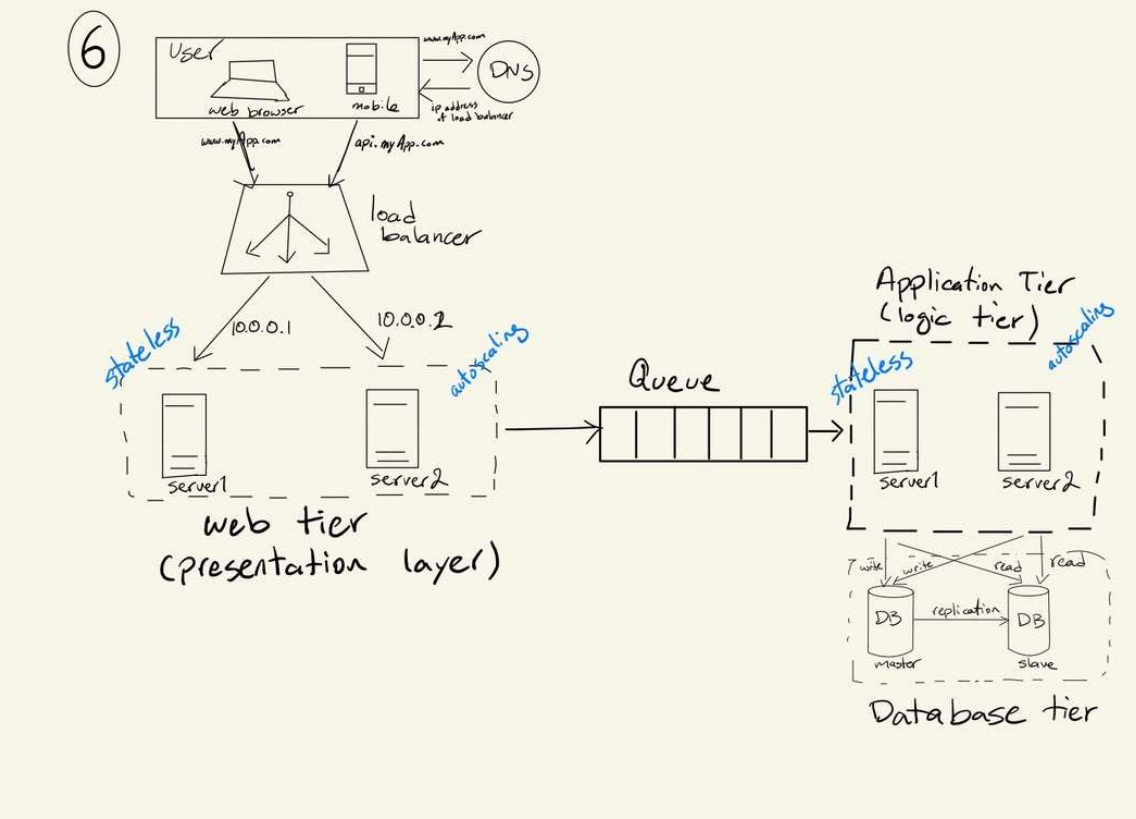
Anatomy of a Web Service Application



Anatomy of a Web Service Application



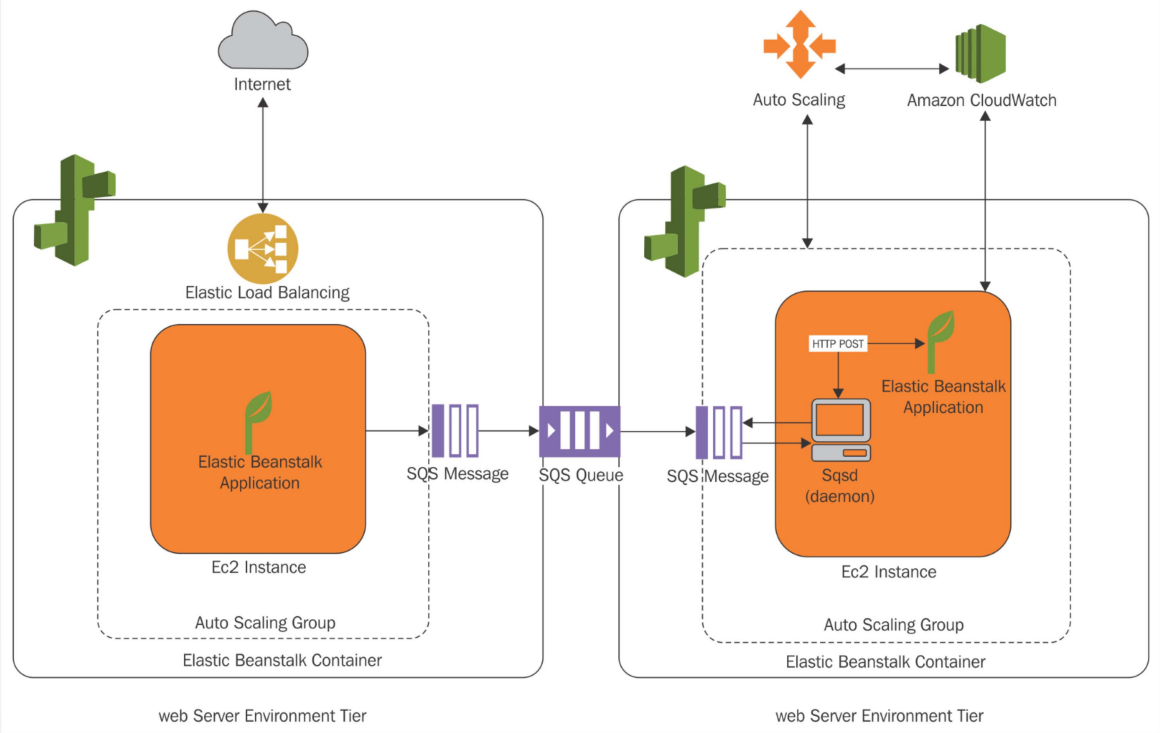
Anatomy of a Web Service Application



Platform as a Service

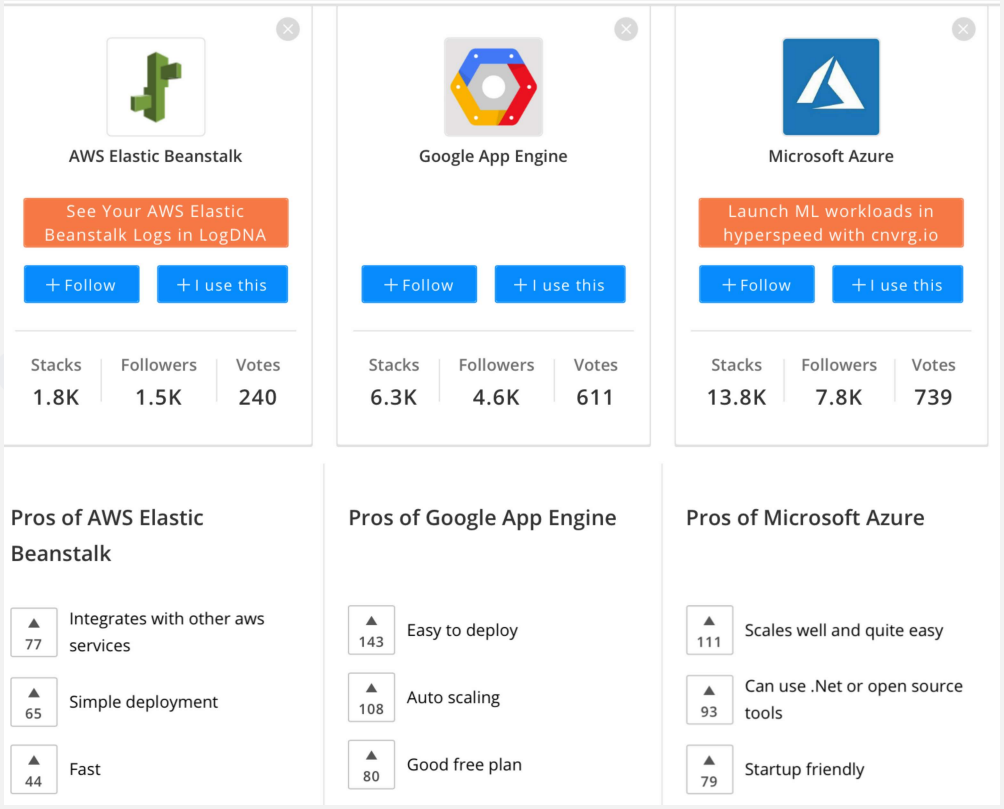
- Platform as a Service provides environments where all the machinery needed for the distributed application is provided by the cloud provider and managed by them
