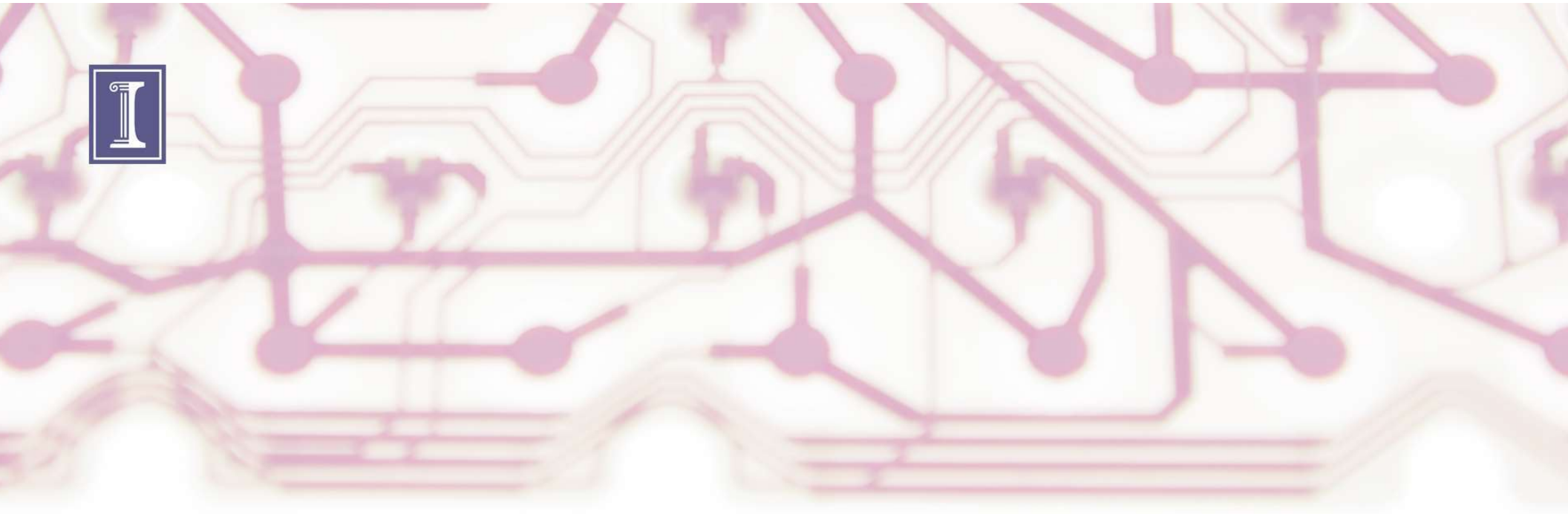
Staging

- A client request to create a file does not reach NameNode immediately
- HDFS client caches the data into a temporary file. When the data reaches an HDFS block size, the client contacts the NameNode
- NameNode inserts the filename into its hierarchy and allocates a data block for it
- The NameNode responds to the client with the identity of the data node and the destination of the replicas (data nodes) for the block
- Then the client flushes it from its local memory

Application Programming Interface

- HDFS provides Java API for application to use
- Python access is also used in many applications
- A C language wrapper for Java API is also available
- A command line interface provided in Hadoop
- The syntax of the commands is similar to bash
- Example: to create a directory /foodir
- /bin/hadoop dfs –mkdir /foodir
- An HTTP browser can be used to browse the files of an HDFS instance



CLOUD COMPUTING APPLICATIONS

YARN, MESOS

Roy Campbell & Reza Farivar

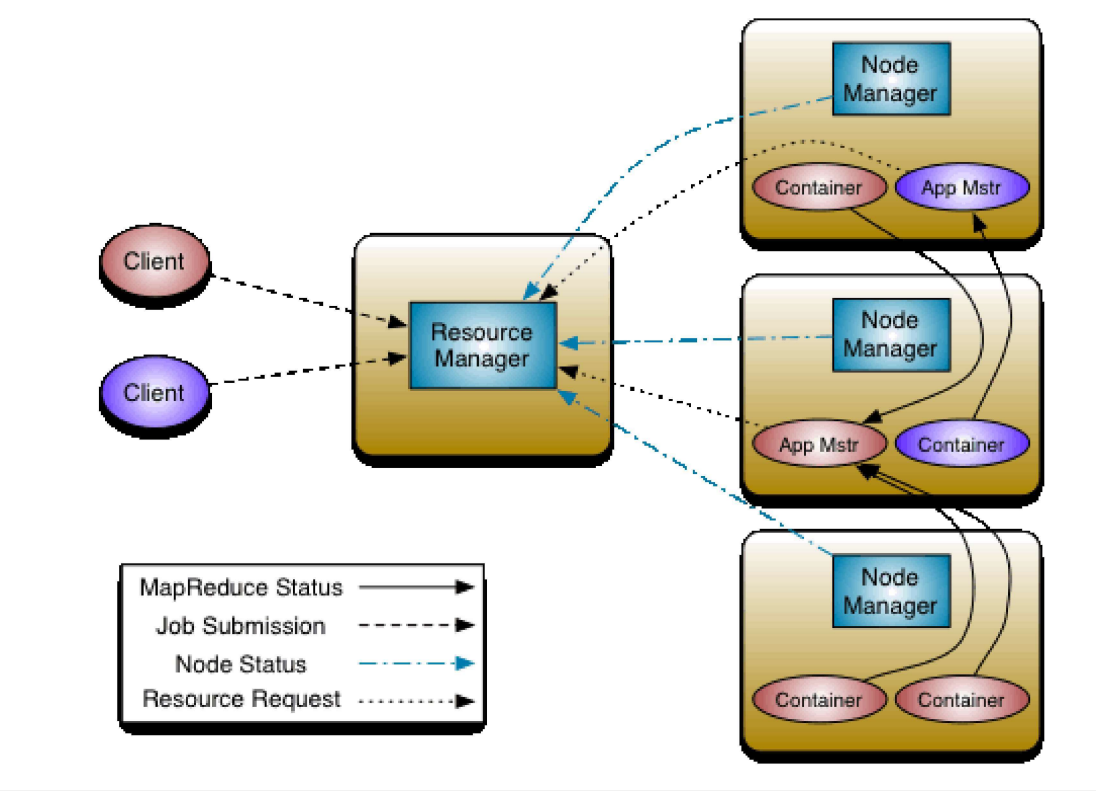
Overview

- Mesos was built to be a scalable global resource manager for the entire data center (Berkeley 2007)
  - Hadoop, MPI,

- YARN was created out of the necessity to scale Hadoop.

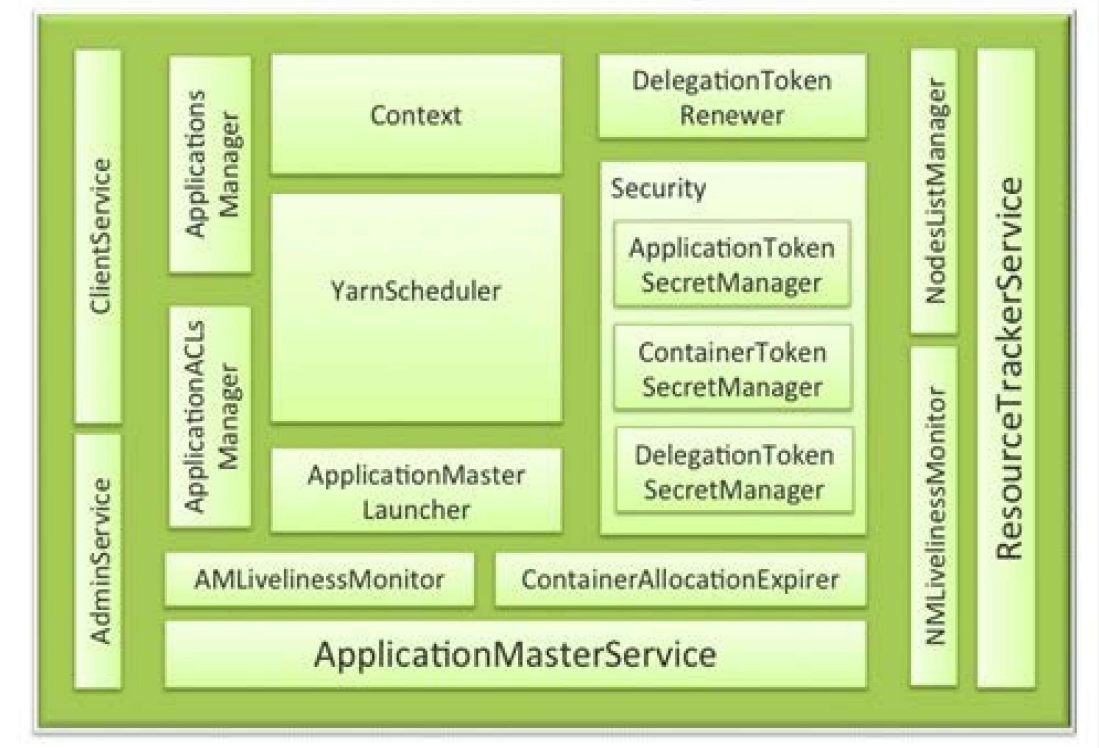
YARN ResourceManager

- Scheduler
  - pluggable policy partitions cluster resources among the various queues, and applications
  - CapacityScheduler and the FairScheduler
- NodeManagers take instructions from the ResourceManager and manage resources available on a single node.
- ApplicationsManager negotiates resources with the resourcemanager and for working with the nodemanagers to start the containers.
- Compatible API with Map/Reduce



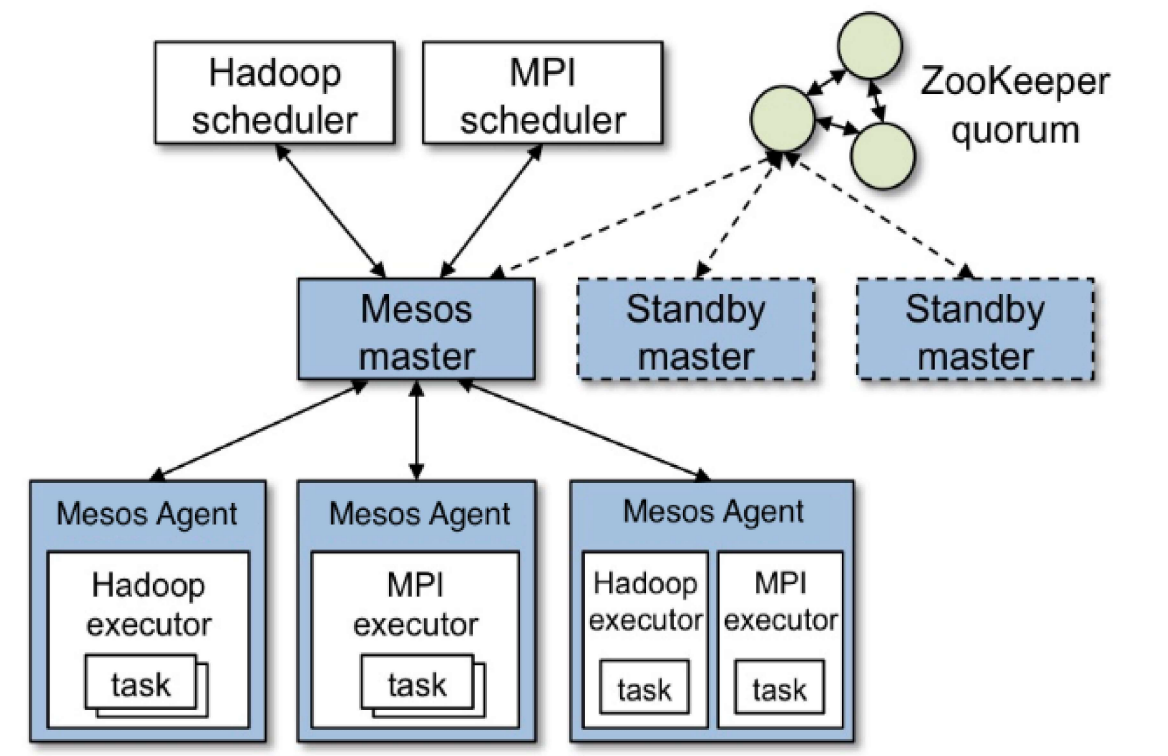
Insides of Yarn

- ResourceTrackerService
  - responds to RPCs from all the nodes.
  - Registration of new nodes, rejection of requests from any invalid/decommissioned nodes,
  - obtaining node-heartbeats and forward them over to the YarnScheduler.



