Cristina Nita-Rotaru

7610: Distributed Systems

Edge Computing

1: Analysis of the github network partition incident

https://blog.github.com/2018-10-30-oct21-post-incident-analysis/

https://www.youtube.com/watch?v=YzhkTu0lSvk

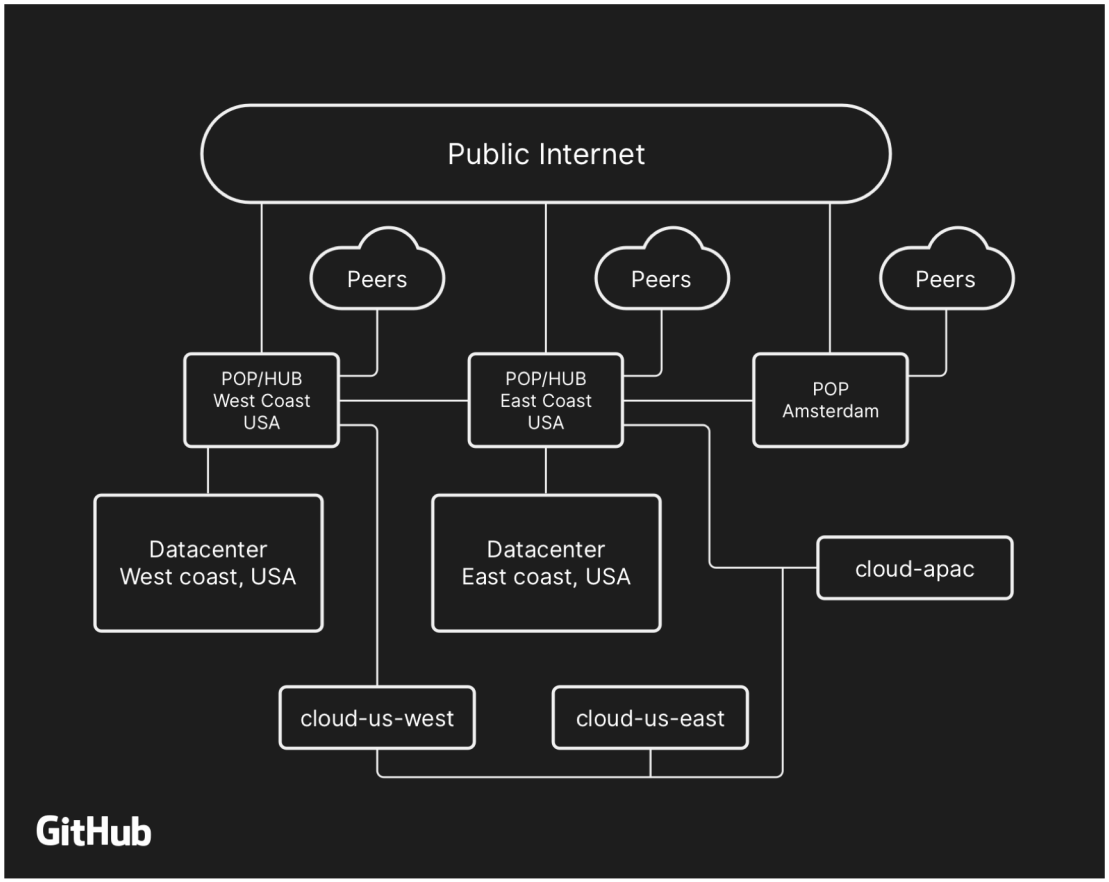
Anatomy of a github incident (Oct 21 2018)

- Result: degraded service for 24 hours and 11 minutes.
- Multiple internal systems were affected which resulted in displaying of information that was out of date and inconsistent.
- No user data was lost; but required manual reconciliation for a few seconds of database writes that took hours
- For the majority of the incident, GitHub was also unable to serve webhook events or build and publish GitHub Pages sites.

WHAT HAPPENED?

Github data centers

- Points of presence (POPs), do not store customer data, rather they are focused on internet and backbone connectivity as well as direct connect and private network interfaces to AWS
- Data centers (DC), East Coast and West store customer data



How does github store data

- MySQL stores metadata: Issues, Pull Requests, comments, organizations, notifications, etc.
  - Git repository data does not need MySQL
  - GitHub's service does. Authentication, API, website
- Use MySQL master-replicas setup, where the master is the single writer, and replicas are used for read.
  - Placing a review, creating a repository, adding a collaborator, require write access to the backend database.
- Several MySQL clusters (hundreds of GB to nearly 5TB), each with up to dozens of read replicas per cluster
  - Different data across different parts of the application is stored on various clusters through sharding.