  - Autoscaling groups
  - Load balancers
  - Queues
  - Daemons managing the queues
  - Consistent data Storage solutions
    - SQL Databases
    - NoSQL solutions

Platform as a Service

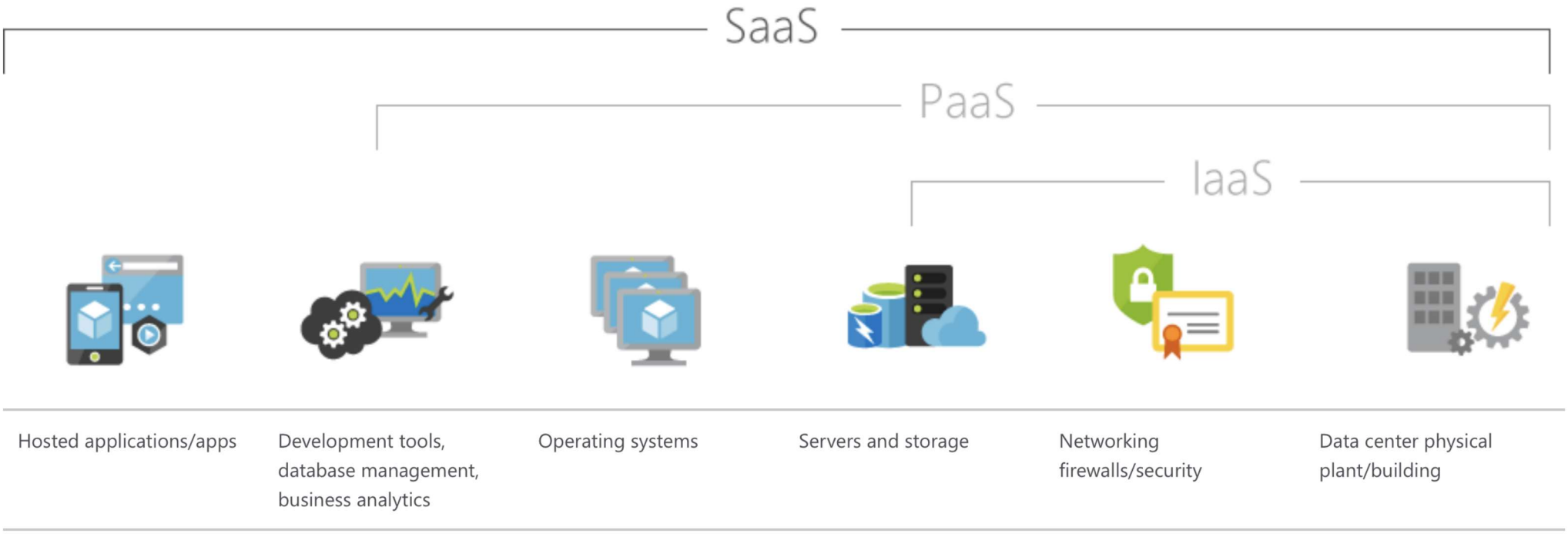


Examples

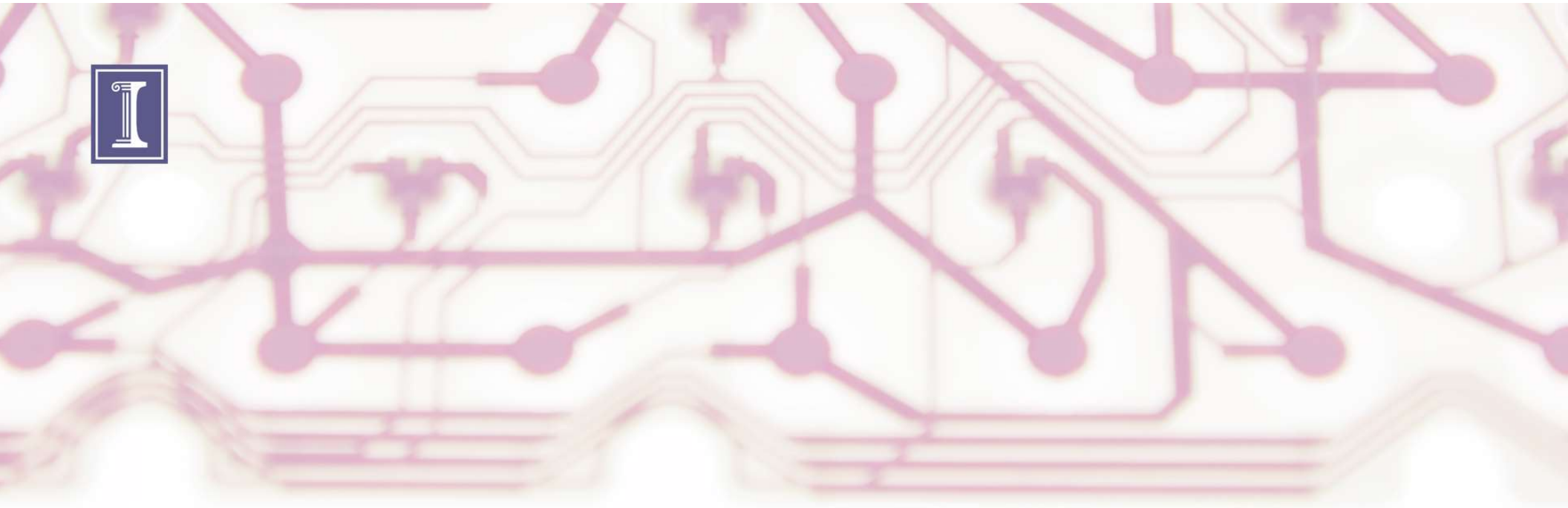
- Microsoft Azure App Service
- Google App Engine
- Amazon Elastic BeanStalk
- Heroku
- IBM Cloud Foundry



