ELE6953E : Cyber-Physical Systems and the Internet of Things

Lecture 9 : Cloud Computing for the Internet of Things

Jérôme Le Ny
Department of Electrical Engineering, Polytechnique Montreal

GERAD
GROUP FOR RESEARCH IN DECISION ANALYSIS

POLYTECHNIQUE MONTRÉAL

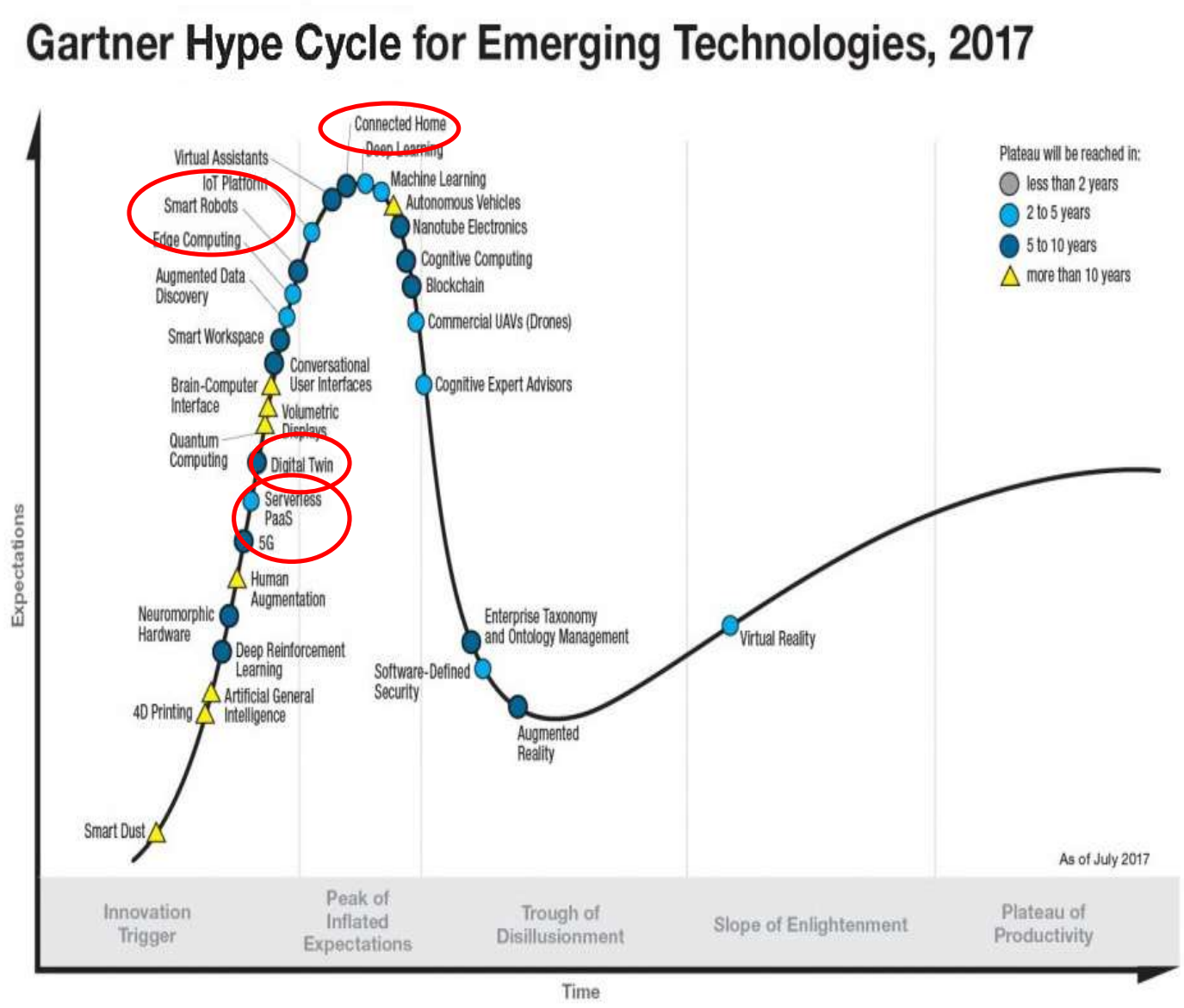
INDUSTRIAL INTERNET CONSORTIUM

POLYTECHNIQUE MONTRÉAL

- IIC, founded in 2014 by AT&T, Intel, IBM, Cisco, GE
- ~260 member companies in 2016
- Focus: Application of "Internet of Things" (IoT) technologies to industries.
- What does that mean?

POSITION ON THE HYPE CYCLE

POLYTECHNIQUE MONTRÉAL



TOPICS

- **"Data Analytics" for the Internet of Things**
  - Next generation SCADA systems / Industrial IoT (IIoT)
  - Back-ends for the Internet of Things: cluster computing for model-based signal processing ("streaming analytics")
  - Overview of some relevant frameworks and technologies for networking, big data storage and processing in "the cloud"
    - Commercial or not
  - Homework: use a small subset of these tools, for a small problem in estimation & fault detection
  - Focus: data processing on the cloud server side (software back-end), only give a few pointers for the data acquisition aspects (hardware front-end, connectivity)
    - "Edge computing" is in some sense covered in earlier part of the course. Software solutions still in early stages (ex: Apache Edgent)

LIMITS OF THIS COURSE

- We'll touch only superficially on computer systems engineering and programming aspects
- No time to go into the details of distributed computing systems
- No time to teach to teach you parallel programming frameworks in any reasonable depth
  - Pick up the minimum you need for the assignment, explore further later on your own if you are interested
  - N.B.: Various computing frameworks come with APIs in a *limited* number of programming languages (Scala, Java, Python, Matlab…)
- Goal is just to give you the flavor of some existing technologies that can be leveraged for automation
- Technologies in this space change very quickly! Will be hard to keep up with the "bleeding edge" for a while
- Landscape of existing tools is messy, competition between service providers is fierce, and choices can be driven by hype
- Lots of ways to delve deeper into the actual software technologies mentioned here (MOOCs, INF8480, blogs, etc.). Follow-up classes will go back to the fundamentals of data processing (algorithms)

OUTLINE

1. Intro to Cloud Computing
   - What is it? How is it relevant to the design of large-scale monitoring and control systems?
2. Cloud Computing Services for Data Acquisition and Analysis
   - Fourth generation SCADA systems (Supervisory Control and Data Acquisition) : Industrial Internet of Things (IIoT) and Cloud Computing
3. Storage for Big Data (databases, distributed file systems)
4. Analyzing big data
   - Cluster computing (ex: Apache Spark)

1. Intro to Cloud Computing

"CLOUD" CONCEPTS

- Deliver computing, storage, software, etc. as services over the internet (or any other network)
  - Ex: Amazon AWS, Gmail, Google Docs, Netflix, …
- For an overview of key "cloud" concepts: [Armbrust et al., 2010]
- A key motivating challenge is scale. Systems that should handle
  - Huge numbers of user requests, coming from anywhere on Earth
  - Huge amounts of data to store and process (Google in 2008: 20 petabytes / day)
- Requires massive distributed computing infrastructure (data centers), enabling "big data" analysis
- Cloud computing providers and simplifying programming models allow many other companies, institutions, governments, etc. to work with big/distributed data as well

NIST DEFINITION

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

- Most of the related concepts are "old" (~1960s), but commercial growth started in the early 2000s, with explosion of web services