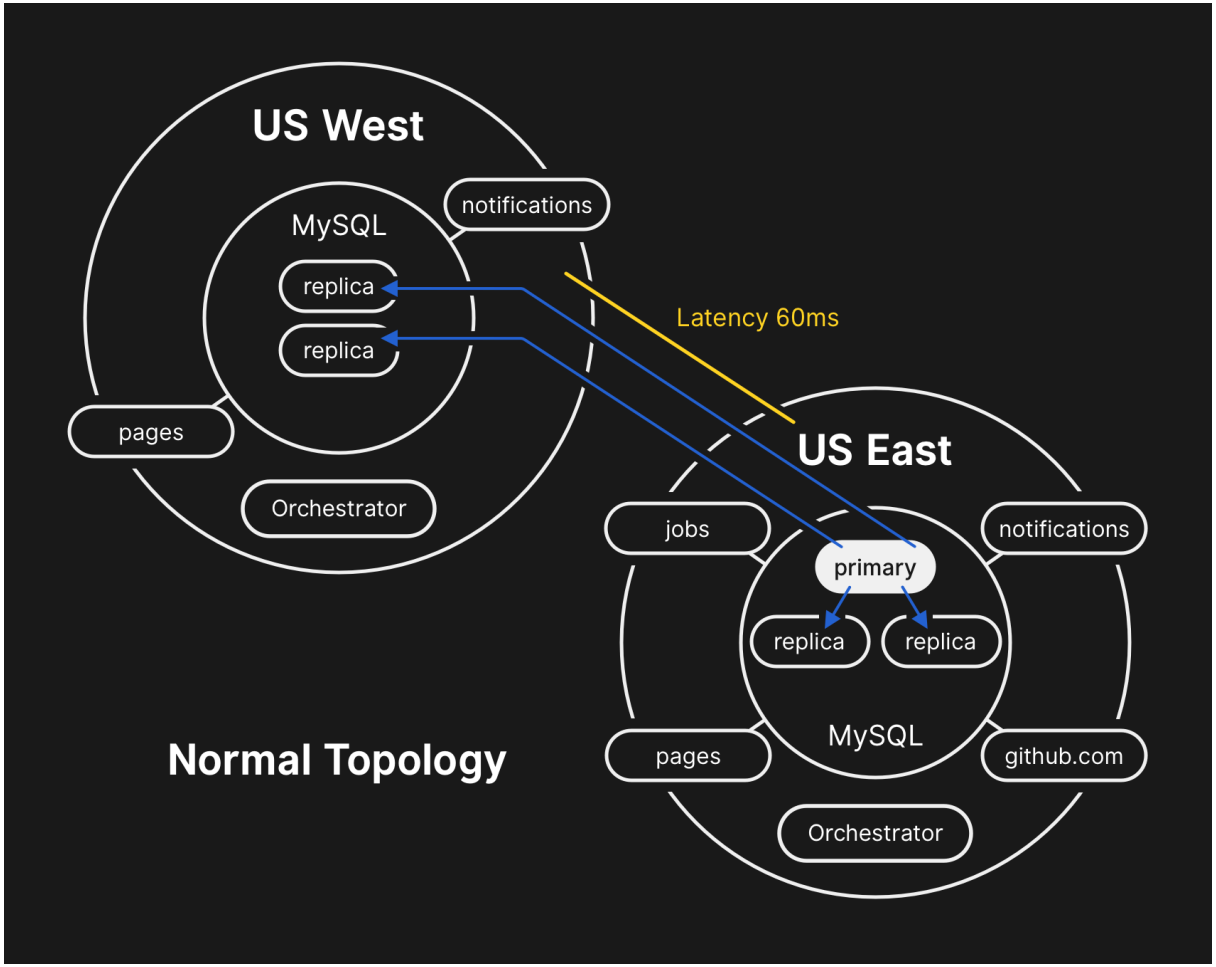
Automated master failover

- Master failures are automatically detected and recovered < 30 seconds
- Orchestrator: Open source MySQL replication management and high availability solution uses RAFT
  - observes MySQL replication topologies
  - auto-detects topology layout and changes
  - understands replication rules across configurations and versions
  - detects failure scenarios and recovers from master and intermediate master failures.
- Semi-sync replication environment: one or more designated replicas are guaranteed to be most up-to-date.

Orchestrator and Apps requirements

- The performance of the replication topology created by Orchestrator might not be aligned with the application-level expectations.

APPS WILL NOT WORK UNDER CERTAIN LATENCY



Normal topology DC East is primary

Orchestrator and RAFT

- Orchestrator uses RAFT for its own availability
- Orchestrator nodes form a RAFT cluster
  - Three nodes, one in each DC and one is a cloud
- Each nodes has its own dedicated MySQL backend
- All nodes probe the topologies
- All nodes ran failures detection
- Only the leader runs failure recovery
- Active datacenter – they want the leader to be at the active datacenter so operations are faster
- Leader step down – if a leader can not ran its own self check, he will step-down , leader deselection; 5 seconds to step-down

Cascading events from a 43 seconds network partition

- At 22:52 UTC on October 21, routine maintenance work to replace failing 100G optical equipment resulted in the loss of connectivity between POP US East Coast and DC East Coast, connectivity between these locations was restored in 43 seconds
- What happened next?
-

events

- Raft consensus triggered a leader step down. The US West Coast DC and US East Coast public cloud Orchestrator nodes figured out the master from East Coast Orchestrator was done, West Coast became leader and all the writes got routed through the US West Coast DC.
- Orchestrator proceeded to organize the US West Coast database cluster topologies.
- Meanwhile, the East Coast DC took some time to figure out that he should not accepting writes, so the US East Coast data center contained a brief period of writes that had not been replicated to the US West Coast facility.
- When connectivity was restored, our application tier immediately began directing write traffic to the new primaries in the West Coast site.
- TWO CONSEQUNCES:
  - Because the database clusters in both data centers now contained writes that were not present in the other data center, they could not switch the primary back over to the US East Coast data center safely.
  - Replication topologies only from West Coast incurred latencies that some applications could not tolerate

investigating the current state

- West Coast database cluster had ingested writes from our application tier for nearly 40 minutes.
- Several seconds of writes that existed in the East Coast cluster that had not been replicated to the West Coast and prevented replication of new writes back to the East Coast.
- Apps that must write to West Coast DC can not keep up
- **Loosing the updates from the West Coast data center, not an option**

plan

- Manually restore East Coast from checkpoint
- Apply all updates from West Coast after the checkpoint
- Manually apply/reconciliate the East Coast updates that were out of sync
- Reconfigure the replication topologies
- However: checkpoints are stored in a public cloud, recovering them over the network took a few hours

Catching up …

- Once all database primaries established in US East Coast again.
  - Site becoming far more responsive as writes were now directed to a database server that was co-located in the same physical data center as our application tier.
- However, many database read replicas were multiple hours delayed behind the primary.
  - resulted in users seeing inconsistent data as they interacted with github services.
  - Spread the read load across a large pool of read replicas to increase chance of hitting a more updates read replica
- Once the replicas were in sync, failover to the original topology
- **Still need to resolve the East Coast updates data inconsistencies cause by the writes during the partition**

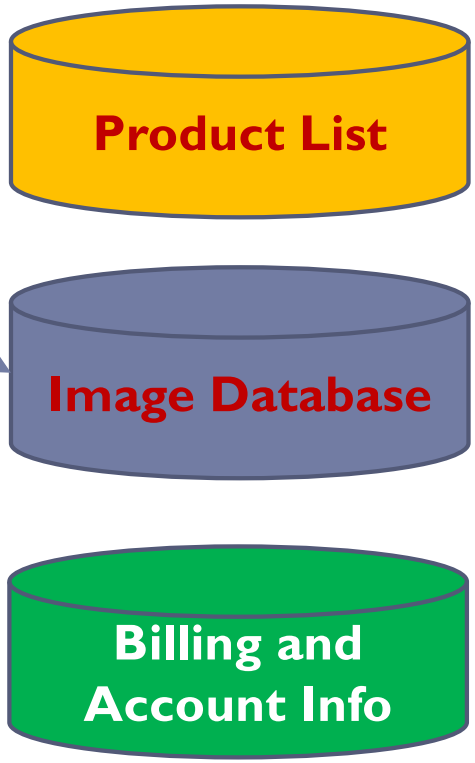
2: Cloud computing evolution

Building web applications (2005)

Computers were mostly desktops

Internet routing was pretty static, except for load balancing

Web Server built the page



Databases held the real product inventory

New Approach (2008)

Devices became more mobile

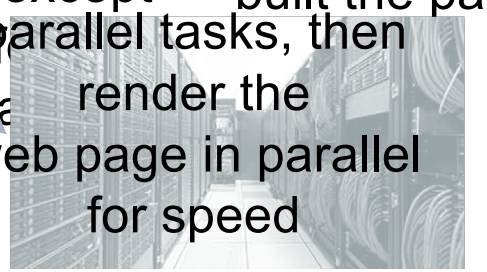
Computers were mostly desktops

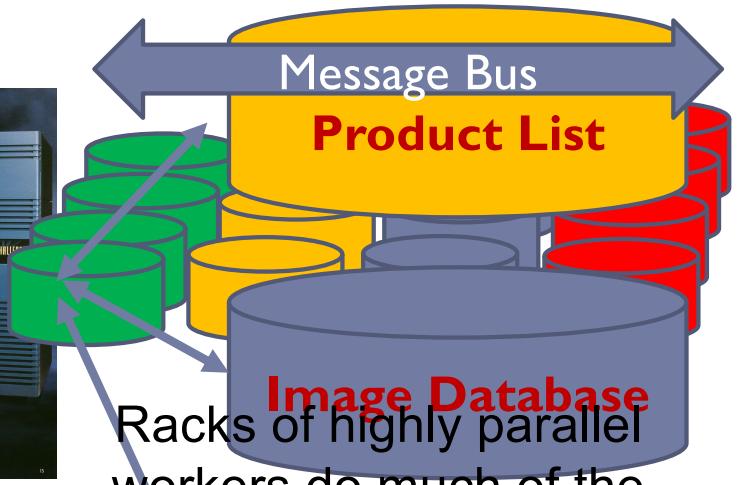
Routed to nearest datacenter, one of many