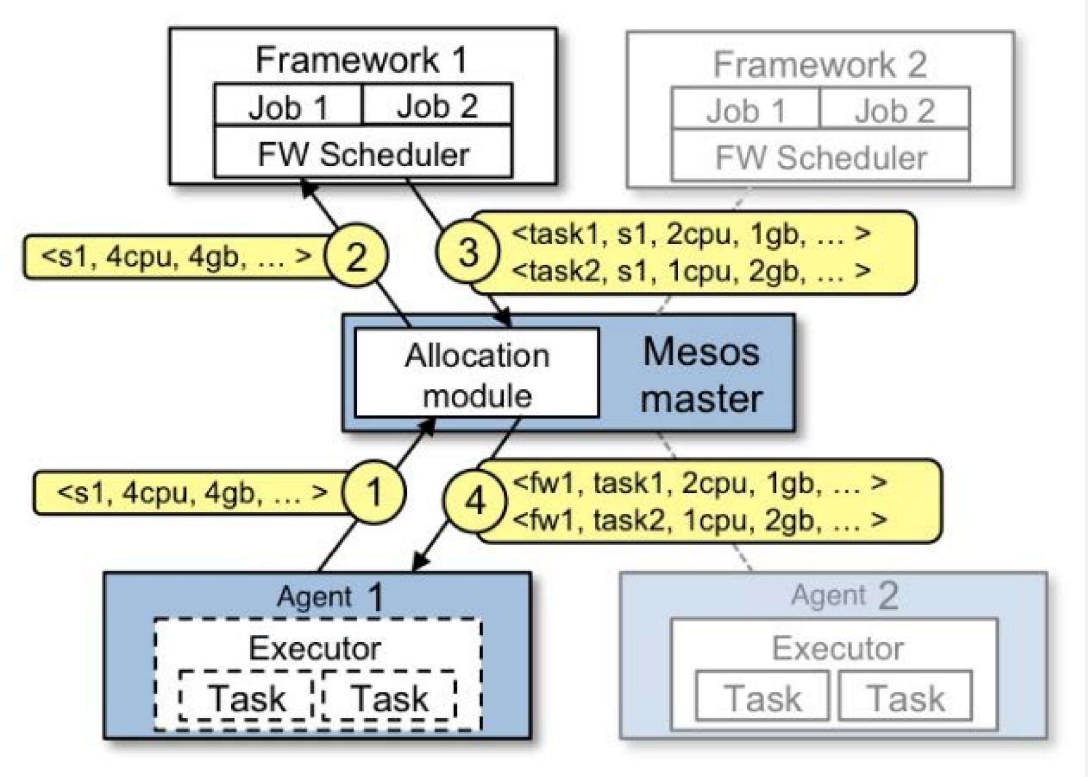
Mesos Architecture

- The master decides *how many* resources to offer to each framework according to a given organizational policy, such as fair sharing or strict priority.



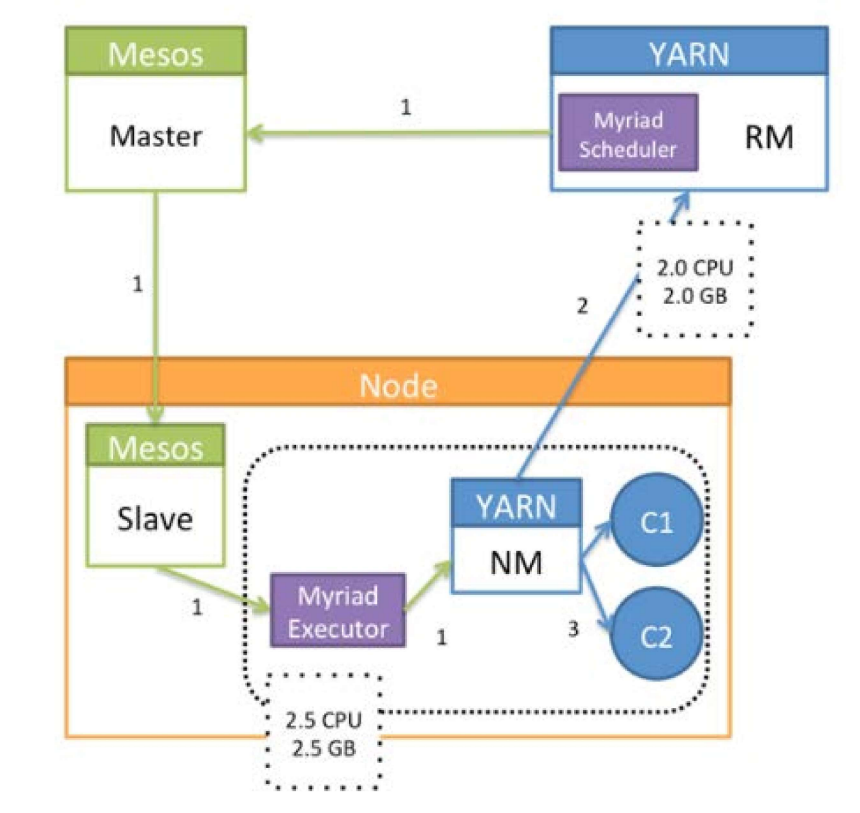
Mesos Resource Offer

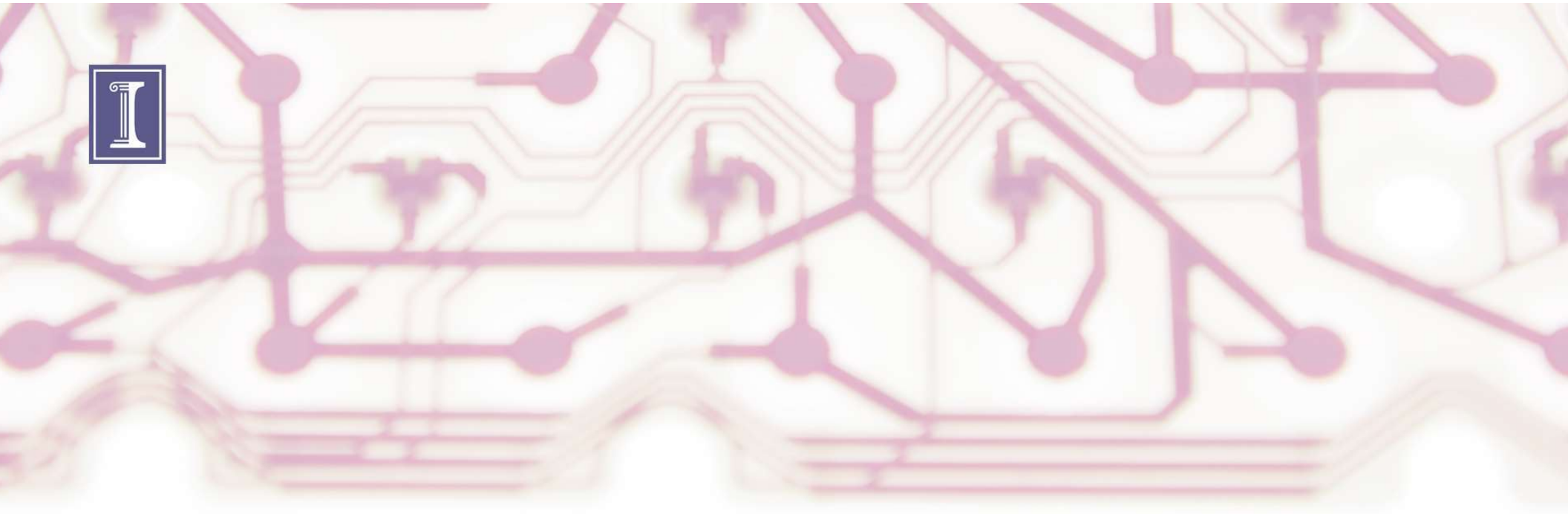
1. Agent 1 reports to the master that it has 4 CPUs and 4 GB of memory free. The master then invokes the allocation policy module, which tells it that framework 1 should be offered all available resources.
2. The master sends a resource offer describing what is available on agent 1 to framework 1.
3. The framework's scheduler replies to the master with information about two tasks to run on the agent, using <2 CPUs, 1 GB RAM> for the first task, and <1 CPUs, 2 GB RAM> for the second task.
4. Finally, the master sends the tasks to the agent, which allocates appropriate resources to the framework's executor, which in turn launches the two tasks (depicted with dotted-line borders in the figure). Because 1 CPU and 1 GB of RAM are still unallocated, the allocation module may now offer them to framework 2.



Mesos and YARN Convergence

- **Project Myriad**
- Mesos framework and a YARN scheduler that enables Mesos to manage YARN resource requests.