CLOUD COMPUTING ECONOMICS

POLYTECHNIQUE MONTRÉAL

- Traditional IT model: company manages its own IT infrastructure, on-premises, even if not primarily an IT company
- Issues: lack of flexibility, not necessarily cost optimal
  - Large fixed cost for hardware investment, slow/expensive to scale up, hard to scale down capacity when not needed any more, cannot handle time-varying loads efficiently, etc.
  - Need large IT team for maintenance, expertise, even if not core business
- On the other hand, some large IT companies (Amazon, Google, IBM, Microsoft, etc.) have large data centers, millions of servers, and deep IT expertise → willing to provide "**utility computing**" for a fee
- Other companies / institutions can then quickly access just the necessary hardware, middleware and software, based on needs
  - Elasticity of on-demand resources for different workloads: customer can use 1000 servers for one hour for the price of 1 server for 1000 hours
  - Power grid / computing analogy: build large power plants, transport power to customer, who is metered and pays just for energy consumed (less efficient for everyone to build his/her own power plant)
- Applies also to distributed monitoring and control systems → IoT (industrial or not), cloud-based SCADA

ECONOMIES OF SCALE

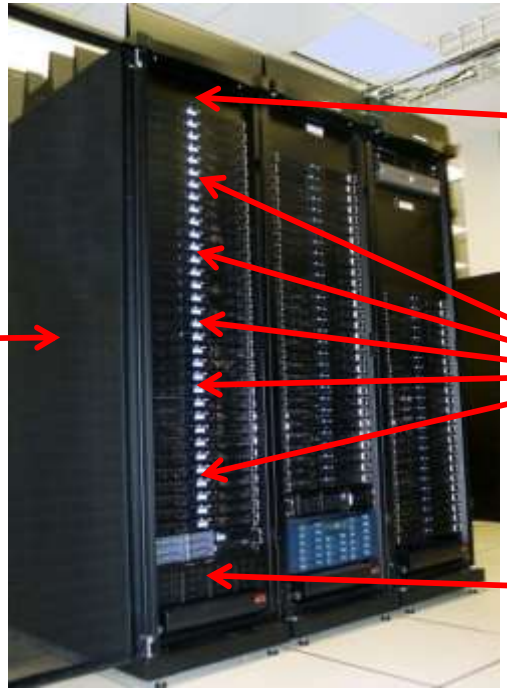
POLYTECHNIQUE MONTRÉAL

Technology	Cost in medium DC (~1,000 servers)	Cost in large DC (~50,000 servers)	Ratio
Network	$95 per Mbit/sec/month	$13 per Mbit/sec/month	7.1
Storage	$2.20 per GByte/month	$0.40 per GByte/month	5.7
Administration	~140 servers/admin	>1,000 servers/admin	7.1

Source: James Hamilton's Keynote, LADIS 2008

- Large internet companies realized their existing data center infrastructures could be used to provide cloud services
- Builds on their investment, they are also selling expertise in services and IT that they have developed first for their own operations
- Pioneer: Amazon Web Services (AWS) launched in 2002, S3 (Simple Cloud Storage Service) and EC2 (Elastic Cloud Compute) in 2006
- 2008: Google App Engine, 2009: Windows Azure Beta

CLUSTERS



Rack

Network switch
(connects nodes with each other and with other racks)

Many nodes/blades
(often identical)

Storage device(s)

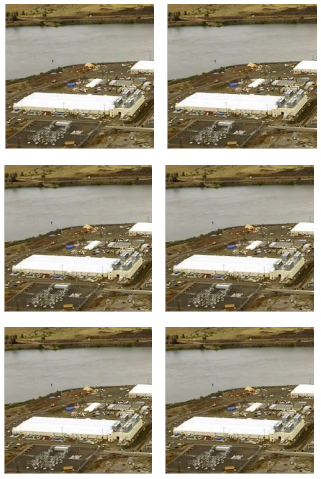
[©Z. Ives & A. Haeberlen, UPenn]

PC

Server

Cluster

Data center



MAIN TYPES OF CLOUD SERVICES

POLYTECHNIQUE MONTRÉAL

- Infrastructure as a Service (IaaS)
  - Provide access to raw computing resources (computing nodes / virtual machines, hard or virtual disks for storage, …)
  - Ex: AWS EC2, EBS (Elastic Block Store)
- Platform as a Service (PaaS)
  - Provide access to middleware, higher abstraction level than IaaS, less configuration needed (and possible), takes care of cluster management, automatic resizing, etc.
  - Application developer builds on top of these services, without worrying about low-level infrastructure (too much)
  - Ex: provide a database management system, on top of infrastructure. AWS RDS (Relational Database Service)
- Software as a Service (SaaS)
  - Provide entire application
  - Ex: Gmail, Google docs, Salesforce.com
- The main companies (Amazon, IBM, Microsoft, Google, etc.) are service providers in all categories
- These services can be provided by public clouds (widely available commercial services), community clouds (ex: Compute Canada Cloud), private clouds (accessible by one organization)

EXAMPLES OF SERVICES

POLYTECHNIQUE MONTRÉAL

Amazon EC2 (IaaS)

Linux | RHEL | SLES | Windows | Windows with SQL Standard | Windows with SQL Web | Windows with SQL Enterprise

Region: US East (N. Virginia)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.nano	1	Variable	0.5	EBS Only	$0.0058 per Hour
t2.micro	1	Variable	1	EBS Only	$0.0116 per Hour
t2.small	1	Variable	2	EBS Only	$0.023 per Hour
t2.medium	2	Variable	4	EBS Only	$0.0464 per Hour
t2.large	2	Variable	8	EBS Only	$0.0928 per Hour
t2.xlarge	4	Variable	16	EBS Only	$0.1856 per Hour
t2.2xlarge	8	Variable	32	EBS Only	$0.3712 per Hour
m5.large	2	10	8	EBS Only	$0.096 per Hour
m5.xlarge	4	15	16	EBS Only	$0.192 per Hour
m5.2xlarge	8	31	32	EBS Only	$0.384 per Hour
m5.4xlarge	16	61	64	EBS Only	$0.768 per Hour
m5.12xlarge	48	173	192	EBS Only	$2.304 per Hour
m5.24xlarge	96	345	384	EBS Only	$4.608 per Hour
m4.large	2	6.5	8	EBS Only	$0.1 per Hour
m4.xlarge	4	13	16	EBS Only	$0.2 per Hour
m4.2xlarge	8	26	32	EBS Only	$0.4 per Hour
m4.4xlarge	16	53.5	64	EBS Only	$0.8 per Hour
m4.10xlarge	40	124.5	160	EBS Only	$2 per Hour
m4.16xlarge	64	188	256	EBS Only	$3.2 per Hour
Compute Optimized - Current Generation					
c5.large	2	8	4	EBS Only	$0.085 per Hour
c5.xlarge	4	16	8	EBS Only	$0.17 per Hour
c5.2xlarge	8	31	16	EBS Only	$0.34 per Hour

IBM Cloud PaaS

IBM Cloud

Dashboard

Cloud Foundry Apps

Name	Route	Memory (MB)
ELE69S3E-hw4-IoT-app	ELE69S3E-hw4-IoT-app.mybluemix.net	512
IoT-class-project	IoT-class-project.mybluemix.net	256