SaaS in Perspective



** Image courtesy of Microsoft Azure*



CLOUD COMPUTING APPLICATIONS

PaaS Providers: Google App Engine

Prof. Roy Campbell

Google App Engine (GAE)

- GAE was developed by Google in 2008 as a PaaS
- It supports multi-tenancy and offers automatic scaling for web applications
- It supports Python, Java, and Go

GAE Frameworks and Tools

- GAE supports Django web framework and the Grails web app framework
- GAE provides infrastructure tools that enable users to deploy code without worrying about infrastructure challenges such as deployment, failover, or scalability
- However, the GAE infrastructure limits the type of applications that can be run

GAE Security, Sandbox

- Applications run in a secure environment
- Isolates applications from hardware and operating system, and imposes security limitations
- For example, application code only runs in response to requests, and a request handler cannot spawn potentially malicious sub-processes after response has been sent

Storing GAE Data

- Users of GAE can use App Engine Datastore, Google Cloud SQL, and Google Cloud Storage

- Users can also harness Google's database technology, such as Bigtable

GAE's Use with Google Services

- Can take advantage of Google's single sign on feature when users want to access their Gmail or Google docs
- Build Chrome and Android games on GAE
- Google Cloud Endpoints to use / access mobile services

Other Services Supported

- App engine Map Reduce
- Search API
- SSL support
- Page speed
- XMPP API
- Memcache API

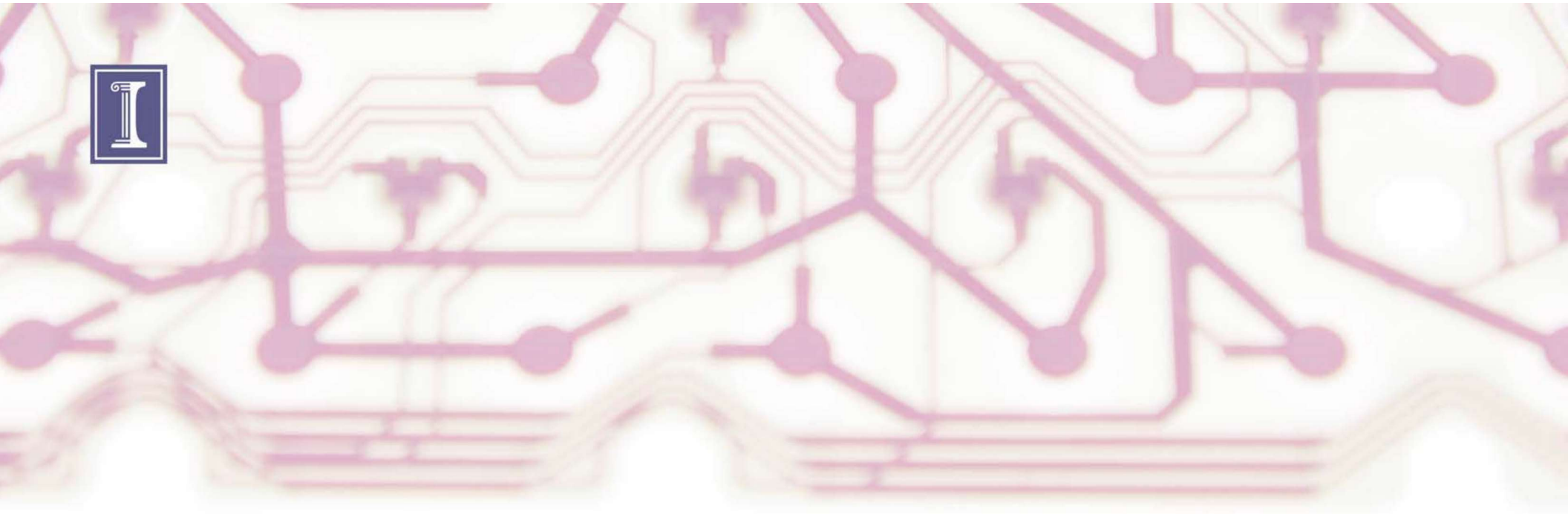
Case Studies of GAE

- BugSense - An application error-reporting service, it used GAE to maintain logs of bugs in software and analyze them
- Ubisoft - Used GAE to build its first web-based game, “From Dust,” on Chrome browser
- Claritics - A small social analytics company of 15 employees, used to analyze game data sets

GAE is Great for Mobile

- Many cell phone apps use GAE for their backend, e.g., Ruzzle and Tap Zoo

- GAE's purpose – being able to scale up for small teams of developers – fits well



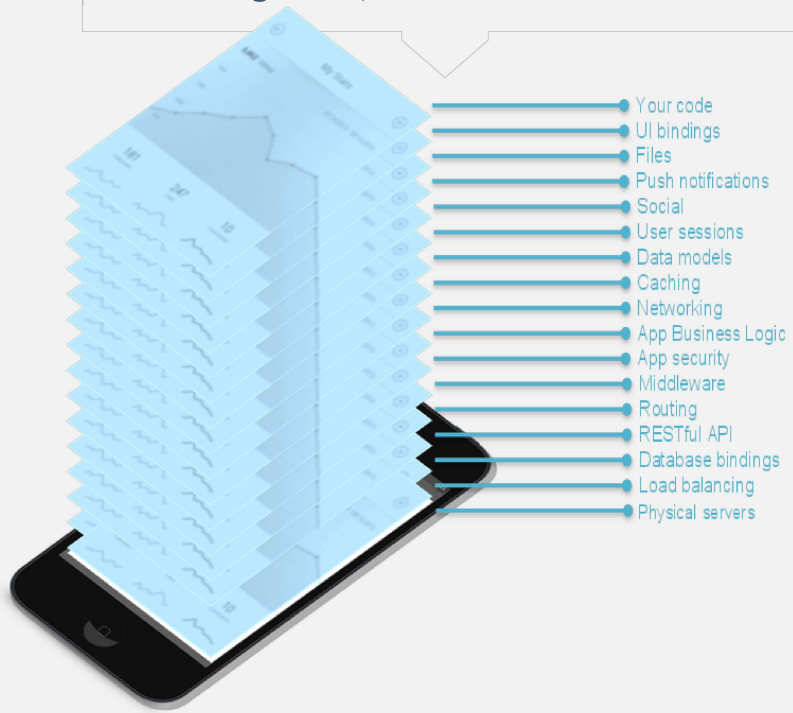
CLOUD COMPUTING APPLICATIONS

Mobile Backend as a Service

Roy Campbell & Reza Farivar

Why MBaaS?

