Data Integration and Large Scale Analysis

Slides credit: Matthias Boehm - Shafaq Siddiqi

08- Cloud Computing Fundamentals

TU Graz
SCIENCE
PASSION
TECHNOLOGY

Lucas Iacono. PhD. - 2024

ISDS

Part B

Large-Scale Data Management & Analysis

- LU3. Cloud Computing
  - Cloud Computing Fundamentals [Nov 29]
  - Cloud Resource Management and Scheduling [Dec 06]
  - Distributed Data Storage[Dec 13]

Part B

Large-Scale Data Management & Analysis

- LU4. Large-Scale Data Analysis
  - Distributed, Data-Parallel Computation [Jan 10]
  - Distributed Stream Processing [Jan 17]
  - Distributed Machine Learning Systems [Jan 24]

Agenda

- Motivation and Terminology
- Cloud Computing Service Models
- Cloud, Fog, and Edge Computing

Motivation and Terminology

Motivation and Terminology

- **How new it is?**

Motivation and Terminology

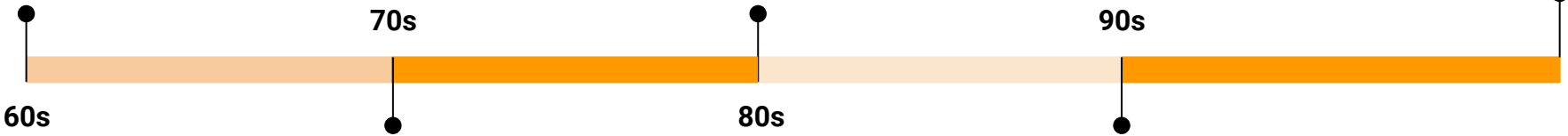
John McCarthy conceptualize the idea of computing as a **public utility** at a lecture at MIT, envisioning "computing on demand."

ARPANET is developed, enabling computational resources sharing over a network.

CompuServe starts offering small-scale cloud-like storage

Tim Berners-Lee invents the World Wide Web

AWS presents EC2 and S3



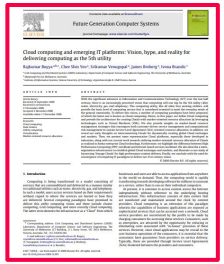
IBM launches CP/CMS (Control Program/Cambridge Monitor System), an early virtualization system that allows multiple users to run applications on a single mainframe.

Distributed computing concept emerges, focusing on resource sharing across multiple machines **(cluster computing)**.

Salesforce launches its CRM SaaS platform

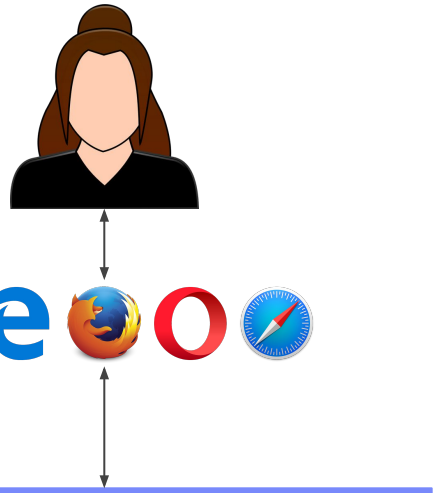
Motivation and Terminology

- **Definition**
  - “A Cloud is a type of **parallel** and **distributed system** consisting of a collection of interconnected and **virtualized computers** that are dynamically provisioned and **presented as one or more unified computing resource**(s) based on service-level **agreements** established through negotiation between the **service provider and consumers**” [*].



Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

Motivation and Terminology



SaaS
Docs, Email, Games, etc.

PaaS
APIs, SDKs, and more

IaaS
VMs, Storage, Network

databricks

Google Workspace

Canva

Red Hat OpenShift

Google App Engine

Elastic Beanstalk

NVIDIA

amazon web services

Motivation and Terminology

- **Transforming IT Industry/Landscape**
  - **Since ~2010** increasing move from on-prem to cloud resources
  - System **software licenses** become increasingly **irrelevant**
  - Few cloud providers dominate IaaS/PaaS/SaaS markets
  - **2023** revenue:
    - Microsoft ($ 111.6B)
    - Amazon AWS ($ 88B)
    - Oracle Cloud ($ 35.3B)
    - IBM Cloud ($ 20.8B)
    - Google Cloud (8.41B)
    - Alibaba Cloud ($ 3.789M)

Motivation and Terminology

- **Argument #1: Pay as you go:**
  - **No upfront cost** for infrastructure
  - **Variable** utilization ➔ over-provisioning
  - **Pay per use** or acquired resources
- **Argument #2: Economies of Scale**
  - **Lower cost** for purchasing and managing IT infrastructure at scale ➔ lower cost (applies to both HW resources and IT infrastructure/system experts)
  - Focus on **scale-out on commodity HW** over scale-up ➔ lower cost
- **Argument #3: Elasticity**
  - System can scale up according to demand
  - **Virtually unlimited resources** allows to reduce time as necessary ((Task time = Time x Resources))
- **Argument #4: Availability**
  - Resources are available 24x7

Characteristics and Deployment Models

- **Characteristics**

  On-demand self service. Users can access and manage computing resources themselves without requiring human assistance.

  Broad network access. Services are accessible from anywhere with an internet connection, using various devices.

  Resource pooling. Computing resources are shared among multiple users through virtualization, securely and efficiently

  Rapid elasticity. Resources can be scaled up or down quickly based on demand, in real time.

  Measured service. Resource usage is monitored and recorded to optimize consumption and bill only for what you use.

Cloud Computing Service Models

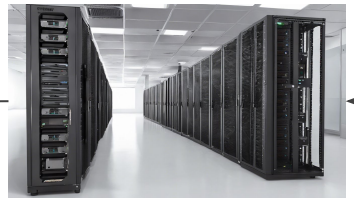
(computing as a utility)

Anatomy of a Data Center

CPUs

Servers (Sockets, RAM, Disks)

Rack
(16-64 servers + rack switch)



Cluster
(++Racks + Cluster switch)

Data center (Google Netherlands)

Fault Tolerance

- **Yearly Data Center Failures**
  - ~0.5 **overheating** (power down most machines in <5 mins, ~1-2 days)
  - ~1 **PDU failure** (~500-1000 machines suddenly disappear, ~6 hrs)
  - ~1 **rack-move** (plenty of warning, ~500-1000 machines powered down, ~6 hrs)
  - ~1 **network rewiring** (rolling ~5% of machines down over 2-day span)
  - ~20 **rack failures** (40-80 machines instantly disappear, 1-6 hrs)
  - ~5 **racks go wonky** (40-80 machines see 50% packet loss)
  - ~8 **network maintenances** (~30-minute random connectivity losses)
  - ~12 **router reloads** (takes out DNS and external vIPs for a couple minutes)
  - ~3 **router failures** (traffic rerouting, ~ 1 hour to stabilize.)
  - ~dozens of minor 30-second DNS issues
  - ~1000 **individual machine failures** (2-4% failure rate, at **least twice**)
  - ~thousands of hard drive failures (**1-5%** of all disks will **die**)

Fault Tolerance

- **Other Common Issues**
  - Configuration issues, partial SW updates, SW bugs.
  - Transient errors: no space left on device, memory corruption, stragglers.
- **Recap: Error Rates at Scale**
  - Cost-effective commodity hardware.
  - Error rate increases with increasing scale.
  - Fault Tolerance for distributed/cloud storage and data analysis.

Failures are inevitable. large-scale systems are designed with **fault tolerance** to ensure they can **detect**, **recover** from, and **mitigate** the impact of errors, ensuring **reliability** and **minimal downtime**.

Fault Tolerance

Cost-effective Fault Tolerance

- BASE:
  - **BAsically available:** system mostly operational even during failures, providing partial functionality if necessary.
  - **Soft state:** state of the system may change over time, even without new input, due to replication or recovery processes.
  - **Eventual consistency:** data may not be instantly consistent across all nodes but will eventually synchronize to the correct state

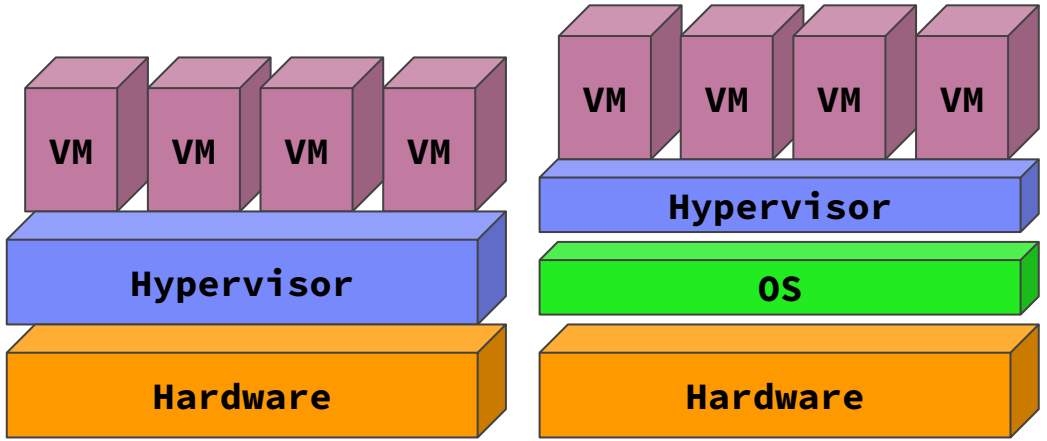
Fault Tolerance

Cost-effective Fault Tolerance

- Data corruption prevention
  - ECC (error correction codes)
  - CRC (cyclic redundancy check)
- Resilient storage
  - Replication (multiple data copies stored across nodes)
  - Erasure coding (data fragmentation + redundancy)
  - Checkpointing (save application state)
  - Lineage (data origin + dependencies)
  - Resilient compute: task re-execution / speculative execution

Virtualization I

- **Native Virtualization**
  - **Simulates** most of the **HW interface**
  - **Unmodified guest OS** to run **as if it were on real HW**
  - **Examples:** VMWare, Virtualbox, AMI (HVM)
- **Advantages**
  - Guess OS **as it is!**
  - High compatibility
  - High security



Virtualization II

- **Para-virtualization**
  - **No HW** interface **simulation**, but special API (**hypercalls**) replacing hardware instructions.
  - Requires modified guest OS to use hyper calls, trapped by hypervisor
  - Examples: Xen, KVM, Hyper-V, AMI (PV)
- **Advantages**
  - Faster and more efficient than native virtualization (skips the overhead of hardware simulation).
  - Ideal for environments where **performance is critical**, and **modifying the OS** is acceptable.

Virtualization III

- **OS-level Virtualization**
  - Allows multiple isolated environments (virtual servers or containers) to run on the OS.
  - Guest OS appears isolated but same as host OS
  - Examples: Solaris/Linux containers, **Docker**
- **Application-level Virtualization**
  - App executed within a virtualized runtime environment that abstracts away dependencies on the host system.
  - Examples: **Java VM (JVM),** Ethereum VM (EVM), Python virtualenv

In brief

Topic	Native Virtualization	Para Virtualization	OS-level Virtualization	Application-level Virtualization
Definition	Simulates hardware fully	Uses hypercalls, no HW emulation	Containers sharing host OS	Virtualizes specific applications
Guest OS	Unmodified	Modified	Same as host OS	NA
Performance	Moderate (HW overhead)	High	Very high	High
Examples	VMware, Parallels	Xen, KVM	Docker, Linux Containers	JVM, Python virtualenv
Use Case	Mixed OS environments	High-performance VMs	Cloud-native apps	App portability across platforms

BONUS: Containerization

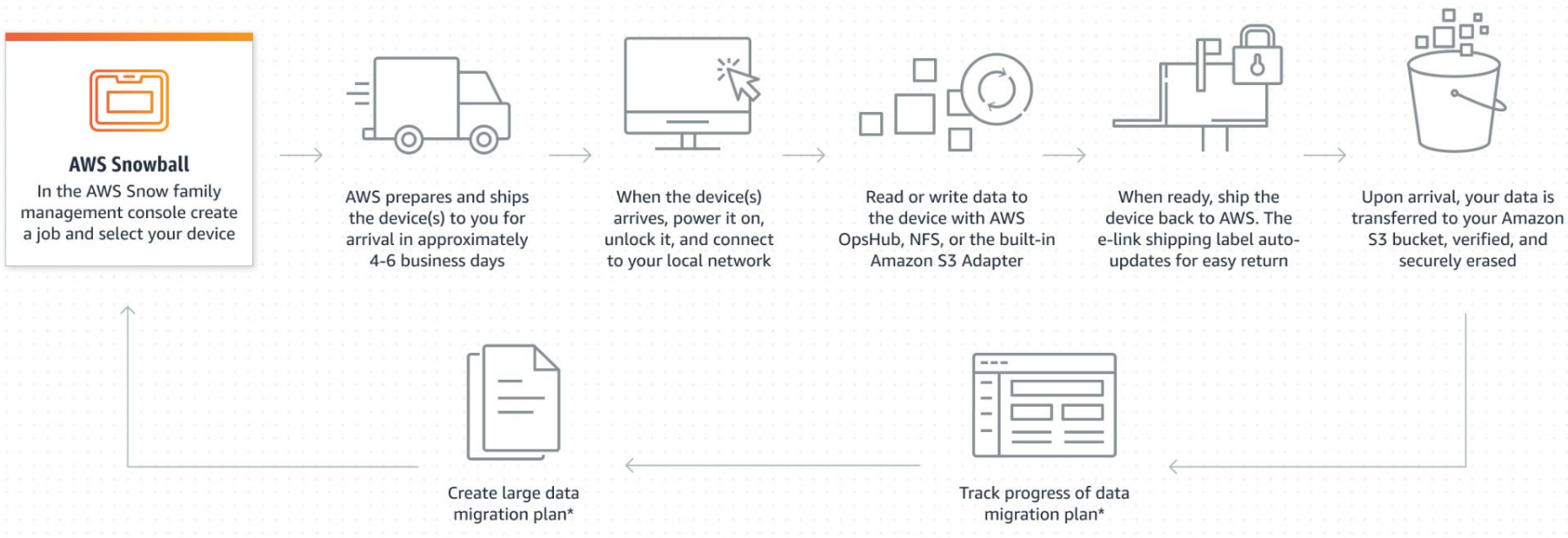
- **Docker Containers**
  - Shipping container analogy
    - Arbitrary, self-contained goods, standardized units
    - Containers reduced loading times ➔ efficient international trade
  - **Self-contained package** of necessary SW and data (read-only image)
  - **Lightweight virtualization** w/ shared OS and resource isolation via control groups

BONUS: Containerization

- **Cluster Schedulers**
  - Container orchestration: scheduling, deployment, and management
  - Resource negotiation with clients
  - Typical resource bundles (CPU, memory, device)
  - Examples: Kubernetes, Mesos, (YARN), Amazon ECS, Microsoft ACS, Docker Swarm

How far can we go?

- **Snowmobile Service:** Data transfer on-premise → cloud via 100PB trucks w/ 1Gb link



How far can we go?

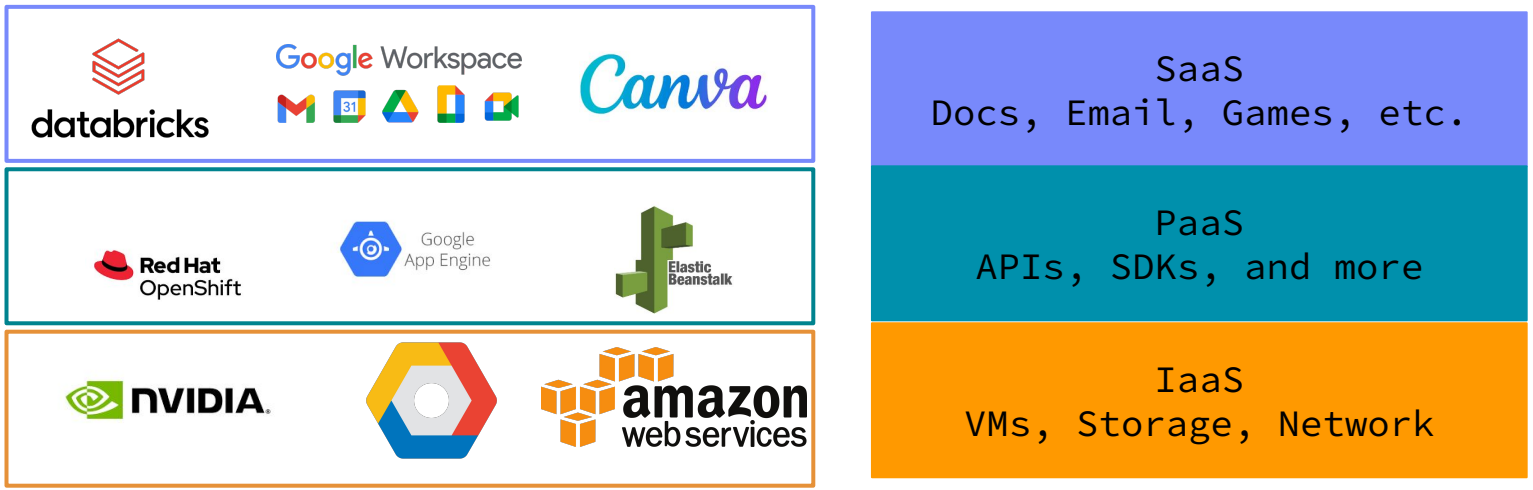
Microsoft Underwater Datacenter

Cloud vs other HPC Technologies

Computing Techniques	Features
Cloud Computing	Cost efficient, almost unlimited storage, backup and recovery, easy deployment
Grid Computing	Efficient use of idle resources, modular, parallelism can be achieved, handles complexity
Cluster Computing	Reduced cost, processing power, improved network technology, scalability, availability

Manvi, S. S., & Shyam, G. K. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. Journal of network and computer applications, 41, 424-440.

Cloud Computing Service Models



IaaS

IaaS
VMs, Storage, Network

Combination of hosting, hardware provisioning, basic services, and other services needed to run a cloud.

Uses of IaaS:

- Provides access to shared resources on need basis, without revealing details like location and hardware to clients.
- Provides details like server images on demand, storage, queuing, and information about other resources.
- Offers full control of server infrastructure, not limited specifically to applications, instances and containers.

Major issues associated with IaaS:

- Resource management.
- Internet access.
- Virtualization.
- Data management.
- APIs.
- Interoperability.

PaaS

PaaS
APIs, SDKs, and more

Provision of a computing platform and the provision and deployment of the associated set of software applications (called a solution stack).

Uses of PaaS:

- Provides developers with tools, frameworks, and runtime environments to build and deploy applications efficiently.
- Removes the need to worry about hardware, operating systems, or server management.
- While IaaS provides control over server infrastructure, PaaS limits control to what's essential for applications, offering an environment tuned specifically for application development and management.

Major issues associated with PaaS:

- Provider reliability
- Resource management
- Internet dependency
- Performance
- Scalability costs

SaaS

SaaS
Docs, Email, Games, etc.

Software as a Service is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network.

Uses of SaaS:

- On-demand software access.
- Scalability
- Cost-effective solution
- Cross device availability
- Collaboration and real-time updates

Major issues associated with SaaS:

- Internet dependency
- Performance
- Long-term cost
- Data Security and Privacy

Other “layers”: Serverless Computing

A cloud computing paradigm where developers can build and run applications without managing server infrastructure. Operational responsibilities like fault tolerance, scaling, and resource allocation are handled by cloud providers.

Uses of SaaS:

- **Function-as-a-Service (FaaS)**: Runs stateless, event-driven functions in response to triggers.
- **Serverless Databases:** Auto-scale capacity as needed and hibernate during periods of inactivity, reducing costs while maintaining availability.

Kounev, S., Herbst, N., Abad, C. L., Iosup, A., Foster, I., Shenoy, P., ... & Chien, A. A. (2023). Serverless computing: What it is, and what it is not?. Communications of the ACM, 66(9), 80-92.


research
Serverless Computing: What It Is, and What It Is Not?

Other “layers”: Serverless

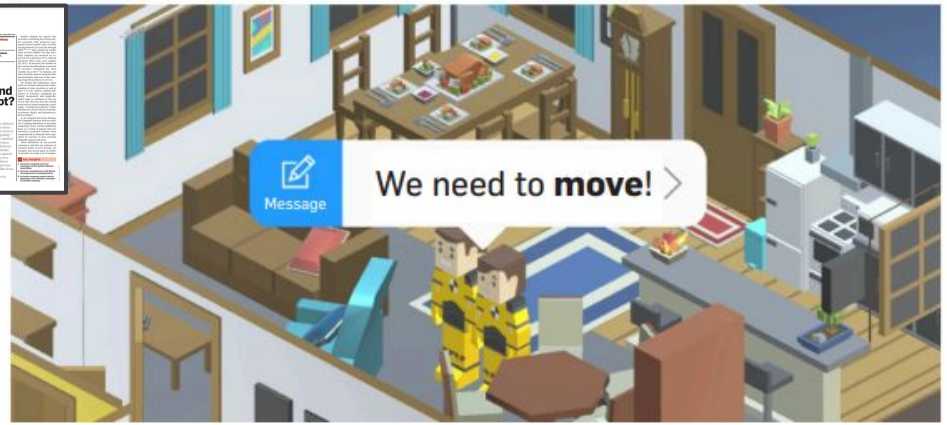
Serverless

IaaS/PaaS/SaaS

Self-hosting


research

Serverless Computing: What It is, and What It Is Not?



	Packaging	Delivery	Operations	Legal	Financial	Personnel
Modern movers	All objects	Any route	All decisions	All covered	Fine-grained Utilization-based	Small team
Traditional movers	Limited support	Major roads	Basic	Basic	Coarse-grained	Large team
Moving it yourself (with family and friends)	Yourself	Yourself	Yourself	Yourself	Yourself	Yourself

GP-GPU, Fog, and Edge Computing

GP-GPU computing involves the usage of a GPU—a processor originally designed for rendering graphics—as a highly parallel programmable processor to handle computationally intensive tasks across various domains.

Uses of GP-GPU Computing:

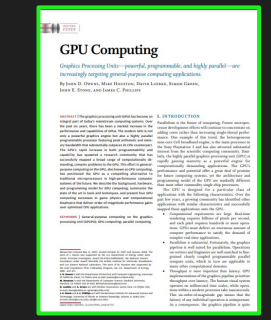
- **Machine Learning and AI:**
  - **Training and inference for deep learning models.**
  - **Accelerating neural network computations.**

- Scientific Applications:
  - Protein folding simulations.
  - Computational biophysics and molecular dynamics.
  - Fluid dynamics and heat transfer.

- Real-Time Graphics:
  - Game physics and rendering.
  - Simulation of physical environments in games.

- Data Processing:
  - Sorting and searching large datasets.

- General Numerical Computations:
  - Differential equation solvers.



GPU Computing

Graphics Processing Units—powerful, programmable, and highly parallel—are increasingly targeting general-purpose computing applications.

By John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. Proceedings of the IEEE, 96(5), 879-899.

Providers (AWS, NVIDIA, Microsoft, Google, IBM, Oracle, Huawei, Tencent)

AWS:

- EC2 with GPU Instances (NVIDIA Tesla V100, T4, K80)
- Deep Learning AMIs (Amazon Machine Images).
- AWS Lambda for GPU-powered serverless workflows.

NVIDIA Cloud

- NVIDIA GPU Cloud (Focused on NVIDIA GPUs, including A100 and V100)
- Optimized software stacks for AI, HPC, and data analytics.
- Pre-built containers for machine learning frameworks.

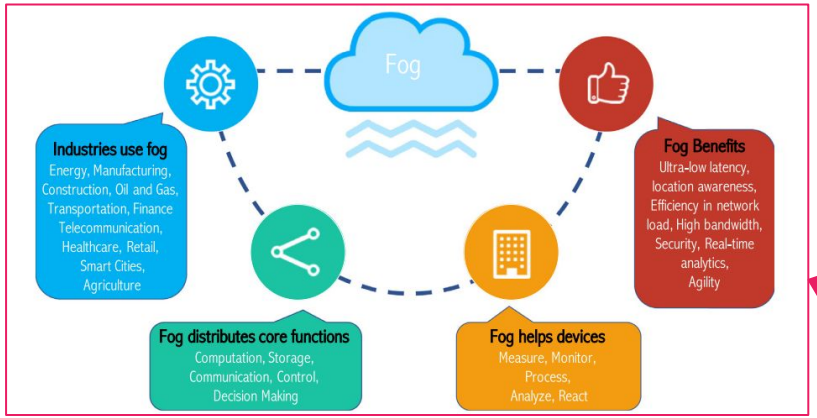
Aspect	Pros	Cons
Performance	High computational power for parallel processing and data-intensive tasks.	Limited performance on tasks that are inherently serial.
Parallelism	Highly efficient for workloads with significant parallelism, such as AI, ML, and scientific simulations.	Inefficient for workloads with low parallelism.
Energy Efficiency	Better performance-per-watt compared to CPUs for parallel tasks.	High energy consumption for sustained operations, especially in large deployments.
Cost Efficiency	Cost-effective for large-scale computations due to faster task completion.	High upfront costs for purchasing and maintaining hardware.
Flexibility	Compatible with various frameworks like CUDA, OpenCL, TensorFlow, and PyTorch.	Requires specialized knowledge for effective programming and optimization.
Applications	Versatile applications in AI, gaming, scientific simulations, and data analytics.	Limited utility in general-purpose or low-demand computing tasks.
Hardware Utilization	Maximizes hardware use with dense and parallel computations.	Requires workloads to be carefully structured to utilize GPU resources efficiently.
Scalability	Can be scaled by adding more GPUs for demanding workloads.	Limited by physical space, thermal constraints, and power requirements.
Memory Bandwidth	High memory bandwidth enables faster data processing for large datasets.	Smaller memory size compared to CPUs can limit large-scale applications.
Programming Tools	Rich ecosystem of programming tools (e.g., CUDA, OpenCL) for developing optimized solutions.	Programming complexity and debugging can be challenging for beginners.
Latency	Excellent for high-throughput applications with less sensitivity to latency.	Latency can be a bottleneck in real-time applications requiring immediate responses.
Hardware Lifespan	Long lifespan with consistent performance for specific tasks.	Rapid hardware obsolescence due to fast advancements in GPU technology.
Data Transfer	Efficient for tasks with high computation-to-data transfer ratios.	Slow data transfer between CPU and GPU can create bottlenecks in some applications.

Aspect	Pros	Cons
Performance	High computational power for parallel processing and data-intensive tasks.	Limited performance on tasks that are inherently serial.
Parallelism	Highly efficient for workloads with significant parallelism, such as AI, ML, and scientific simulations.	Inefficient for workloads with low parallelism.
Energy Efficiency	Better performance-per-watt compared to CPUs for parallel tasks.	High energy consumption for sustained operations, especially in large deployments.
Cost Efficiency	Cost-effective for large-scale computations due to faster task completion.	High upfront costs for purchasing and maintaining hardware.
Flexibility	Compatible with various frameworks like CUDA, OpenCL, TensorFlow, and PyTorch.	Requires specialized knowledge for effective programming and optimization.
Applications	Versatile applications in AI, gaming, scientific simulations, and data analytics.	Limited utility in general-purpose or low-demand computing tasks.
Hardware Utilization	Maximizes hardware use with dense and parallel computations.	Requires workloads to be carefully structured to utilize GPU resources efficiently.
Scalability	Can be scaled by adding more GPUs for demanding workloads.	Limited by physical space, thermal constraints, and power requirements.
Memory Bandwidth	High memory bandwidth enables faster data processing for large datasets.	Smaller memory size compared to CPUs can limit large-scale applications.
Programming Tools	Rich ecosystem of programming tools (e.g., CUDA, OpenCL) for developing optimized solutions.	Programming complexity and debugging can be challenging for beginners.
Latency	Excellent for high-throughput applications with less sensitivity to latency.	Latency can be a bottleneck in real-time applications requiring immediate responses.
Hardware Lifespan	Long lifespan with consistent performance for specific tasks.	Rapid hardware obsolescence due to fast advancements in GPU technology.
Data Transfer	Efficient for tasks with high computation-to-data transfer ratios.	Slow data transfer between CPU and GPU can create bottlenecks in some applications.

Fog Computing

Bridges the gap between the cloud and end devices (e.g., IoT nodes) by enabling computing, storage, networking, and data management on network nodes within the close vicinity of IoT devices.

Computation, storage, networking, decision making, and data management not only occur in the cloud, but also occur along the IoT-to-Cloud path as data traverses to the cloud (preferably **close to the IoT devices**).



Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., ... & Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 98, 289-330.

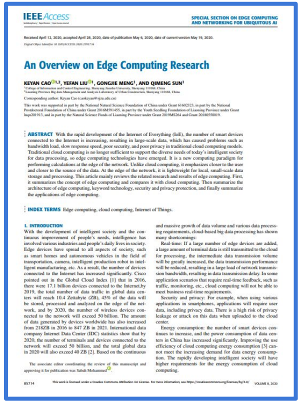
Edge Computing

Enhances the management, storage, and processing capabilities for data generated by connected devices like mobile cloud computing (MCC) does for mobile devices.

The main difference with MCC is that edge computing operates at the edge of the network, positioned near IoT devices—typically one hop away.

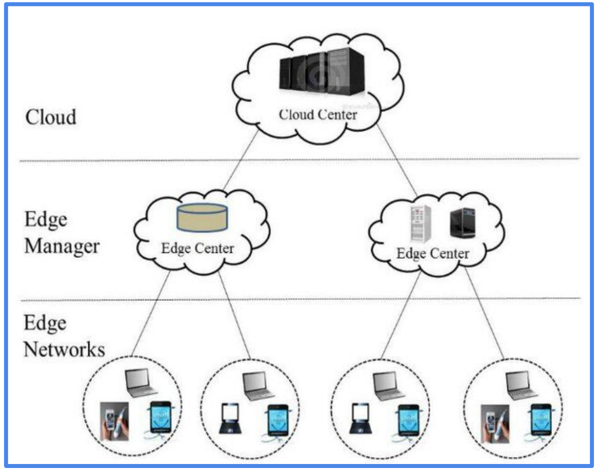
The edge is not directly on the IoT devices!

Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An overview on edge computing research. *IEEE access*, *8*, 85714-85728.


IEEE Access
SPECIAL SECTION ON EDGE COMPUTING AND NETWORKING FOR UBIQUITOUS AI

An Overview on Edge Computing Research

KEYAN CAO, YEFAN LIU, GONGJIE MENG, AND QIMENG SUN



Ren, Y., Zhu, F., Qi, J., Wang, J., & Sangaiah, A. K. (2019). Identity management and access control based on blockchain under edge computing for the industrial internet of things. Applied Sciences, 9(10), 2058.

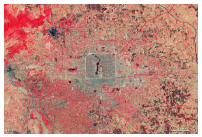
A short view about a public cloud...

Connecting...

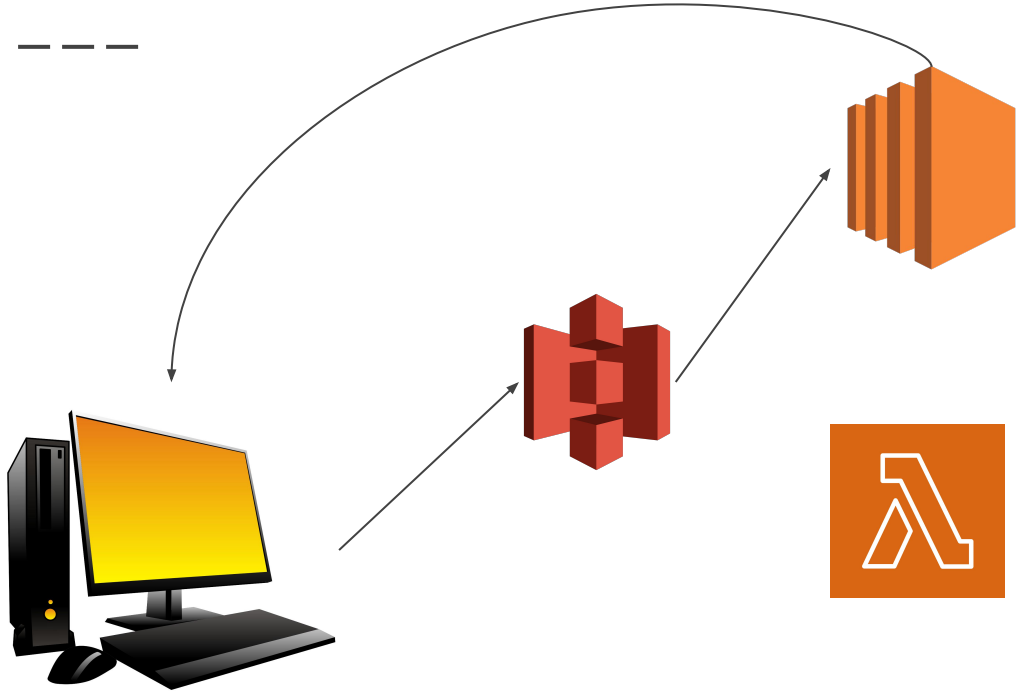
https://us-west-2.console.aws.amazon.com/

```
ssh -i <my-key> ubuntu@<my-instance-ip-public-address>
```