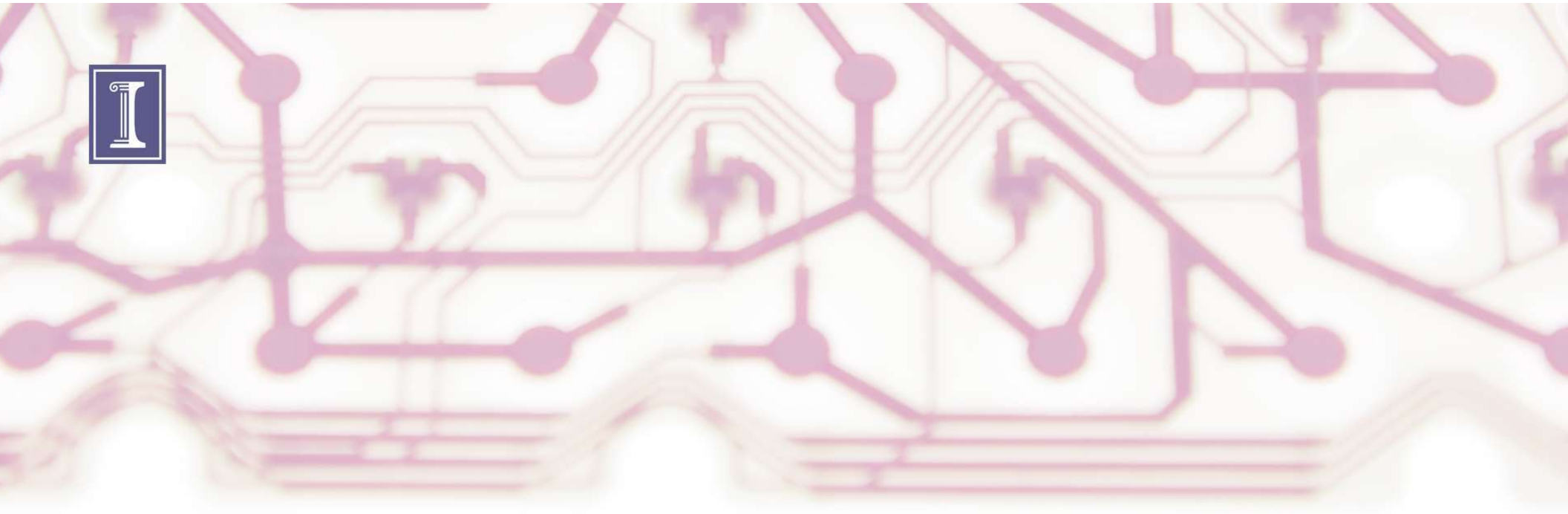
CLOUD COMPUTING APPLICATIONS

Introduction to Distros

Roy Campbell & Reza Farivar

Distribution of Data Analytics Tools

- Hortonworks
- Cloudera
- MapR



CLOUD COMPUTING APPLICATIONS

Hortonworks

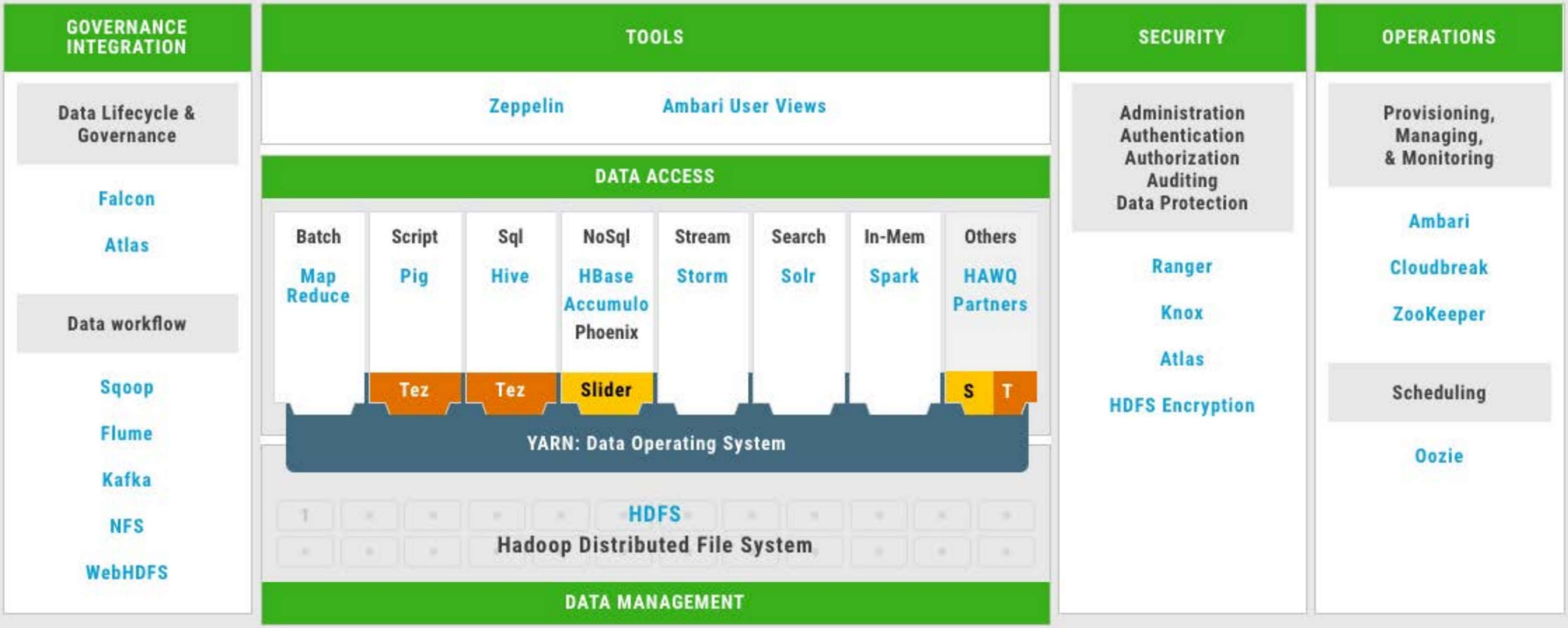
Roy Campbell & Reza Farivar

Pieces of the Puzzle

Connected Data Strategy

- HDP: Apache Hadoop® is an open source framework for distributed storage and processing of large sets of data on commodity hardware. Hadoop enables businesses to quickly gain insight from massive amounts of structured and unstructured data.
- HDF enables real-time data collection, curation, analysis and delivery of data to and from any device, source or system, either on-premise and in the cloud

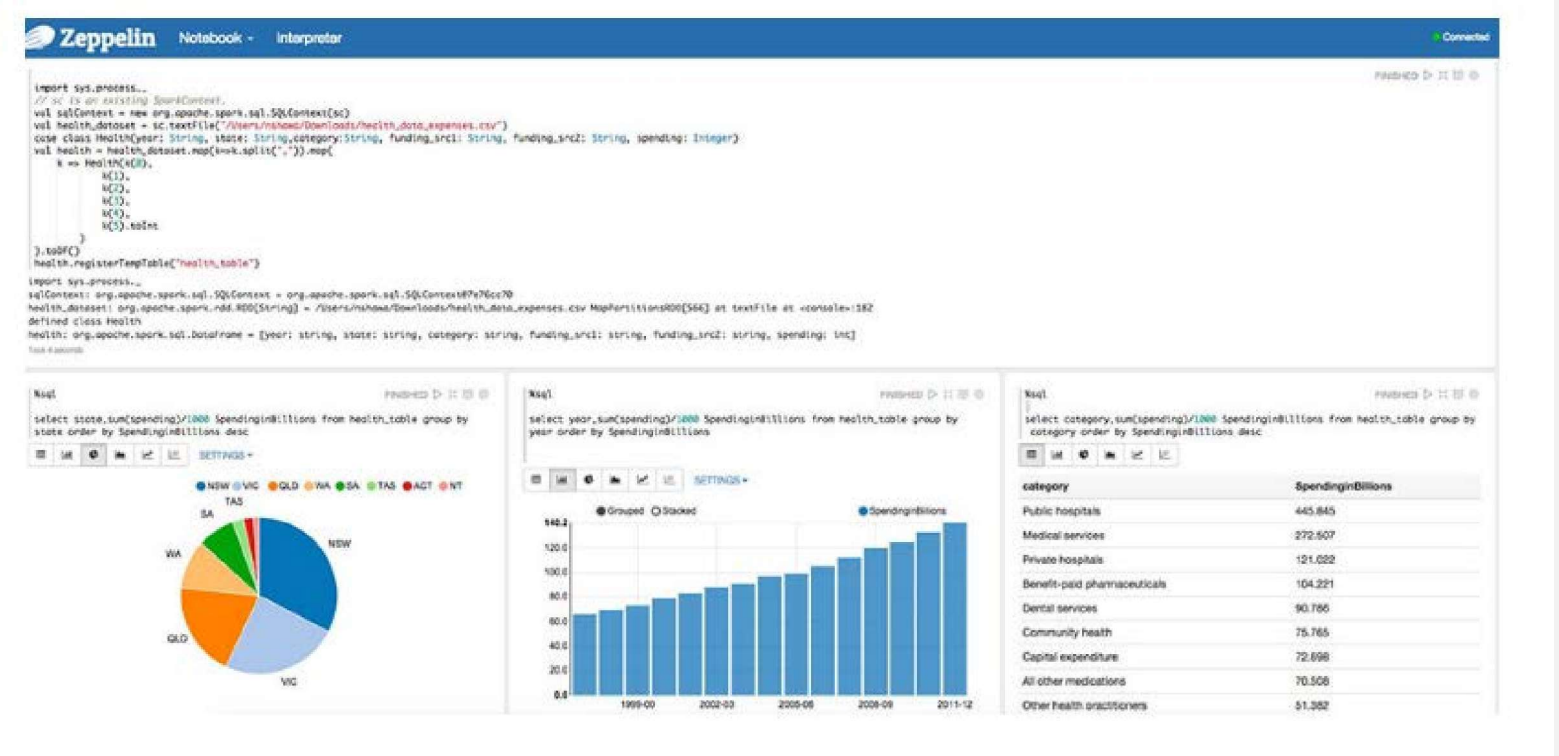
Hortonworks Data Platform (HDP)



HDP Tools

- Apache Zeppelin: open web-based notebook that enables interactive data analytics
- Apache Ambari: source management platform for provisioning, managing, monitoring and securing Apache Hadoop clusters.

Apache Zeppelin (Python, Spark, Scala…)



Apache Zeppelin (Python, Spark, Scala…)

User View	Description
Tez	The **Tez View** helps you understand and optimize your cluster resource usage. Using the view, you can optimize and accelerate individual SQL queries or Pig jobs to get the best performance in a multi-tenant Hadoop environment.
Hive	**Hive View** allows the user to write & execute SQL queries on the cluster. It shows the history of all Hive queries executed on the cluster whether run from Hive view or another source such as JDBC/ODBC or CLI. It also provides graphical view of the query execution plan. This helps the user debug the query for correctness and for tuning the performance. It integrates **Tez View** that allows the user to debug any Tez job, including monitoring the progress of a job (whether from Hive or Pig) while it is running. This view contribution can be found here.
Pig	**Pig View** is similar to the Hive View. It allows writing and running a Pig script. It has support for saving scripts, and loading and using existing UDFs in scripts. This view contribution can be found here.
Capacity Scheduler	**Capacity Scheduler View** helps a Hadoop operator setup YARN workload management easily to enable multi-tenant and multi-workload processing. This view provisions cluster resources by creating and managing YARN queues. This view contribution can be found here.
Files	**Files View** allows the user to manage, browse and upload files and folders in HDFS. This view contribution can be found here.

Data Access

- YARN: Data Operating System
  - MapReduce. Batch application framework for structured and unstructured data
  - Pig. Script Extract-transform-load (ETL) data pipelines, Research on raw data, and Iterative data processing.
  - SQL. Hive interactive SQL queries over petabytes of data in Hadoop
  - NoSql (Hbase Accumulo) non-relational (NoSQL) database on top of HDFS
  - Storm (Stream) distributed real-time --large volumes of high-velocity data.
  - Solr (Search) -- full-text search and near real-time indexing.
  - Spark (In-Mem)
- Data Management
  - Hadoop Distributed File System (HDFS)

MapReduce

Benefit	Description
Simplicity	Developers can write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run
Scalability	MapReduce can process petabytes of data, stored in HDFS on one cluster
Speed	Parallel processing means that MapReduce can take problems that used to take days to solve and solve them in hours or minutes
Recovery	MapReduce takes care of failures. If a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.
Minimal data motion	MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed.

