BIG DATA AND CLOUD FILE SYSTEMS

Center for Intelligent Technologies
Department of Cybernetics and Artificial Intelligence
Technical University of Košice

DCAI
Department of Cybernetics and Artificial Intelligence

CIT
Center for Intelligent Technologies

CONTENTS

CEPH – MOTIVATION

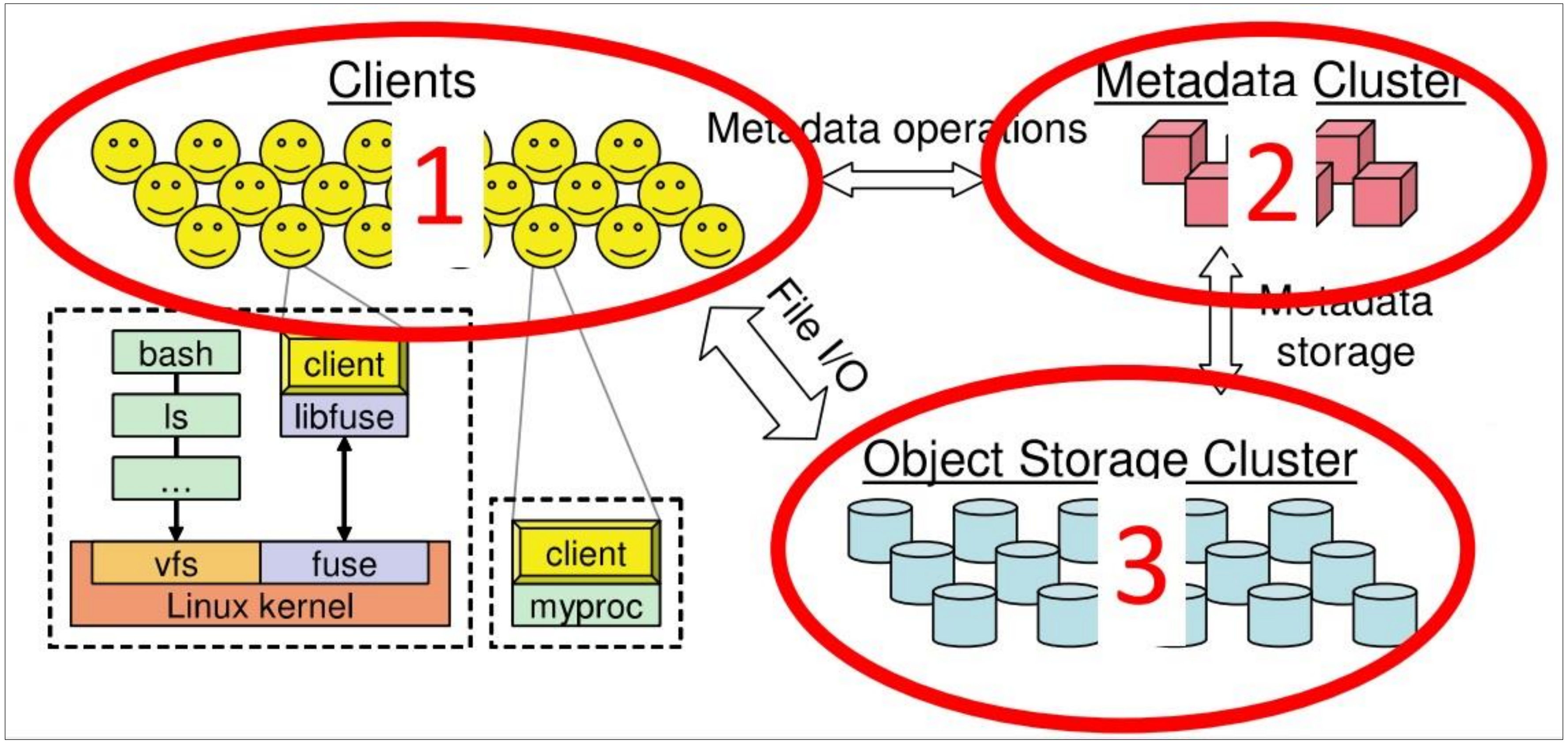
- Ceph is an emerging technology in the production-clustered environment
- designed for:
  - performance – striped data over data servers
  - reliability – no single point of failure
  - scalability – adaptable metadata cluster
  - more general than HDFS
    - smaller files
- GlusterFS has more or less similar ideas

CEPH – OVERVIEW

- MDS – Meta Data Server
- ODS – Object Data Server
- MON – Monitor
- **decoupled data and metadata**
  - I/O directly with object servers
- **dynamic distributed metadata management**
  - multiple metadata servers handling different directories (subtrees)
- **reliable autonomic distributed storage**
  - ODS’s manage themselves by replicating and monitoring

CEPH – COMPONENTS

- ordered: clients, meta data, object storage



CEPH - DECOUPLING DATA AND METADATA

- increases performance by limiting interaction between clients and servers
- decoupling is common in distributed filesystems: HDFS, Lustre, Panasas, …
- in contrast to other file systems, Ceph uses a function to calculate the block locations

CEPH - DYNAMIC DISTRIBUTED METADATA MANAGEMENT

- metadata is split among cluster of servers
- distribution of metadata changes with the number of requests to even load among metadata servers
- metadata servers also can quickly recover from failures by taking over neighbors’ data
- improves performance by leveling metadata load

CEPH - RELIABLE AUTONOMIC DISTRIBUTED STORAGE

- data storage servers act on events by themselves
- initiates replication and
- improves performance by offloading decision making to the many data servers
- improves reliability by removing central control of the cluster (single point of failure)

CEPH - USERS

HIVE- BACKGROUND

- started at Facebook
- data was collected by nightly cron jobs into Oracle DB
- “ETL” via Python
- HQL, a variant of SQL
- translates queries into MapReduce
- note
  - no UPDATE or DELETE support
  - focuses primarily on the query part of SQL

HIVE - EXAMPLE

- Hive looks similar to an SQL database
- relational join on two tables:
  - table of word counts from Shakespeare collection
  - table of word counts from Homer

HIVE - EXAMPLE (2)