When you are thinking to develop a new app, you have lot of steps like server setup, database creation, routing, social integration, UI binding, file management, etc. that need to be solved:



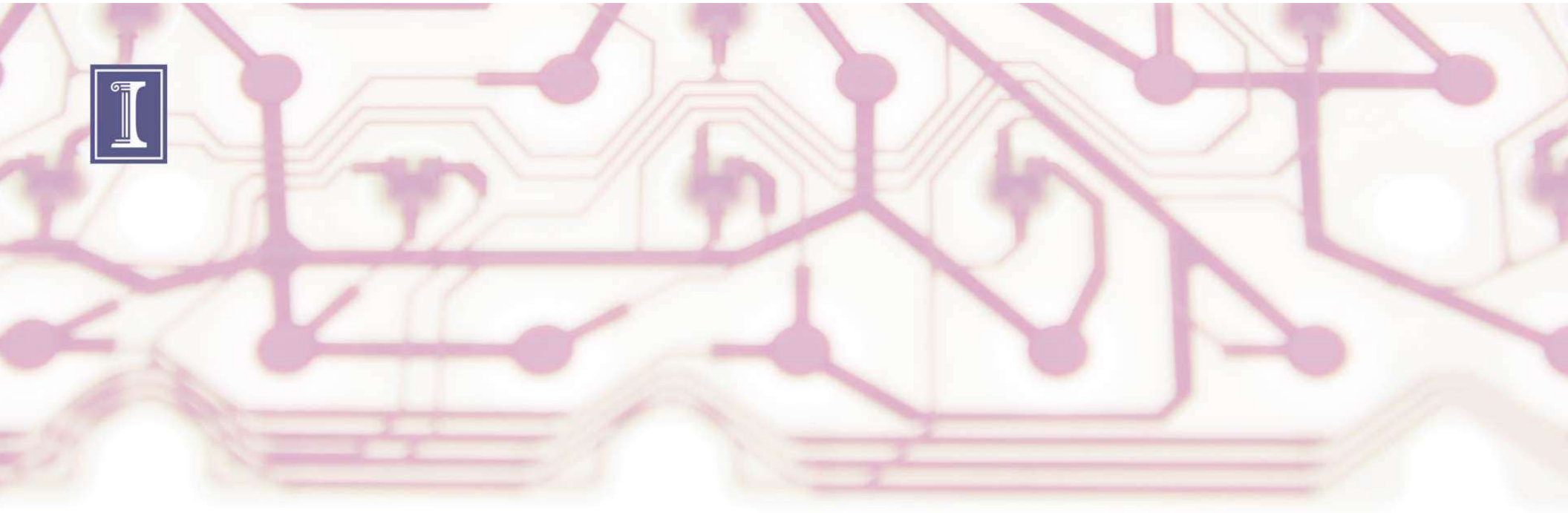
Just imagine, you focus only on your frontend code and the rest will bind together as a service as follows:

Introduction to MBaaS

- General idea: mobile apps need common services that can be shared among apps instead of being custom developed for each
- Enable web and mobile app developers to link their applications to backend cloud storage and backend APIs
  - Cloud storage
  - User management
  - Push notifications
  - Integration with social networking services
- Provide all of these in a one-shop model

MBaaS Examples

- Appcelerator, AnyPresence, Appery, Built.io, FeedHenry, Kinvey, TruMobi, Apple CloudKit (iCloud), etc.
- Many commonalities
  - E.g. many use MongoDB to serve JSON objects
  - REST API common
  - MicroServices
  - DevOps
  - Frontend design framework
- Different levels of enterprise integration
- Either on-premise or in private clouds
- Some support compliance with HIPAA, PCI, FIPS, and EU data security standards



CLOUD COMPUTING APPLICATIONS

Software as a Service

Reza Farivar

Software Distribution

- Once upon a time, retail was the main method of getting access to software
- With the advent of Cloud Computing, that model is now dated
- Then came software purchase over internet as downloads
  - Mobile App Store
  - Steam
  - Etc.
- Third wave is SaaS, which mainly rely on Browser capability and broadband internet
  - Browser is the new Operating System
    - Marc Andresen, 1995

Software as a Service

- Bring a complete software solution to the consumer with zero setup in a browser
- Examples:
  - Web-based email (gmail, Yahoo! Mail)
  - Internet Search Portals (Google, Yahoo!, Bing)
  - Project Management (Atlasian JIRA)
  - Office Productivity (MS Office 365, Google Docs)
  - Document Signing (Docusign)
  - Customer Relations Management (Salesforce)
  - Tax Services (TurboTax, H&R Block, TaxAct)
  - Remote education (Coursera ☺)

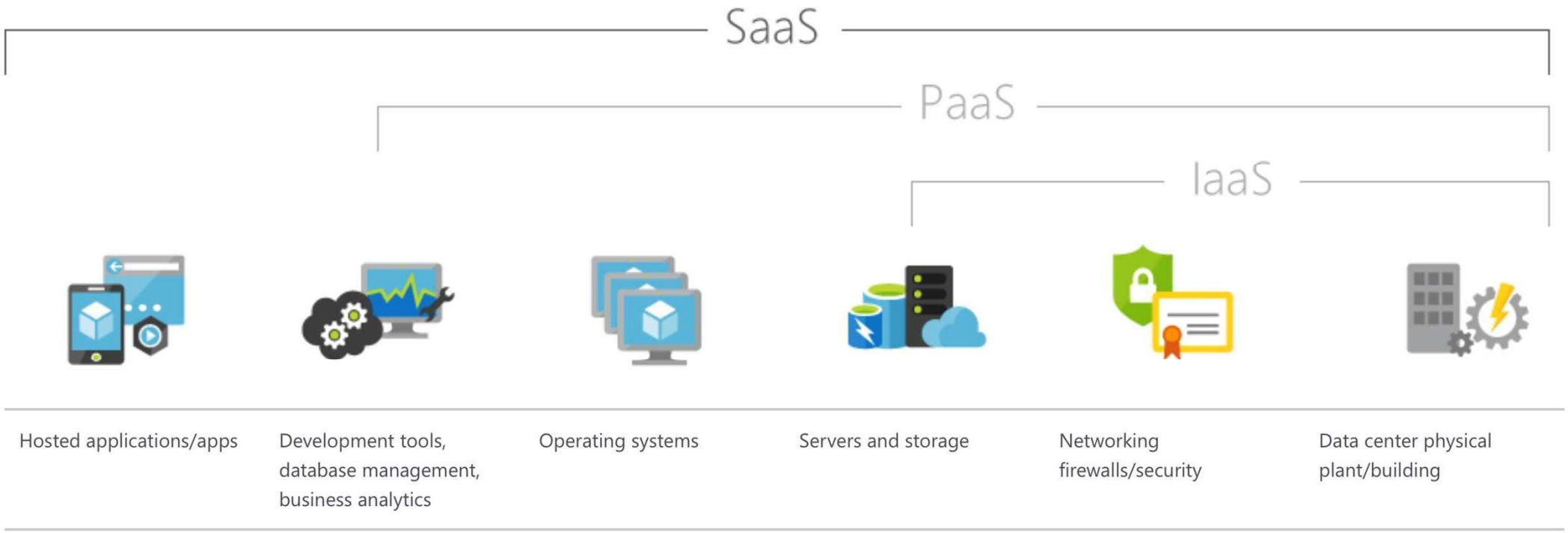
Software as a Service

- Multitenant Architecture
  - Same Software for all customers
  - Data is specific for individual user
- Client-side software runs in users' browser
- Server-side software runs in the cloud
  - Many SaaS solutions themselves build on IaaS and PaaS services
- Customer's data may be stored locally, in the cloud or both locally and in the cloud
- Because SaaS applications cannot access a company's internal systems, many SaaS solutions allow API access to further customize their offerings
  - HTTP, REST, SOAP