Pig

Characteristic	Benefit
Extensible	Pig users can create custom functions to meet their particular processing requirements
Easily programmed	Complex tasks involving interrelated data transformations can be simplified and encoded as data flow sequences. Pig programs accomplish huge tasks, but they are easy to write and maintain.
Self-optimizing	Because the system automatically optimizes execution of Pig jobs, the user can focus on semantics.

Hive

Feature	Description
Familiar	Query data with a SQL-based language
Fast	Interactive response times, even over huge datasets
Scalable and Extensible	As data variety and volume grows, more commodity machines can be added, without a corresponding reduction in performance
Compatible	Works with traditional data integration and data analytics tools.

Nosql HBase

Characteristic	Benefit
Fault tolerant	• **Replication** across the data center • **Atomic and strongly consistent** row-level operations • **High availability** through automatic failover • **Automatic sharding and load balancing**s of tables
Fast	• **Near real time lookups** • **In-memory caching** via block cache and bloom filters • **Server side processing** via filters and co-processors
Usable	• **Data model** accommodates wide range of use cases • **Metrics exports** via File and Ganglia plugins • **Easy Java API** as well as Thrift and REST gateway APIs

Nosql Accumulo

Feature	Benefit
Table design and configuration	• Includes cell tables for cell-level access control • Large rows need not fit into memory
Integrity and availability	• Master fail-over with ZooKeeper locks • Write-ahead logs for recovery • Scalable master metadata store • Fault tolerant executor (FATE)
Performance	• Relative encoding to compress similar consecutive keys • Speed long scans with parallel server threads • Cache recently scanned data
Data Management	• Group columns within a single file • Automatic tablet splitting and rebalancing • Merge tablets and clone tables

Storm

Feature	Description

Fast – benchmarked as processing one million 100 byte messages per second per node

Scalable – with parallel calculations that run across a cluster of machines

Fault-tolerant – when workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node.

Reliable – Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.

Easy to operate – standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

Solr

Feature	Description

Advanced full-text search
Near real-time indexing
Standards-based open interfaces like XML, JSON and HTTP
Comprehensive HTML administration interfaces
Server statistics exposed over JMX for monitoring
Linearly scalable, auto index replication, auto failover and recovery
Flexible and adaptable, with XML configuration

Spark

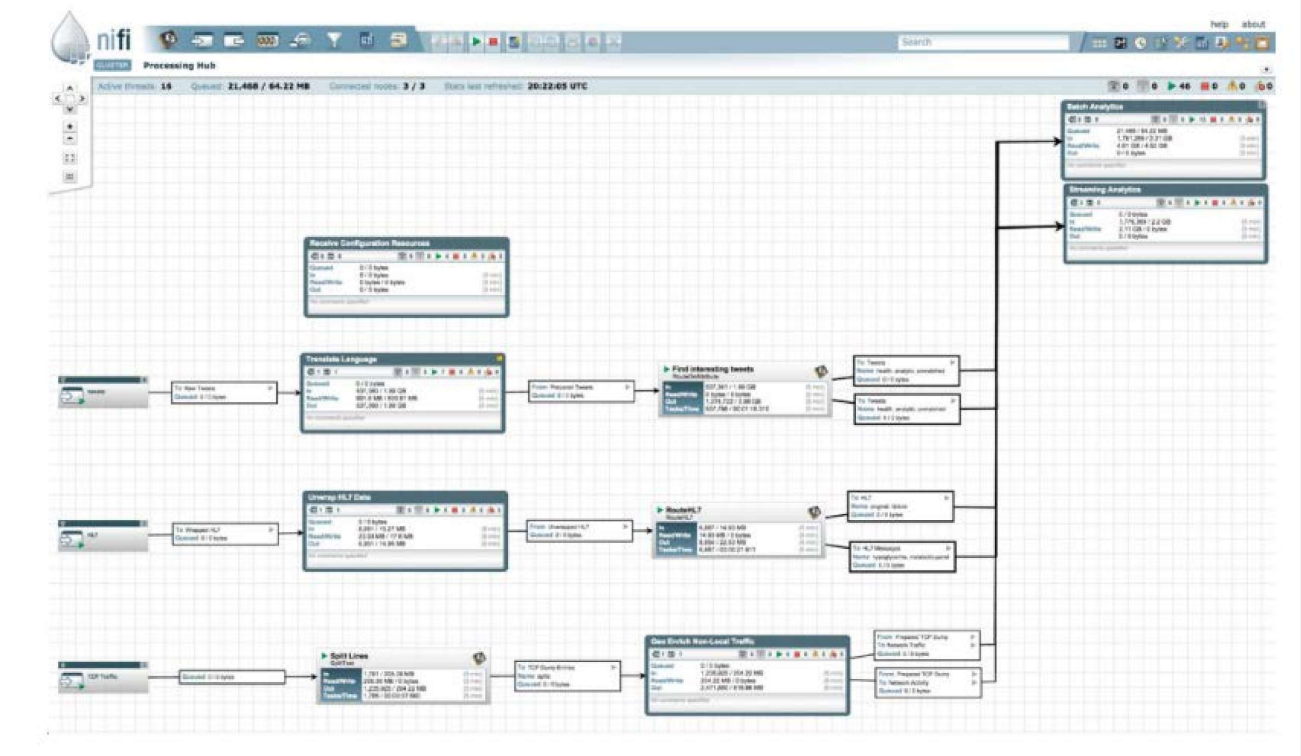
Feature	Description

- In memory Spark Core
  - Programming Scala, Java, Python, R, APIs
- Libraries
  - Mlib
  - Spark SQL
  - Spark Streaming
  - Graphx
  - ML Pipeline for data science workflow

Hortonworks Data Flow (HDF)

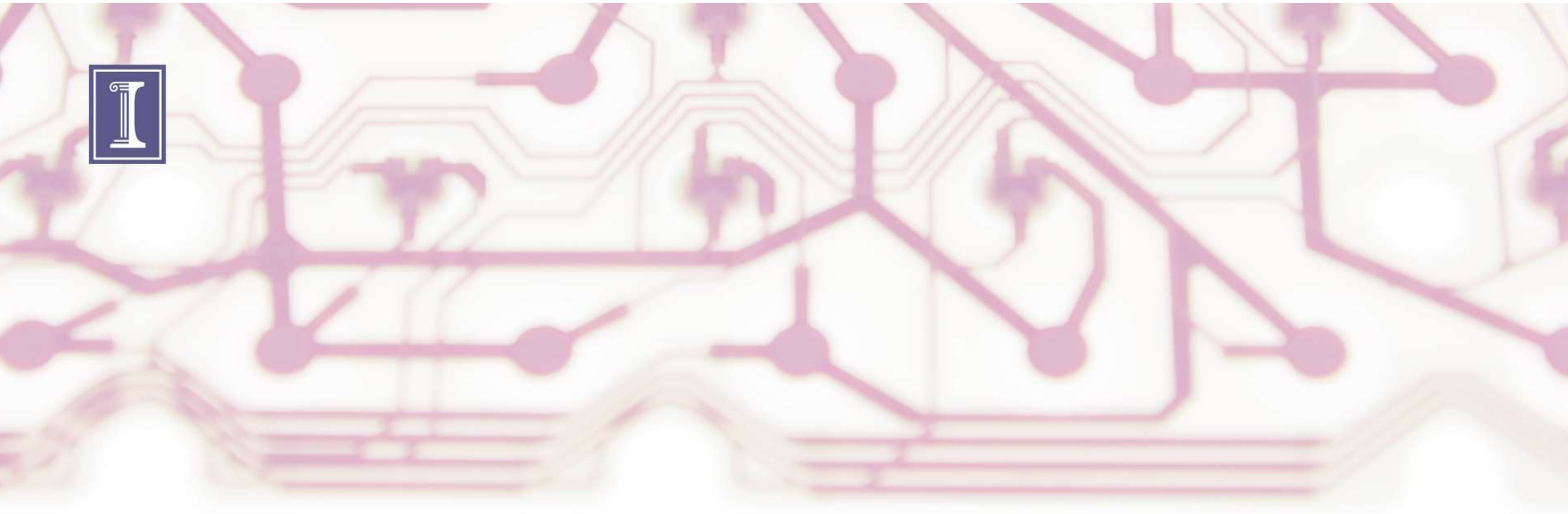
- Apache NiFi, Kafka and Storm provide real-time dataflow management and streaming analytics.
- HDF enables real-time data collection, curation, analysis and delivery of data to and from any device, source or system, either on-premise and in the cloud.
- Tools
  - KAFKA: fast, scalable, durable, and fault-tolerant publish-subscribe messaging
  - NIFI, integrated data logistics and simple event processing
  - STORM

NiFi


nifi
Processing Hub
Search
help
about
Batch Analytics
Streaming Analytics
Resolve Configuration Resources
Translate Language
Find interesting tweets
Unwrap HL7 Data
RouteHL7
Split Lines
Geo Enrich Non-Local Traffic

NiFi

NiFi Term	FBP Term	Description
FlowFile	Information Packet	A FlowFile represents each object moving through the system fand for each one, NiFi keeps track of a map of key/value pair attribute strings and its associated content of zero or more bytes.
FlowFile Processor	Black Box	Processors actually perform the work. In [eip] terms a processor is doing some combination of data Routing, Transformation, or Mediation between systems. Processors have access to attributes of a given FlowFile and its content stream. Processors can operate on zero or more FlowFiles in a given unit of work and either commit that work or rollback.
Connection	Bounded Buffer	Connections provide the actual linkage between processors. These act as queues and allow various processes to interact at differing rates. These queues then can be prioritized dynamically and can have upper bounds on load, which enable back pressure.
Flow Controller	Scheduler	The Flow Controller maintains the knowledge of how processes actually connect and manages the threads and allocations thereof which all processes use. The Flow Controller acts as the broker facilitating the exchange of FlowFiles between processors.
Process Group	Subnet	A Process Group is a specific set of processes and their connections, which can receive data via input ports and send data out via output ports. In this manner process groups allow creation of entirely new components simply by composition of other components.