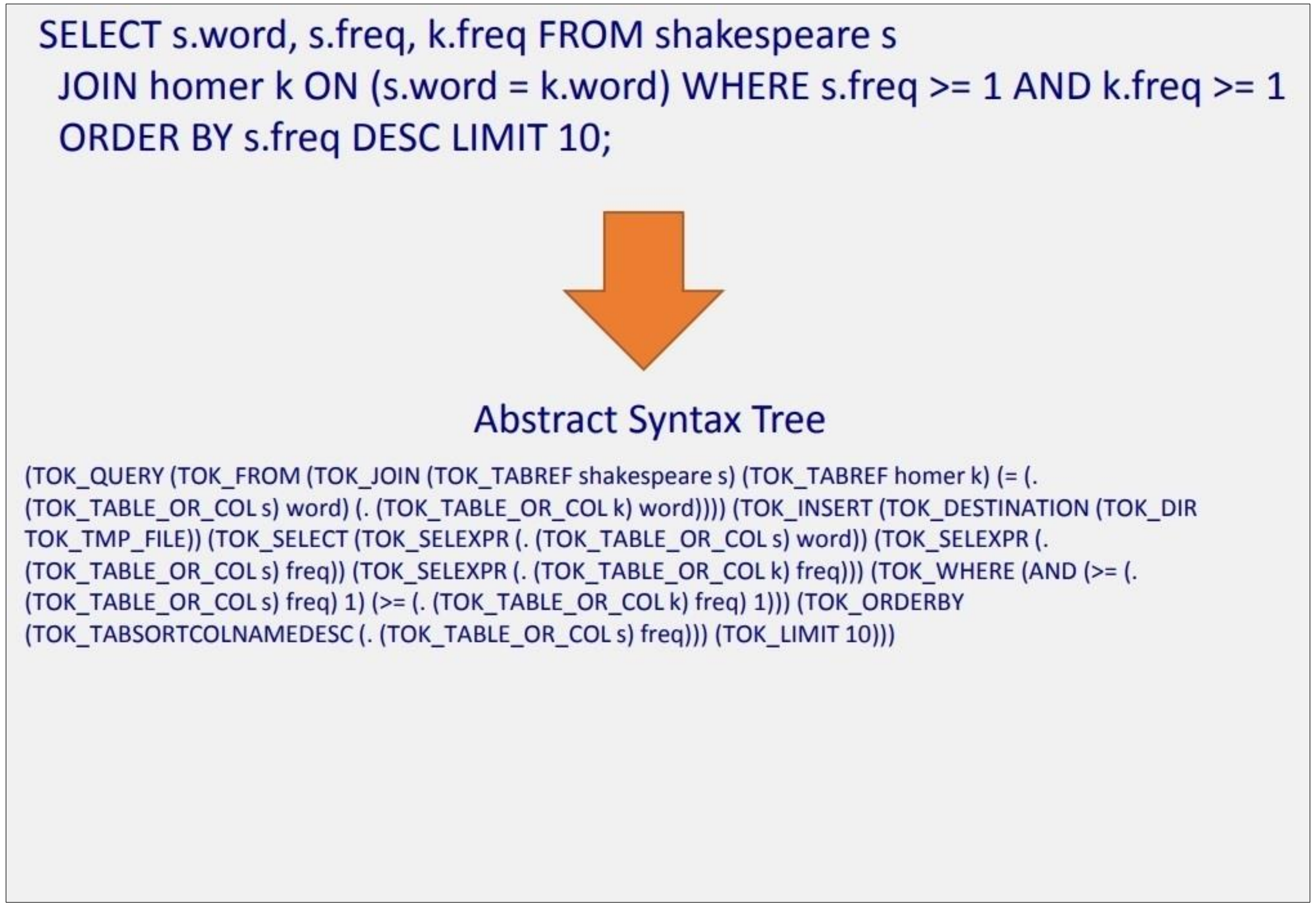
```
SELECT s.word, s.freq, k.freq FROM shaespeare s
JOIN homer k ON (s.word=k.word) WHERE
s.freq >= 1 AND k.freq >= 1 ORDER BY s.freq DESC LIMIT 10;
```

```
the     25848   62394
I       23031    8854
and     19671   38985
to      18038   13526
Of      16700   34654
A       14170    8057
You     12702    2720
My      11297    4135
In      10797   12445
Is      8882     6884
```

HIVE - BEHIND THE SCENES

```
SELECT s.word, s.freq, k.freq FROM shakespeare s
  JOIN homer k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
  ORDER BY s.freq DESC LIMIT 10;
```

HIVE - BEHIND THE SCENES

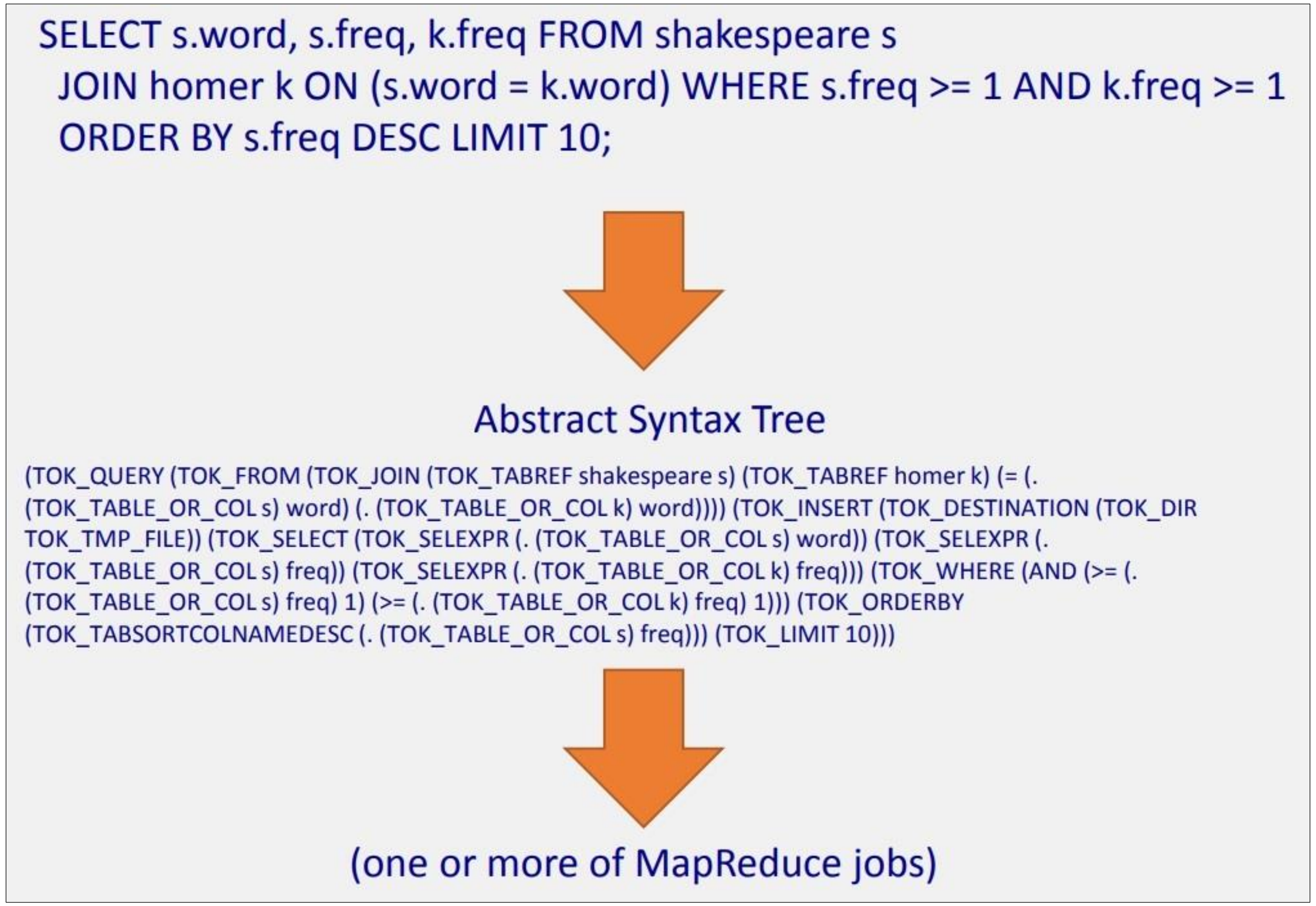


```
SELECT s.word, s.freq, k.freq FROM shakespeare s
  JOIN homer k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
  ORDER BY s.freq DESC LIMIT 10;
```