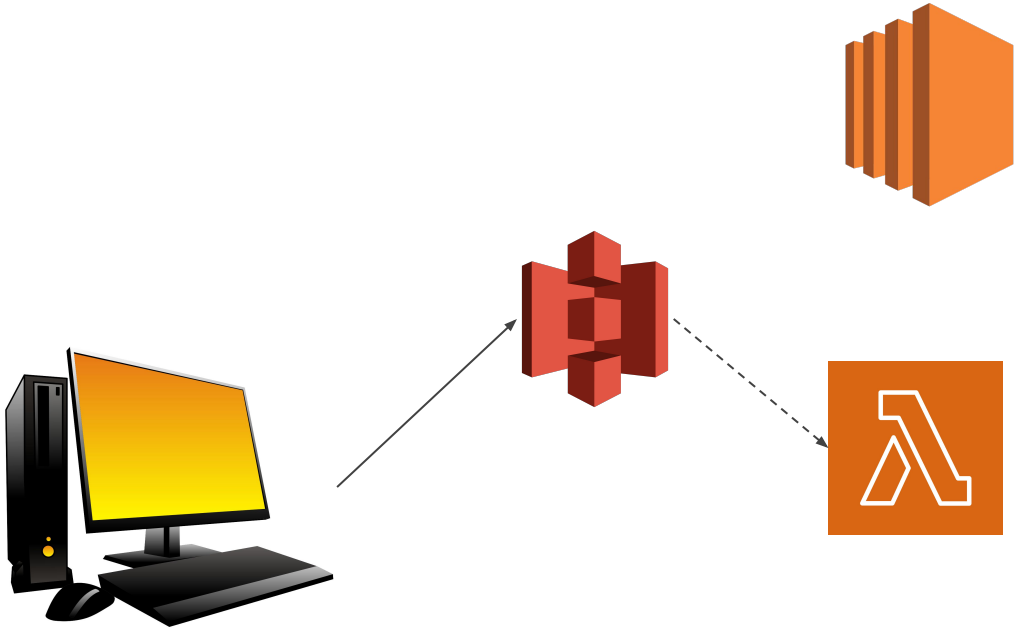
A pipeline



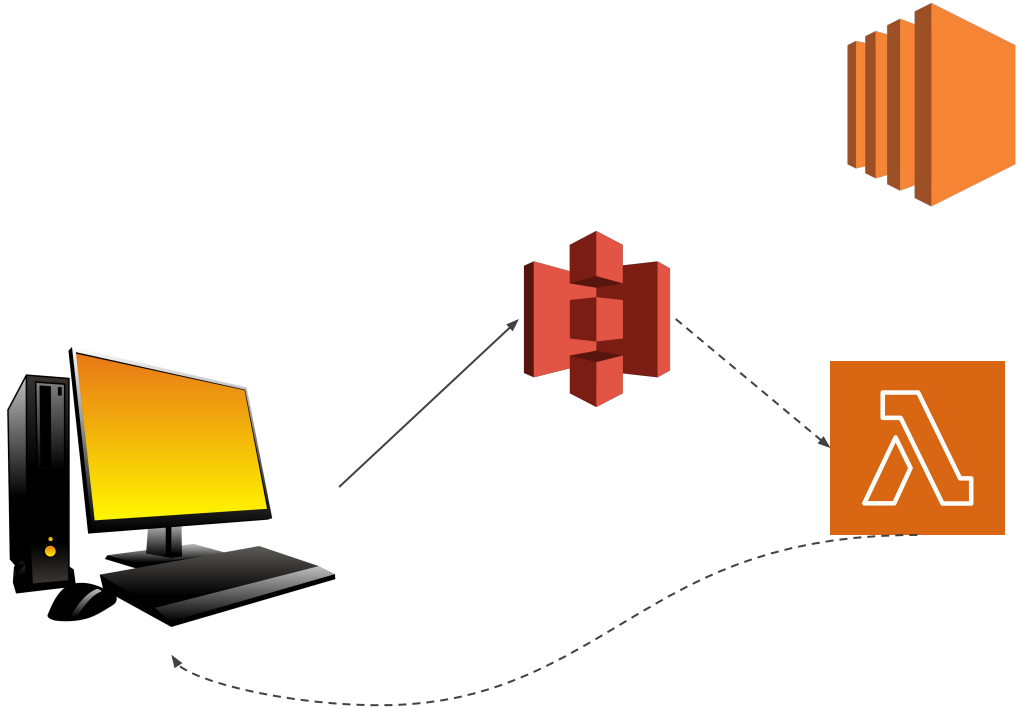
A pipeline



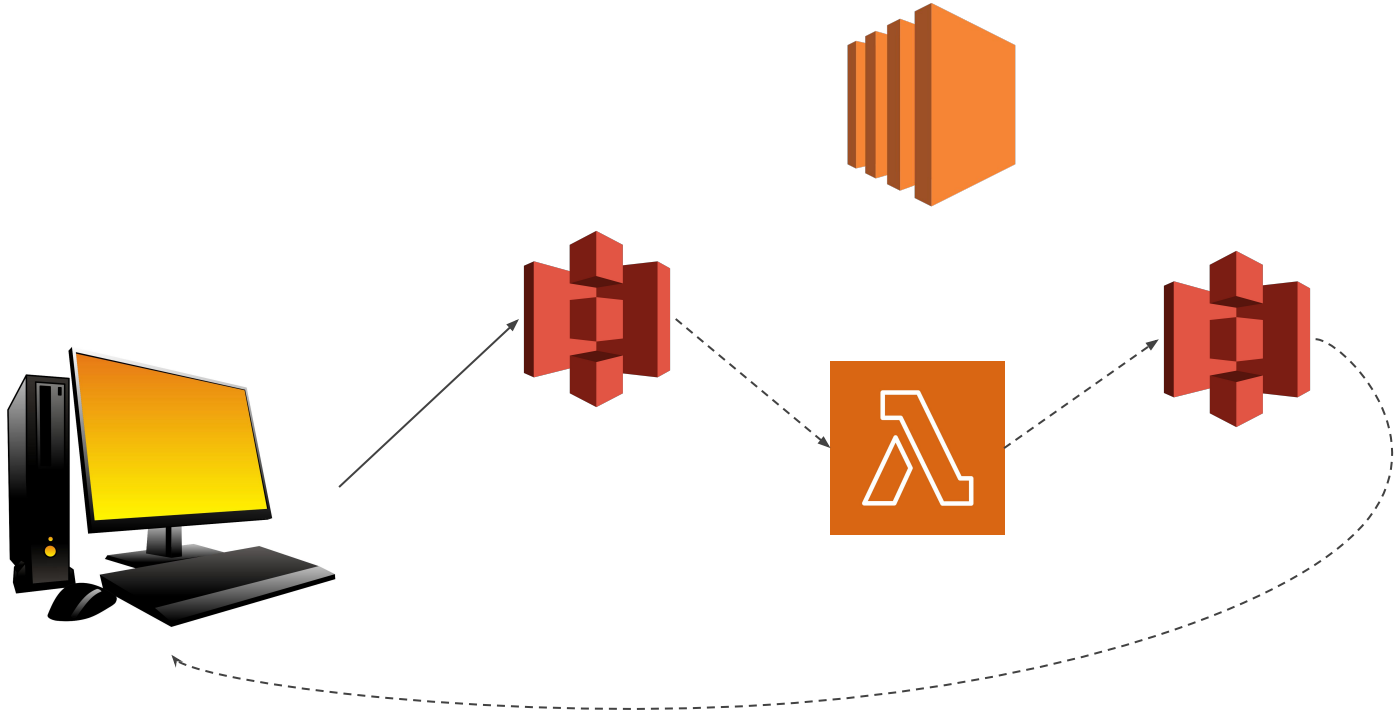
A pipeline



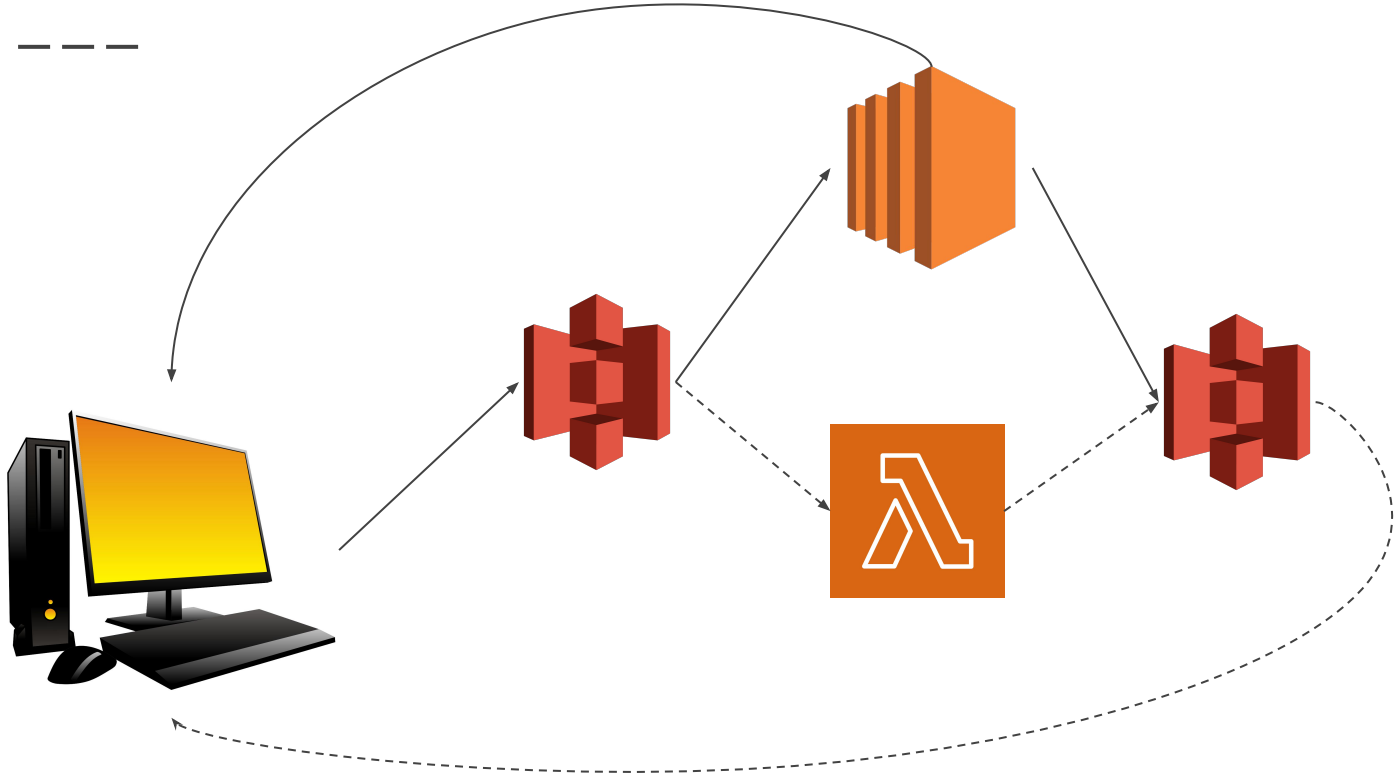
A pipeline



A pipeline



A pipeline



Summary and Q&A

Summary and Q&A

- **Summary and Q&A**
  - Cloud Computing Motivation and Terminology
  - Cloud Computing Service Models
  - Cloud, Fog, and Edge Computing
- **Next Lectures**
  - 08 Cloud Resource Management and Scheduling [Dec 06]

Vielen Dank!