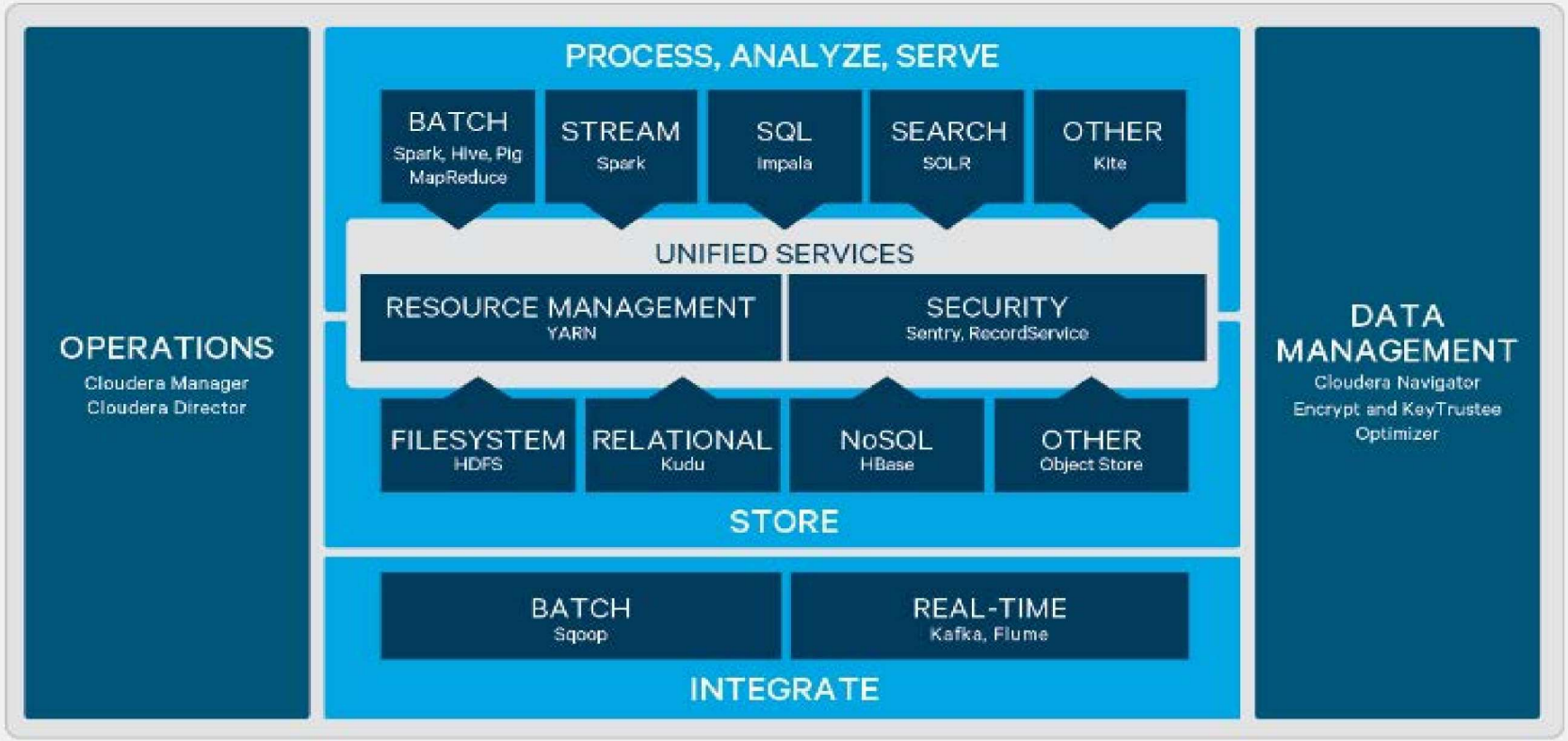


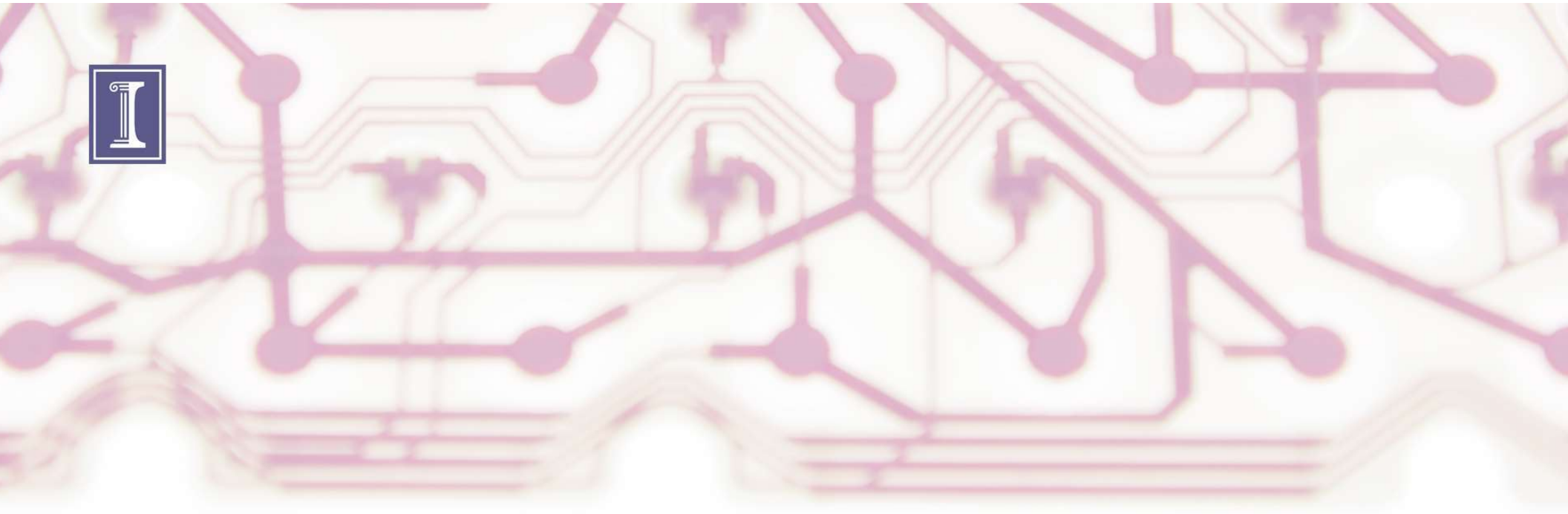
CLOUD COMPUTING APPLICATIONS

Cloudera CDH

Roy Campbell & Reza Farivar

Cloudera Enterprise Data Hub (EDH)





CLOUD COMPUTING APPLICATIONS

MapR Distro

Roy Campbell & Reza Farivar

Platform for Big Data

- MapReduce (Hadoop written in C/C++)
- NFS
- Interactive SQL (Drill, Hive Spark SQL, Impala)
- MapR-DB
- Search (Apache Solr)
- Stream Processing (MapR Streams)

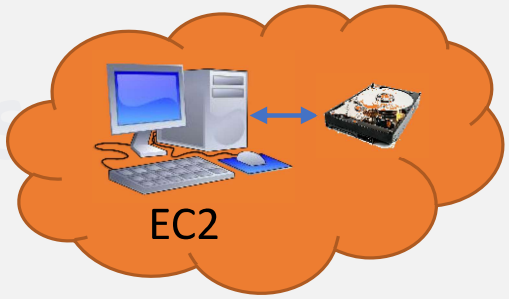
CLOUD COMPUTING APPLICATIONS

Cloud Storage: Block Storage

Prof. Reza Farivar

Cloud Storage Category 1: Block Storage – Instance Stores

- The physical machine running the virtual machine is physically connected to the storage device
  - virtual devices whose underlying hardware is physically attached to the host computer for the instance
  - Data transfer is limited by SATA / NVMe bandwidth
  - I3en.metal: 8 x 7,500 GB (60 TB)



- Since the machine may be rented to someone else in the next minute, the data stored on the drive does not persist, it's **ephemeral**.

- Example:
  - Amazon AWS: Instance Store
  - Google: Local SSD

Cloud Storage Category 1: Block Storage – Instance Stores

- The physical machine running the virtual machine is physically connected to the storage device
  - virtual devices whose underlying hardware is physically attached to the host computer for the instance
  - Data transfer is limited by SATA / NVMe bandwidth
  - I3en.metal: 8 x 7,500 GB (60 TB)

Instance Size	Local Storage (GB)	Read MBps	Write GBps
i3.large *	1 x 475 NVMe SSD	391	137
i3.xlarge *	1 x 950 NVMe SSD	806	273
i3.2xlarge	1 x 1,900 NVMe SSD	1,611	703
i3.4xlarge	2 x 1,900 NVMe SSD	3,223	1,406
i3.8xlarge	4 x 1,900 NVMe SSD	6,445	2,813
i3.16xlarge	8 x 1,900 NVMe SSD	12,891	5,469
i3.metal	8 x 1,900 NVMe SSD	12,891	5,469

** Throughputs are a snapshot in time, might be different now*

- Since the machine may be rented to someone else in the next minute, the data stored on the drive does not persist, it's **ephemeral**.

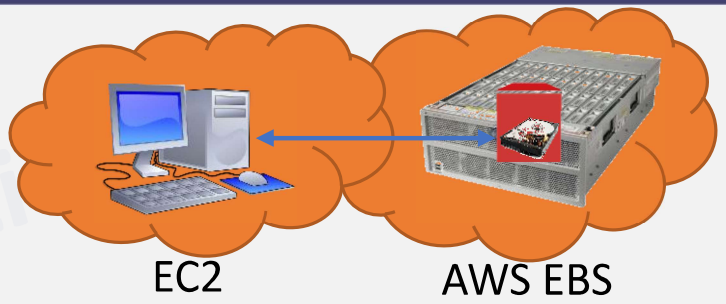
- Example:
  - Amazon AWS: Instance Store
  - Google: Local SSD

Storage Virtualization

- The process of presenting a logical view of the physical storage resources to a host computer system
  - Block Virtualization
  - File Virtualization

Cloud Block Storage – Virtual Block Stores

- Simulate a hard drive or SSD
- The physical machine running the virtual machine is separate from the physical machine hosting the data
  - NVMe over Fabric
    - NVMf or NVMe-oF
  - Data transfer is limited by network bandwidth



- The Hypervisor on the host machine has middleware that intercepts the network comm. and presents the network stream of bytes to the virtual machine as a “Block Storage Device”
  - E.g. in AWS, EBS as an NVMe device
  - E.g.in Google Cloud: Persistent Disk

Cloud Block Storage – Virtual Block Stores

- Typically less storage bandwidth than the previous option (EBS)
- You can also select how much bandwidth you are willing to pay for
  - Selecting IOPS for io1 type
- gp2 types are preselected and fixed
  - 3 IOPS/ gigabyte, 16KBps / IOPS, min 100, max 16,000
  - Bursting support
- Note: A single EC2 instance can be attached to more than 1 EBS volume
- Some instance types are EBS optimized
  - Bandwidth for EBS access is separate from network bandwidth
  - Other (micro, small, older generations) share network bandwidth

Instance Size	Instance Storage	EBS Bandwidth (MBps)
m5.large	EBS-Only	Up to 594
m5.xlarge	EBS-Only	Up to 594
m5.2xlarge	EBS-Only	Up to 594
m5.4xlarge	EBS-Only	594
m5.8xlarge	EBS Only	850
m5.12xlarge	EBS-Only	1,188
m5.16xlarge	EBS Only	1,700
m5.24xlarge	EBS-Only	2,375
m5.metal	EBS-Only	2,375

** Throughputs are a snapshot in time, might be different now*

AWS Elastic Block Storage

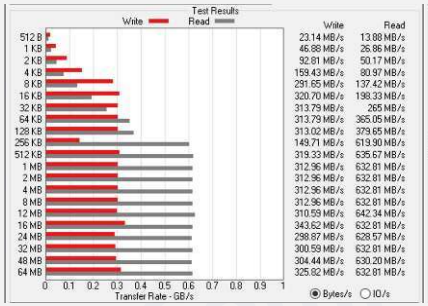
- Depending on IO requirements, different offerings and prices
  - gp2 up to 250 MB/s @ $0.10 per GB-month
  - io1 up to 1,000 MB/s @ $0.125 per GB-month AND $0.065 per IOPS-month
  - …
- Note: A single EC2 instance can be attached to more than 1 EBS volume
  - Max Throughput/Instance for T3 class: 2,375 MB/s

** Prices and bandwidths are a snapshot in time, might be different now*

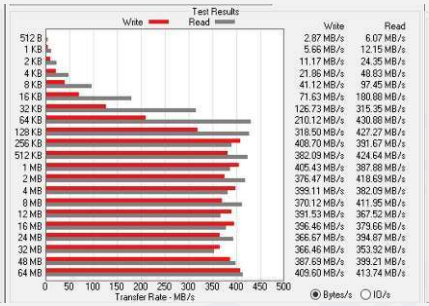
Instance Store vs. EBS Throughput

Experiment on an AWS m5ad.2xlarge instance (General Purpose, 8 vCPU, 32 GB RAM, 10GB network, EBS optimized)

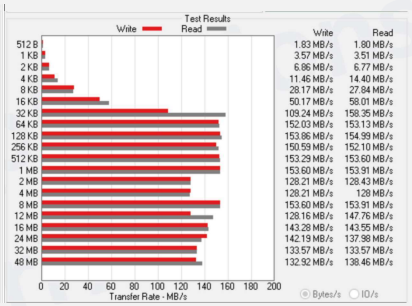
Instance Store 300GB



Io1 @ 200GB with 10000 IOPS



gp2 @ 30GB with 100/3000 IOPS

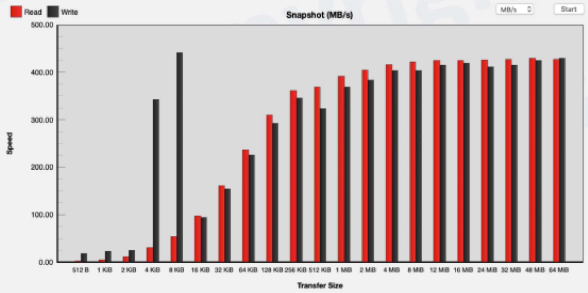


100*16KB = 1.56 MBps
3000*16KB = 45 MBps

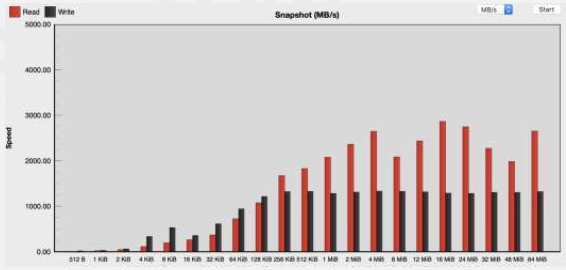
- *Bursting*
- *IO Consolidation at EBS Backend*

Experiment on laptop (Macbook Pro)

SSD via USB 3.0 (5 Gbit/s)

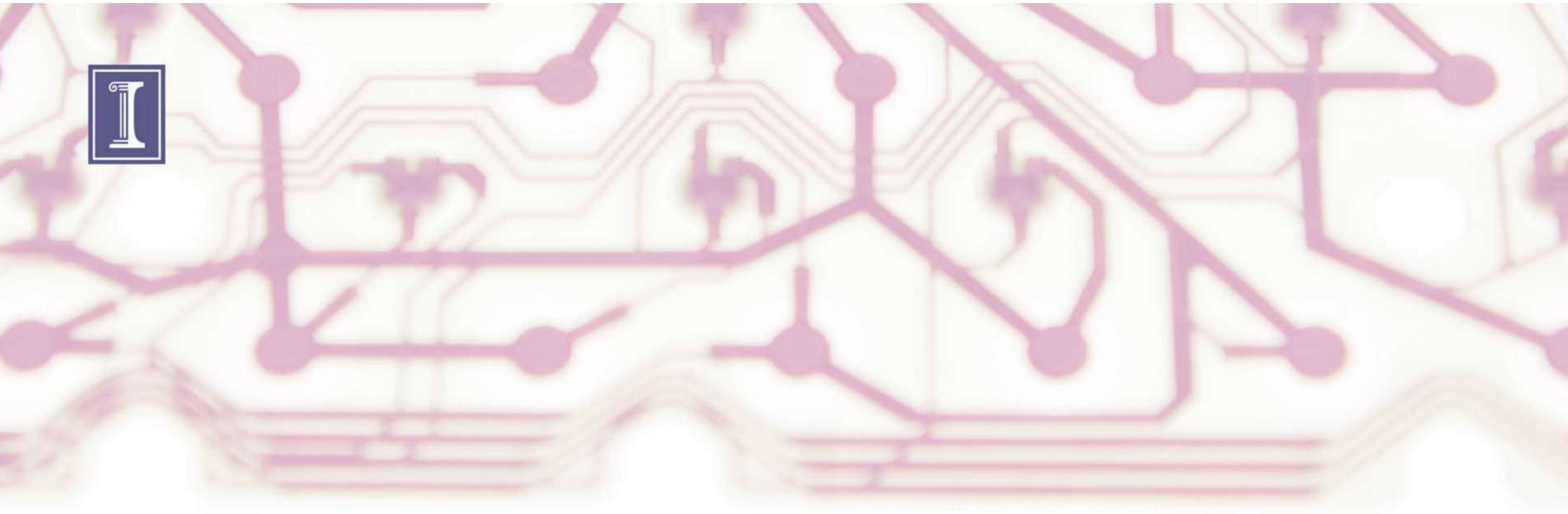


NVMe SSD (PCIe 3.0 × 4 8.0 GT/s (31.5 Gbit/s))



Summary

- Block Storage
- Instance Store
- Virtual Block Store
- Performance Comparison



CLOUD COMPUTING APPLICATIONS

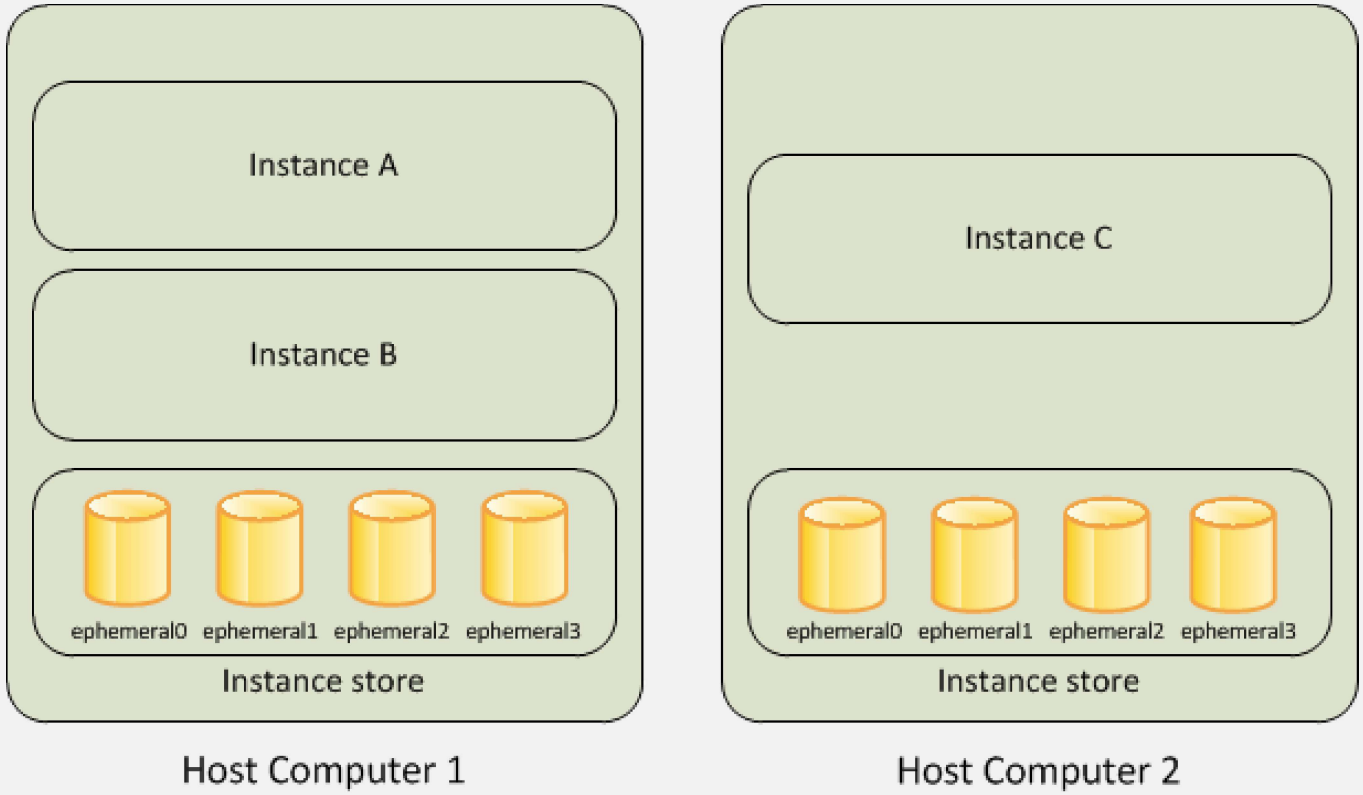
Amazon AWS Block Stores

Roy Campbell & Reza Farivar

Amazon AWS Instance Store

- Instance stores are another form of cloud-hosted **temporary** block-level storage
  - These are provided as part of an 'instance', such as an Amazon EC2
- Their contents will be lost if the cloud instance is stopped.
  - But offer higher performance and bandwidth to the instance.
  - Located on disks that are physically attached to the host computer
- They are best used for temporary storage such as caching or temporary files, with persistent storage held on a different type of server.

Amazon AWS Instance Store



Instance Store Lifetime

- Data persists only during the lifetime of its associated instance
  - It persists a reboot
- Data lost if
  - The underlying disk drive fails
  - The instance stops
  - The instance terminates
- You can get reliability by
  - A distributed file system (e.g. HDFS)
  - Backup to S3 or EBS

Instance Store Size

- A typical instance store is small
  - SSD: can be any where from around 80 GB to 320 GB SSD, up to 3,840 GB on x1.32xlarge
  - HDD: when available (on older generation instances), up to 1,680 GB

Amazon AWS EBS

- EBS Volumes are highly available and reliable
- Can be attached to running instances in the same availability zones
  - Persist independently of the life of an instance
- Use when data must be quickly accessible and requires long-term persistence
- Support encryption
- Up to 16TB in size

Amazon AWS EBS

- Different types
  - General Purpose SSD (gp2)
    - 100 IOPS/GiB, burst up to 10,000, 160 MB/s throughput
  - Provisioned IOPS SSD (io1)
    - Provision a specific level of performance
    - up to 20,000 IOPS and 320 MB/s of throughput
  - Throughput Optimized HDD (st1)
    - Low cost magnetic storage
    - Throughput of up to 500 MB/s
    - Large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing
  - Cold HDD (sc1)
    - Inexpensive magnetic
    - Throughput of up to 250 MB/s

CLOUD COMPUTING APPLICATIONS

Cloud Storage: Object Storage

Prof. Reza Farivar

Cloud Object Storage

- As we have seen, Distributed File Systems are not easy
  - Considerable overhead, complexity, cost
  - Main reason: maintaining consistency while providing transparency
- CAP: Consistency, Availability, Partition Tolerance
- Transparency: "invisible" to client programs, which "see" a system which is similar to a local file system.
  - Behind the scenes, the distributed file system handles locating files, transporting data, and potentially providing other features listed below
  - While transparency may seem trivial, these semantics can incur a significant performance penalty at scale despite not being strictly necessary

Internet Scale Storage: Breaking the Chains of Transparency and Consistency

- What if we want to scale to “unlimited Storage”?
- Solution: Stop trying to have all 3 components of CAP
  - Availability: Important to keep, otherwise customer data may be unavailable
  - Partition Tolerance: Networks do fail, cloud providers have to be resilient
  - Consistency: Can be scarified
- The typical BLOB storage by a cloud provider can scale “infinitely” by being “eventually consistent”
- In addition, they typically are not POSIX compliant
- The access model is through REST APIs
  - GET, PUT, DELETE
- Examples:
  - AWS S3
  - OpenStack Swift

AWS S3 Consistency Model

- Objects have a URI, and are accessible by REST API calls
  - https://cloudApplications.s3.us-west-2.amazonaws.com/photos/picture1.jpg
- If you PUT to an existing key, a subsequent read might return the old data or the updated data, but it never returns corrupted or partial data.
- For Availability, data will be replicated across AWS datacenters (Availability Zones)
  - If a PUT request is successful, your data is safely stored. But temporarily:
    - A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
    - A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 might return the previous data.
    - A process deletes an existing object and immediately tries to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
    - A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.
  - Amazon S3 does not currently support object locking. If two PUT requests are simultaneously made to the same key, the request with the latest timestamp wins
  - Updates are key-based. There is no way to make atomic updates across keys.

Cloud Object Storage

- By relaxing the consistency model, building the distributed storage system becomes much simpler
- The Storage costs are significantly cheaper
- Bandwidth can be quite high
  - Up to 25 GB/s
- Unlike the previous models (Block Storage, Managed File System), data can be accessible from outside the cloud
  - Your mobile app customers all over the world
  - The small number of computers you personally own

AWS S3 tiers comparison

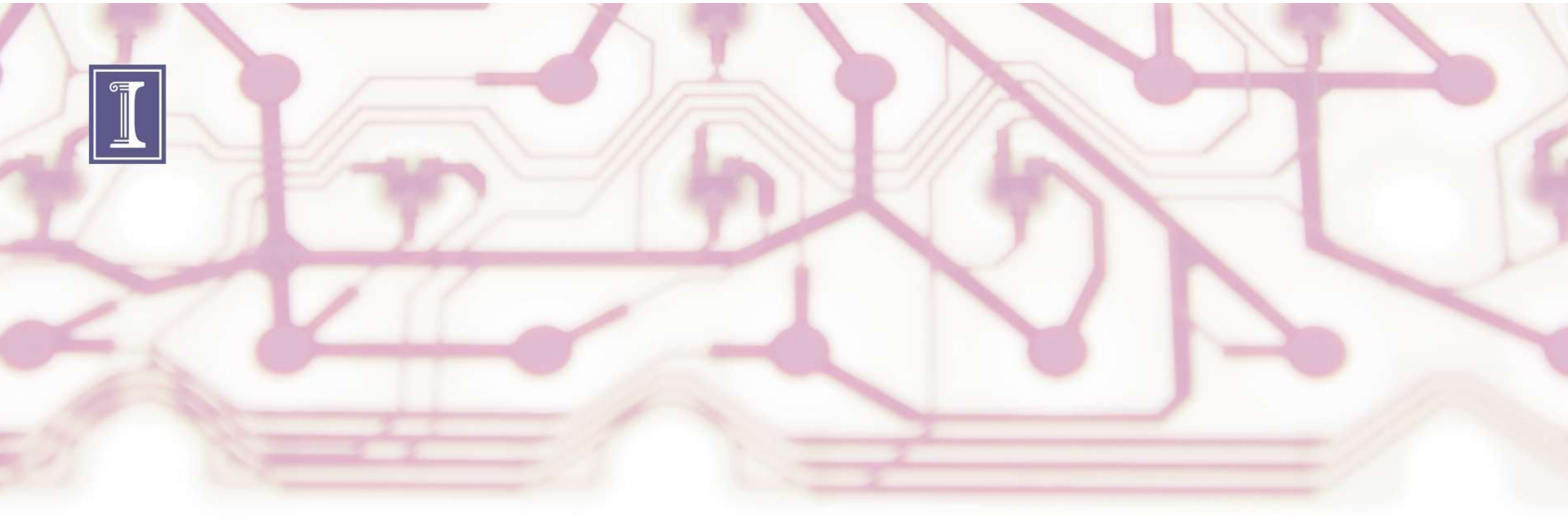
	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	millseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

Service	Cost per 1TB / month
AWS EFS	$300
AWS FSx Lustre	$290
EBS gp2	$100
AWS EFS infrequent access	$25
S3 standard	**$23**
S3 infrequent	**$13**
S3 Glacier	$4
S3 Glacier deep archive	$1

** Prices and bandwidths are a snapshot in time, might be different now*

Summary

- Cloud Object Storage
- Consistency Model
- API
- Tiers



CLOUD COMPUTING APPLICATIONS

SWIFT – An Object Store

Roy Campbell & Reza Farivar

Definition

A **Binary Large OBject** (**BLOB**) is a collection of binary data stored as a single entity in a database management system.

(The blob was a science fiction movie featuring Steve McQueen.)

https://en.wikipedia.org/wiki/Binary_large_object

Use case

- Store unstructured object data like text or binary data
- Images
- Movies
- Audio, Signal data
- Large queue of messages
- Example is LinkedIn data in a user page (Uses Ambry)
- Usually accessible over the web

Examples

- Windows Azure Blob Storage
- Ambry – LinkedIn
- Facebook's Warm BLOB Storage System
- Amazon Simple Storage Service (S3)*
- Apache Open Stack Blob Service (SWIFT)

Goals

- Data growth ~ 50% a year
- 50%-70% data is unstructured or archival
- RESTful API (HTTP)
- High availability (no single point of failure)
- Agile data centers
- Open Source
- Multi-region, geographic distribution of data
- Storage policies
- Erasure Coding