Abstract Syntax Tree

```
(TOK_QUERY (TOK_FROM (TOK_JOIN (TOK_TABREF shakespeare s) (TOK_TABREF homer k) (= (.
(TOK_TABLE_OR_COL s) word) (. (TOK_TABLE_OR_COL k) word)))) (TOK_INSERT (TOK_DESTINATION (TOK_DIR
TOK_TMP_FILE)) (TOK_SELECT (TOK_SELEXPR (. (TOK_TABLE_OR_COL s) word)) (TOK_SELEXPR (.
(TOK_TABLE_OR_COL s) freq)) (TOK_SELEXPR (. (TOK_TABLE_OR_COL k) freq))) (TOK_WHERE (AND (>= (.
(TOK_TABLE_OR_COL s) freq) 1) (>= (. (TOK_TABLE_OR_COL k) freq) 1))) (TOK_ORDERBY
(TOK_TABSORTCOLNAMEDESC (. (TOK_TABLE_OR_COL s) freq))) (TOK_LIMIT 10)))
```

HIVE - BEHIND THE SCENES



```
SELECT s.word, s.freq, k.freq FROM shakespeare s
  JOIN homer k ON (s.word = k.word) WHERE s.freq >= 1 AND k.freq >= 1
  ORDER BY s.freq DESC LIMIT 10;
```

Abstract Syntax Tree

```
(TOK_QUERY (TOK_FROM (TOK_JOIN (TOK_TABREF shakespeare s) (TOK_TABREF homer k) (= (.
(TOK_TABLE_OR_COL s) word) (. (TOK_TABLE_OR_COL k) word)))) (TOK_INSERT (TOK_DESTINATION (TOK_DIR
TOK_TMP_FILE)) (TOK_SELECT (TOK_SELEXPR (. (TOK_TABLE_OR_COL s) word)) (TOK_SELEXPR (.
(TOK_TABLE_OR_COL s) freq)) (TOK_SELEXPR (. (TOK_TABLE_OR_COL k) freq))) (TOK_WHERE (AND (>= (.
(TOK_TABLE_OR_COL s) freq) 1) (>= (. (TOK_TABLE_OR_COL k) freq) 1))) (TOK_ORDERBY
(TOK_TABSORTCOLNAMEDESC (. (TOK_TABLE_OR_COL s) freq))) (TOK_LIMIT 10)))
```

(one or more of MapReduce jobs)

HIVE - COMPONENTS

- shell: allows interactive queries
- driver: session handles, fetch, execute
- compiler: parse, plan, optimize
- execution engine
- metastore: schema, location in HDFS, etc.

HIVE - METASTORE

- Hive uses a traditional database to store its metadata:
  - A namespace containing a set of tables
  - Holds table definitions (column types, physical layout)
  - Holds partition information
- can be stored in MySQL, Oracle, and many other relational databases

TEZ- DESIGN GOALS

- **empowering end users by:**
  - expressive dataflow definition APIs
  - flexible Input-Processor-Output runtime model
  - data type agnostic
  - simplifying deployment
- **execution performance**
  - performance gains over MapReduce
  - optimal resource management
  - plan reconfiguration at runtime
  - dynamic physical data flow decisions

TEZ- INTEGRATES HIVE AND PIG

- Hive, Pig creates
  - extra MapReduce jobs when implemented with Hadoop compared with using Tez

OBJECT STORAGES - BLOB DEFINITION

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

(not the 1958 sci-fi movie)

USE CASES OF OBJECT STORAGES

- store unstructured object data like text or binary data
- images
- movies
- audio, signal data
- large queue of messages
- usually accessible over the web
- example is LinkedIN data in a user page

EXAMPLES OF OBJECT STORAGES

- Microsoft Azure Blob Storage
- Ambry used by LinkedIN
- Facebook’s Warm Blob Storage System
- Amazon Simple Storage Service (S3)
- Apache Open Stack Blob Service (SWIFT)

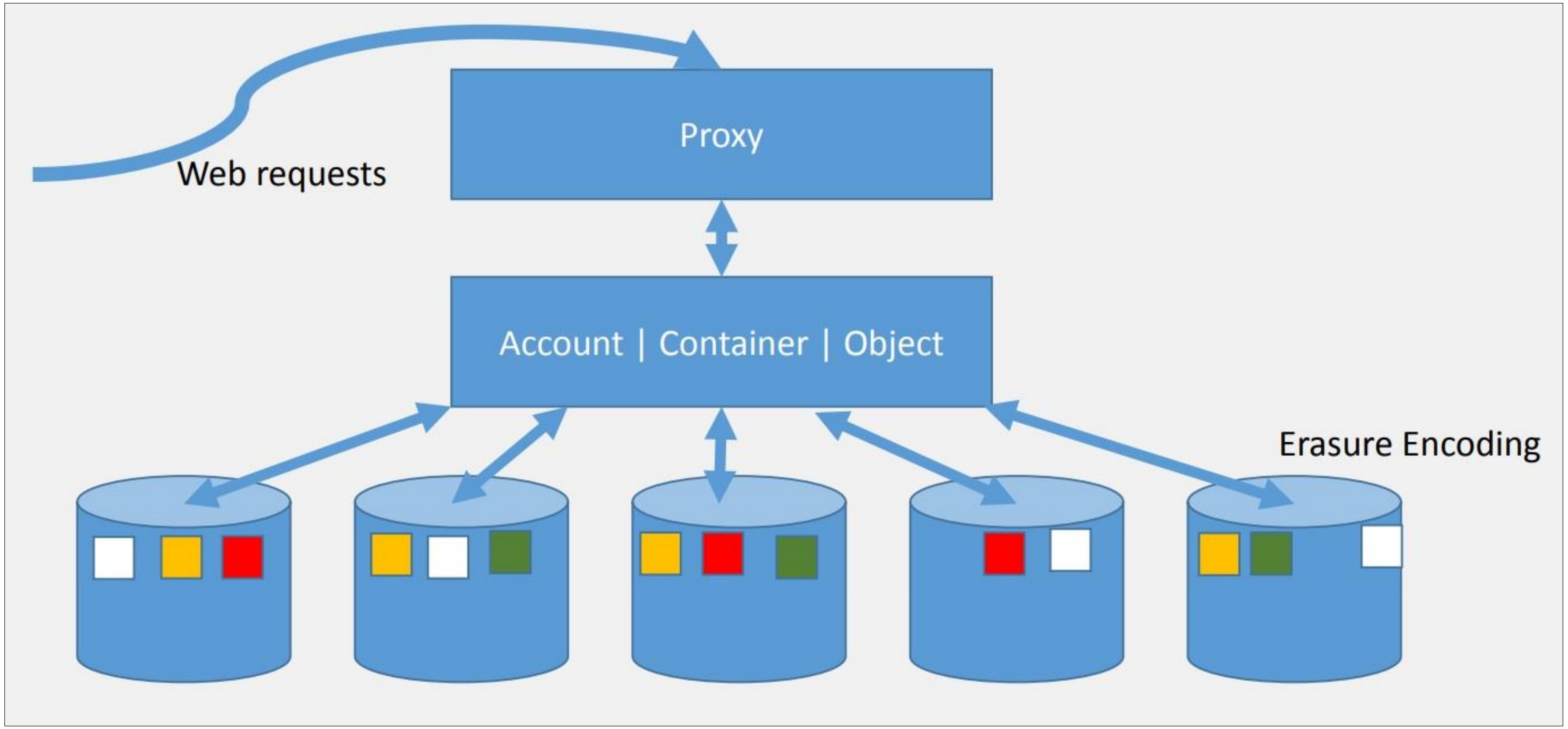
WHY DO WE NEED OBJECT STORAGES?

- data growth approx. 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

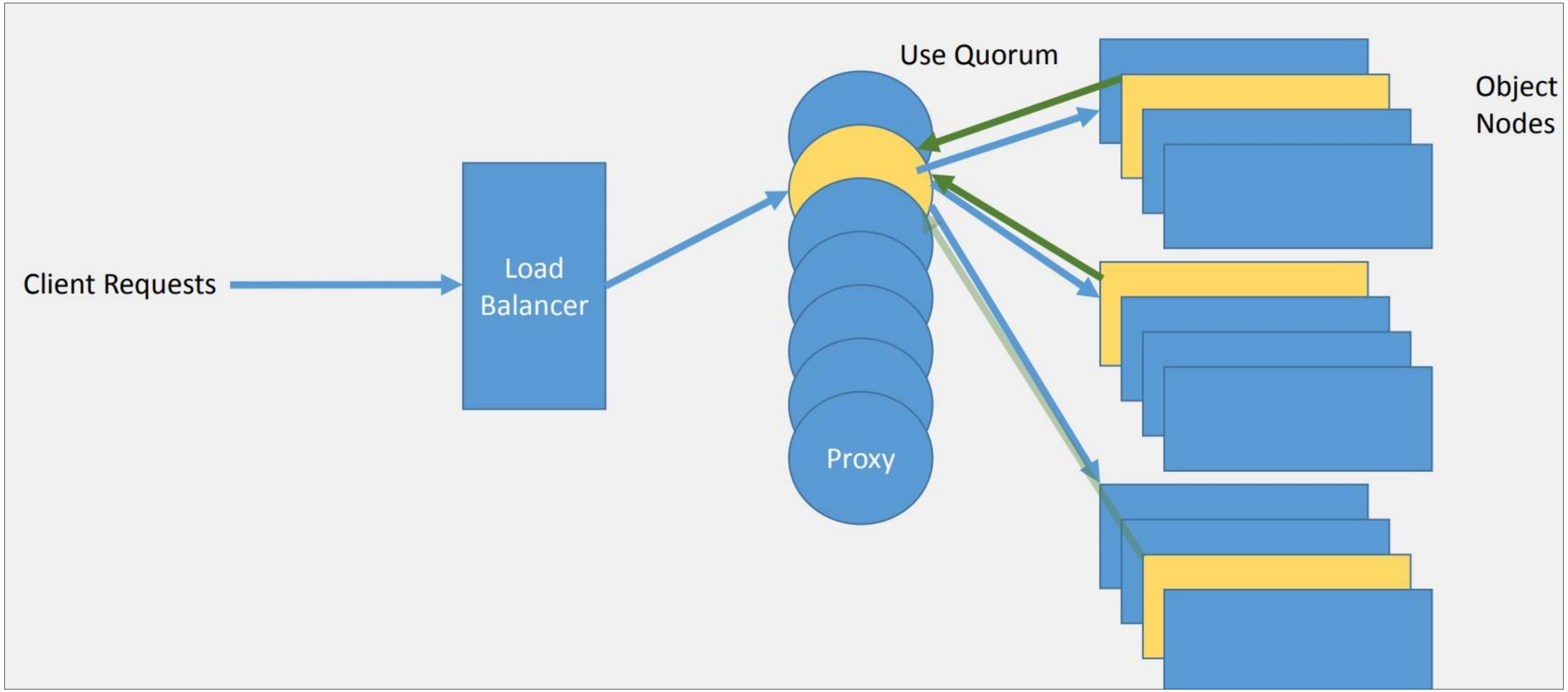
SWIFT API

- https://tuke.sk/blob_storage/auth_account/container/object
- PUT /blob_storage/g_magyar/myblobs/lecture5
- GET /blob_storage/g_magyar/myblobs/lecture5