Advantages and Disadvantages

- Advantages:
  - No need to install software, licensing, maintenance and support
  - Flexible Payments
  - Scalable Usage
  - Automatic Updates
  - Access anywhere
- Disadvantages
  - You lose control
    - Might not be such a bad thing, see above advantages
  - No access to source code (as opposed to Open Source)
  - Provider service disruptions will impact you

SaaS in Perspective



** Image courtesy of Microsoft Azure*

CLOUD COMPUTING APPLICATIONS

PAAS Providers: SalesForce

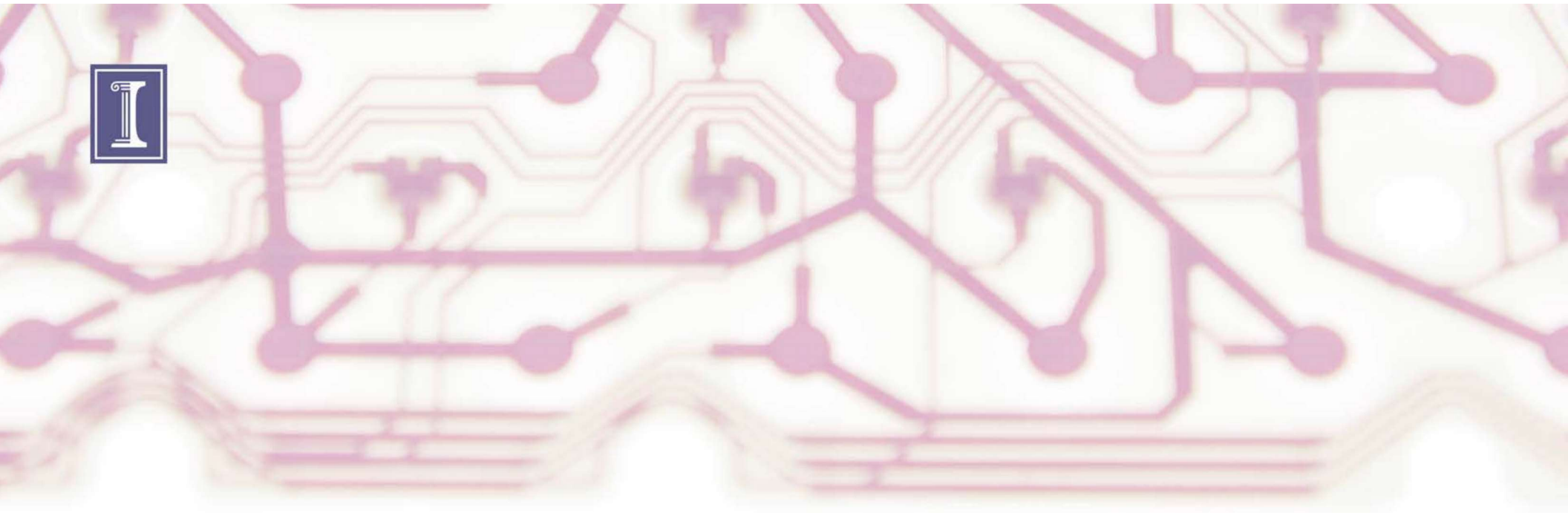
Prof. Roy Campbell

Main CRM Services (Mainly SaaS, PaaS)

- Sales Cloud, Service Cloud, Marketing Cloud, Analytics Cloud, Community Cloud, Lightening
- Social Software in the Workplace (Microsoft, IBVM, Jive, Salesforce)
- Find out about each other personally or professionally
- Mine their networks of contacts and acquaintances for advice, references and referrals
- Form teams, communities or informal groups
- Collaborate on the same work objects

Main CRM Services (Mainly SaaS, PaaS)

- Discuss and comment on their work
- Organize work from an individual or group perspective
- Identify relevant information
- Discover other people with common interests
- Alert users to information or events that might be relevant to them
- Learn from others' expertise
- Wave, Saleforce Analytics Cloud



CLOUD COMPUTING APPLICATIONS

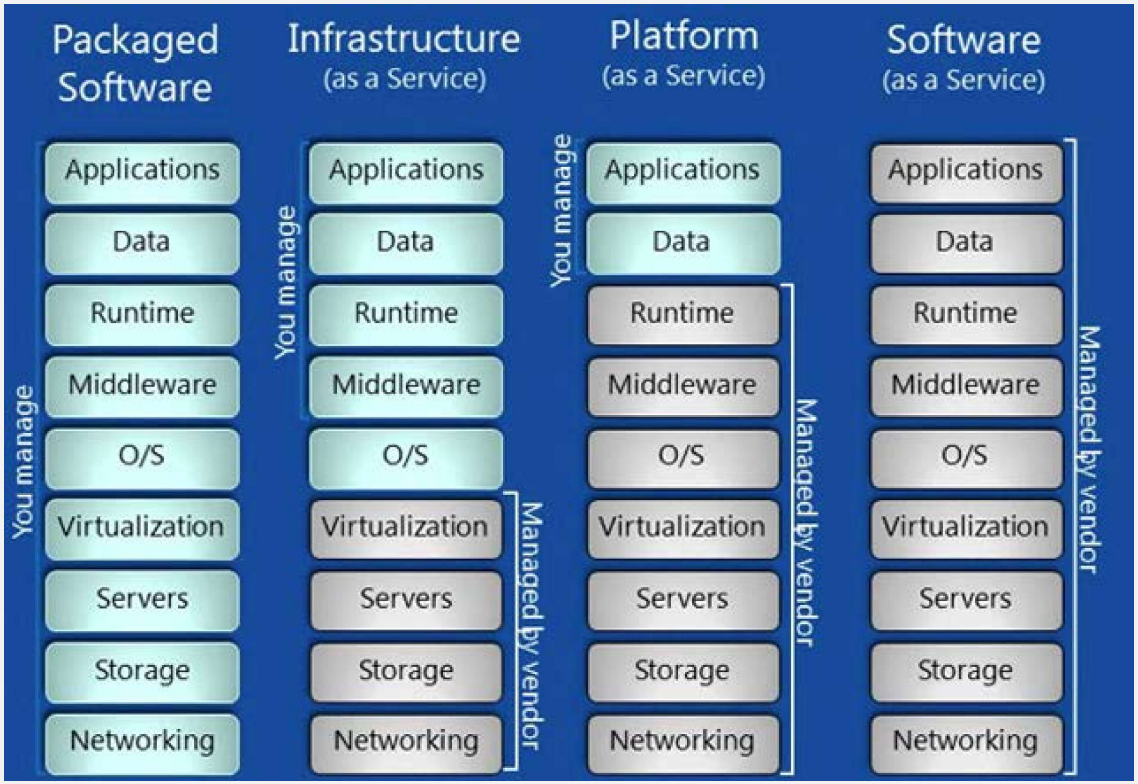
CLOUD SERVICES

Roy Campbell & Reza Farivar

Objective

- Compare IaaS, PaaS, and SaaS
- Look at what services major Cloud companies provide and how they provide them