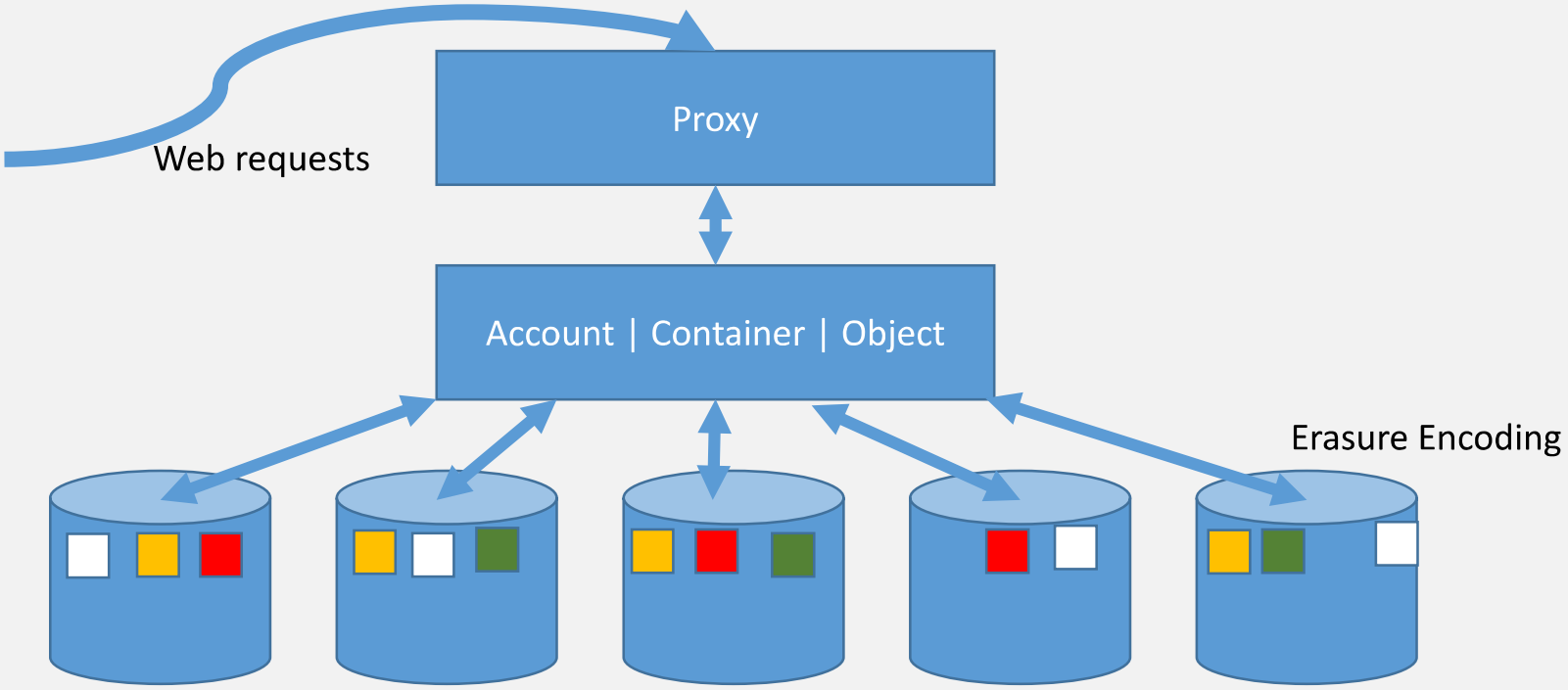
Swift API

https://swift.illinois.edu/version2/auth_account/container/object

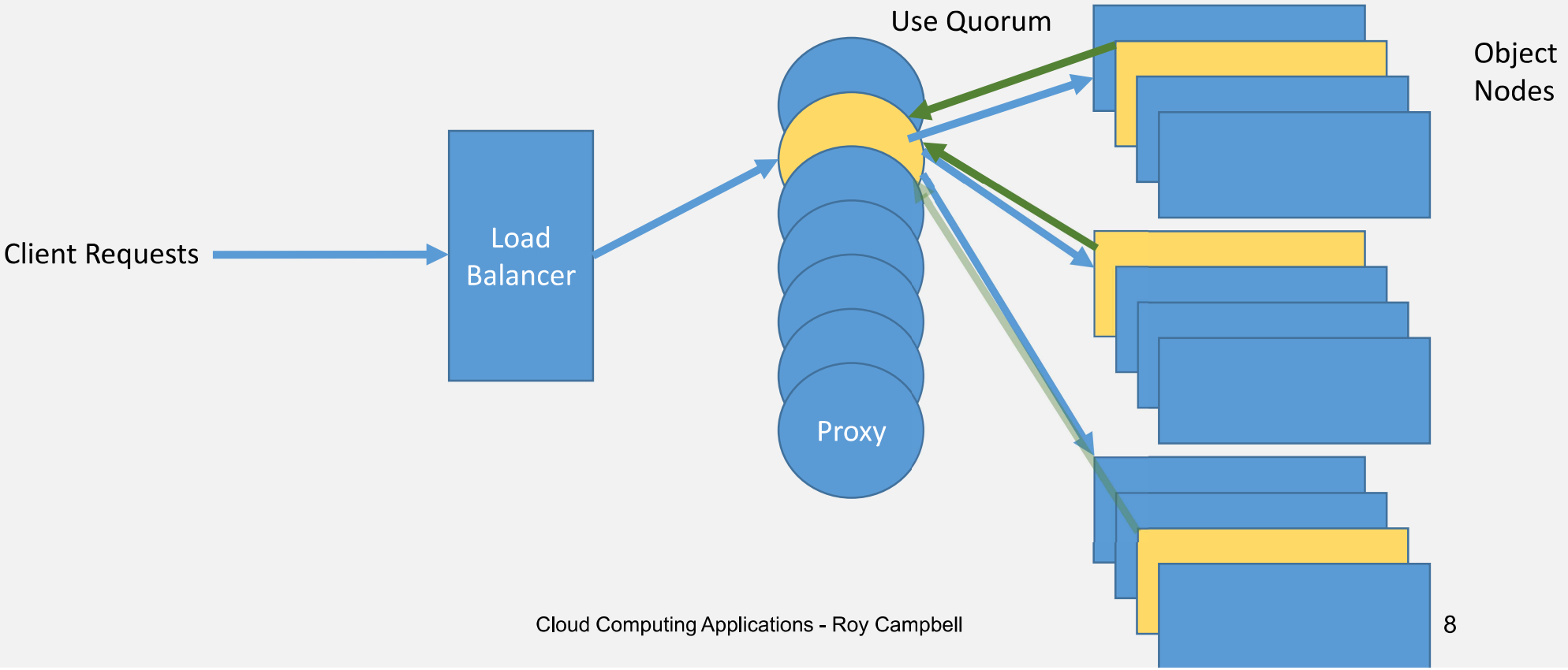
PUT /version2/roy/myblobs/classvideo1

GET /version2/reza/hisblobs/yesterdaysdataforhadoop

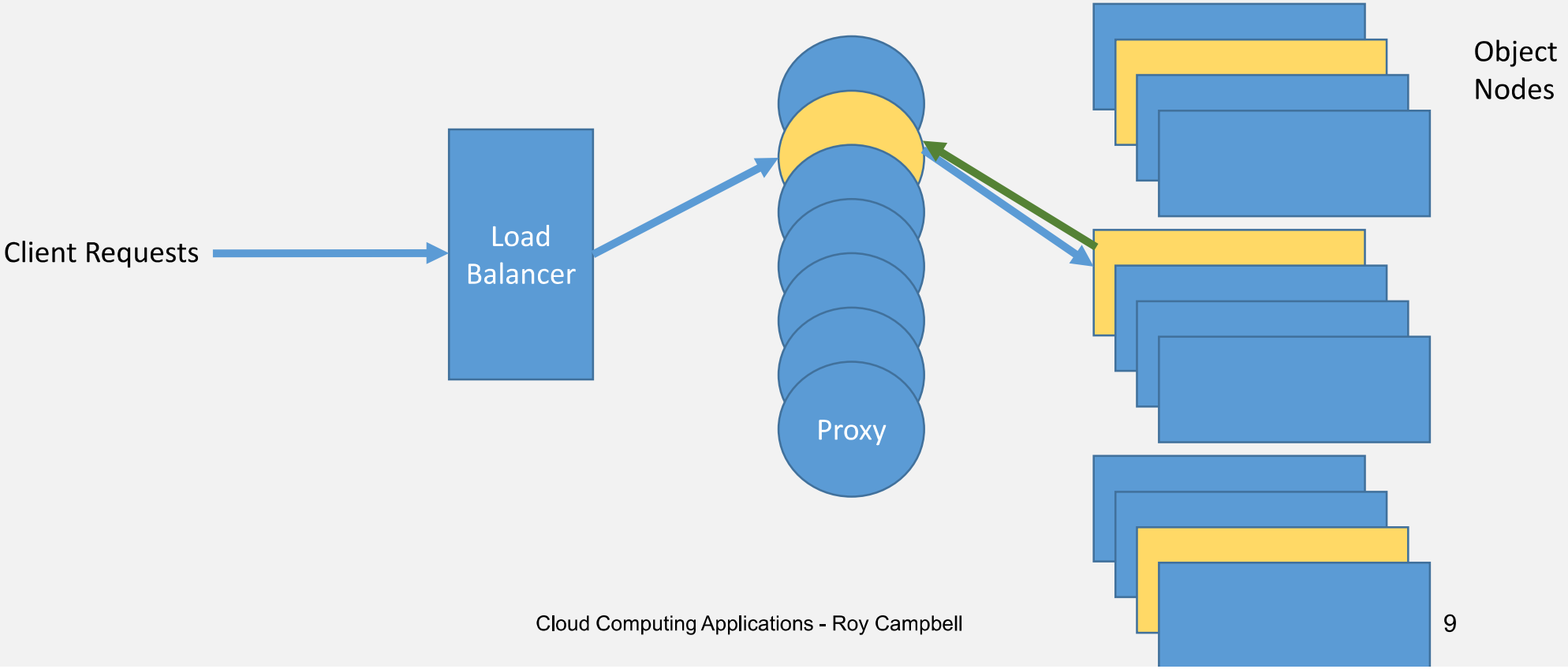
Swift Components



Write Requests - Load Balancer and Proxy



Read Requests - Load Balancer and Proxy

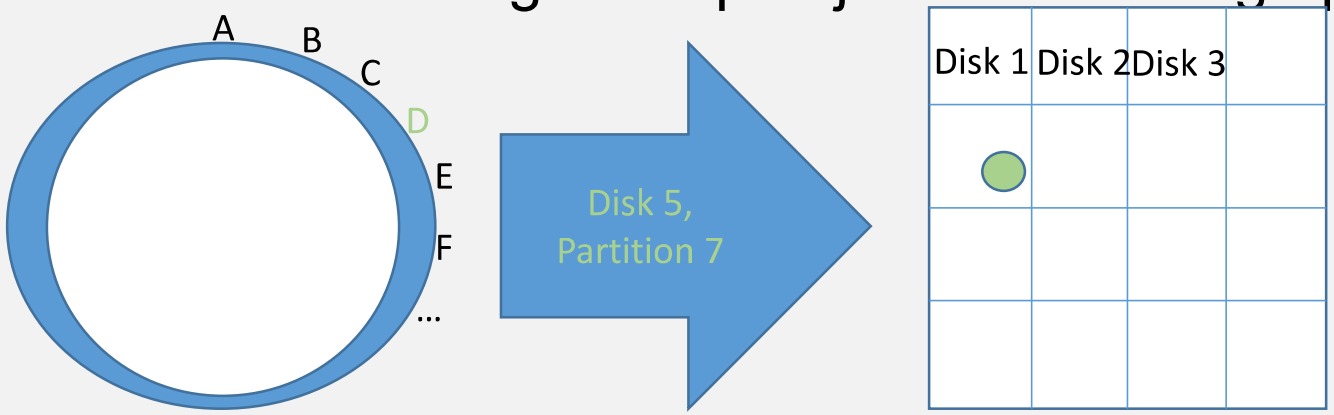


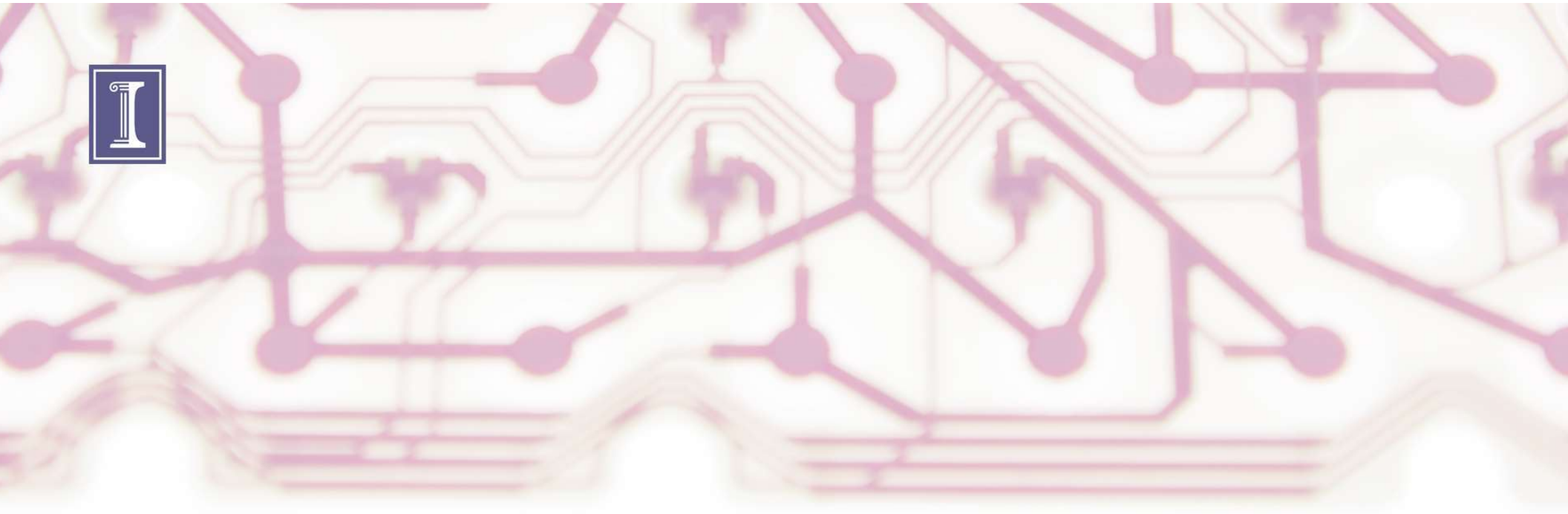
Details

- MD5 Checksums with each object
- Auditing and active replication
- Any sized disks

Swift Partitions

- 1 Node, 8 Disks, 16 Partitions per disk, 8*16 = 128 partitions
- 2 Nodes, 8 Disks each, 8 Partitions per disk, 8*16 = 128 partitions
- Use Hash into Ring to map objects into storage partitions





CLOUD COMPUTING APPLICATIONS

Amazon S3 BLOB Storage

Roy Campbell & Reza Farivar

Definition

Online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web service interfaces REST, SOAP, BitTorrent. (wikipedia)

https://aws.amazon.com/s3/

Use case

- Scalability, high availability, low latency – 99.99% availability
- Files up to 5 terabytes
- Objects stored in buckets owned by users
- User assigned keys refer to objects

- Amazon Machine Images (exported as a bundle of objects)
- SmugMug, Hadoop file store, Netflix, reddit, Dropbox, Tumbler

Simple Storage Service (S3)

- A **bucket** is a container for objects and describes location, logging, accounting, and access control.
  - A bucket has a name that must be **globally unique**.
  - http://*bucket*.s3.amazonaws.com
  - http://*bucket*.s3-*aws-region*.amazonaws.com.

- A bucket can hold any number of **objects**, which are files of up to 5TB.
  - http://bucket.s3.amazonaws.com/object
  - http://johnsmith.s3.amazonaws.com/photos/puppy.jpg

Fundamental operations corresponding to HTTP actions:

http://**bucket**.s3.amazonaws.com/**object**

- POST a new object or update an existing object.
- GET an existing object from a bucket.
- DELETE an object from the bucket
- LIST keys present in a bucket, with a filter.

A bucket has a **flat directory structure**

S3 Weak Consistency Model

“Updates to a single key are **atomic**….”

“Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However:

- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report "key does not exist."
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.”

S3 Command Line Interface

```
aws    s3   mb    s3://bucket
. . .       cp    localfile   s3://bucket/key
            mv    s3://bucket/key    s3://bucket/newname
            ls    s3://bucket
            rm    s3://bucket/key
            rb    s3://bucket

aws   s3    help
aws   s3    ls help
```

CLOUD COMPUTING APPLICATIONS

Cloud Storage: Managed File Systems

Prof. Reza Farivar