Internet routing was pretty static, except for

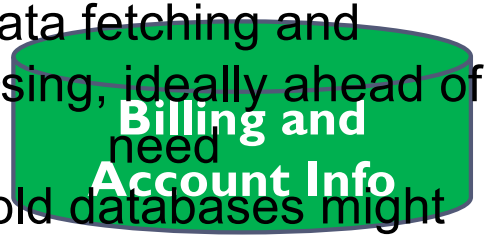
Massive racks of inexpensive web server machines dispatch parallel tasks, then render the web page in parallel for speed

Web Server built the page





Racks of highly parallel workers do much of the data fetching and processing, ideally ahead of need



The old databases might still be present in this layer, or may have been split into lots of services

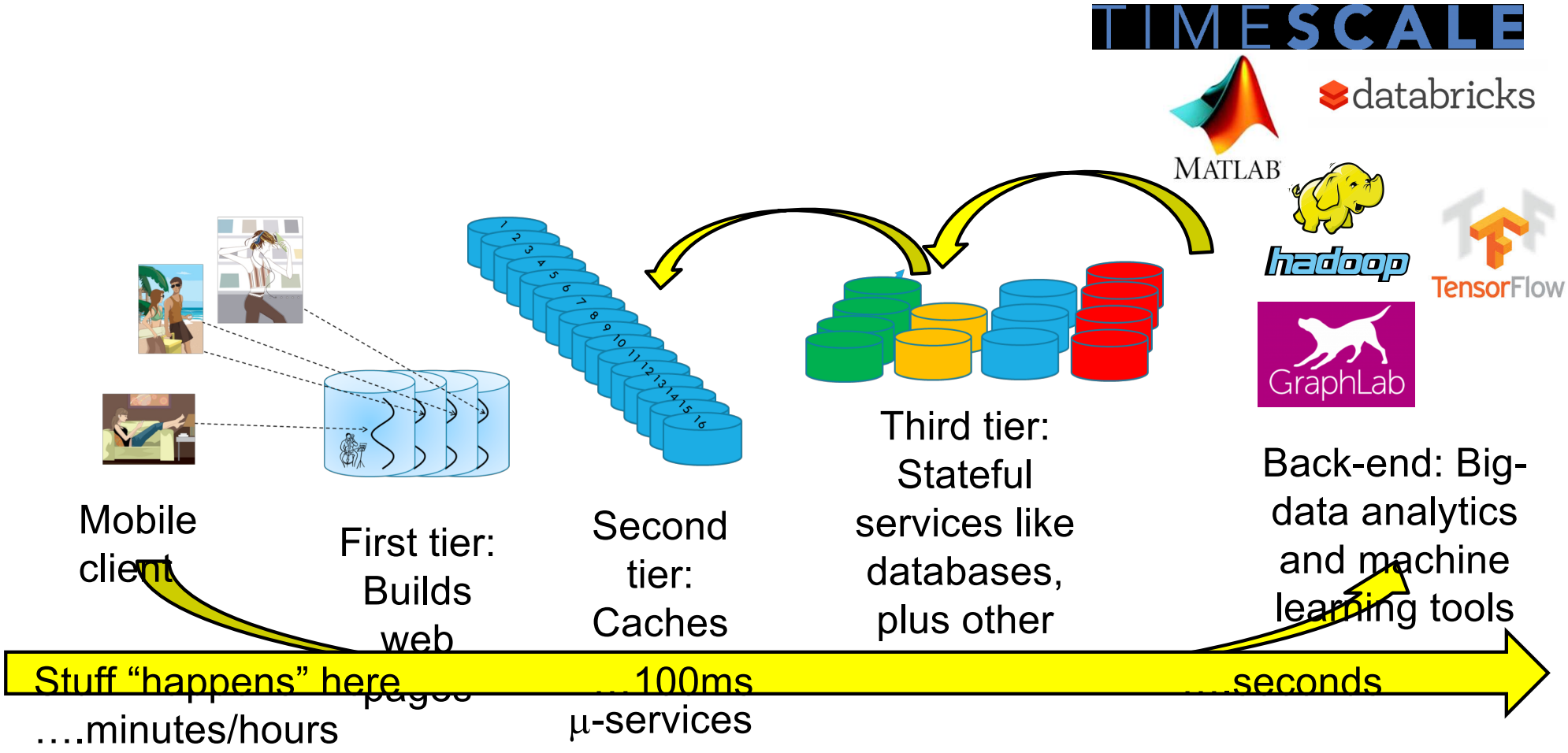
Databases held the real product inventory

Today's Cloud

- Tier one runs on very lightweight servers:
  - They use very small amounts of computer memory
  - They don't need a lot of compute power either
  - They have limited needs for storage, or network I/O

Tier two μ-Services specialize in various aspects of the content delivered to the end-user. They may run on somewhat "beefier" computers.

Cloud computing architecture



http://www.cs.cornell.edu/courses/cs5412/2018sp

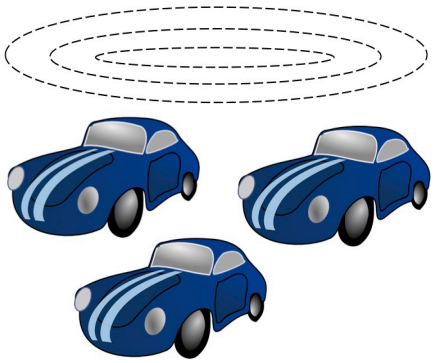
1: Emerging applications

Connected Cars Deployment

- General Motors:
  - Available in Cadillac CTS sedans since 2017
- Toyota:
  - Toyota and Lexus enabled with DSRC-based V2V communications in Japan since 2015
  - Announced plans to begin deployment of V2V and V2I technology in the U.S. market starting in 2021
- Volkswagen:
  - Announced in 2017 that will have DSRC in Europe beginning in 2019

Connected cars

- DSRC enables access to the Internet (and the "cloud")
- Sensors that allows them
  - Sense the physical environment around them
  - Interact with other vehicles
  - Interact with road infrastructure
- Underlying services
  - Driver assistance systems (ADAS)
  - Human-machine interface (HMI)
  - Infotainment support
  - Access to connectivity, computing, and the cloud enable a variety of applications



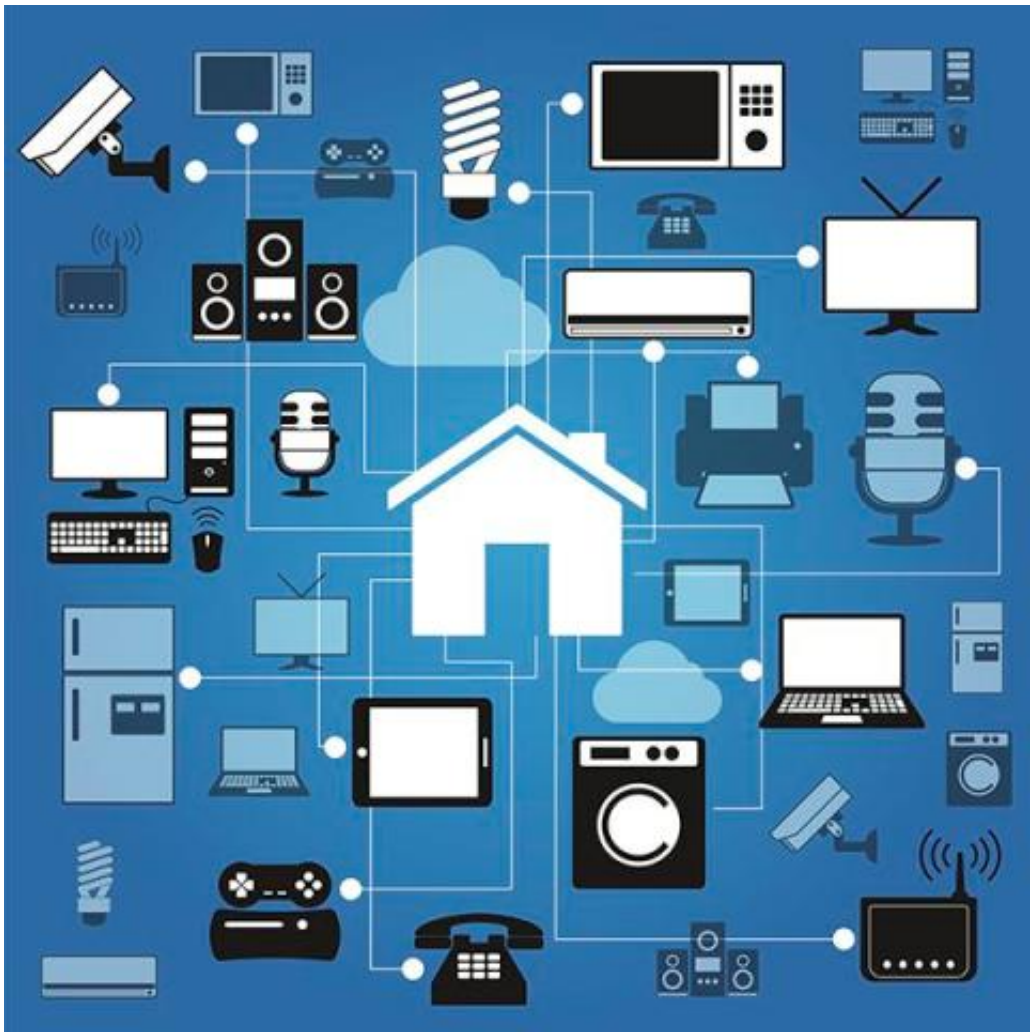
Applications

- Safety applications
  - Traffic and congestion control
  - Collision avoidance
  - Intersection management
  - Assisted-turn
  - Collaborative adaptive cruise control
- Entertainment
- Shopping

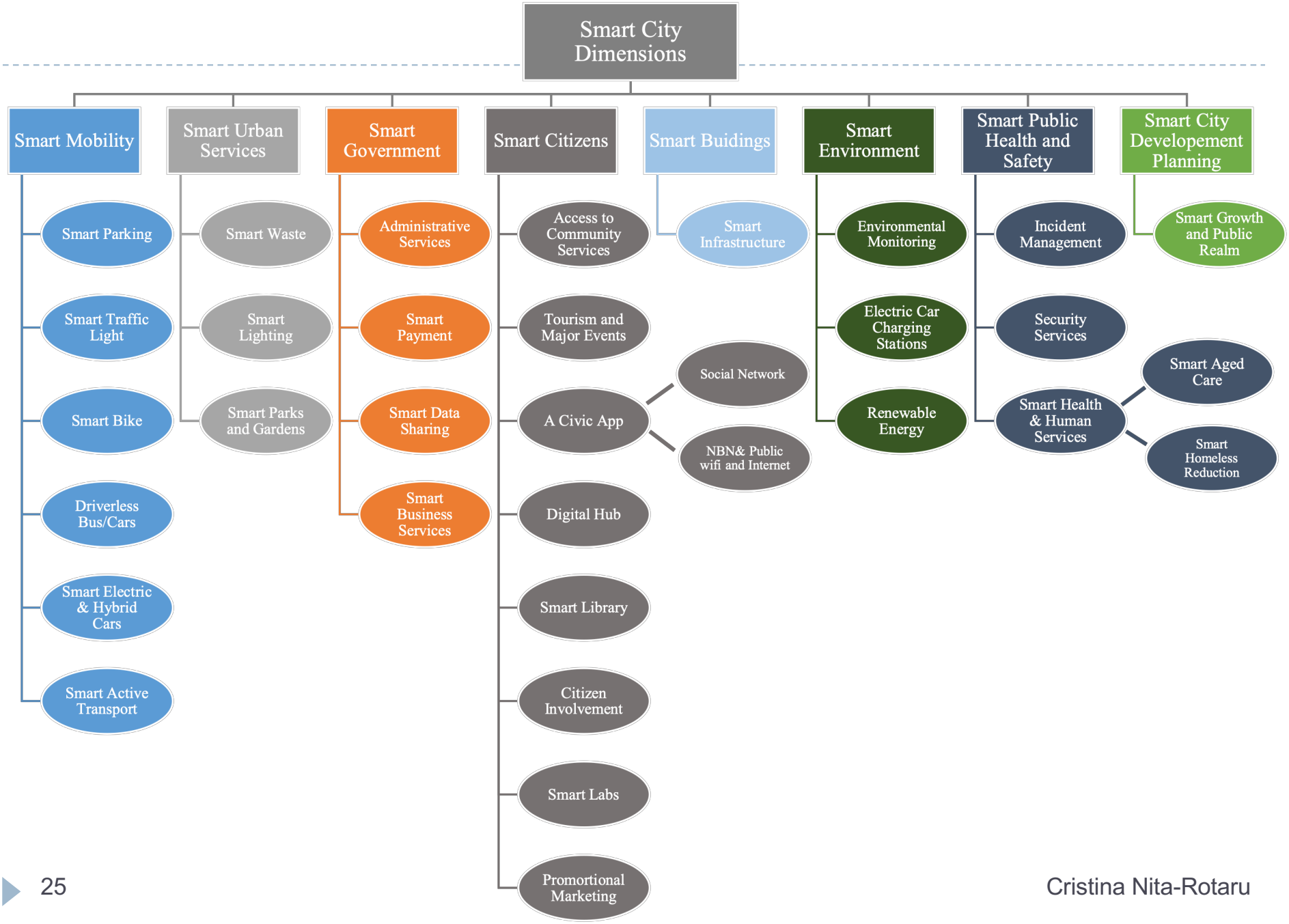
A group of self-driving cars successfully formed a platoon
https://www.volpe.dot.gov/

Smart Homes

- Enables various devices around the home to be connected to the internet and controlled via smartphone
- Control:
  - Lighting, TV functions, thermostat, appliances, door locks, door bell, garage, sprinklers, automotive…
- Alexa or Google Home:
  - Weather, Shopping, Music, News, To do lists, Jokes, Timers, Math, Definitions, Calendar…

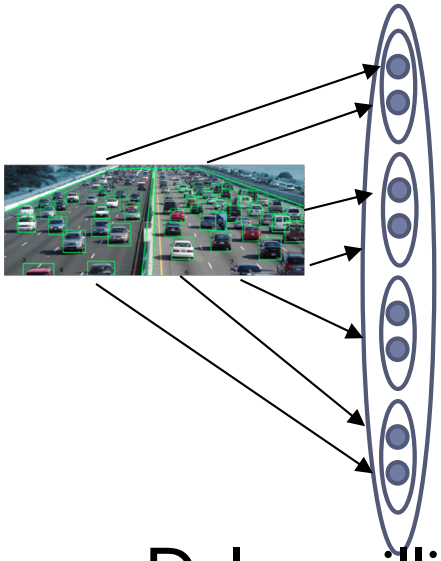


Smart Cities



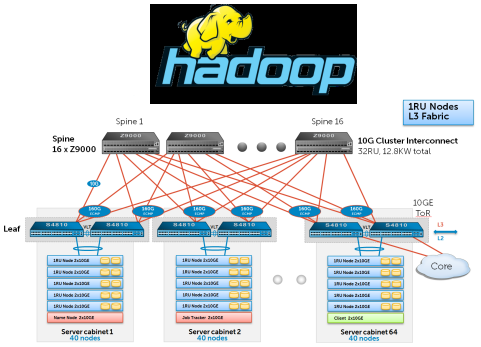
Today: a very "long" pipeline

- Data acquisition…. Global File System… Hadoop jobs



Delay: milliseconds…

Seconds….



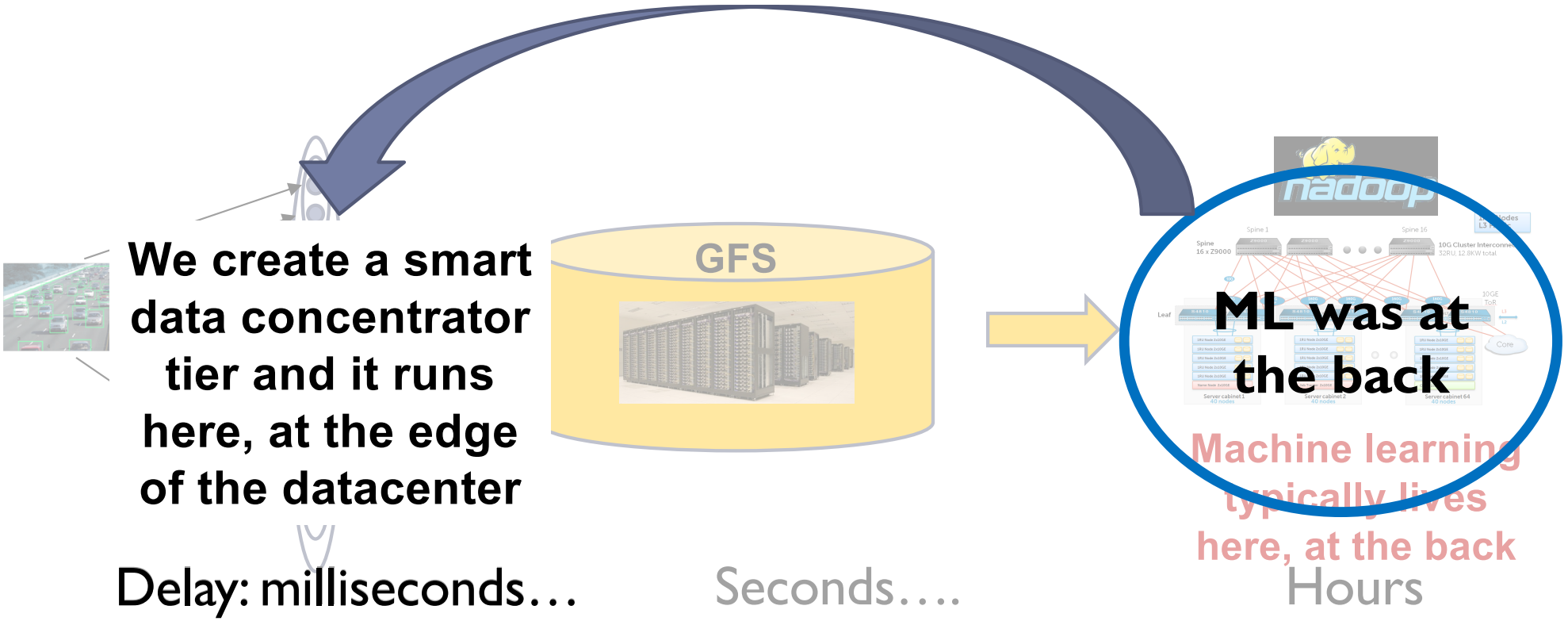
Machine learning typically lives here, at the back

Hours

http://www.cs.cornell.edu/courses/cs5412/2018sp

New: Move ML to the edge Of the cloud

- Data acquisition…. Global File System… Hadoop jobs



http://www.cs.cornell.edu/courses/cs5412/2018sp

In the future cloud, needs will differ

- Smart highways and cars, smart homes and cities, smart grid:
- Generate huge amounts of data, even if the sensors do basic preprocessing
- Will still need to be rapidly reactive, like today's cloud
- And will need to ***base those actions on the most up-to-date data.***

2: Edge computing, a new paradigm

NSF Workshop Report on Grand Challenges in Edge Computing

Edge computing

- Resources for **communication, computation, control and storage are placed at the Internet's edge** in close proximity to mobile devices or sensors
- These nodes are referred to as cloudlets, micro datacenters (mDC), or fog nodes
- Edge computing generalizes and extends the CDN concept by leveraging cloud computing infrastructure.
  - As with CDNs, the proximity of cloudlets to end users is crucial
  - Unlike CDN, instead of being limited to caching web content, a cloudlet can run arbitrary code just as in cloud computing

What is really an edge computing system?

- Combines computing and communication, and typically performs sensing and actuating as well.
- Multiple perspectives:
  - **Network location point of view**: edge refers to the part of the network outside the Core network, but could include Metro/Access networks.
  - **Data point of view**: edge computing is done at or close to where the *sensory data is generated*, and/or where the *users/consumers of the data are*.
  - **Control-decision point of view**: edge processing inherently involves some distributed computing.

Edge devices

- Edge device could be a smartphone sitting between body sensors and the cloud, or an mDC or cloudlet between a mobile device and the cloud.
- Data edge: end device not only consumes data but also produces data.
- Network edge: devices not only request services and information from the cloud but also handle computing tasks—including processing, storage, caching, and load balancing—on data sent to and from the cloud.

Applications

- Characteristics of applications that can benefit from edge computing:
  - Are time-critical
  - Require a very low computing and communication latency
  - Ran in an environment with limited bandwidth, limited computing power, possibly lots of data to be processed or limited energy that powers the edge computing device.

Examples of applications

- Vehicular applications:
  - Generate and send safety warnings/alerts, suggest driving speeds, routes and other driving behaviors to improve travel efficiency or reduce fuel consumption and emission, and provides telemetry and in-vehicle entertainment services
- Smart city applications:
  - Monitor and manage buildings, infrastructures and governance processes, in addition to transportation and public safety
- Video surveillance apps for public safety:
  - Distributed cameras collect and process data for face/object recognition with real-time processing (based on e.g. video analytics)

Examples of applications

- Mobile and wearable applications:
  - Smartphones and wearable devices provide cognitive personal assistance information gathering and decision making
- Disaster-relief apps:
  - Mobile devices such as drones can aid search and rescue, or damage/infrastructure assessment.
- Bedside clinical apps:
  - Combine measurements from devices attached to the patient with contextual information (e.g., medical history, and diagnosis) to assist in fast medical intervention

Examples of applications

- Games:
  - Incorporate location information and require real-time response: examples include VR/AR technologies for sports and training.
- Smartgrid and microgrid applications:
  - Distributed monitoring, management and control of various renewable energy sources, currents, voltage, and demands can yield a more efficient and resilient system.
- Privacy-protection for cloud storage:
  - Clients shred a file into many pieces and spread them over multiple public cloud storage infrastructures.

Examples of applications

- CDN in the sky:
  - Imagery is coming from hi-res satellites which are bandwidth constrained, with sensitive info, restricted mobility, and limited power
- Camera intelligence:
  - Cameras data usually is not uploaded to the cloud because of privacy concerns or the cost of transferring the information. With edge computing, the data could be pushed to the many edge devices in a target area. They could search the data they receive and report the findings to the cloud, yielding the results much faster than using only cloud computing.

Challenges for applications

- Real-time processing and communications:
  - While most edge computing applications require this, it is particularly challenging to achieve real-time processing for video analytics, and activities analysis, due to limited computing and communication abilities of edge computing devices.
- Security and privacy:
  - On one hand, since edge is close to data and users, there is an opportunity for stronger security and privacy. On the other hand, the sheer number of heterogeneous devices which need to process data from and for nearby users makes it challenging to achieve those goals

Challenges for Applications

- Incentives and Monetization:
  - Edge computing services still needs better-developed business models, and incentive schemes to encourage users to adopt the edge computing apps.
- Adaptive application development:
  - Given the variety of the application scenarios, it is challenging to develop applications which can adapt to various environment and handle graceful degradation of performance/services.
- Tools for the development and testing of apps in edge computing

Challenges for Architectures: Cage-Level Security

- The paramount challenge toward computational resources out of the massive data centers under complete control of the operator is **how to guarantee the same level of security and safety as those under complete control of the operator.**
  - Is it possible to achieve the same security guarantees as a physical cage via only hardware and software support?
  - What are the implications of this level of security as the resource moves to entirely to the edge?
- Recent reports of DDoS attacks stemming from traffic cameras and other internet-connected things are a reminder of the vulnerability of anything accessible over the internet, but some of these things will be physically accessible to attackers.

Challenges for Architectures: Embracing Approximation

- Edge computing will involve high volume data.
- How can this data be reduced and how can processing be improved?
  - The more the data is compressed close to the source, the less data has to travel through the network, but the less useful the data may be for end applications. Need new data compression schemes.
- How can the usefulness of data be analytically modeled or represented?
- How can the timeliness of data be analytically modeled or represented as some data may have a lifetime or shelf life?

Challenges for Architectures: Embracing Approximation

- The answers are likely to be application-dependent.
- One possible approach to helping with efficient processing of high-volume data is self-descriptive data, also known as smart objects.
- Data processing at the edge is likely to introduce uncertainty in the data itself.
  - How can this uncertainty be expressed in a probabilistic manner? Probabilistic computing is not yet very mature, though there has been significant work in the past 15- 20 years on stream processing, which does best-effort processing on data as it flows through a system.

Challenges for Architectures: Theorem for Tradeoffs

- Brewer's "CAP Theorem" has been an important design principle for distributed systems.
- What are the fundamental properties and design principles about tradeoffs for edge computing?
- There may be a fundamental tradeoff between the following four factors, i.e. one might be able to achieve any three of them but not all four, or there may be pairs in direct opposition: (1) Mobility, (2) Latency, (3) Capability, and (4) Privacy.
- "CAP theorem for Edge Computing" ?

Challenges for Architectures: Data Provenance

- Provenance refers to how data is generated: the input sources, programs, users involved, and other factors.
  - Provenance has never been tackled at scale.
- Edge computing also introduces another aspect, how data will be used. For instance, law enforcement will want to know who has viewed a particular surveillance video, even if the video hasn't been modified.
- Data is likely to go through the computational resources of various administrative domains and different levels of trust, provenance is further complicated.
- Integrity of the provenance data itself is critically important.

Challenges for Architectures: Enabling QoS on Network Edge

- As an application/service leverages computational resources available at various points between the end users and the massive data centers, how can end-to-end quality-of-service (QoS) be guaranteed?
- New mechanisms are necessary to incentivize providers for cooperation and to incentivize application developers to efficiently utilize resources.
  - Without providers cooperating amongst themselves, if a certain level of end-to-end latency is desired, the application developer must reason about all application's computational and communication requirements, negotiate an SLA separately with each provider, and correspondingly map different parts of it to different computation resources so that the end-to-end latency requirement is met.

Capabilities and Services

- Edge devices: Devices that end users use and interact with.
  - Examples: smartphones, tablets, wearables, laptops, IoT devices, and sensors.
- Backend clouds: backends that service providers own and use.
  - Examples: Google services, Microsoft services, Facebook, Amazon clouds, etc.
- Edge infrastructure: infrastructure that sits between edge devices and backend clouds.
  - Examples: extreme-edge computing resources. in-car servers that are built in a vehicle or a building.

Challenges for Services

- Naming, identifying, and discovering resources:
  - Since edge services are distributed at the edge, traditional challenges of distributed systems all apply
- Standardized APIs:
  - APIs need to be standardized to get proper communication and synergy between different edge services.
  - Support computation migration. Computations can migrate in edge computing. For example, virtual machines and containers could migrate between different nodes of an edge computing network. Achieving efficient virtual machine and container migration is important.

Challenges for Services

- Intelligent Edge Services:
  - Edge services will have data processing, analytics capabilities, and intelligent cognitive capabilities. For example an edge computing application might have communications and computations that are dependent on the specific physical location of the device (e.g., a drone) -- connectivity with a major network or a wireless ad hoc network.
- Security and Trust:
  - There will be many edge services, and methods are needed to assess trustworthiness and deal with untrusted services. Since edge infrastructure may run critical services and are close to end users, it is also of critical importance to achieve high availability and reliability.

Challenges for Services

- Create an Edge Service Ecosystem:
  - Having capabilities similar to app stores for edge services would incentivize end users to purchase and own edge computing resources (e.g., servers).
  - Open ecosystem where (1) third parties package up and distribute their edge services, similar to mobile apps, and (2) end users browse, download, and install various edge services as they see fit on their own edge infrastructure, similar to the way end users use online app stores to download various apps.
  - Edge service brokers may become important. An edge service broker will help a client pick the best services for its needs, taking cost into consideration.

2: Cloudlet architecture

The emergence of edge computing. M. Satyanarayanan, https://www.cs.cmu.edu/~satya/docdir/satya-edge2016.pdf

Two Level Architecture

- First level:
  - Today's unmodified cloud infrastructure;
- Second level:
  - Consists of dispersed elements called cloudlets with state cached from the first level.
  - Using persistent caching instead of hard state simplifies the management of cloudlets despite their physical dispersal at the Internet edge.

New challenges and opportunities arise as the consolidation of cloud computing meets the dispersion of edge computing.

How cloudlet proximity can help

- Highly responsive cloud services.
  - Makes it easier to achieve low end-to-end latency, high bandwidth, and low jitter to services located on the cloudlet.
  - Valuable for applications that offload computation to the cloudlet.
- Scalability via edge analytics.
  - Lower the cumulative ingress bandwidth demand into the cloud from a large collection of high-bandwidth IoT sensors, such as video cameras
  - Analyze raw data is analyzed on cloudlets and send only the extracted information and metadata to the cloud.

How cloudlet proximity can help

- Privacy-policy enforcement.
  - By serving as the first point of contact in the infrastructure for IoT sensor data, a cloudlet can enforce the privacy policies of its owner prior to release of the data to the cloud.
- Masking cloud outages.
  - If a cloud service becomes unavailable due to network failure, cloud failure, or a denial-of-service attack, a fallback service on a nearby cloudlet can temporarily mask the failure.

Highly responsive cloud services

- Humans are very sensitive to delay
- Human performance
  - Face recognition takes 370–620 ms, depending on familiarity
  - Speech recognition takes 300–450 ms for short phrases, and it requires only 4 ms to tell that a sound is a human voice.
- VR applications that use headtracked systems require latencies of less than 16 ms to achieve perceptual stability.
- Edge computing brings energy-rich high-end computing within one wireless hop of mobile devices, thereby enabling new applications that are **computation intensive and latency-sensitive**

Wearable cognitive assistance

- Combines a device like Google Glass with cloudlet-based processing to guide users through a complex task.

- On the cloudlet these applications consist of two phase:.
  - First: the sensor inputs are analyzed to extract a symbolic representation of task progress. This is an idealized representation of the input sensor values relative to the task, and excludes all irrelevant detail.
  - Second: comparing the symbolic representation to the expected task state generates user guidance for the next step

Example

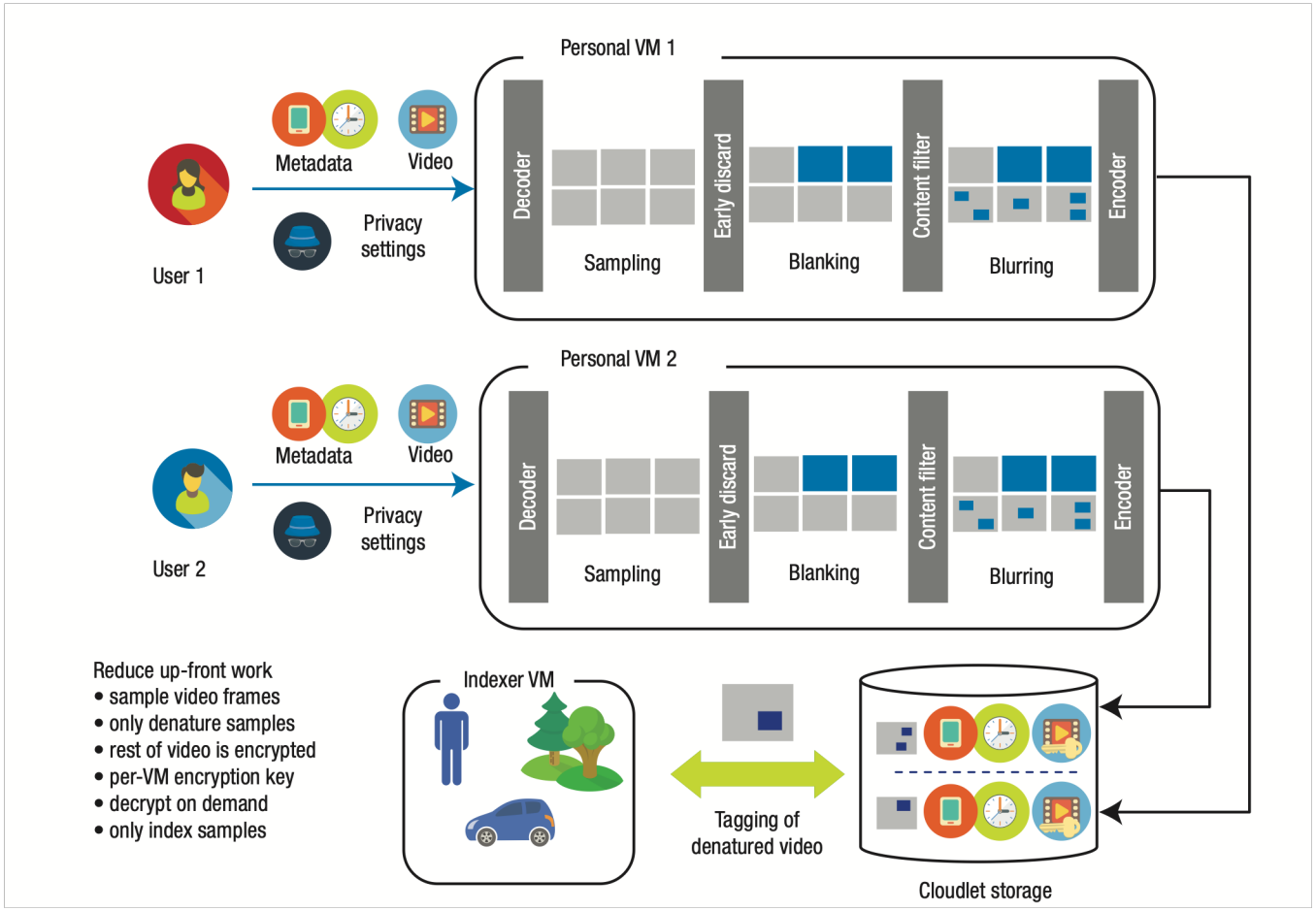
- Jogs user's memory of a familiar face whose name cannot be recalled.:
  - Detects and extracts a tightly cropped image of each face, then applies popular open source face recognizer. Whispers name of person.
- Helps novice pool player aim correctly:
  - Gives continuous visual feedback (left arrow, right arrow, or thumbs up) as user turns cue stick. Correct shot angle is calculated based on widely used fractional aiming system. Uses color, line, contour, and shape detection. Symbolic representation describes positions of cue ball, object ball, target pocket, and top and bottom of cue stick.