IaaS, PaaS, and SaaS Comparison



Cloud Fundamentals

- Infrastructure as a Service (IaaS): basic compute and storage resources
  - On-demand servers
  - Amazon EC2, VMWare, vCloud
- Platform as a Service (PaaS): Cloud application infrastructure
  - On-demand application-hosting environment
  - For example, Google AppEngine, Salesforce.com, Windows Azure, Amazon
- Software as a Service (SaaS): Cloud applications
  - On-demand applications
  - For example, GMail, Microsoft Office Web Companions

Platform as a Service (PaaS)

- PaaS is a Cloud computing service that offers a platform for users to run applications on the Cloud

- PaaS is a level above IaaS because unlike IaaS, PaaS does not require users to develop their own operating system environment

Platform as a Service (PaaS)

- Middle ground between SaaS and IaaS
- Development platform
  - Customers use it to develop applications that benefit from the scalability of the Cloud without fully developing their own solution using an IaaS provider
- Offers an application development platform that will automatically scale with demand

The Benefits of the Cloud

The Cloud is about cheap, on-demand capacity

✓ = Managed for You	Standalone Servers	IaaS	PaaS	SaaS
Applications	✗	✗	✗	✓
Runtimes	✗	✗	✓	✓
Database	✗	✗	✓	✓
Operating system	✗	✗	✓	✓
Virtualization	✗	✓	✓	✓
Server	✗	✓	✓	✓
Storage	✗	✓	✓	✓
Networking	✗	✓	✓	✓

Platform as a Service (PaaS)

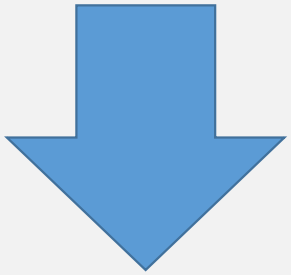
Official definition of PaaS from NIST standard

“The capability provided to the consumer is to deploy, onto the Cloud infrastructure, consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying Cloud infrastructure, including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.”

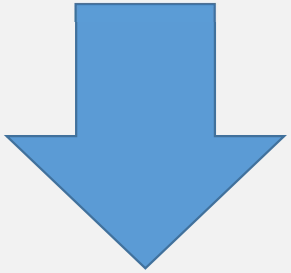
Example: Google

PaaS

Runtime environment, database, development



Google App Engine



Amazon AWS, EC2

Example: Windows Azure

- PaaS
  - Application platform in the Cloud
- Provides
  - *Compute*
    - Web, worker, and VM role
  - *Storage*
    - Blob, Table, Queue, and Azure SQL server
  - *Application fabric*
    - Service bus, access control
    - Future: cache, integration, and composite

More Cost Effective

- PaaS can be better for costs than IaaS, because systems are optimized to run applications efficiently

- IaaS may only provide hardware, and thus, clients must be in charge of load balancing and networking

Multi-tenancy

- PaaS is better suited for **multi-tenancy** because the PaaS provider optimizes its infrastructure for use by many providers

- Multi-tenancy means that many users may share the same physical computer and database

Multi-tenancy

- PaaS is better suited for multi-tenancy than an IaaS because an IaaS may (1) provide each user with his own virtual machine and (2) create a clear separation of resources

- However, in a PaaS, users may share the same machine, database, etc.

Vendor Lock-in

- PaaS may lock in applications by requiring users to develop apps using proprietary interfaces and languages
- This means that it may be difficult for users to go to another vendor to host their app
- Businesses may risk their future on the dependability of the PaaS

Development Tools

- Often, a PaaS will offer browser-based development tools
- In this way, developers can create their own applications online
- Ease of deployment: the platform takes care of the scaling for you

Principles of Software Development

- As a developer, your objective is to create an application in the quickest, most effective way possible
- You should not create applications using convoluted methods that may take a long time to complete
- The user only sees the end product, not the development process

PaaS vs. IaaS

- When you use the Cloud, remember that your decisions have long-term consequences
- If you choose to use a PaaS and get your application vendor locked in, then your business may fail if the PaaS greatly increases the vendor's prices
- You will not be able to move to another Cloud since your app cannot be easily migrated to somewhere else

PaaS vs. IaaS

- An app that is used to fulfill a temporary need may be handled by a PaaS solution
- An app that needs to be deployed quickly may be faster developed by a PaaS
- If your software team is small, it may be better to develop a PaaS and let the PaaS provider handle the OS and networking for your team

CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Introduction

Prof. Reza Farivar

Cloud Computing Glue

- Cloud Computing is all about running your compute tasks and storing your data on other computers
- At the core, it's about communication between computing sources
  - Your client + some server on the cloud, through internet
- Have we seen this problem before?
  - Inter-process communication
  - Communication on a local network
  - Communication on internet

Overview

1. Intro
2. Communication
3. Internet Protocol, HTTP and RTC on HTTP
4. Service Oriented Architecture and SOAP
5. RESTful Architecture
6. Asynchronous RPC, WebSocket
7. HTTP2 Push, Streaming Video
8. JSON, comparison with XML
9. RPC semantics and implementation
10. Protocol Buffers and Thrift

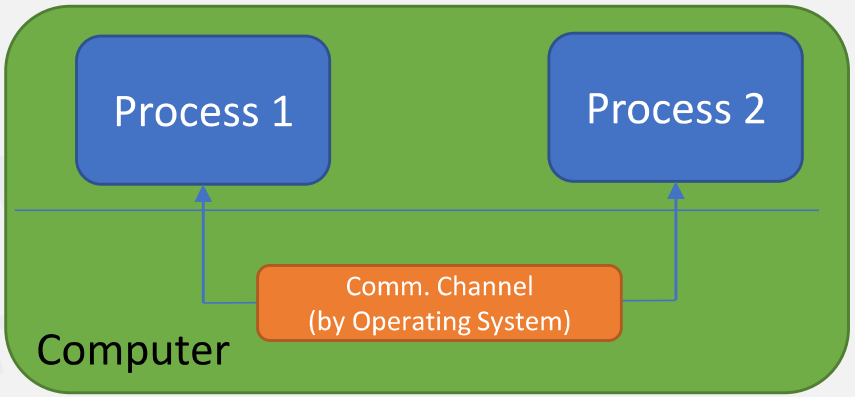
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Communication