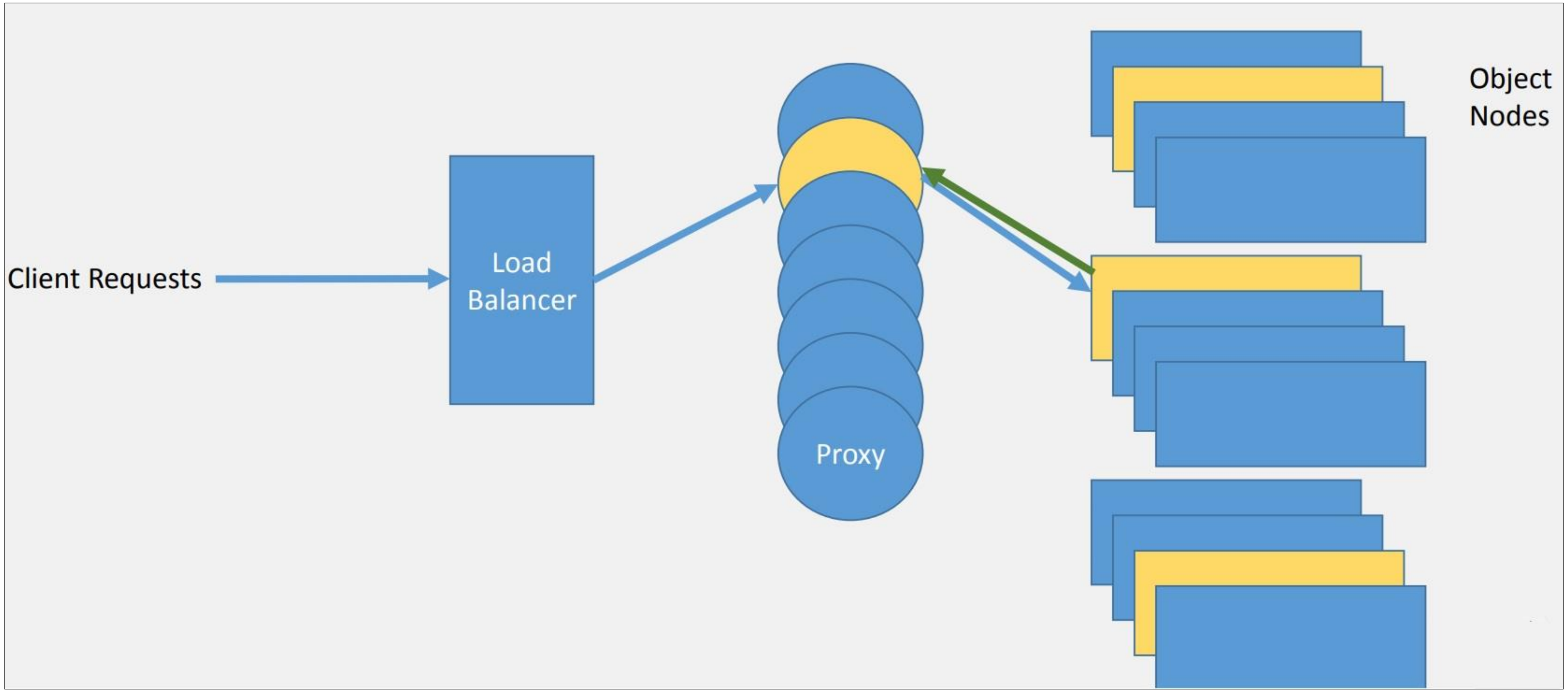
SWIFT COMPONENTS



SWIFT WRITE REQUESTS



SWIFT READ REQUESTS



SWIFT - DETAILS

- checksums with each object
- auditing and active replication
- any sized disks

AMAZON S3 - DEFINITION

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

AMAZON S3 - USE CASES

- 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)
- Netflix, reddit, Dropbox, etc.

AMAZON 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 5 TB.
  - http://bucket.s3.amazonaws.com/object
  - http://johnsmith.s3.amazonaws.com/photos/puppy.jpg

AMAZON S3 - FUNDAMENTAL OPERATIONS

- 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

AMAZON S3 - WEAK CONSISTENCY MODEL

- updates to a single key are atomic
- Amazon S3 achieves high availability by replicating data across multiple servers within Amzon'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 propagates, Amazon S3 might report "key does not exist."
  - a process writes a new object to Amazon S3 and immediately lists keys within its bucked. 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.

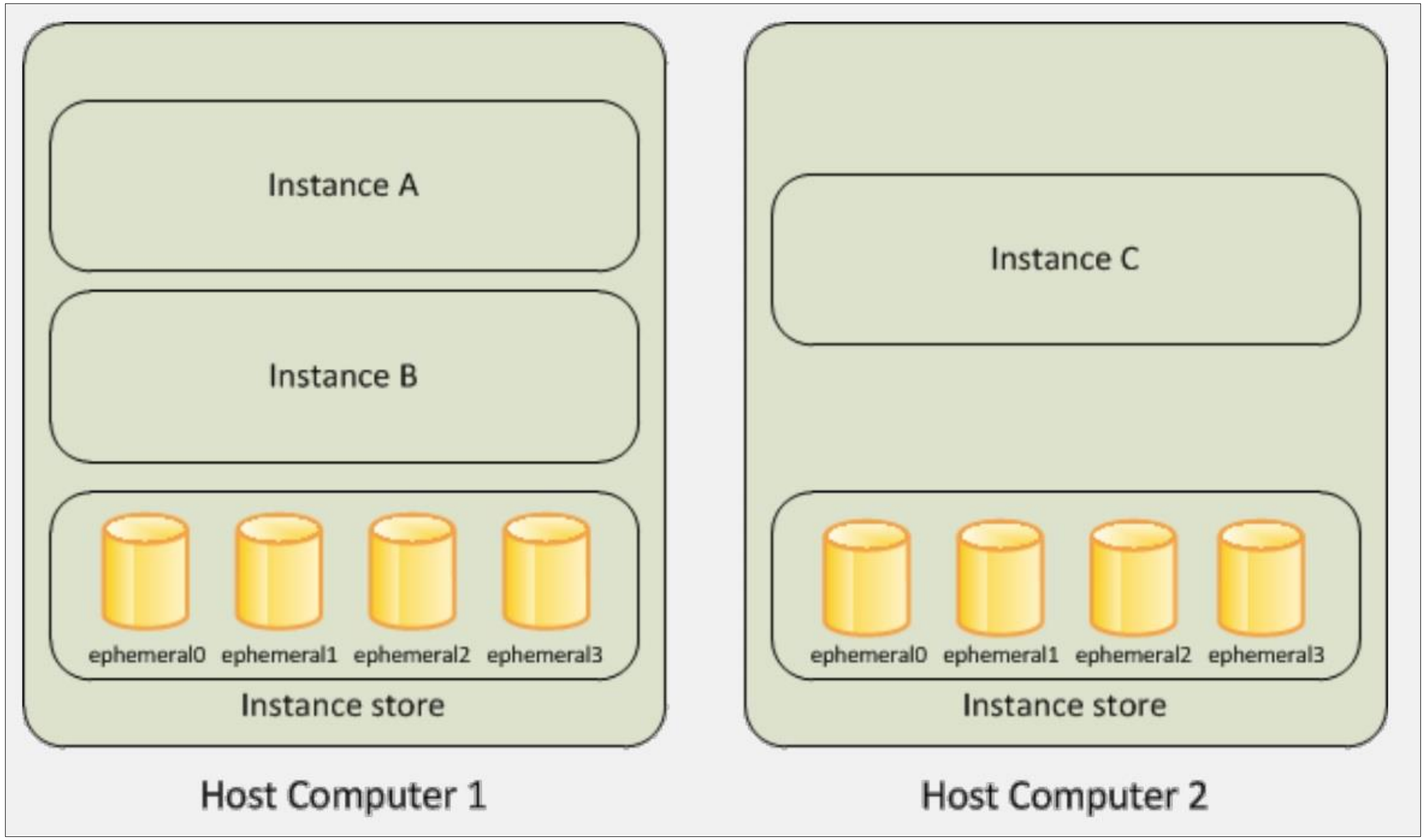
AMAZON S3 - COMMAND LINE INTERFACE

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

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 (2)



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
  - backup to S3 or EBS

INSTANCE STORE SIZE

- a typical instance store is small
  - SSD: can be anywhere from around 80 GB to 320 GB SSD, up to 3840 GB
  - HDD: when available up to 1680 GB

AMAZON AWS EBS

- EBS volumes are highly available and reliable
- can be attached to running instances in the same availability zones
- use when data must be quickly accessible and requires long-term persistence
- support encryption
- up to 16 TB in size

AMAZON AWS EBS (2)

- general purpose SSD
- provisioned SSD
  - provision a specific level of performance
- throughput optimized HDD
  - low cost magnetic storage
  - large, sequential workloads, such as log processing
- cold HDD
  - inexpensive magnetic storage

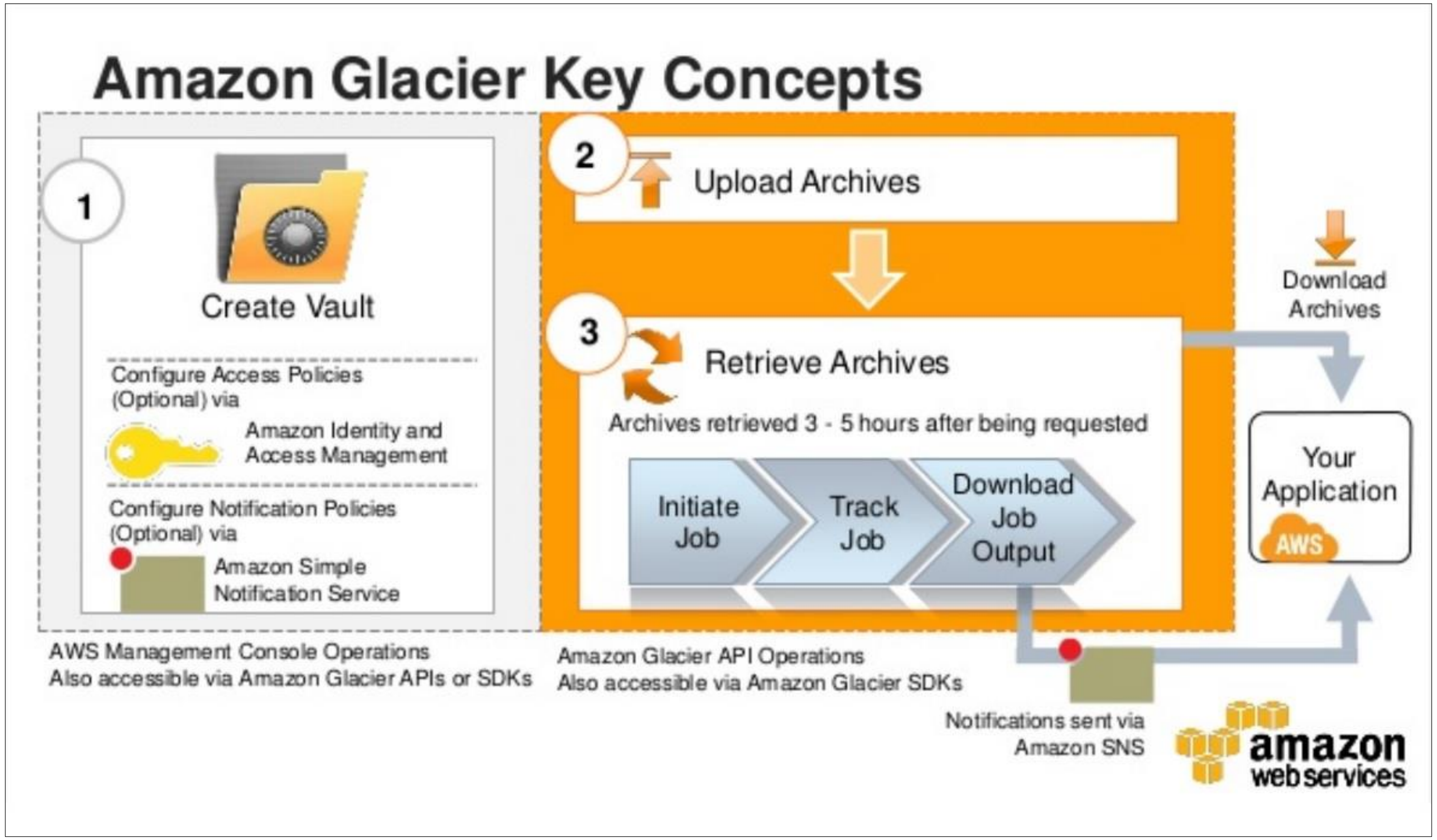
AMAZON AWS GLACIER

- allows you to archive your data
- very low cost, $0,007 per GB per month
- very durable
  - average annual durability of 99.999999999%
- each single archive up to 40 TB

AMAZON AWS GLACIER (2)

- archives stored in vaults
- the main access point to glacier is S3
- typically takes between ***3 to 5 hours*** to prepare a download request
  - after that you have 24 hours to download from the staging location

AMAZON AWS GLACIER (3)


Amazon Glacier Key Concepts
1
Create Vault
Configure Access Policies (Optional) via
Amazon Identity and Access Management
Configure Notification Policies (Optional) via
Amazon Simple Notification Service
AWS Management Console Operations
Also accessible via Amazon Glacier APIs or SDKs
2
Upload Archives
3
Retrieve Archives
Archives retrieved 3 - 5 hours after being requested
Initiate Job
Track Job
Download Job Output
Amazon Glacier API Operations
Also accessible via Amazon Glacier SDKs
Download Archives
Your Application
AWS
Notifications sent via Amazon SNS
amazon webservices

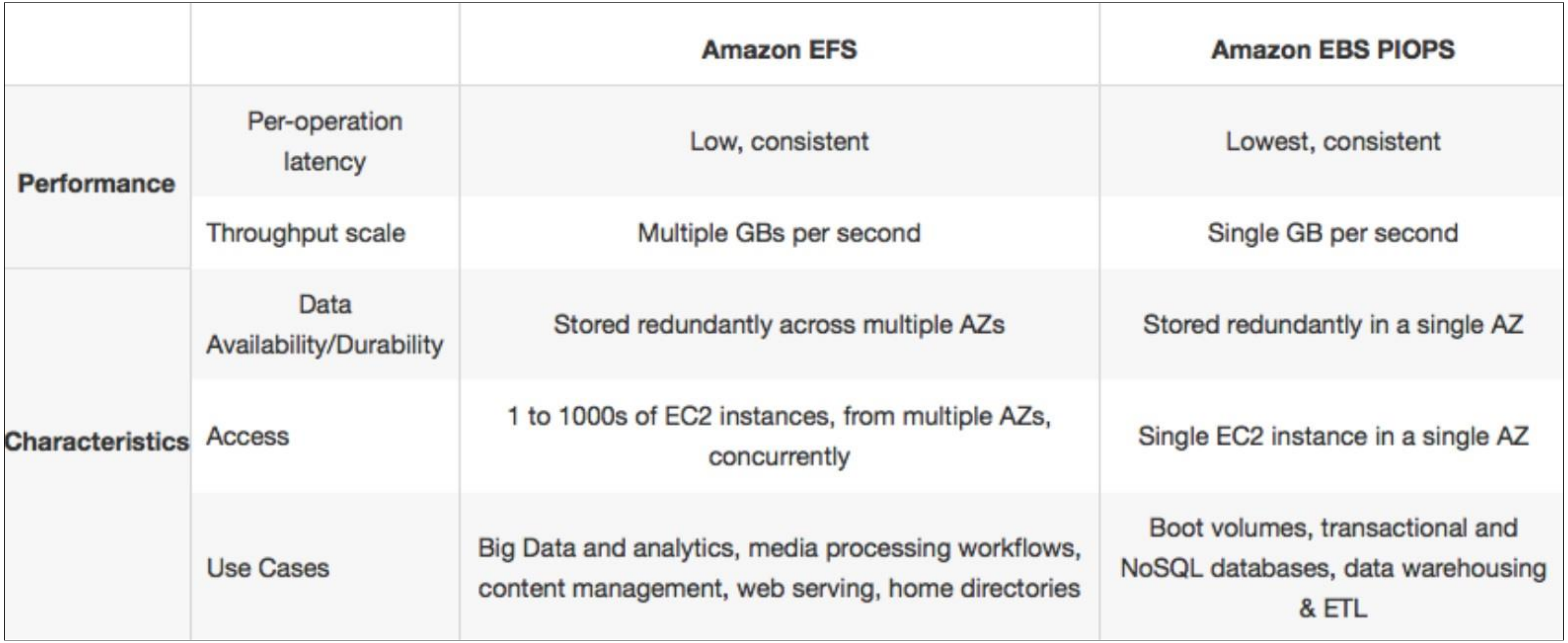
AMAZON AWS EFS

- Elastic File System
- motivation: enterprise customers need a large distributed file system
  - S3 is large and distributed, but it is an object store without performance guarantees and eventual consistency model
  - block storage (EBS, instance store) are small
    - enterprise can build a distributed file system on top of these, but it requires operational expertise
  - glacier: good for only archival storage
- EFS provides a fully NFSv4 compliant network file system

AMAZON AWS EFS (2)

- SSD backed
- highly available and highly durable
  - files, directories, and links are stored redundantly across multiple Availability Zones within an AWS region
- grow or shrink as needed
  - no need to pre-provision capacity

AMAZON AWS EFS VS EBS

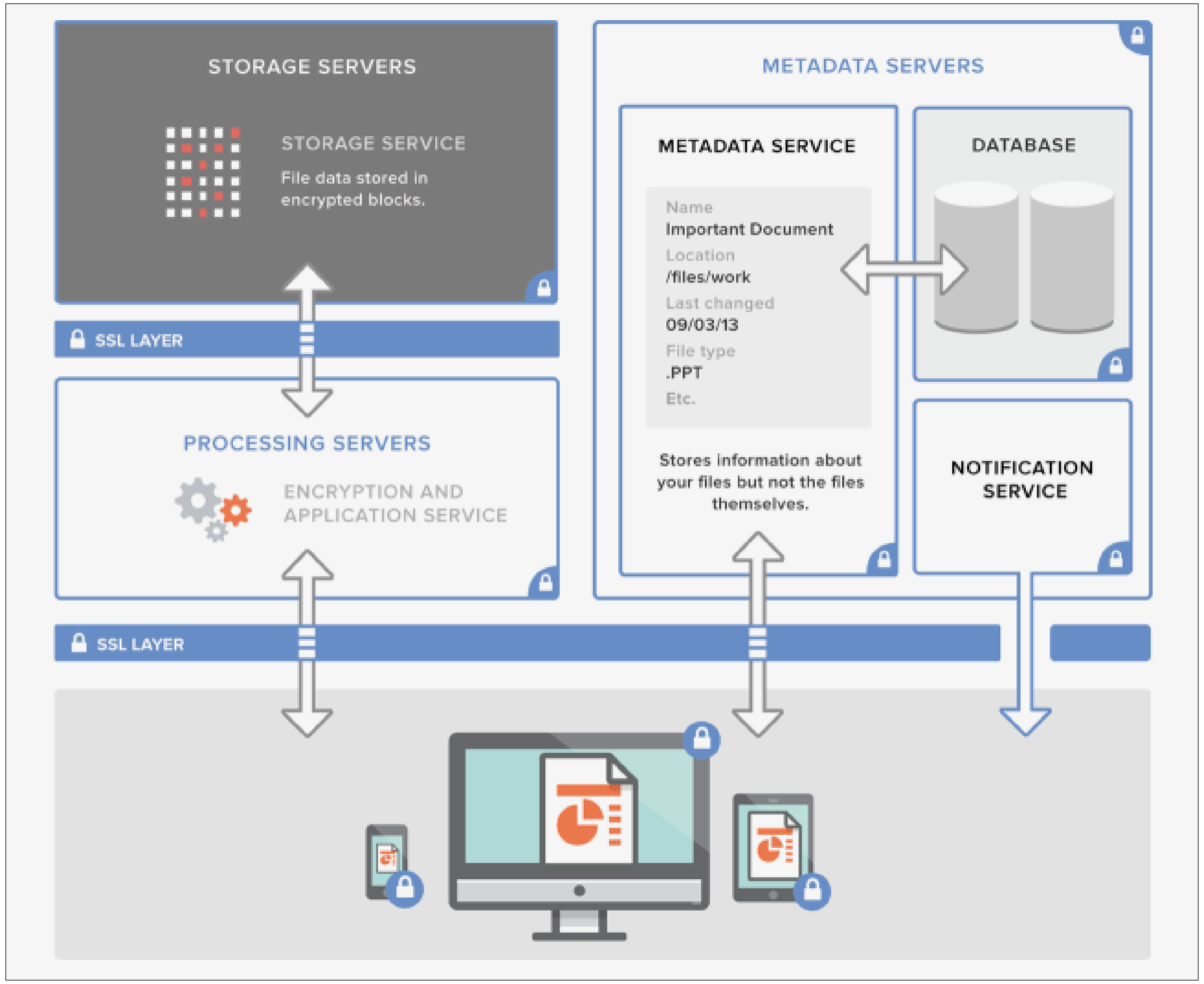


		Amazon EFS	Amazon EBS PIOPS
Performance	Per-operation latency	Low, consistent	Lowest, consistent
	Throughput scale	Multiple GBs per second	Single GB per second
Characteristics	Data Availability/Durability	Stored redundantly across multiple AZs	Stored redundantly in a single AZ
	Access	1 to 1000s of EC2 instances, from multiple AZs, concurrently	Single EC2 instance in a single AZ
	Use Cases	Big Data and analytics, media processing workflows, content management, web serving, home directories	Boot volumes, transactional and NoSQL databases, data warehousing & ETL

DROPBOX

- interesting case study
- Dropbox offers cloud file storage
  - easily synced across multiple devices
  - accessible through web interface, mobile apps, and directly integrated with the file system on PCs
- Dropbox itself uses clouds!
  - metadata stored in Dropbox servers
  - actual files stored in Amazon S3
  - Amazon EC2 instances run the logic

DROPBOX ARCHITECTURE

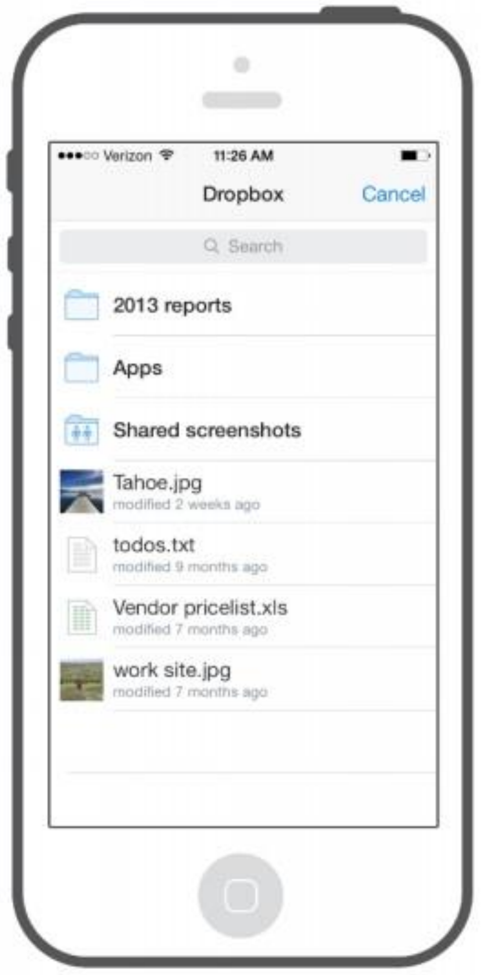


DROPBOX API

- two levels of API access to Dropbox
  - drop-ins
    - cross-platform UI components that can be integrated in minutes
    - ***chooser*** allows instant access to files in Dropbox
    - ***saver*** makes saving files to Dropbox easy
  - core API
    - support for advanced functionality like search, revisions, and restoring file
    - better fit for deeper integration

DROP-IN API

- simple objects
  - ***chooser*** available for JavaScript, Android and iOS
  - ***saver*** on web and mobile web
- handles all the authentication (OAuth) and file browsing
- chooser object returns the following:
  - link: URL to access the file
  - file name
  - file size
  - icon
  - thumbnails
- saver
  - pass in URL, filename and options



CORE API

- many languages and environments
    - Python, Ruby, PHP, Java, Android, iOS, OS X, HTTP
- based on HTTP and OAuth
    - OAuth v1, OAuth v2
- Low-level calls to access and manipulate a user’s Dropbox account
    - create URL schemes
    - upload files
    - download files
    - list files and folders
    - delta
    - metadata access
    - create and manage file sharing