Cloud Foundry Services

Name	Service Offering	Plan
AnalyticsEngine-ELE69S3E	Analytics Engine	Lite
availability-monitoring-auto	Availability Monitoring	Lite
DataScienceExperience-ELE69S3E	Data Science Experience	Lite
ELE69S3E-hw4-IoT-app-cloudantNoSQLDB	Cloudant NoSQL DB	Lite
ELE69S3E-hw4-IoT-app-iotf-service	Internet of Things Platform	Lite
IoT-class-project-cloudantNoSQLDB	Cloudant NoSQL DB	Lite
IoT-class-project-iotf-service	Internet of Things Platform	Lite
ObjectStorage-1	Object Storage	Lite
Spark-1	Apache Spark	Lite

Services

Name	Resource Group	Service Offering
cloud-object-storage-ELE69S3E	default	Cloud Object Storage

EXAMPLES OF APPLICATIONS AND SOME LIMITATIONS

POLYTECHNIQUE MONTRÉAL

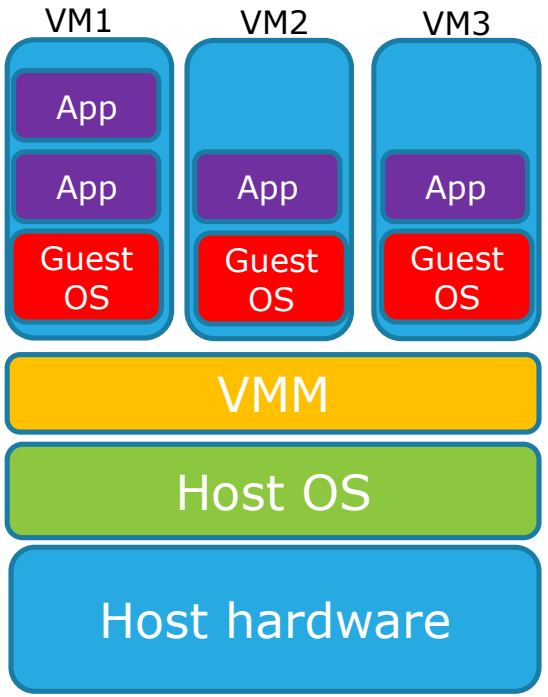
- Web and application hosting
- Backup, storage
- E-commerce
- High-performance computing
- Many other…

Can be problematic when

- Data is sensitive (ex: medical, financial), contains IP
- Need (auditable) compliance to laws and regulations related to data storage, data transfer, etc.
- Need guaranteed availability (What if the cloud is down? Can I run a safety-critical application in the cloud?)
- Need predictable computing / comm. performance, hard real-time
- Large data transfer / data storage for short term usage (cost)
- Need to install specific licensed software
- …

CORE TECHNOLOGY: VIRTUALIZATION

POLYTECHNIQUE MONTRÉAL



- A physical machine can be "divided" into multiple virtual machines (VM)
  - Each VM associated with one or several physical CPUs, some amount of RAM, disk space, etc.
  - Each VM is isolated from the others, can be provided to customer
- A Virtual machine monitor (VMM)/hypervisor translates requests from VMs for virtual resources to physical resources
- VMs can be easily migrated to a different physical machine if needed (for maintenance, geographical redeployment, etc.), with no impact for customer
- VMs can *time-share* physical resources, in effect giving more virtual resources to sell for the provider (but challenge for VM isolation / performance prediction, e.g., if a customer changes its workload suddenly)

SOME FUNDAMENTAL CHALLENGES IN DISTRIBUTED SYSTEMS / CLOUD COMPUTING

- Clusters and data centers provide access to potentially very large numbers of computing nodes are storage devices
  - Networking is complex (ex: many protocols) and imperfect: delays, packet loss, etc.
  - **Faults become very common when dealing with large numbers of (commodity) devices**. How to deal with faulty machines (crash, slow, etc.), disk failures & data corruption, variable memory latency, etc.?
  - Parallelizing programs can be hard, algo. dependent, not always possible
  - Distributed computing is difficult (Ex: Dining Philosophers Problem), especially in the presence of faults, which can be random, caused by manipulation, byzantine/anything (Ex: Byzantine General's Problem)
- Fortunately, some system and software abstractions hide many of these complexities to application developers (up to a point)
  - Frameworks for fault-tolerant data replication, distributed computing, networking, etc.
  - Still, important to be aware of underlying issues

CLOUD COMPUTING AND CPS

POLYTECHNIQUE MONTRÉAL

- For CPS, cloud computing services offer a natural solution to
  - Collect data from large numbers of distributed sensors
  - Process that data at scale, in a "virtual centralized server"
  - Store historical sensor data
  - Run batch or streaming computing jobs on that data: e.g., for fault / anomaly detection.
  - Optimize process objectives, close high-level feedback loops in real-time. With caveats:
    - Not for fast real-time, nor hard real-time, due to internet network time-varying delays
    - Cost of communication bandwidth to data centers, of running servers in the cloud, advantage of centralized view must be weighted against possibility of running some computations locally in low-cost but still very capable embedded systems ("edge computing")
- → Main cloud computing providers often provide services targeted at "IoT" applications

2. Cloud Computing Services for Data Acquisition and Analysis

SCADA SYSTEM EVOLUTION

POLYTECHNIQUE MONTRÉAL

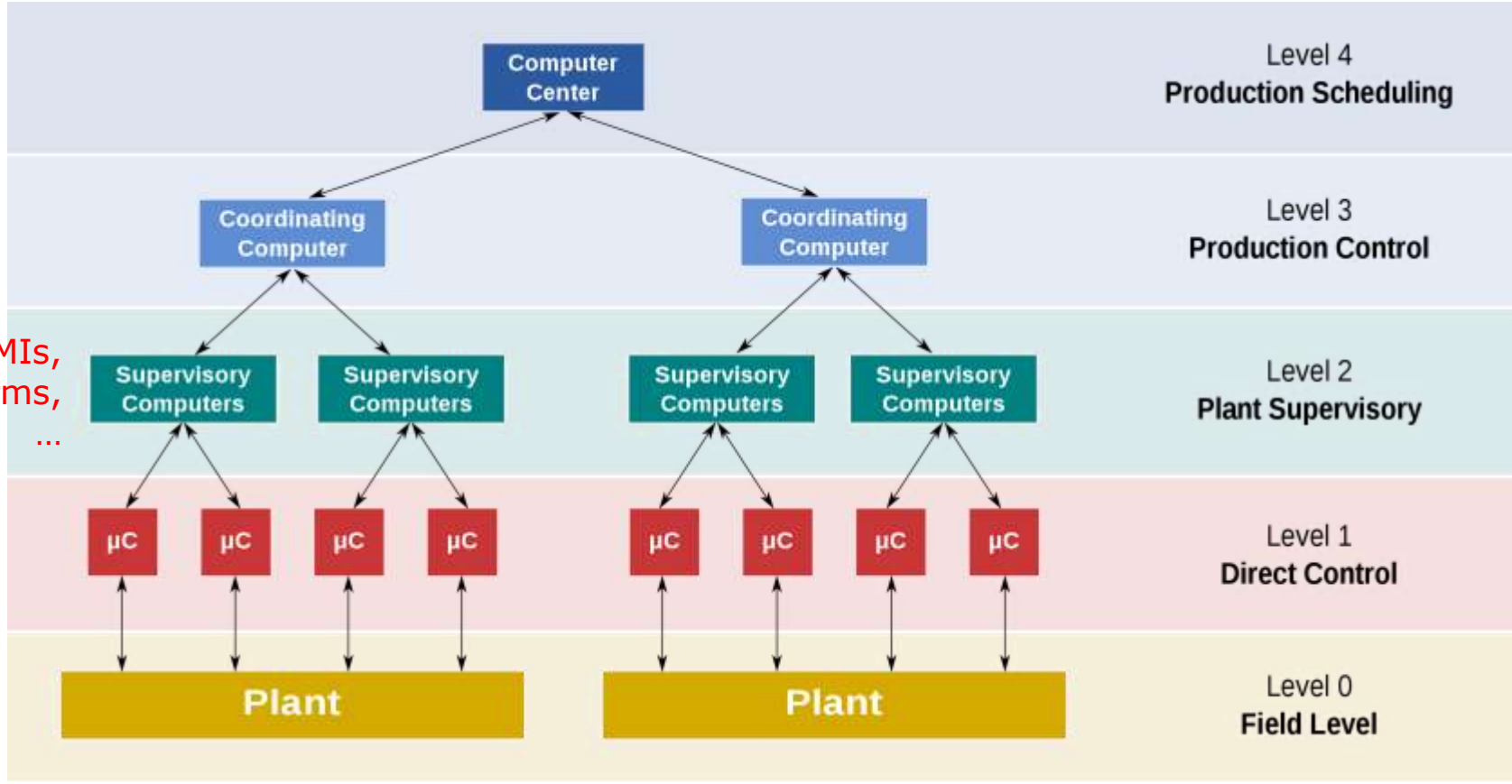
- SCADA: System to obtain real-time data from geographically distributed sites, to monitor a control process / automated system
  - Interface with local controllers (Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs))
  - Human-machine interfaces (HMIs) on the monitoring side
- Industries: oil & gas, water and wastewater treatment, power, chemical, pharmaceutical, automotive, electronics, etc.
- Expensive systems, vendor lock-in. Vendors: ABB, Emerson Electric, Honeywell International, Schneider Electric, Siemens, GE, etc.
- How do new "cloud" technologies change things?
  - Cloud computing platforms widely available, economically viable
  - Easily deployable and accessible IoT services (commercial and open-source)
  - Democratization of cluster computing / big data analytics
  - But: reliability, safety, real-time performance, security, privacy issues

TYPICAL ARCHITECTURE OF INDUSTRIAL DISTRIBUTED CONTROL SYSTEMS

POLYTECHNIQUE MONTRÉAL

SCADA HMIs, alarms, ...

PLCs, RTUs



[Wikipedia]

A LARGE SCALE SCADA SYSTEM: THE POWER GRID

POLYTECHNIQUE MONTRÉAL

[ISO New England]

[Northeast blackout, 2003]

NETWORKING FOR INDUSTRIAL DCS

POLYTECHNIQUE MONTRÉAL

- Many different technologies and protocols developed over time to connect to industrial devices from different vendors. Ex:
  - Industrial Ethernet (IE) can be used to connect PLCs to supervisory layer. One protocol for IE is PROFINET
  - Modbus (1979: a serial protocol originally for Schneider Electric PLCs. Now a standard to connect PLCs/RTUs with supervisory layer, for example over IE (general trend from serial comms to IP nets)
  - Profibus (1989), a type of fieldbus, industrial computer protocol generally to connect PLCs to lower level sensors, actuators, etc. Used by Siemens. Should not be confused with PROFINET…
  - Wireless: WirelessHART, ISA100.11a, WIA-PA, ZigBee, Wi-Fi, etc.
- Networks for Industrial Control Systems (ICS) are different from traditional IT networks, and have somewhat different QoS reqs.
  - Equipment can be old, harsh / non controlled environment
  - Delivering data on time most of the time (availability) can be more important than delivering always correct data but with a delay (integrity)

DATA SHARING STANDARDIZATION EFFORTS FOR DCS

POLYTECHNIQUE MONTRÉAL

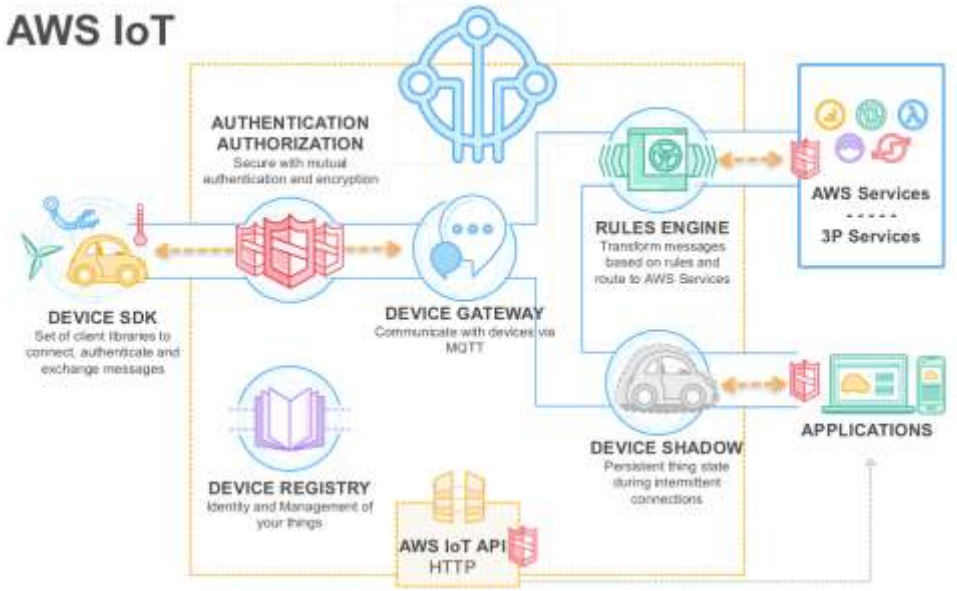
- OPC (Open Platform Communications) UA (Unified Architecture)
  - Standard developed by the OPC foundation, first version in 1996
  - Initially standardized interface to connect HMI/SCADA to PLC specific protocols → goal is to support device interoperability
  - OPC UA specification released in 2008, goal remains platform/vendor independent information sharing between devices for industrial applications, with advanced functionalities (device discovery, security, etc.).
  - Client-server and publish-subscribe communication models
- DDS
  - More recent standard developed by the Object Management Group (OMG), "Data-Centric Publish Subscribe" (DCPS) model
  - Middleware simplifying network programming (declarative programming), creates a decentralized data space
  - Various QoS parameters, adresses (hard?) real-time distributed sys.
- OPC-UA vs. DDS: article, discussion

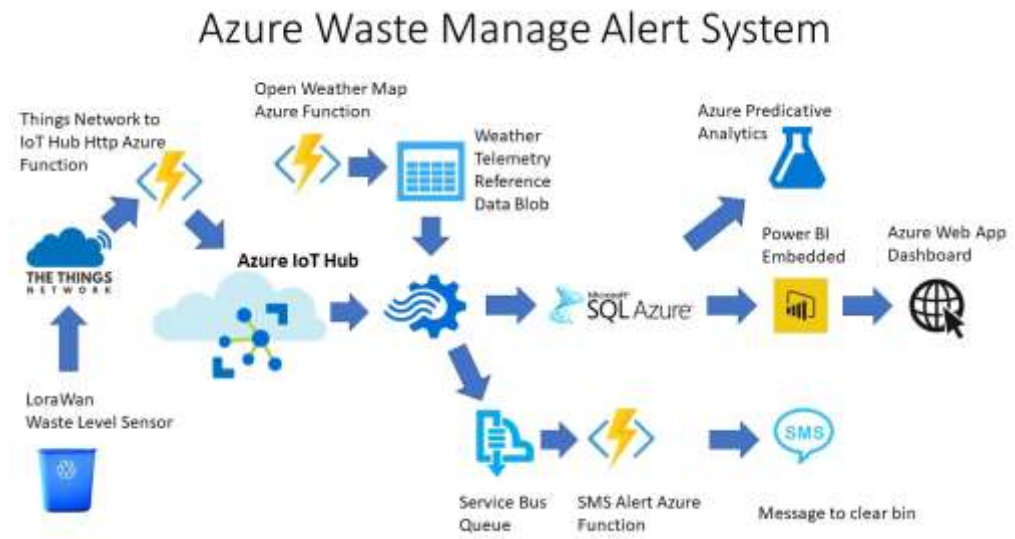
CLOUD-BASED SCADA

- Growing demand for cloud-based SCADA
- CC should provide scalable, flexible (elastic) way to set-up infrastructure to collect, store, monitor, process sensor data
- In a sense, some existing systems with SCADA servers in a remote (safer) location already constitute private clouds
- Extends to very large-scale systems: smart buildings, smart communities, smart cities and infrastructure
- Public, private of hybrid cloud?
  - Ex: Private cloud at process level, public cloud at corporate level
- Risks: security, reliability, performance
- Read this Manufacturing Automation Article, Nov. 2017. Regarding IIoT: "*a common architecture has emerged - one in which field sensors are connected to gateways that move the data to a public Cloud where it is analyzed and individuals or software packages may access it with standard APIs*"

STANDARD IOT SERVICES

- Typically more benign applications than ICS
- Provide libraries for connecting devices (ex: Amazon FreeRTOS)
- Device Management
- Often offers connection via MQTT protocol
  - Lightweight pub/sub protocol (with message broker)
  - Targets low bandwidth data links, intermittent connection
- Authentication, crypto
- Easy connection with PaaS back-end services
- List of IoT Platforms

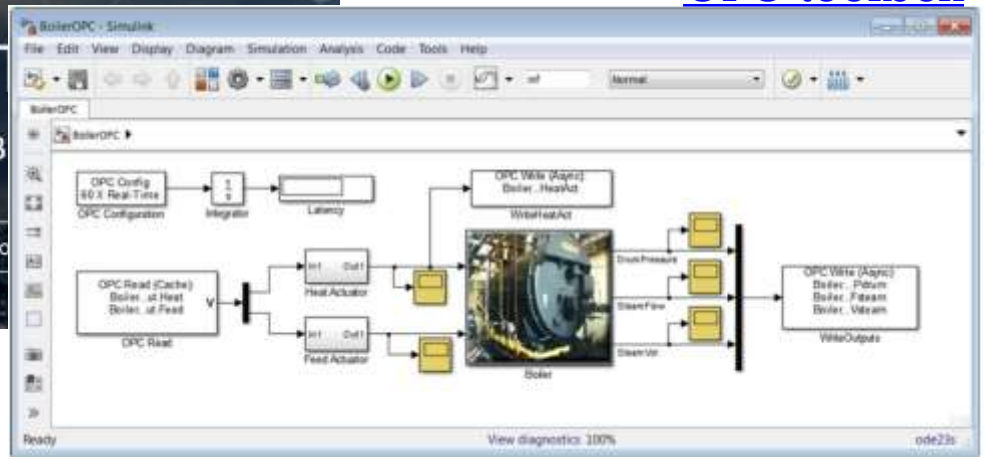




EXAMPLE
MATHWORKS + THINGSPEAK

POLYTECHNIQUE MONTRÉAL

[OPC toolbox]



ARCHITECTURE ISSUES

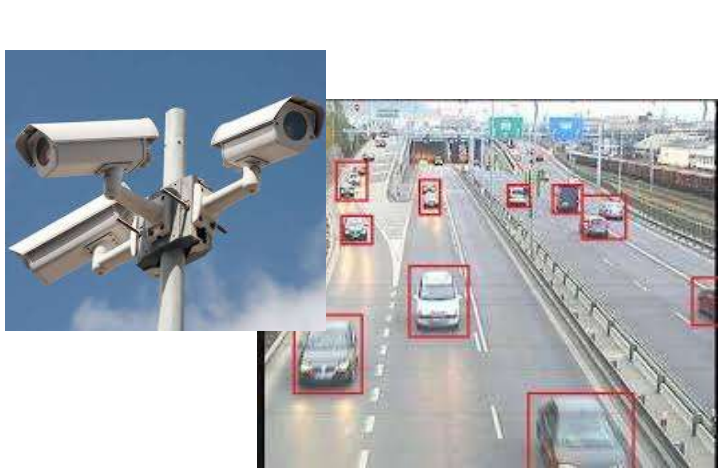
Where to process your data?

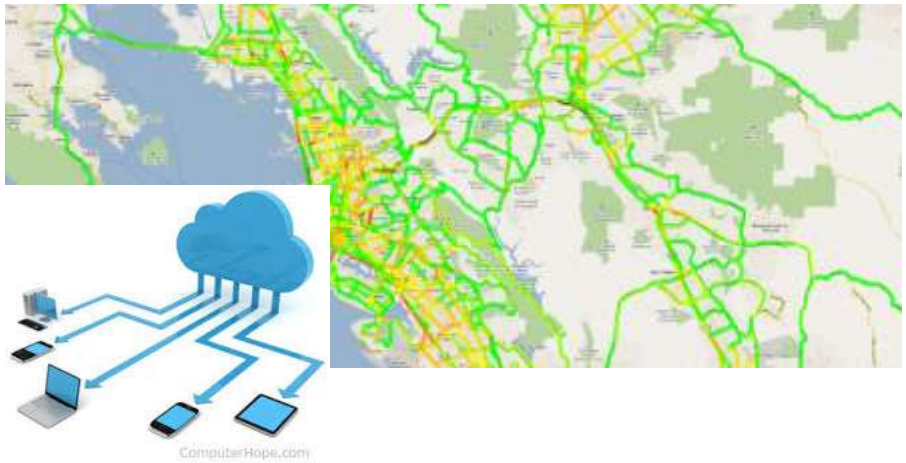
- Features of local processing ("edge" computing, "fog" computing)
  - Lower costs for data tx to cloud,
  - Less power consumption for communications (esp. if wireless connection)
  - Lower latency
  - High computation or storage requirements need to be provisioned
    - Potentially not feasible (embedded devices)
    - Potentially expensive
    - Not flexible (ex: requirement changes)
    - Redundancy, fault-tolerance, etc., difficult and expensive
- Features of cloud-based processing
  - Can have a more global view of the system (beyond local network)
  - Access to large computing infrastructure
  - Access to large data storage (historical databases, etc.)
  - Fault-tolerant computing and storage provided as a service
  - But: high communication requirements (high bandwidth, low latency) can be expensive or impossible to satisfy at the performance level required
- In general, a good architecture needs a combination of local computing (ex: for low latency feedback control) and cloud computing (ex: for anomaly detection, long-term performance analysis, data logging for future audit, etc.)

EXAMPLE: TRAFFIC ESTIMATION FROM VIDEO ANALYSIS

Example: analyze car traffic patterns on a road network

- Edge computing: attach a small single board computer (SBC, ex: raspberry pi) to each camera for real-time computer vision algorithm detecting cars
- Only transmit from SBC car detection events with time stamps, or the number of cars detected in a time window, etc., to the cloud for further analysis of traffic patterns
- Cloud computing: cloud can collect counts from many sensors and provide a global estimate of the traffic on the road network, distribute results to client computers
- Alt.: would you transmit raw video directly to the cloud? [Pricing]

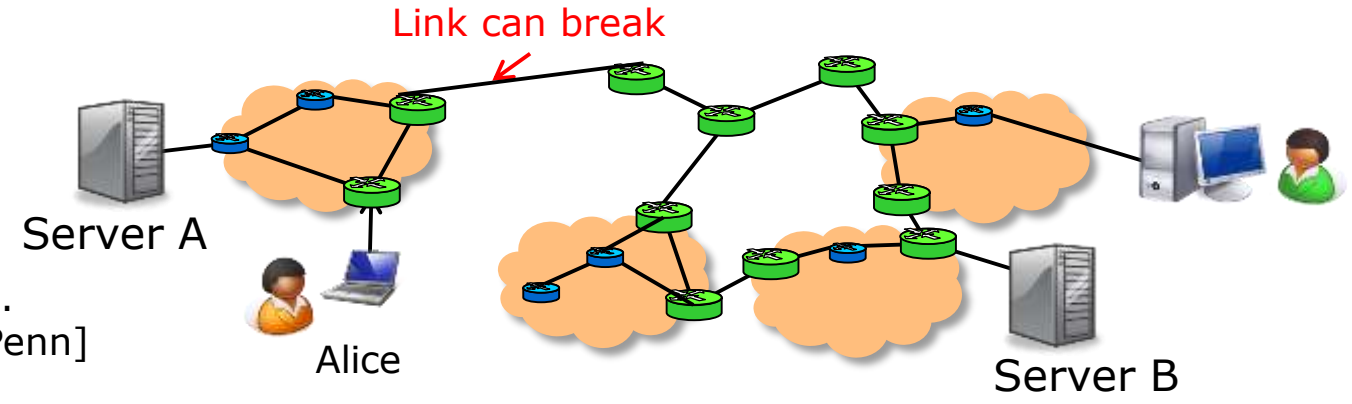




3. Storing and Managing Big Data

DISTRIBUTED DATA STORES

- **Databases** collect information for efficient retrieval (a basic form could be just a file system…)
- Nowadays, many applications require storing large volume of data on multiple (many) nodes in a network, often with **replication** for fault-tolerance: **distributed data store**
- Makes data management much more difficult. Want
  - **Consistency**: Every read receives the most recent write or an error => shared view of data by all clients, despite concurrent updates
  - **Availability**: Every request receives a (non-error) response (without guarantee to obtain the most recent write), even if there are faults
  - **Partition tolerance**: system works even when the network partitions (global connectivity between nodes is lost)



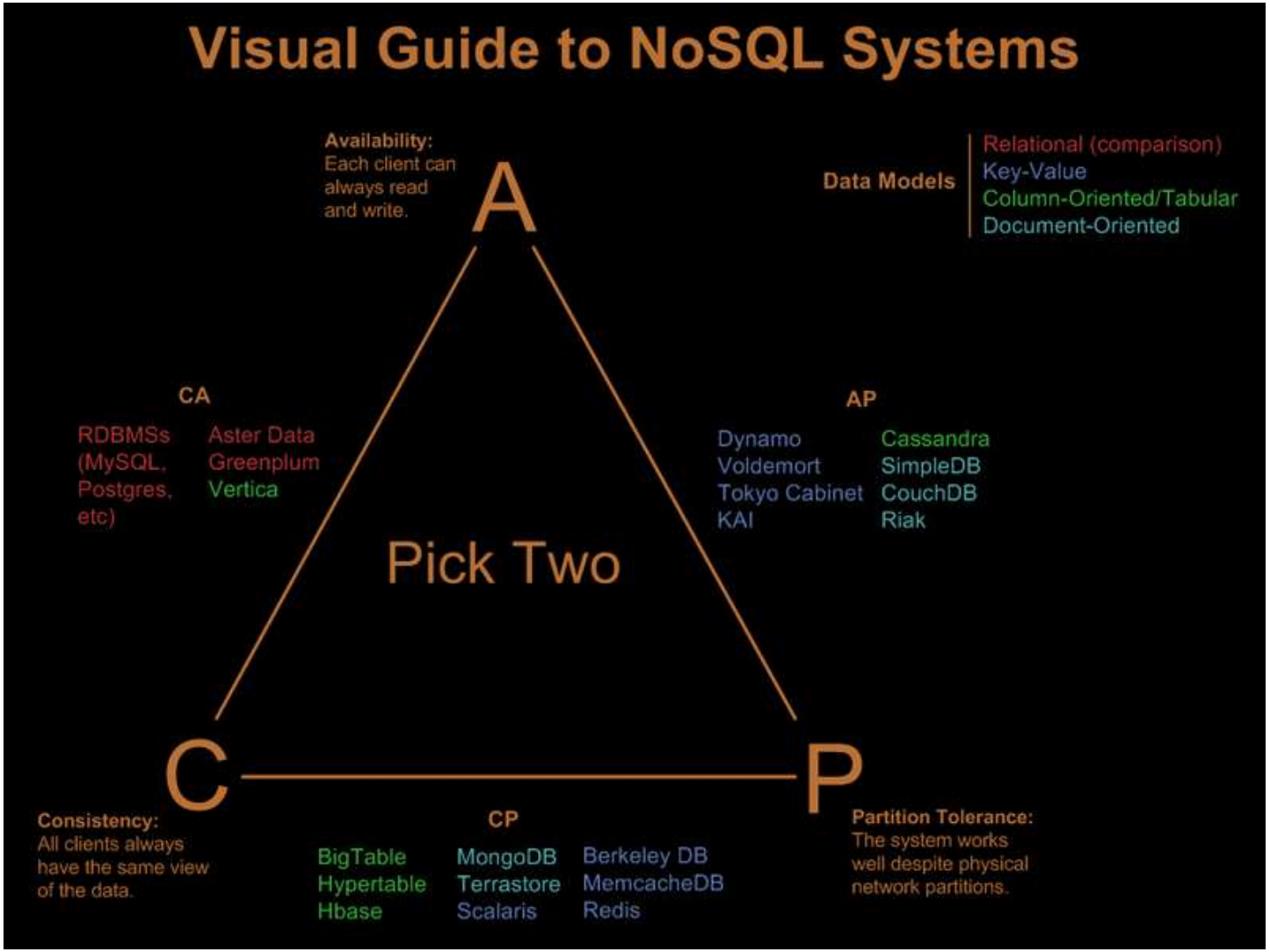
[©Z. Ives & A. Haeberlen, UPenn]

CAP THEOREM AND CONSEQUENCES

- **CAP Theorem** [Brewer conjecture, proved by Gilbert and Lynch]: we can have at most two of the previous properties simultaneously
  - Ex: Generally, a distributed system must tolerate partitions. Then we need to choose between consistency and availability
  - Consequence: different data management systems make different trade-offs, appropriate for different data models and applications. Relational DBMS, key-value store, document store, graph DBMS, etc.
- If choose consistency. Ex: return time-out if partition makes data unavailable, no new write req. during partition, etc.
- If choose availability. During network partition, different relaxations of consistency:
  - Weak consistency: updated value not guaranteed to be returned immediately, only after some conditions are met (inconsistency window)
  - Eventual consistency: weak consistency, such that if no new updates are made to the object, eventually all accesses will return the last updated value
  - Accepting write req. during partitions, state can be inconsistent

CAP THEOREM AND TYPES OF DATABASES

POLYTECHNIQUE MONTRÉAL



[Nathan Hurst]

RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

POLYTECHNIQUE MONTRÉAL

- Relational databases: traditional way of storing data against which we want to run queries (= ask questions) efficiently
- Relational model of data: store data into one or more tables
  - Each column has a name and a data type (text, number, date, …). RDBMS schema
  - Rows of tuples or records
- SQL (Structured Query Language): language for storing, manipulating and retrieving data in databases
  - Declarative programming model
  - Ex: **SELECT** CustomerName, City **FROM** Customers
- SQL also usable for Relational Data Stream MS (RDSMS). Ex: IBM System S (commercial: IBM Streams)
- Traditional RDBMS choose consistency over availability
  - ACID guarantees (Atomicity, Consistency, Isolation, Durability)
  - Hard to scale to distributed systems, relatively high storage cost

DBMS WITH RELAXED CONSISTENCY

POLYTECHNIQUE MONTRÉAL

- Many more recent distributed database management systems for big data and real-time apps choose availability over consistency
  - BASE guarantees (Basically Available, Scalable, Eventually consistent)
  - Ex: Shopping cart
- Generally NoSQL databases, store data by means other than relational model (key-value, wide column, graph, document, …)
  - May still support an SQL-like query language
  - Simpler to design than RDBMS, cheaper storage
- Ex :
    - Key-value stores: (dictionaries) key uniquely identify a record. Supports get, put, delete
      - Ex: Redis, Amazon SimpleDB
    - Document stores: subclass of key-value stores. "document" is DB specific
      - Amazon DynamoDB, Apache CouchDB / IBM Cloudant
    - Many others

OTHER TYPES OF STORAGE

- File system: manage data as file hierarchy
  - Ex: Hadoop Distributed File System (HDFS)
- Block-level storage (ex: Amazon EBS): blocks of data
  - "Virtual hard disk", can be attached to various computing instances
- Object Storage: manage data as objects (ex: Amazon S3)
  - Can store very heterogeneous, unstructured data, including for example audio, images, video
  - Fault tolerance
  - Scalable
  - Similar to key-value store

SUMMARY

Data storage decision matrix

	SQL	NoSQL	ObjectStorage
Storage cost	high	low	very low
Scalability	low	high	very high
Query Speed	high	low	very low
Flexibility	low	high	very low

©Romeo Kienzler, IBM

- Sensor data is mostly time series data
- Main issue is cost of storage ($/GB/month depends on storage solution)
  - Ex: ~3$/GB/month for SQL DB, ~1$/GB/month for NoSQL DB, a few cents/GB/month for Object Storage

4. Processing Big Data: Parallel Computing on Clusters and Data Centers

PARALLEL PROGRAMMING AT SCALE

- Cluster consists of potentially many nodes, which need to coordinate to compute over massive data, itself stored in a distributed filesystem or database
- A general cluster computing organization: 1 driver node, organizing computations with many worker nodes
  - At a lower level, there is also a cluster manager for job scheduling, resource management, etc.
- Traditional parallel programming frameworks for high performance scientific computing such as OpenMP or MPI have not been used much for "data science"
  - Programming level is too low (ex: specify every exchange of data)
  - Does not include fault-tolerance (rebalancing workload when a node becomes slow of fails, etc.), but this is critical when computing infrastructure scales up
  - See this opinion article

APACHE HADOOP

- Collection of open-source software modules to store (HDFS) and process (Hadoop MapReduce) big data

From the Hadoop page

- "The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing" http://hadoop.apache.org/
- 4 main modules

Hadoop Common: The common utilities that support the other Hadoop modules.

Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data.

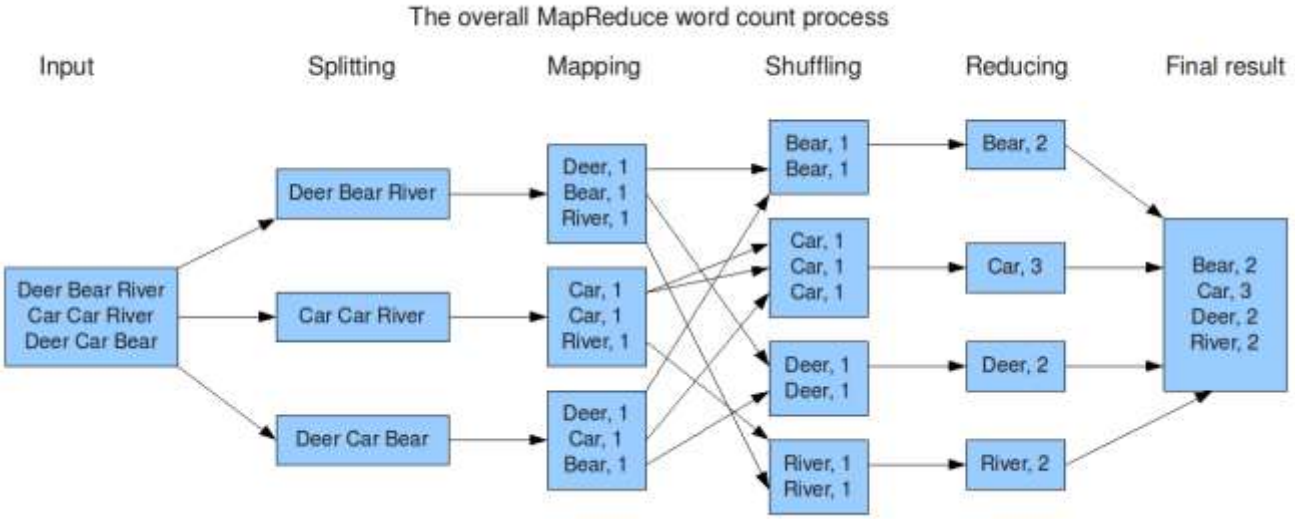
Hadoop YARN: A framework for job scheduling and cluster resource management.

Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.

MAPREDUCE PROGRAMMING MODEL

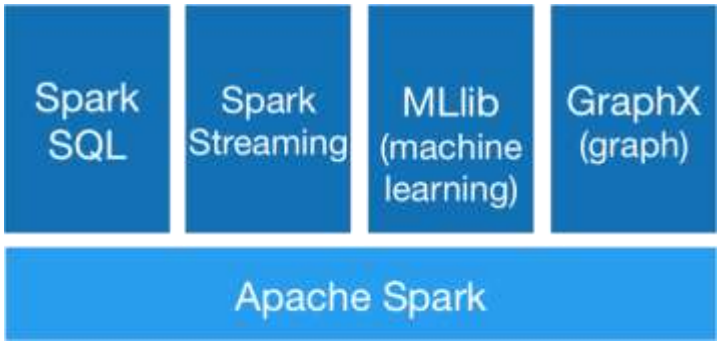
POLYTECHNIQUE MONTRÉAL

- Programming model for parallel distrib. computing on clusters
- Originally a Google technology, open-source implementation in Hadoop
- Constrains the structure of programs to parallelize them:
  - Operate on (key, value) pairs
  - Map step: worker nodes apply a function on their local data elements
  - Shuffle step: data redistrib. among workers based on associated keys
  - Reduce step: summary operation by the workers (ex: summing)



HADOOP MAPREDUCE VS. SPARK

- MapReduce model very powerful for certain types of jobs but constraining, restrictive, low-level / difficult to use
- Not well adapted to iterative/repeated queries common in machine learning, revisiting dataset multiple times
  - In particular, requires rewriting data on disks multiple times, slow
- Hadoop MapReduce (HMR) essentially attached to the rest of the Hadoop Ecosystem
- Apache Spark is a more recent very popular framework facilitating cluster computing
- Generally much faster than HMR for big data processing (using in-memory computing) & more convenient to use
- Spark Core + several libraries targeting different types of applications



APACHE SPARK

- Very actively developed open source engine for parallel data processing on (large-scale) computer clusters
- Project started at UC Berkeley in 2009 at the AMPLab
- Initial paper in 2010 by Matei Zaharia et al.
- Goal was to overcome issues with MapReduce, e.g., for machine learning
- For the end user: API based on functional programming to express multistep applications
- Under the hood: Engine to perform computations efficiently, in-memory data sharing
- Implemented in Scala, runs on Java Virtual Machines (JVM)

LANGUAGE BINDINGS FOR SPARK

Summary

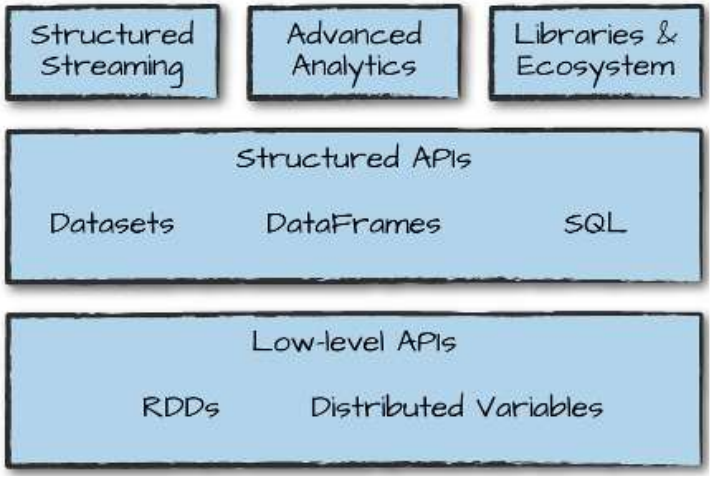
	Scala	Java	R	Python
Spark API support	complete	complete	very limited	limitted
Ease of use	low	very low	high	very high
Speed	very high	high	very low	low
3rd party libs	few	few	many	many

©Romeo Kienzler, IBM

- Can also switch seamlessly between several different data storage solutions, use different cluster managers, …

SPARK DATA TYPES

- Basic data type: Resilient Distributed Dataset (RDDs)
  - Immutable partitioned fault-tolerant collection of records (Java/Scala/Python objects), which can be operated on in parallel
  - Distribute data in nodes' memory (lazily, if action performed)
- Since Spark 2.0, user should use Structured Data Types wrapping RDDs when possible rather than RDDs directly, for better optimization of workloads. But everything still compiled to RDDs
  - DataFrames: maintains a schema on top of RDD, support SQL queries
  - DataSets: typed DataFrames, only in Scala & Java
  - Structured Streaming



[Chambers and Zaharia, 2018]

FUNCTIONAL PROGRAMMING PARADIGM FOR PARALLEL PROGRAMMING

POLYTECHNIQUE MONTRÉAL

- Mathematical foundation of FP: lambda calculus, expresses computations as application of functions
- Data is immutable, functions applied to create new data
- Example of FP languages: Haskell, OCaml, Lisp, Scheme…
- Python supports some basic FP constructs (lambda functions)
- Scala supports FP but is multi-paradigm (imperative, OO)
- FP has many advantages for parallel computations (ex: can send the function to each worker node and apply it on data there)
- Example:

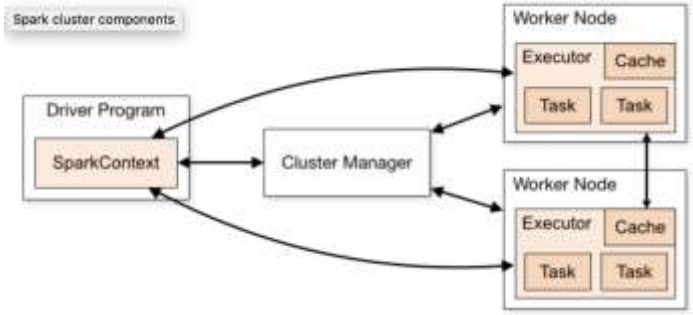
```
In [29]: test_RDD = sc.parallelize(range(100))

In [30]: test_RDD.map(lambda x: x+1).take(5)
Out[30]: [1, 2, 3, 4, 5]

In [33]: test_RDD.reduce(lambda a,b : a+b)
Out[33]: 4950
```

SPARK APPLICATIONS
TRANSFORMATIONS VS. ACTIONS

- Spark app: driver process (assigns work) + executor processes (carry work, report back)

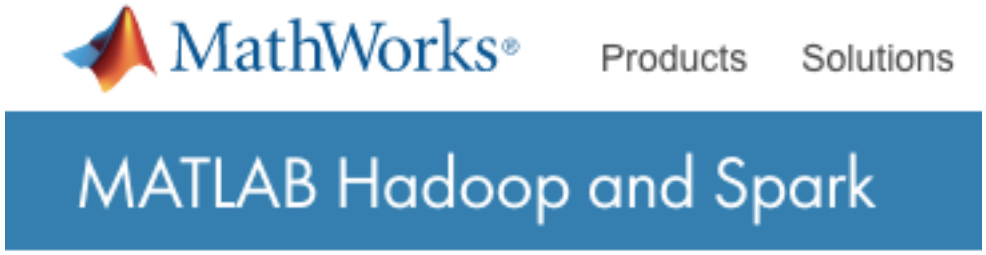


- Developing applications in Spark uses the following paradigm
  - Define **Transformations**: instruct how to modify data, but do not actually perform comp. ("evaluate lazily")
  - Perform **Actions**: evaluate eagerly, actually manipulate data in a distributed fashion
- Ex: `map` transformation defines a function to apply to each element of a dataset, `collect` action brings dataset back in the memory of the driver node
  - Will crash driver node if too big to fit in its memory!

USING MATLAB FOR BIG DATA PROCESSING

POLYTECHNIQUE MONTRÉAL

- Matlab Tall Arrays (data type)
  - Columnar data that does not necessarily fit in memory
  - Support data parallelism storage (HDFS)
  - Parallel computing with parallel computing toolbox
- Can use Matlab compiler to execute applications on Spark
- Link



CONCLUSIONS - RECAP

- Cloud computing provides new technologies useful for distributed automation
- Demand for cloud based SCADA + IoT solutions available from many vendors: convergence?
- Various frameworks, services (commercial or not) simplify the management of computing clusters, distributed data stores, parallel fault-tolerant programming on clusters, etc.
- Architecture (local distributed vs. cloud computing) should still be made carefully. Consider
  - Network reliability, latency
  - Cost of data transfer
  - Cost of computing nodes, renting vs. buying, maintenance, etc.
  - IP and data sensitivity issues (security, privacy, secrecy), compliance
  - Etc.

HOMEWORK ASSIGNMENT AND CHOICE OF CLOUD PROVIDER

- We'll use IBM Bluemix (Watson IoT Platform, Data Science Experience, etc.) for HW Assignment 4
- No fundamental reason, IBM not involved in this course beyond providing standard academic credits (not course specific)
- Bluemix more focused on PaaS compared to Amazon: services/software perhaps easier to set-up for beginner
- AWS provides IaaS and PaaS services, has more or less equivalent services to those used in assignment
- For a new application, would have to decide platform and service level based on your team: experience with platforms, personnel dedicated to infrastructure set-up and maintenance, etc.