Prof. Reza Farivar

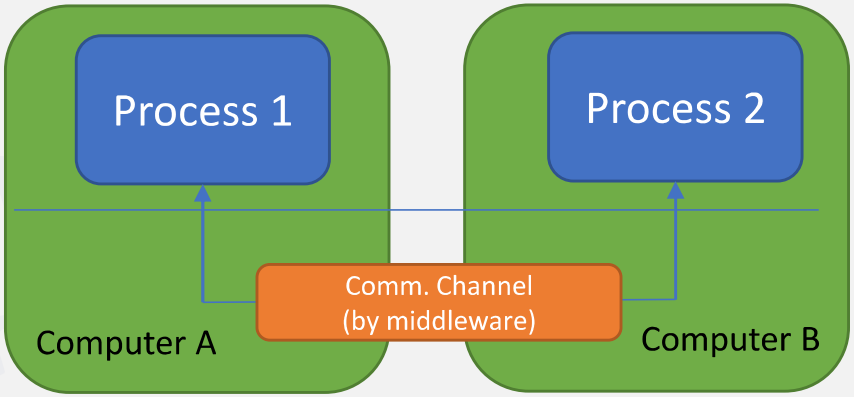
Communication in a single machine

- Communication Channel provided by the Operating System
- Shared memory block
- Shared File System
- Signal
- POSIX Socket, aka. Berkeley Socket
  - Port numbers
  - *SOCK_STREAM (compare to TCP)*
  - *SOCK_DGRAM (compare to UDP)*
- Remote Method Invocation (RMI)
  - Method invocations between objects *in different processes (processes may be on the same or different host)*
  - *From one JVM to another*
- Message Queue
- Message Passing
  - Unix Pipe
  - *Actor Model*
  - *Pi Calculus*

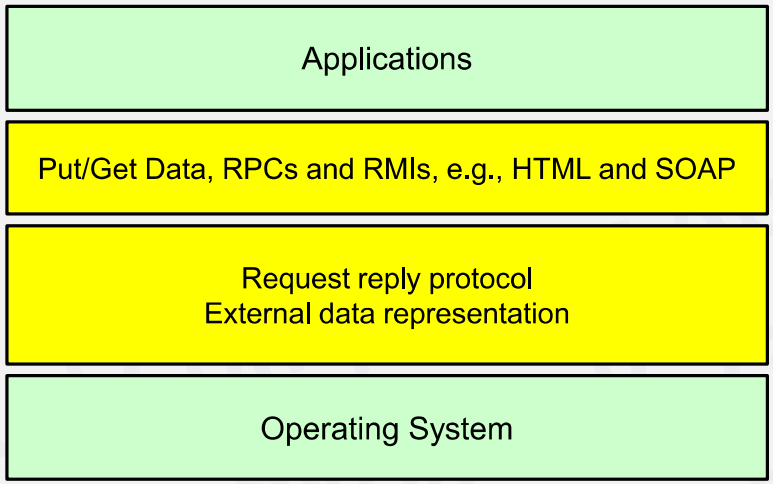


Middleware Layer Definition

- *Software that provides services to applications beyond those generally available at the operating system*
- Middleware implements functionalities that are common across many different applications
  - No need to reinvent the wheel (e.g., message parsing) every time you need to do something
- A Middleware Layer can provide the same abstractions to distributed applications!
  - Building distributed systems while maintaining our code is not very different from a single-node program



Middleware Layers



Middleware layers

Provide support to the application

Run at all servers at user level

RMI = Remote Method Invocation
CORBA = Common Object Request Brokerage Architecture
SOAP = Simple Object Access Protocol

Communication in a local network

- Scientific Computing
  - Message Passing Interface (MPI)
  - Simple model: Send() and Receive()
  - No native support for fault tolerance
  - Programming interface is complicated
    - Race, deadlock, etc.
- Business Sector
  - Remote Procedure Calls
    - RPC Semantics (behavior in presence of network failures)
    - RPC Implementation
  - Remote Method Invocation (RMI)
    - Between two JVMs on a network

Communication in Big Data Deployments

- Need scaling from the start
  - Sometimes many 10s of thousands of nodes on the network
  - Ad hoc solutions do not work
- RPC Frameworks
  - Google Protocol Buffer
    - You define the functions that will be called remotely
    - Then Compile
    - The system automatically generates interfaces functioning as communication stubs in your choice of programming language
      - Many languages are supported: C++. C#, Objective C, Java, Python, etc.
    - You import the generated header / code files into your project
    - At run-time, your program just calls the function locally. The auto-generated code takes care of serialization and marshalling of the function parameters, making the network calls (handling any network errors), and transferring the function call in the target system.
  - Apache Thrift
    - Apache HDFS, Hadoop, Spark, Storm, etc. extensively use Thrift
- Consistency
  - Paxos
  - Zookeeper

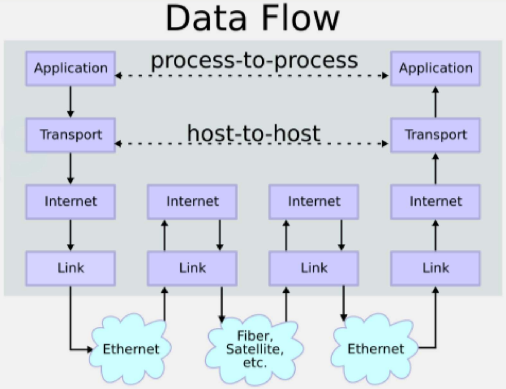
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Internet Protocol, HTTP and RPC on HTTP

Prof. Reza Farivar

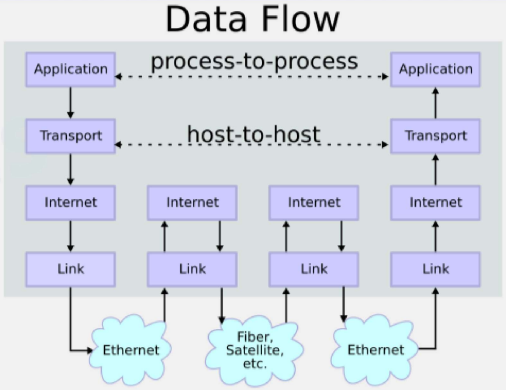
Internet Protocol Stack: Link and Internet Layers

- The second level, “internet level”, is where internet switches and routers operate
  - *Gets your message from one machine to another*
  - Responsible for addressing **host interfaces**
  - Encapsulating data into datagrams (including fragmentation and reassembly)
  - Routing datagrams across one or more IP networks
  - *IP address, IPV4, IPV6*
- EE and Network engineers live at the bottom level, the Link level
  - Out of scope our discussion



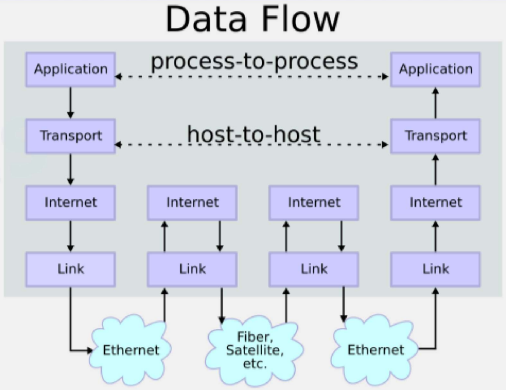
Internet Protocol Stack: Transport Layer