Scalability via edge analytics

- Cloudlets can reduce ingress bandwidth into the cloud.
- Example:
  - Consider an application in which many colocated users are continuously transmitting video from their smartphone to the cloud for content analysis.
  - Cumulative data rate for even a small fraction of users in a modest-size city would saturate its metropolitan area network: 12,000 users transmitting 1080p video would require a link of 100 gigabits per second; a million users would require a link of 8.5 terabits per second

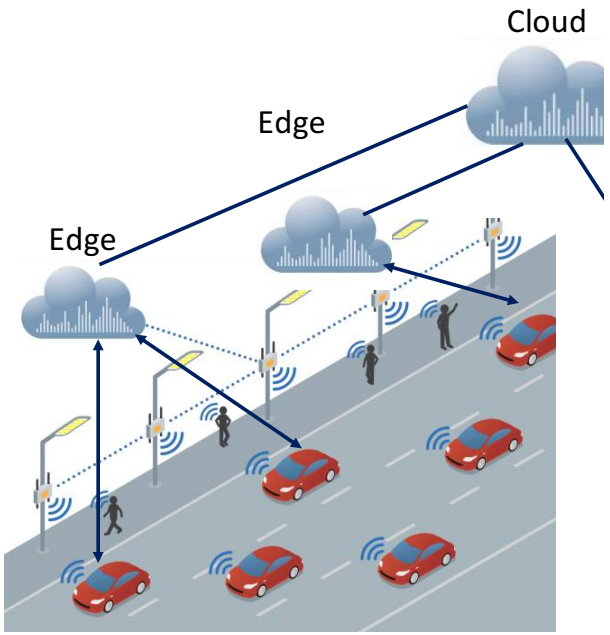
Scalability via edge analytics

- GigaSight framework. A cloudlet performs computer vision analytics on video from mobile devices in near real time and only sends the results along with metadata to the cloud, sharply reducing ingress bandwidth into the cloud.



Edge analytics for connected vehicles

- Should cloudlets be in automobiles or part of the telco infrastructure
- Vehicles's cloudlet
  - Multiplayer video game
  - Real-time analytics of high-data-rate sensor streams from the engine and other sources to alert the driver to imminent failure or the need for preventive maintenance.
- Telco cloudlet:
  - Collaborative real-time avoidance of road hazards

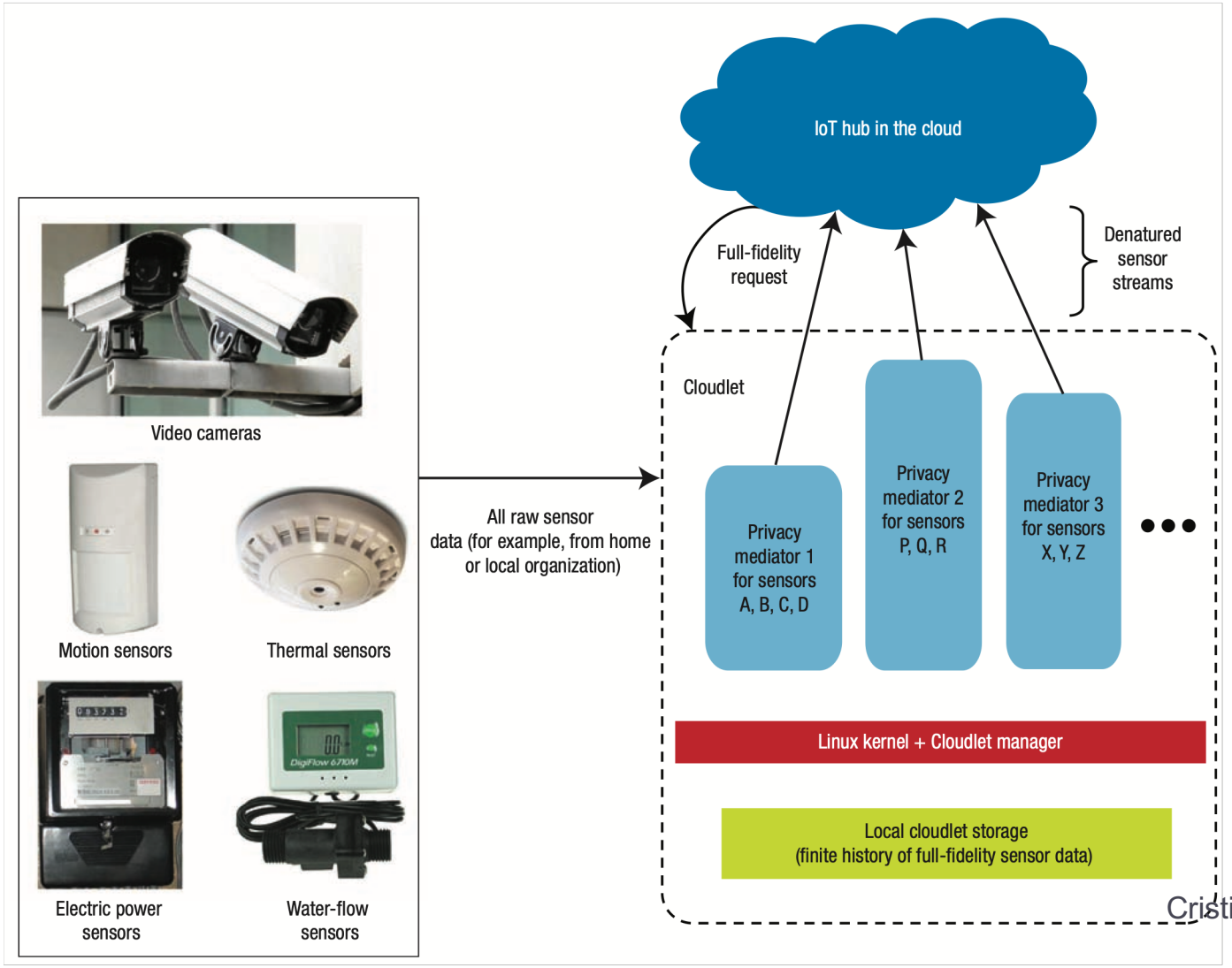


Privacy-policy enforcement

- Cloudlets could help with concerns over data privacy arising from IoT system over centralization.
- Users and organizations desire finer-grain control over release of that data.
- Example:
  - A user should be able to delete or denature a subset of sensor data he or she deems sensitive. Denatured sensor data is safe to release to the outside world: faces in images can be blurred, sensor readings can be coarsely aggregated
  - Today's IoT architectures, in which data is transmitted directly from sensors to a cloud hub, make such fine grain control impossible

Internet of Things (IoT) privacy architecture

Software modules execute on a cloudlet within a sensor owner's trust domain to perform denaturing and privacy-policy enforcement on the sensor streams.



Masking cloud outages

- Cloudlets can provide a fallback service to temporarily mask cloud inaccessibility.
- During failures, a cloudlet can serve as a proxy for the cloud and perform its critical services.
- Upon repair of the failure, actions that were tentatively committed to the cloudlet might need to be propagated to the cloud for reconciliation. See CODA system

CODA

- Coda is a Distributed File System developed as a research project at Carnegie Mellon University since 1987, descended directly from an older version of AFS (AFS-2)
- Features
  - disconnected operation for mobile computing
  - high performance through client side persistent caching
  - server replication
  - security model for authentication, encryption and access control
  - continued operation during partial network failures in server network
  - network bandwidth adaptation
  - good scalability
  - well defined semantics of sharing, even in the presence of network failures

Opportunities: SDN/NFV and 5G

- Software-defined networking (SDN) and the associated concept of network function virtualization (NFV), which must be supported by some of the same virtualized infrastructure as edge computing.
- Ultra-low-latency (one millisecond or less) wireless networks for a new class of tactile applications. Ultra- low latency is one of the proposed attributes for future 5G networks.
- Continuing improvement in the computing capabilities of wearables, smartphones, and other mobile devices that represent the Internet's extreme edge.