- The transport level handles packetizing of data
- Gets your message from a process in one machine to another process in another machine
- Typically implemented in the Operating System
- Provide port numbers (similar to Unix sockets)
- TCP
  - Keeps track of data segments, retransmission, acknowledgement
    - Handle network congestion
    - Traffic load balancing
    - Unpredictable network behavior
    - Lost, duplicated, or delivered out of order IP packets
  - Guarantees that all bytes received will be identical and in the same order as those sent
- UDP
  - "User Datagram Protocol", aka. "Unreliable Datagram Protocol"
  - Very simple



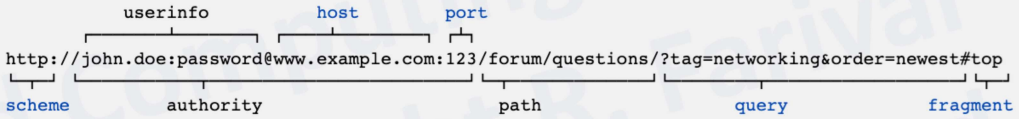
Internet Protocol Stack: Application Layer

- Application level handles “what to send”
  - HTTP, HTTPS
    - RESTful APIs
  - FTP
  - WebSocket
  - SMTP
  - IMAP
  - SSH
  - DHCP
  - DNS
  - Bit Torrent



HTTP Protocol

- Originally targeted static web pages: *Client Requests, Server Responds, Connection is closed*
- Works on top of TCP for reliable transport
- Client (user agent) can be a web browser, or any other software
- HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes http and https



- In HTTP/1.0 a separate connection to the same server is made for every resource request
  - Establishment of TCP connections has overhead
- HTTP/1.1 can reuse a connection multiple times to download images, scripts, stylesheets, etc. after the page has been delivered
  - Persistent Sessions
  - Lower latency

HTTP Message Format

- Request “verbs”
  - GET: retrieve data
  - POST: server should accept the call parameter as a new value for the resource specified in the URL
  - PUT: server should store the enclose entity under the supplied URL
  - DELETE: server should delete the specified resource by the URL
  - PATCH: server should apply partial modification to the resource
  - …
- Request message
  - a request line (e.g., GET /dataset/inventory.htm HTTP/1.1, which requests /dataset/inventory.htm resource from the server)
  - request header fields
  - an empty line
  - an optional message body
- Response message
  - a status line which includes the status code and reason message (e.g., HTTP/1.1 200 OK, which indicates that the client's request succeeded)
    - *Informational 1XX | Successful 2XX | Redirection 3XX | Client Error 4XX | Server Error 5XX*
  - response header fields (e.g., Content-Type: text/html)
  - an empty line
  - an optional message body

Client Request

```
GET / HTTP/1.1
Host: www.example.com
```

Session

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

RPC on HTTP

- Remote Procedure Calls built on HTTP
  - For many types of RPC, the client/server conversation model of HTTP works just fine
  - Just replace the HTML markup with an XML or JSON representation of the data
  - XML-RPC
  - JSON-RPC
    - E.g. Bitcoin server
    - Commands encoded as JSON, sent over HTTP

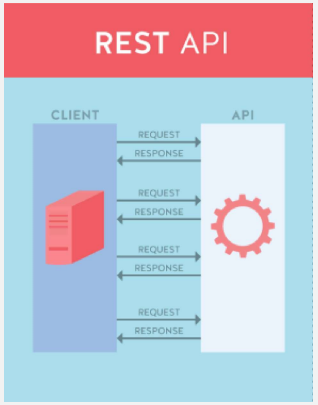
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: RESTful Architecture

Prof. Reza Farivar

Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web
- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification
- The motivation for REST was to capture those characteristics of the Web that made the Web successful
  - URI-addressable resources
  - HTTP
  - Make a request – receive response – display response
- A collection of network architecture principles that outline how resources are defined and addressed
  - Based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data
  - Request/response between client and server, like a conversation
  - Something is requested, something is done, and then something is sent in return



RESTful API

- Uses HTTP verbs: GET, POST, PUT, PATCH, DELETE
  - Exploits the use of the HTTP beyond HTTP POST and HTTP GET
    - HTTP PUT and DELETE are not even supported in HTML
  - GET is safe (does not change state)
  - GET, PUT and DELETE are idempotent (you can execute them more than once and get the same state change result)
  - Example request:
    - `curl -X POST https://api.github.com/user/repos`
    - Response:

```
{
  "message": "Requires authentication",
  "documentation_url": "https://developer.github.com/v3/repos/#create"
}
```

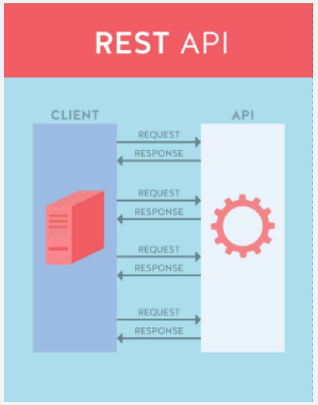


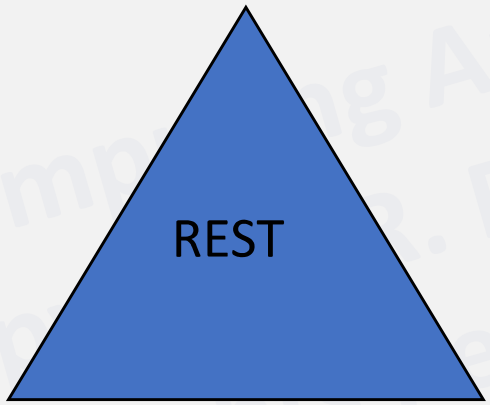
Image from https://nordicapis.com/rest-vs-streaming-apis-how-they-differ/

REST – Not a Standard

- There is no “official standard”, REST is an architectural style
  - JSR 311: JAX-RS: The Java™ API for RESTful Web Services
- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc. (resource representations)
  - Text/xml, text/html, image/gif, image/jpeg, etc. (resource types, MIME types)
- Huge adoption for “Web mashup” applications, operations on When entities
- Many cloud SaaS and PaaS services
- LinkedIn, Twitter,

Main Concepts

Nouns (resources)
unconstrained
i.e., http://example.com/employees/12345



Verbs
constrained
i.e., GET

Representations
constrained
i.e., XML

Resources

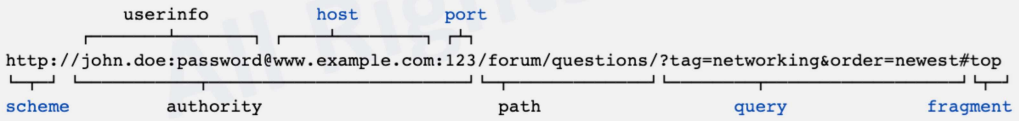
- The key abstraction of information in REST is a resource

- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g., "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g., a person), and so on

- Represented with a global identifier (URI in HTTP)
  - http://www.boeing.com/aircraft/747

Naming Resources

- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data



Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE
- HTTP PATCH

HTTP GET

- How clients ask for the information they seek
- Issuing a GET request transfers the data from the server to the client in some representation

- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP POST, HTTP PUT

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties