Data Storage Services

Essential Cloud Infrastructure: Core Services

CLOUD STORAGE, CLOUD SQL, CLOUD SPANNER, CLOUD DATASTORE, CLOUD BIGTABLE, BIGQUERY

CLOUD STORAGE, CLOUD SQL, CLOUD DATASTORE

Last modified 2018-1-29

Data storage services

Data Processing Services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values, HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Google Cloud

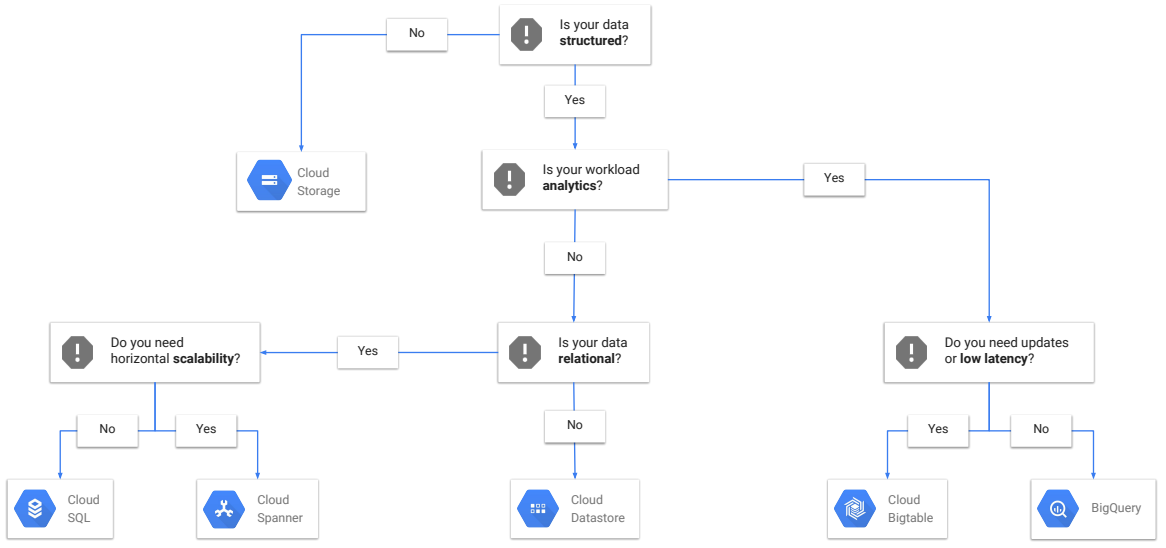
Virtually all applications need persistent and durable storage to accomplish their purpose.
Applications vary in their storage requirements, so Google Cloud Platform offers many persistent storage services.

Notice that BigQuery is grayed out. BigQuery sits on the edge between data storage and data processing. You *can* store data in BigQuery, but the usual reason to do this is to use BigQuery's big data analysis and interactive querying capabilities. For this reason, the course discusses BigQuery in the Managed Services module.

All data in GCP is encrypted while at rest and encrypted in flight.

These services provide ranges of scaling, performance, and data structure characteristics. Variations within each service complicate things. For example, Cloud Storage stores large objects. However, one type of Cloud Storage is good for streaming video, while another type is intended to archive data that will be accessed no more than once a year.
https://cloud.google.com/storage-options/

Storage and database decision chart



Google Cloud

Different applications and workloads require different storage and database solutions. Google offers a full suite of industry-leading storage services that are price performant and meet your needs for structured, unstructured, transactional, and relational data. This decision chart helps you identify the solutions that fit your scenarios.

For more information, including mobile solutions, see https://cloud.google.com/storage-options/

Scope

Infrastructure Track

- Infrastructure
- Service differentiators
- When to consider using each service
- Basic knowledge for setting up and connecting to a service
- Administration tasks

Data Engineering Track

- How to use a database system
- Design, organization, structure, schema, and use for an application
- Details about how a service stores and retrieves structured data

Google Cloud

Google offers an entire learning track on Data Engineering. This module is not a duplicate of that content. The purpose here is to understand, from an infrastructure perspective, what services are available and when to consider using them. This material is not intended to teach you how to use database systems.

Agenda

- **Google Cloud Storage**
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Datastore
- Lab
- Cloud Bigtable
- Quiz

Data storage services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Google Cloud

Cloud Storage stores objects in buckets. There are several differences between Cloud Storage and a file system.

1. A file system has a hierarchical structure. Cloud Storage is unstructured. It is a flat system of buckets (not directories) that cannot be nested.
2. An object name may consist of up to 222 characters. A valid character in an object name include '/' (forward slash). Using this character in object names can simulate some of the hierarchical structure of a file system, even though the slash is not a functionally significant entity.
3. Objects are replicated and distributed for availability. However, there is no distributed equivalent of a file lock. Therefore, the last entity to write to an object "wins." If you use Cloud Storage in a distributed application, the application is responsible for locking and serialization of access.
4. Cloud Storage treats objects as an unstructured series of bytes.

Storage classes

	Regional	Multi-Regional	Nearline	Coldline
Design Patterns	**Data that is used in one region** or needs to remain in region	**Data that is used globally** and has no regional restrictions	**Backups** Data that is accessed no more than once a month	Archival or **Disaster Recovery** (DR) data that is accessed no more than once a year
Feature	Regional	Geo-redundant	Backup	Archived or DR
Availability	99.9%	99.95%	99.0%	99.0%
Durability	99.999999999%	99.999999999%	99.999999999%	99.999999999%
Duration	Hot data	Hot data	30-day minimum	90-day minimum
Retrieval cost	none	none	$	$$

Google Cloud

Multi-Regional = Data is stored redundantly in multiple locations separated by at least 100 miles. Actual locations are not specified.
Multi-Regional is only available in "Multi-Regional" locations:
https://cloud.google.com/storage/docs/bucket-locations#location-mr

Regional is typically lower cost than Multi-Regional. When you select Regional, you must choose a location.
https://cloud.google.com/storage/pricing

Nearline and Coldline storage incur an "early deletion" charge of the minimum number of days, if an object is deleted before that time.

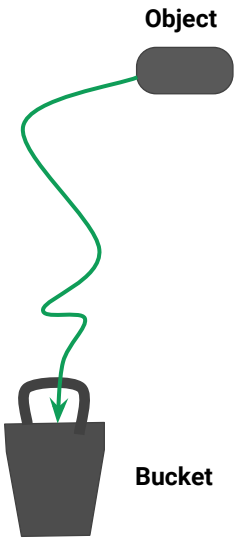
Regional and Multi-Regional storage classes typically return the first byte in less than a second (and often in tens of milliseconds). Nearline and Coldline storage may take seconds before the first byte is retrieved.

Cloud Storage overview

Buckets
- Naming requirements
- Cannot be nested

Objects
- Inherit storage class of bucket when created
- No minimum size; unlimited storage

Access
- `gsutil` command
- (RESTful) JSON API or XML API



Google Cloud

Name requirements:

- globally unique
- lowercase, #s, -, . (3-63 chars)
- URI = DNS CNAME

- Access control
  - ACL
  - Signed access
- Encryption at rest
- No minimum size; unlimited storage
- Pay for use
- 99.999999999% durability
- Low latency (time to first byte is typically tens of milliseconds)
- API access

The dot "." is also a valid character in a bucket name. It can be used to create domain-named buckets, such as mybucket.example.com. However, there is a verification process required to prove that you are the owner of the domain before this kind of bucket can be created. Max URI bucket name is 222 characters, with max 63 characters between dots.
https://cloud.google.com/storage/docs/domain-name-verification

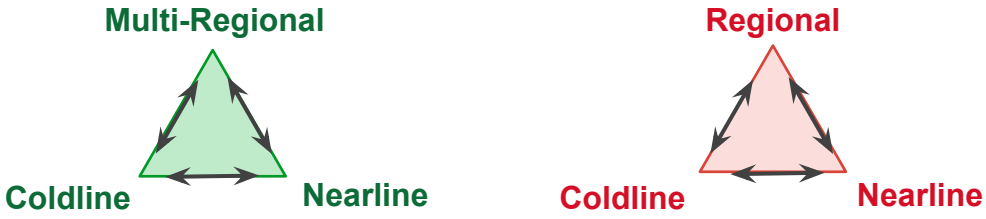
The JSON REST API is the preferred API. The XML API is a subset, it doesn't support all features, and it is used commonly with third-party tools.
`https://cloud.google.com/storage/docs/json_api/`
`https://cloud.google.com/storage/docs/xml-api/overview`

`gsutil` is the command line tool (a python application) used for managing Cloud Storage.
`gsutil` was developed separately from and is not assimilated into the `gcloud` tool.
https://cloud.google.com/storage/docs/gsutil

For more information on how subdirectories work, see
https://cloud.google.com/storage/docs/gsutil/addlhelp/HowSubdirectoriesWork

Note: If you have an authenticated and authorized domain, it is possible to use the Google DNS service to host a website in a bucket. The bucket does not have the ability to terminate SSL sessions, so it can natively serve only HTTP web traffic. However, as you will see in a later module, it is possible to terminate an SSL session with a load balancer.

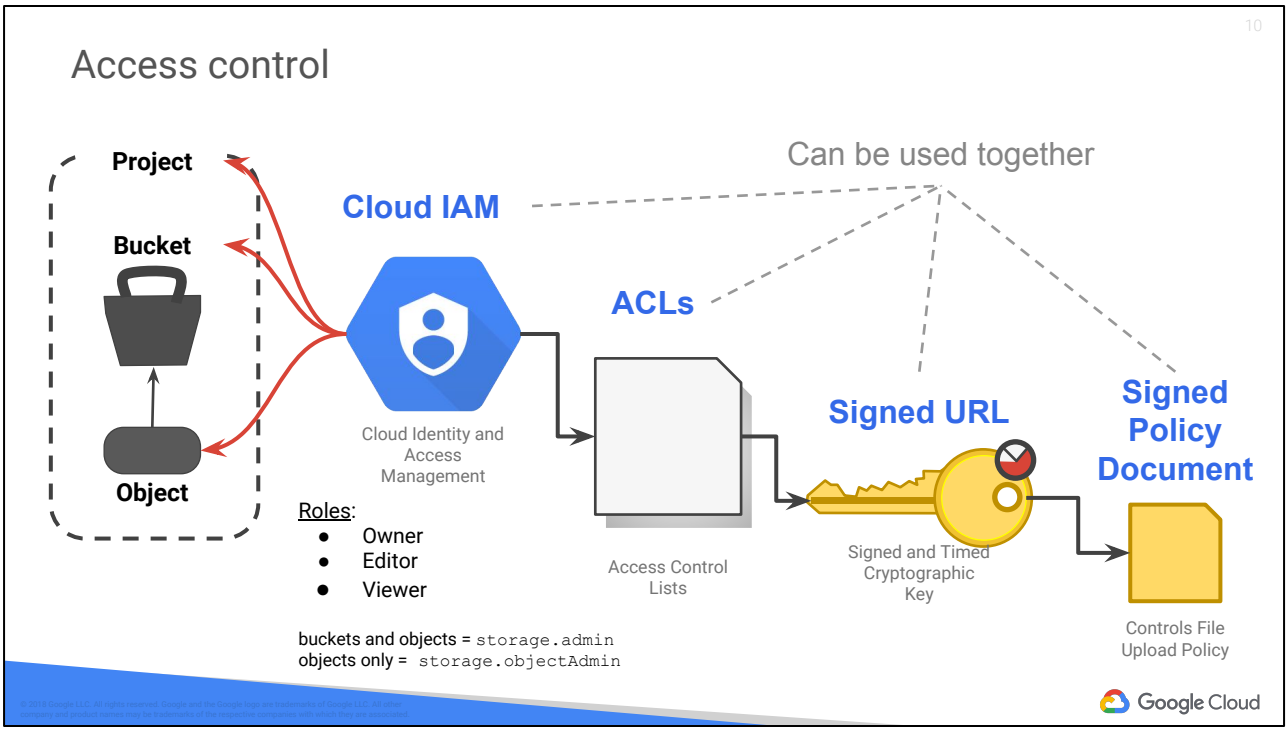
Changing default storage classes



- You can change the default storage class of a bucket.
- The default class is applied to objects as they are created in the bucket.
- The change only affects new objects added after the change.
- A Regional bucket can never be changed to Multi-Regional.
- A Multi-Regional bucket can never be changed to Regional.
- Objects can be moved from one bucket to another bucket with the same storage class from the GCP Console; however moving objects to buckets of *different* storage classes requires using the `gsutil` command from CloudShell.

Google Cloud

Moving buckets: Although you can change the storage class associated with a bucket, you cannot directly change a bucket's name or location after it is created. If there is no data in the bucket, you can simply delete the bucket and create it again with a new name or in a new location. If the bucket has data in it, you can create a new bucket with a different name and, if desired, in a new location, then copy data from the old bucket to the new bucket. The steps are here: https://cloud.google.com/storage/docs/moving-buckets



Cloud Storage offers layers of increasingly granular access control. For most purposes, Cloud IAM is sufficient, and roles are inherited from project to bucket to object. Access control lists (ACL) offer finer control. And for detailed control, signed URLs provide a cryptographic key that gives time-limited access to a bucket or object. A signed policy document further refines the control by determining what kind of file can be uploaded by someone with a signed URL.

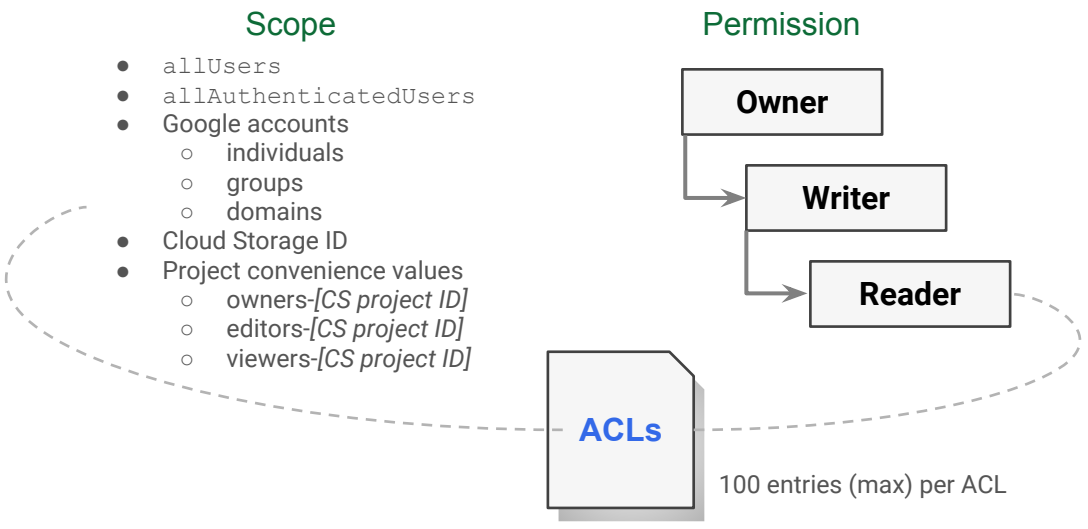
https://cloud.google.com/storage/docs/access-control/

Cloud IAM
https://cloud.google.com/storage/docs/access-control/iam
Works with Cloud Storage just as with using Cloud IAM with any other resource. Project Owners are automatically granted Bucket Owner role for all buckets in the project.

Note that ACLs and Cloud IAM are independent, so Project-level Cloud IAM permissions will not appear in bucket or object ACLs.

Signed URLs
A signed URL gives you the ability to grant access to a bucket without Cloud IAM user authentication for a limited period of time.
https://cloud.google.com/storage/docs/access-control/signed-urls

Access control lists (ACLs)



Predefined ACL *project-private* is applied by default to all new buckets and objects.

Google Cloud

ACL permissions are concentric, meaning that the greater access level includes the lesser.
Permission is the action that can be performed.
Scope (sometimes called "grantee") is the identity that can perform the action.
Applied to bucket or to object.
"Owner" is called "FULL_CONTROL" in the API.
Predefined ACLs provide a convenient way to change permissions for common scenarios, for example, revoking access from everyone.
https://cloud.google.com/storage/docs/access-control/lists

Signed URLs

- "Valet key" access to buckets and objects via ticket:
  - Ticket is a cryptographically signed URL
  - Time-limited
  - Operations specified in ticket: HTTP GET, PUT, DELETE (not POST)
  - Program can decide when or whether to give out signed URL
  - Any user with URL can invoke permitted operations
- Example using private account key and gsutil:

```
gsutil signurl -d 10m path/to/privatekey.p12 gs://bucket/object
```


Google Cloud

For some applications, it is easier and more efficient to grant limited-time access tokens that can be used by any user, instead of using account-based authentication for controlling resource access (e.g., when you don't want to require users to have Google accounts).

Signed URLs allow you to do this for Cloud Storage. You create a URL that grants read or write access to a specific Cloud Storage resource and specifies when the access expires. That URL is signed using a private key associated with a service account. When the request is received, Cloud Storage can verify that the access-granting URL was issued on behalf of a trusted security principal (the service account), and delegates its trust of that account to the holder of the URL.

After you give out the signed URL, it is out of your control. People copy it or steal it. So you want the signed URL to expire after some reasonable amount of time.

The types of operations that could be performed by using signed URL slightly align with the XML APIs.

Example: Signed URL

```
http://google-testbucket.storage.googleapis.com/testdata.txt?
GoogleAccessId=1234567890123@developer.gserviceaccount.com&
Expires=1331155464&
Signature=BClz9e...LoDeAGnfzCd4fTsWcLbal9sFpqXsQI8IQi1493mw%3D
```

- Query parameters
  - GoogleAccessId: Email address of the service account
  - Expires: When the signature expires in UNIX epoch
  - Signature: Signature is cryptographic hash of composite URL string
- The string that is digitally signed must contain:
  - HTTP verb (GET)
  - Expiration
  - Canonical resource (/bucket/object)

Google Cloud

This is an example signed URL that authorizes reading an object.

The URL contains:

- GCS path to the object
- GoogleAccessID - used by Cloud Storage to get the corresponding public key to decrypt the signature
- Expiration time (Unix) epoch
- Cryptographic signature

The string you sign is based on the following:

- HTTP Verb + '\n'*
- Content_MD5 + '\n'
- Content_Type + '\n'
- Expiration + '\n'*
- Canonicalized_Extension_Headers + '\n'
- Canonicalized_Resource + '\n'*

* = required

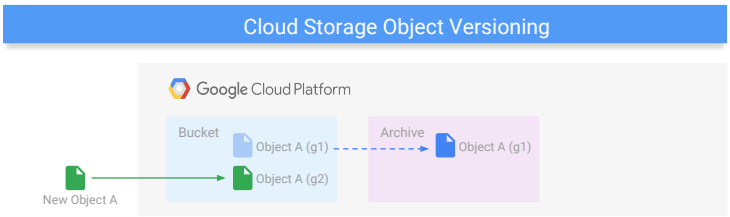
Cloud Storage features

- Customer-supplied encryption key (CSEK)
  - Use your own key instead of Google-managed keys
- Object Lifecycle Management
  - Automatically delete or archive objects
- Object Versioning
  - Maintain multiple versions of objects*
- Directory synchronization
  - Synchronizes a VM directory with a bucket
- Object Change Notification
- Data import
- Strong consistency

Google Cloud

*Note: You are charged for the versions as if they were multiple files.

Object Versioning



- Buckets with Object Versioning enabled maintain a history of modifications (overwrite/delete) of objects.
- Bucket users can list archived versions of an object, restore an object to an older state, or delete a version.
- Supports conditional updates.

Google Cloud

For more information, see: https://cloud.google.com/storage/docs/object-versioning

Object Lifecycle Management

- Lifecycle management policies specify actions to be performed on objects that meet certain rules.
- Examples include:
  - Downgrade storage class on objects older than a year
  - Delete objects created before a specific date
  - Keep only the 3 most recent versions of an object
- Object inspection occurs in asynchronous batches, so rules may not be applied immediately.
- Changes to lifecycle configurations can take 24 hours to apply.

Google Cloud

Lifecycle actions include:

- Delete
- SetStorageClass

Lifecycle conditions include:

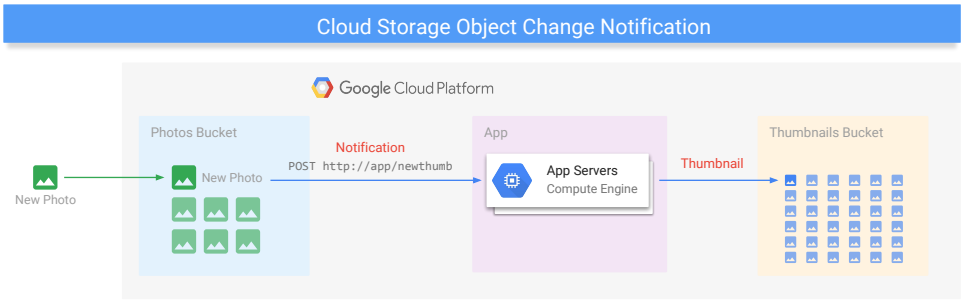
- Age
- CreatedBefore
- IsLive
- MatchesStorageClass
- NumberOfNewerVersions

Lifecycle configurations can be updated using the XML API, the JSON API, or the gsutil utility.

For more information, see: https://cloud.google.com/storage/docs/lifecycle

Object Change Notification

Cloud Storage can watch a bucket and send notifications to external applications when objects change (through a web hook).



Cloud Pub/Sub Notifications for Cloud Storage is the recommended way to track changes.

Google Cloud

For more information, see:
https://cloud.google.com/storage/docs/object-change-notification

Cloud Pub/Sub Notifications are the recommended way to track changes to objects in your Cloud Storage buckets because they're faster, more flexible, easier to set up, and more cost-effective.

Data import

- **Storage Transfer Service** enables high-performance imports of online data into Cloud Storage buckets
  - Imports from another bucket, an S3 bucket, or web source
  - Can schedule transfers, create filters, etc.
  - Accessible via console or APIs
- **Google Transfer Appliance** enables secure transfer of up to a petabyte of data by leasing a high capacity storage server from Google
- **Offline Media Import** is a service where physical media is sent to a third-party provider who uploads the data

Google Cloud

Transfer Appliance (Beta) is a high-capacity storage server that you lease from Google. You connect it to your network, load it with data, and then ship it to an upload facility where the data is uploaded to Cloud Storage. Transfer Appliance enables you to securely transfer up to a petabyte of data on a single appliance.

For more information, see:
https://cloud.google.com/transfer-appliance/docs/introduction
https://cloud.google.com/storage/docs/offline-media-import-export

Strong consistency

Cloud Storage provides strong global consistency, including both data and metadata:

- Read-after-write
- Read-after-metadata-update
- Read-after-delete
- Bucket listing
- Object listing
- Granting access to resources

Google Cloud

When you upload an object to Cloud Storage and you receive a success response, the object is immediately available for download and metadata operations from any location where Google offers service. This is true whether you create a new object or overwrite an existing object. Because uploads are strongly consistent, you will never receive a 404 Not Found response or stale data for a read-after-write or read-after-metadata-update operation.

In addition, when an upload request succeeds, it means your data is replicated in multiple data centers. The latency for writing to Cloud Storage's globally consistent, replicated store may be slightly higher than for a non-replicated or non-committed store. This is because a success response is returned only when multiple writes are complete, not just one.

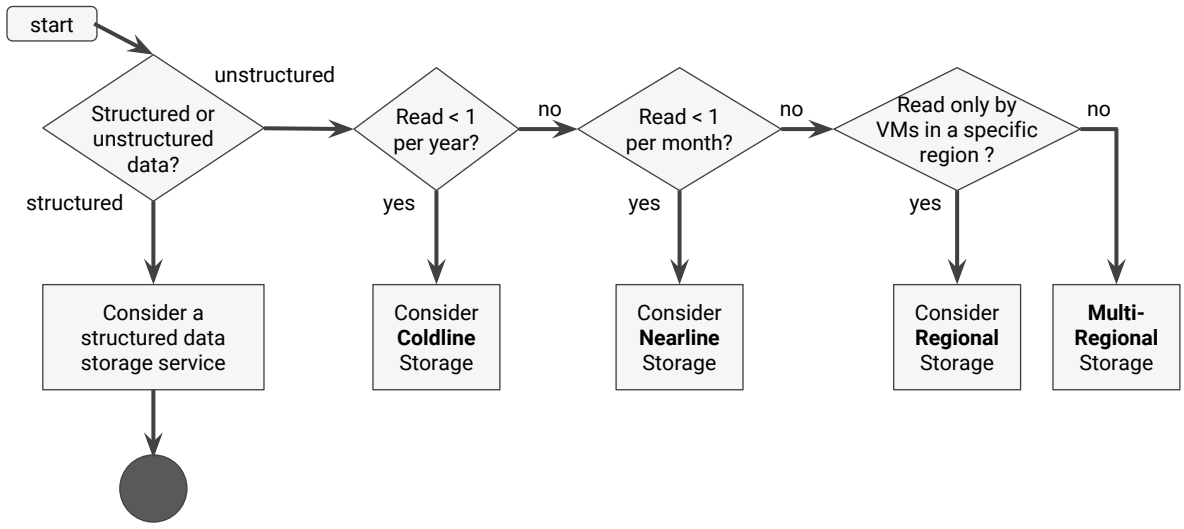
Strong global consistency also extends to deletion operations on objects. If a deletion request succeeds, an immediate attempt to download the object or its metadata will result in a 404 Not Found status code. You get the 404 error because the object no longer exists after the delete operation succeeds.

Bucket listing is strongly consistent. For example, if you create a bucket, then immediately perform a list buckets operation, the new bucket appears in the returned list of buckets.

Object listing is also strongly consistent. For example, if you upload an object to a bucket and then immediately perform a list objects operation, the new object appears

in the returned list of objects.

Choosing Cloud Storage



Google Cloud

Another reason to choose Regional storage is if you want to ensure that your data stays within a specific geographic or political region, to comply with legal requirements, for example.

Agenda

- Google Cloud Storage
- **Lab**
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Datastore
- Lab
- Cloud Bigtable
- Quiz

Lab: Cloud Storage

Objectives

In this lab, you learn how to perform the following tasks:

- Create and use buckets
- Set access control lists to restrict access
- Use your own encryption keys
- Implement version controls
- Use directory synchronization
- Share a bucket across projects using Cloud IAM

Completion: 60 minutes

Access: 120 minutes

Lab review

In this lab you learned to create and work with buckets and objects, and you learned about the following features for Cloud Storage:

- CSEK: Customer-supplied encryption key
  - Use your own encryption keys
  - Rotate keys
- ACL: Access control list
  - Set an ACL for private, and modify to public
- Lifecycle management
  - Set policy to delete objects after 31 days
- Versioning
  - Create a version and restore a previous version
- Directory synchronization
  - Recursively synchronize a VM directory with a bucket
- Cross-project resource sharing using Cloud IAM
  - Enable access to resources across projects

Agenda

- Google Cloud Storage
- Lab
- **Cloud SQL**
- Lab
- Cloud Spanner
- Cloud Datastore
- Lab
- Cloud Bigtable
- Quiz

Google Cloud

Data storage services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values, HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Google Cloud

Cloud SQL is a hosted MySQL service.
It provides:

- Rich query language
- Primary and secondary indexes
- ACID transactions
- Relational integrity
- Stored procedures

Cloud SQL

- Fully managed MySQL and PostgreSQL databases
  - Fully managed instances
  - Patches and updates automatically applied
  - You still have to administer MySQL users
- Cloud SQL supports many clients
  - `gcloud beta sql`
  - App Engine, G Suite scripts
  - Applications and tools
    - SQL Workbench, Toad
    - External applications using standard MySQL drivers

Cloud SQL

Google Cloud

You can't reuse an instance name for 7 days after it is deleted.
There are several unsupported features and statements:
https://cloud.google.com/sql/docs/features

Cloud SQL supports: Stored procedures, Triggers, and Views
Cloud SQL does not support: User-defined functions, Internal MySQL replication, statements and functions related to files and plugins

Cloud SQL generations

- Second Generation (recommended)
  - Up to 7x throughput and 20x storage capacity of First Generation
  - Up to 208 GB of RAM and 10 TB of data storage
  - MySQL 5.6 or 5.7
  - InnoDB only
- First Generation
  - Up to 16 GB of RAM and 500 GB of data storage
  - MySQL 5.5
  - IPv6 connectivity and On Demand activation policy
- *Some MySQL features not supported; UDFs*

Google Cloud

Second Generation compares with First Generation:

- Up to 7x greater throughput and 20x storage capacity
- Less expensive for most use cases
- Failover and replica options

For more differences, see
https://cloud.google.com/sql/docs/mysql/1st-2nd-gen-differences?hl=en_US&_ga=2.111246068.-1171753218.1503062023#differences

Cloud SQL services

Replica services
- Read replicas
- Failover replicas
- External replicas
- External master to Cloud SQL first gen replica

Backup service
- On demand
- Scheduled/automated

Auto-quiesce

Import/Export
- Import data to Cloud SQL
- Export data from Cloud SQL

Scaling
- Scale up (change machine capacity)
  - Requires restart
- Scale out
  - Via read replicas

Google Cloud

Replication
Instances are replicated across zones in a region. Automatically fails over to another zone if an outage occurs.
Options:
1) synchronous replication for slower, more reliable writes
2) asynchronous replication for faster writes that could result in loss of latest updates in the unlikely event of a data center failure
https://cloud.google.com/sql/docs/mysql/replication/tips

Backup schedule: Daily backups over a 7-day period; the time slot can be configured

Auto-quiesce can be disabled, on by default. Quiesces MySQL after a period with no traffic. Restarts when traffic arrives.
Feature to reduce cost when traffic is bursty or infrequent. Or disable it to run "hot" for faster first responses.

Connecting to Cloud SQL

- Basic connection
  - IP address
  - Add authorized network
- Secure access
  - Whitelist IP addresses
  - Add to authorized networks list
  - Configure SSL
  - Static IP
- Instance level access
- MySQL database level access
  - MySQL users
  - Database access

Google Cloud

For more information, see:
https://cloud.google.com/sql/docs/mysql/configure-ssl-instance
https://cloud.google.com/sql/docs/mysql/connect-admin-ip

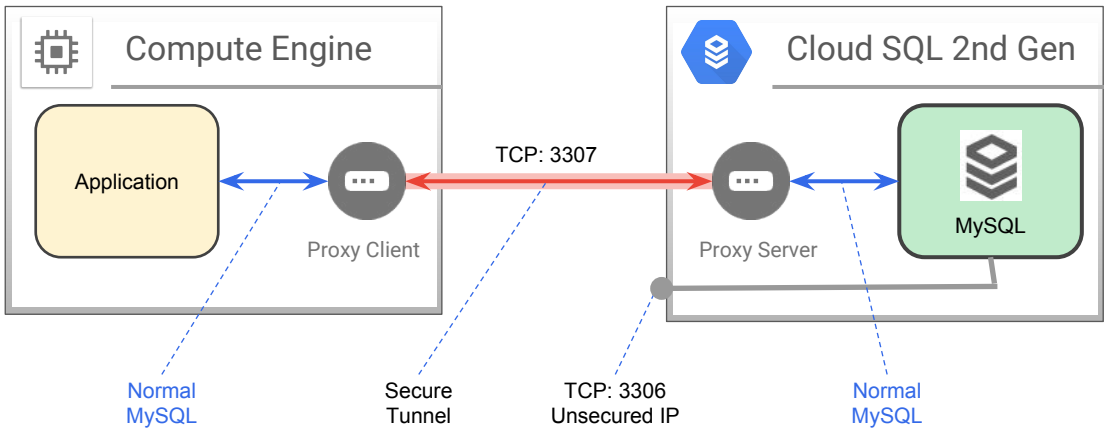
Connecting from outside Google Cloud Platform:

1. Create Cloud SQL instance in desired region.
2. Determine the IP address of your machine.
3. Add external IP address of machine in Cloud SQL instance authorized networks list.
4. Assign Cloud SQL an IPv4 address.
5. Connect with username and password to Cloud SQL address.

Connecting from Compute Engine:

1. Optionally create Cloud SQL instance in same zone as Compute Engine instance.
2. Assign Compute Engine external IP address.
3. Add external IP address of Compute Engine instance in Cloud SQL instance authorized networks list.
4. Assign Cloud SQL an IPv4 address.
5. Connect with username and password to Cloud SQL address.

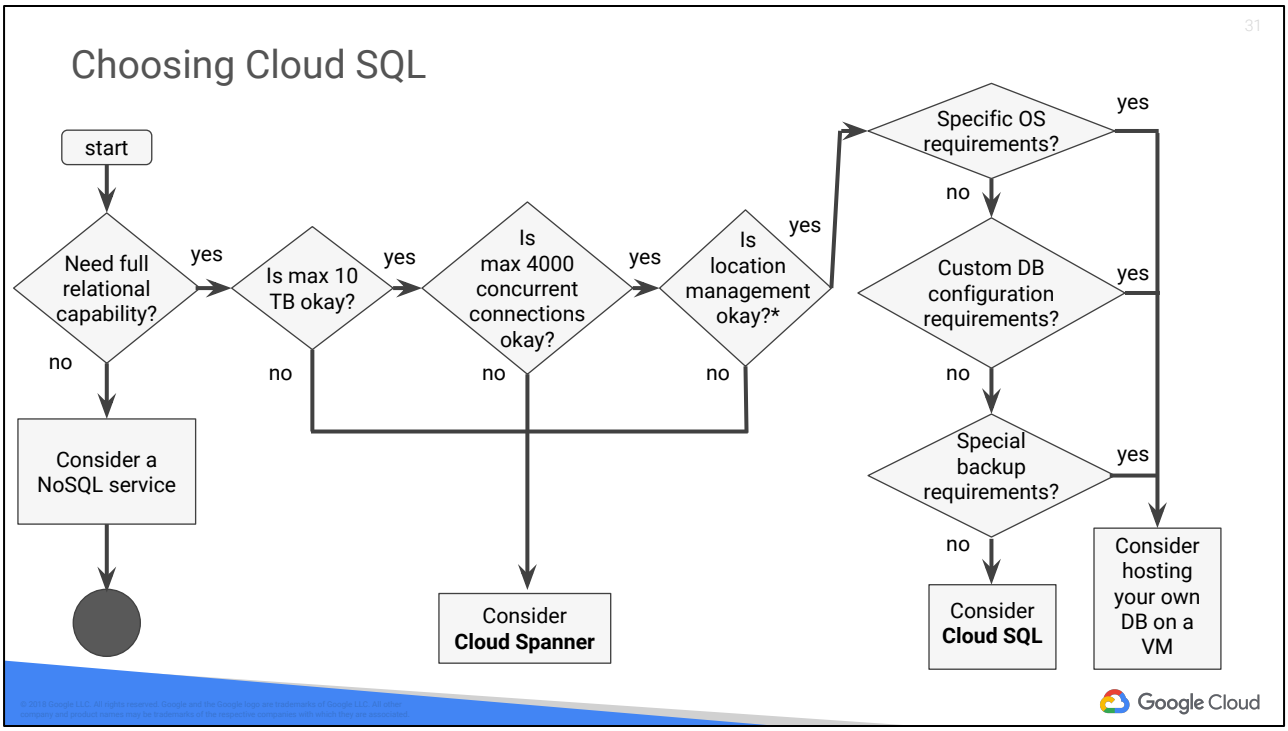
Cloud SQL Proxy and Secure SSL



Google Cloud

https://cloud.google.com/sql/docs/mysql-connect-proxy
https://cloud.google.com/sql/docs/sql-proxy

If you are not using Cloud SQL Proxy, you are strongly encouraged to use Cloud SSL:
https://cloud.google.com/sql/docs/mysql/configure-ssl-instance



Cloud SQL First Generation has a maximum database size of 250 GB and supports MySQL 5.5 and 5.6.
If you need version compatibility and your application won't surpass the 250-GB size, consider Cloud SQL First Generation.

*Note that you can get Multi-Regional capability from MySQL by creating MySQL replicas. Cloud Spanner was built with global scale in mind. The real question here is, if you are scaling up globally, do you want your application design to be responsible for scaling, availability, and location management? Or do you want to delegate those complicated problems to a service? Cloud Spanner allows you to expand horizontally globally, and the service handles many of the details.

Agenda

- Google Cloud Storage
- Lab
- Cloud SQL
- **Lab**
- Cloud Spanner
- Cloud Datastore
- Lab
- Cloud Bigtable
- Quiz

Lab: Cloud SQL

Objectives

In this lab, you learn how to perform the following tasks:

- Create a Cloud SQL instance
- Create a VM to serve as a database client and install software
- Restrict access to the Cloud SQL instance to a single IP address
- Download sample GCP billing data in *.csv format and load that into the database
- Configure the Cloud SQL instance and the client to use SSL encryption

Completion: 45 minutes

Access: 90 minutes

Lab Review

In this lab you:

- Configured Cloud SQL for use by a client.
- Improved security by requiring SSL certificates.

Agenda

- Google Cloud Storage
- Lab
- Cloud SQL
- Lab
- **Cloud Spanner**
- Cloud Datastore
- Lab
- Cloud Bigtable
- Quiz

Data storage services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values, HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Google Cloud

Before Cloud Spanner, architects were forced to choose between scalability using a NoSQL solution or transactional integrity using a Relational solution.

Cloud Spanner

- Strong consistency
  - Strongly consistent secondary indexes
  - Globally consistent
- SQL support
  - SQL (ANSI 2011 with extensions)
  - ALTER statements for schema changes
- Managed instances
  - High availability through data replication

Cloud Spanner

Google Cloud

Cloud Spanner is suited for applications that require relational database support, strong consistency, transactions, and horizontal scalability.
Natural use cases include financial applications and inventory applications traditionally served by relational database technology.

Characteristics

	Cloud Spanner	Relational DB	Non-Relational DB
Schema	Yes	Yes	No
SQL	Yes	Yes	No
Consistency	Strong	Strong	Eventual
Availability	High	Failover	High
Scalability	Horizontal	Vertical	Horizontal
Replication	Automatic	Configurable	Configurable

Google Cloud

Cloud Spanner is the first horizontally scalable globally consistent database. It is proprietary, not open source.
Cloud Spanner is used for applications that use relational, structured, and semi-structured data and require high availability, strong consistency, and transactional reads and writes.

Mission-critical use cases include:

- Powering customer authentication and provisioning for multi-national businesses
- Building consistent systems for transactions and inventory management in the financial services and retail industries
- Supporting high-volume systems that require low latency and high throughput in the advertising and media industries

Cloud Spanner

Open standards

- Standard SQL (ANSI 2011)
- Encryption, Audit logging, Identity and Access Management
- Client libraries in popular languages
- Java, Python, Go, Node.js
- JDBC driver

Workloads

- Transactional
- Scale-out
- Global data plane
- Database consolidation

Google Cloud

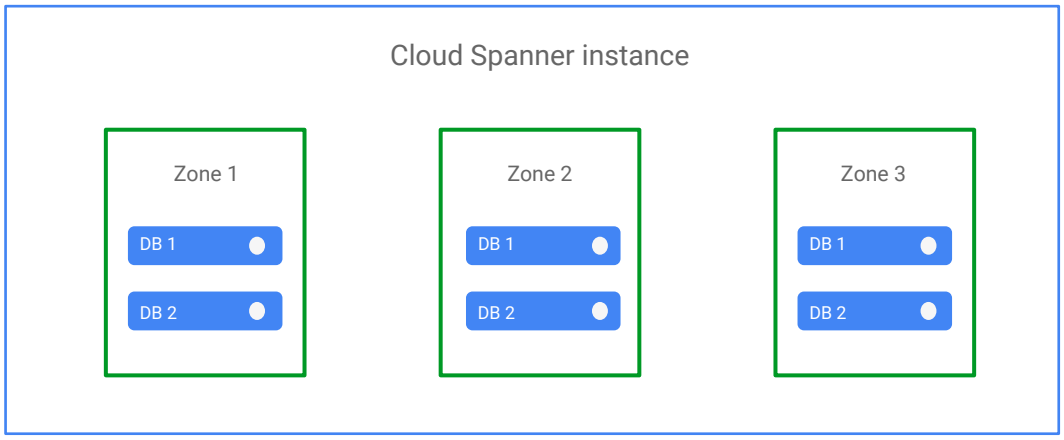
Transactional
Companies that have outgrown their single-instance RDBMS and have already moved to NoSQL solution, but need transactional consistency, or they are looking to move to a scalable solution

Scale-out
Companies currently sharding databases because they need more read or write throughput than can be placed on a single node

Global data plane
Companies and/or developers building applications that have global data and need strong consistency

Database consolidation
Companies that store their business data in multiple database products with variable maintenance overheads and capabilities and need consolidation of their data

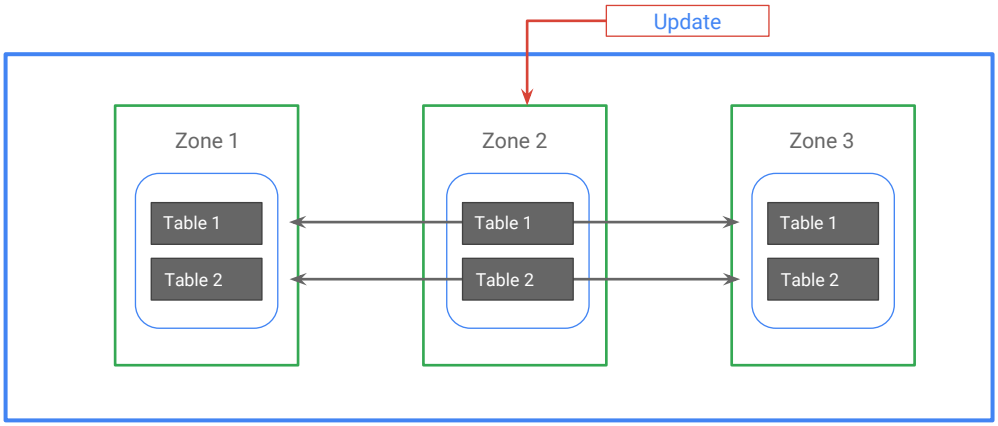
Architecture



Google Cloud

This architecture allows HA and global placement
Data is replicated in N cloud zones which can be within one region or across several regions.
DB placement is configurable. You can choose which region to put your DB in.

Data replication



Google Cloud

Writes are synchronous. Data is always consistent and has ACID properties like any other relational DB

Relational semantics
High throughput → data is sharded within the zone
HA built-in → No manual intervention needed on zone failure
Paxos replication (leslie lamport paper) - only require majority (not all zones) for commits

How Cloud Spanner works: https://research.google.com/pubs/pub45855.html

Cloud Spanner best practices: https://cloud.google.com/spanner/docs/best-practices

Cloud IAM roles

You can apply Cloud IAM roles to individual databases

- `spanner.admin`
- `spanner.databaseAdmin`
- `spanner.databaseReader`
- `spanner.databaseUser`
- `spanner.viewer`

Google Cloud

The course can't cover everything about Cloud Spanner here; the course could spend 3 to 4 days on Cloud Spanner alone.
For more information, see https://cloud.google.com/spanner/docs/

Feature details

- Tables
  - Optional parent-child relationships between tables
  - Interleaving of child rows within parent rows on primary key
- Primary keys and secondary keys
- Database splits
- Transactions
  - Locking read-write, and read-only
- Timestamp bounds
  - Strong, Bounded staleness, Exact staleness, Maximum staleness

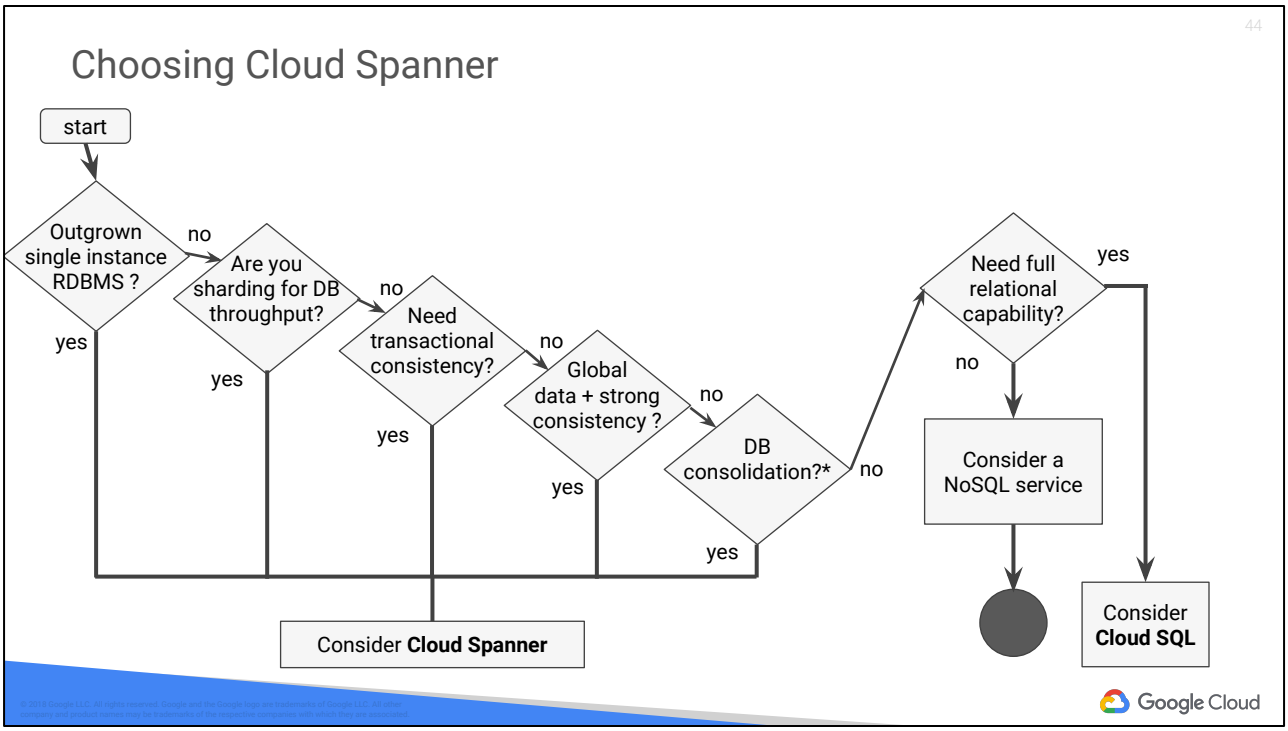
https://cloud.google.com/spanner/

Google Cloud

https://cloud.google.com/spanner/docs/schema-and-data-model

Cloud Spanner divides your data into chunks called "splits," where each split can move independently from another and get assigned to different servers, which could be in different physical locations.

Timestamp bounds
Strong: Cloud Spanner provides a bound type for strong reads. Strong reads are guaranteed to see the effects of all transactions that have committed before the start of the read.
Bounded staleness: Bounded staleness modes allow Cloud Spanner to pick the read timestamp, subject to a user-provided staleness bound.
Exact staleness: The timestamp can either be expressed as an absolute Cloud Spanner commit timestamp or a staleness relative to the current time.



*DB Consolidation
Some companies evolve complex database applications targeted to specific customer needs. The maintenance overhead for managing the multiple systems may be significant and variable. Cloud Spanner may be a candidate to consolidate the systems into a single managed service.

Agenda

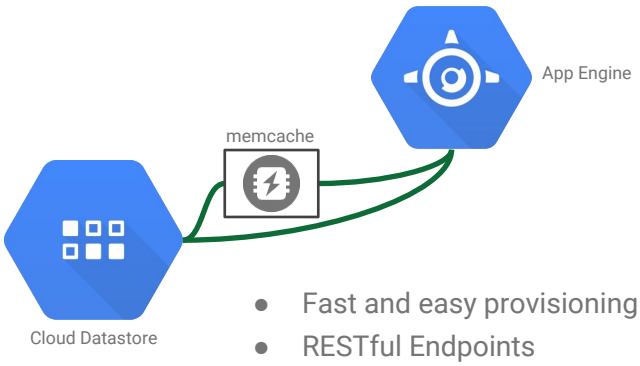
- Google Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- **Cloud Datastore**
- Lab
- Cloud Bigtable
- Quiz

Data storage services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values, HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Cloud Datastore

- API support for several languages
- Data replicated across several data centers
- Queries scale with the size of result set, not the size of data set
- Autoscale
- Schemaless access
- SQL-like capabilities
- Authentication, secure sharing



- Fast and easy provisioning
- RESTful Endpoints
- ACID Transactions
- Local Development tools
- Built-in Redundancy

Google Cloud

Yes, schema-less from the perspective of being able to store any entity
Schema matters a lot from the perspective of performance
Read performance to match the engineering design point of AppEngine (massively fast web reads)
Write performance is slow (this *is* engineering after all, and engineering is all about making careful tradeoffs).
The system is a key-entity storage system, not a key-value storage system.

Cloud Datastore is used with App Engine applications. It is good for structured pure-serve use cases. Cloud Datastore is not a good choice for relational database needs or for analytical data processing. Consider Cloud SQL and Cloud Bigtable, respectively.
Cloud Datastore is paired with a memcache service to increase performance for repeatedly read data. Typically in development, the App Engine application will try memcache first, and then on a cache-miss, access Cloud Datastore. This strategy radically improves performance and reduces costs.

JSON API, Java (JPA, JPO, Objectify), Python (NDB), Ruby, Node.js

Charges for storage and for read/write operations.

Information on consistency models in Cloud Datastore:
https://cloud.google.com/datastore/docs/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/

Cloud Datastore

Kind: category of an object
Entity: a single object
Property: individual data for an object
Key: unique ID

Entity groups and ancestor queries

- An "ancestor query" enables a strong consistency result, versus a non-ancestor query, which yields an eventually consistent result.
- Enables application developers to balance requests.

Google Cloud

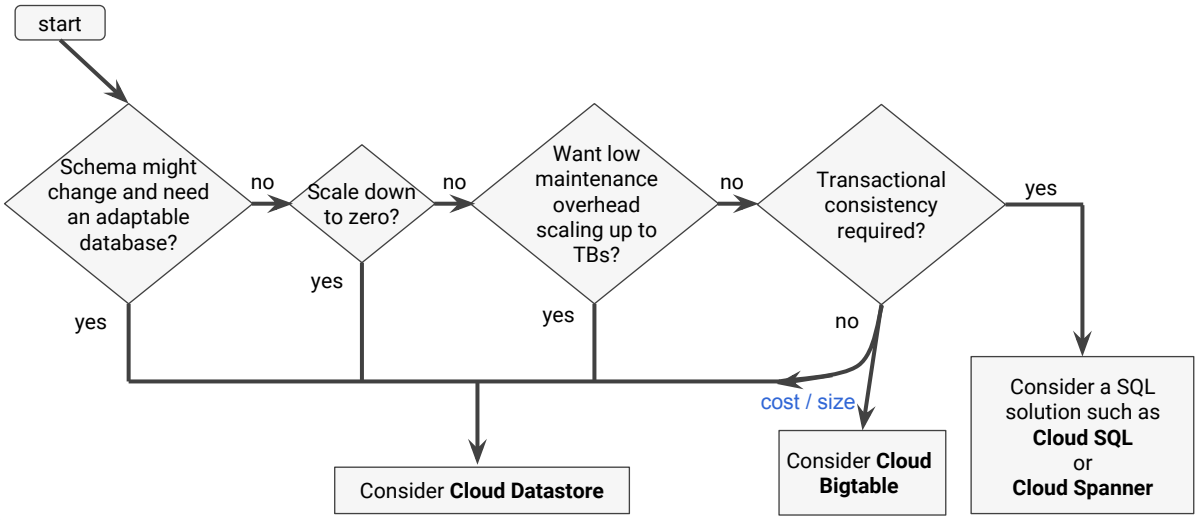
Compared to a relational database, here are the rough equivalents:
Kind = Table
Entity = Row
Property = Field
Key = Primary Key
https://cloud.google.com/datastore/docs/datastore-api-tutorial
https://cloud.google.com/datastore/docs/concepts/overview

Fully managed service, no provisioning required
Regional (US or Europe)
Transactions supported with some restrictions
Secondary indexes
Strong or Eventual consistency depending on design decisions
JSON API, Java (JPA, JPO, Objectify), Python (NDB), Ruby, Node.js
10s ms R/W latency. QPS and throughput are data model-dependent
< 200 TB
Charged for storage and read/write operations

Cloud Datastore replication

- Multiple locations
  - Multi-Regional
    - Multi-region redundancy; higher availability
  - Regional locations
    - Lower write latency; co-location with other resources
  - Global points of presence: lower latency for the end user
- Service-level agreement
  - Regional (multi-zone) and Multi-Regional replication enable high availability with an SLA covering:
    - 99.95% for Multi-Regional locations
    - 99.9% for Regional locations

Choosing Cloud Datastore



Google Cloud

Agenda

- Google Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Datastore
- **Lab**
- Cloud Bigtable
- Quiz

Lab: Cloud Datastore

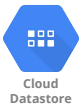
Objectives

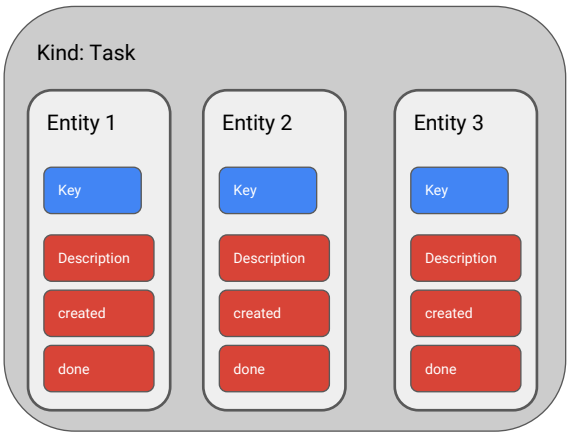
In this lab, you learn how to perform the following tasks:

- Initialize Cloud Datastore
- Create content in the database
- Query the content using both GQL and Kind queries
- Access the Cloud Datastore Admin console

Completion: 40 minutes

Access: 80 minutes





Google Cloud

Lab Review

In this lab you:

- Created a Cloud Datastore database
- Populated the database with data entities
- Ran
  - Query by kind
  - Query by GQL queries
- Enabled the Cloud Datastore Admin console so that you could learn how to:
  - Clean up data
  - Remove test data

Agenda

- Google Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Datastore
- Lab
- **Cloud Bigtable**
- Quiz

Google Cloud

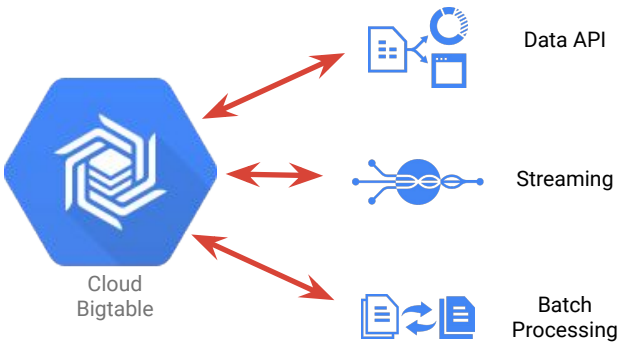
Data Storage Services

	Cloud Storage	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	BigQuery
Capacity	Petabytes +	Terabytes	Petabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Globally scalable RDBMS	Persistent Hashmap	Key-values, HBase API	Relational
Read	Have to copy to local disk	SELECT rows	transactional reads and writes	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row		put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	SQL, Schemas ACID transactions Strong consistency High availability	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud		Structured data from App Engine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Cloud Bigtable

- Fully managed NoSQL database
- Petabyte-scale with very low latency
- Seamless scalability for throughput
- Learns and adjusts to access patterns

Cloud Bigtable



- Flexible scalable data service
- High throughput, low latency
- High capacity (Petabytes)
- Ideal for storing single-keyed data with very low latency
- Excellent for big data applications
- Can be access via HBase extension
- Benefits over HBase
  - Scales by machine count, not limited by QPS
  - Low Ops: admin tasks are automatic
  - Resize in seconds

Google Cloud

NoSQL database, originally written to support search, now supports dozens of products.

Cloud Bigtable is a sparsely populated table that can scale to billions of rows and thousands of columns, allowing you to store terabytes or even petabytes of data. A single value in each row is indexed; this value is known as the row key. Cloud Bigtable is ideal for storing very large amounts of single-keyed data with very low latency. It supports high read and write throughput at low latency, and it is an ideal data source for MapReduce operations.
https://cloud.google.com/bigtable/docs/overview

- Cloud Bigtable is a NoSQL database system
- Petabytes in size
- A sparse, distributed, persistent multidimensional sorted map, indexed by row key, column key, and timestamp
- Each value in the map is an array of bytes
- Eventually consistent
- Cross-zone replication

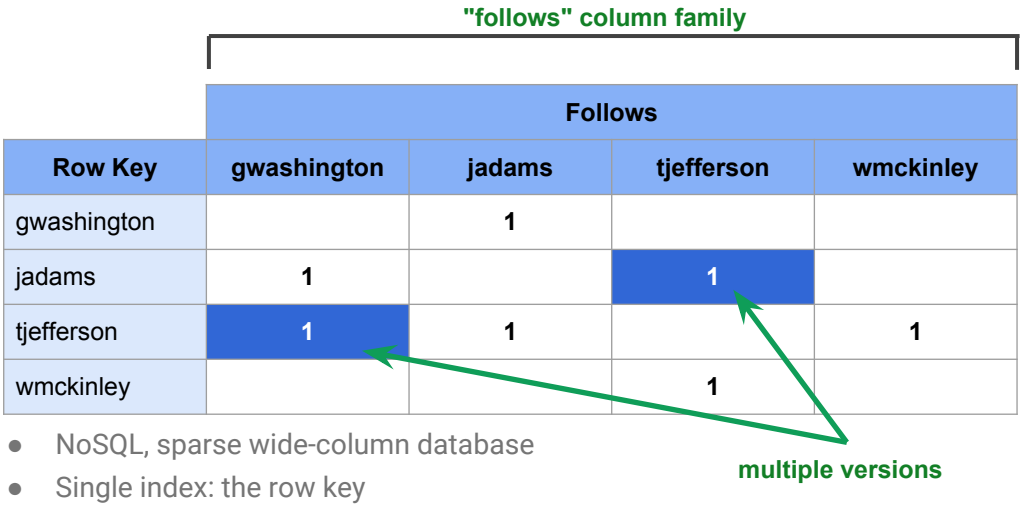
Data API
Data can be read from and written to Cloud Bigtable through a data service layer like: Managed VMs, the HBase REST Server, a Java Server using the HBase client. Typically this will be to serve data to applications, dashboards, and data services.

Streaming
Data can be streamed in (written event by event) through a variety of popular stream processing frameworks like Dataflow Streaming, Spark Streaming, or Storm.

Batch Processing
Data can be read from and written to Cloud Bigtable through batch processes like Hadoop MapReduce, Dataflow, or Spark. Often, summarized or newly calculated data is written back to Cloud Bigtable or to a downstream database

Cloud Bigtable
Need to create clusters and add nodes if more QPS is needed
Zonal (zones in us-central1, eu-west1 or asia-east1 regions).
Strongly consistent in one zone at a row level
HBase API (with some differences), Go Client library (gRPC)
Highly predictable. Up to 10,000 queries per second (QPS) for each node, throughput of up to 10 MB/s per node. Single ms R/W latency
2 TB–10 PB
Charged for storage and provisioned nodes

Structure and schema

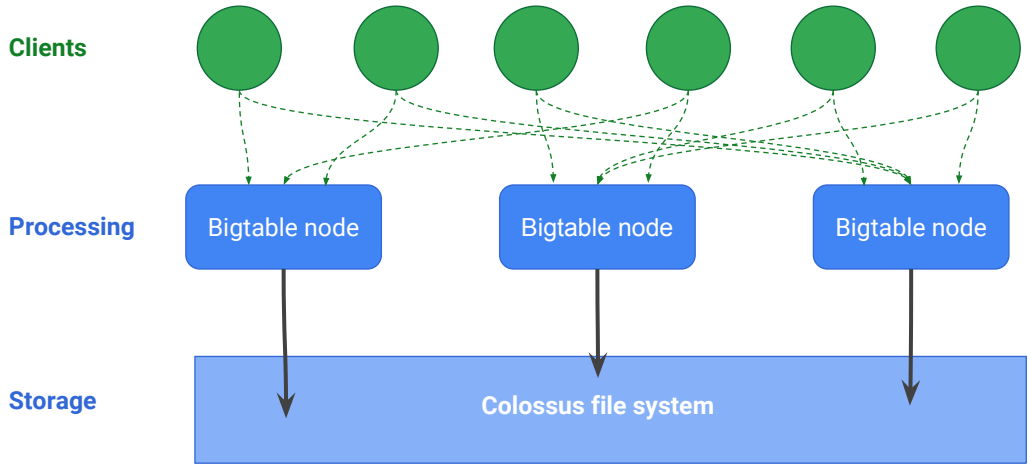


- NoSQL, sparse wide-column database
- Single index: the row key
- Atomic single-row transactions

Google Cloud

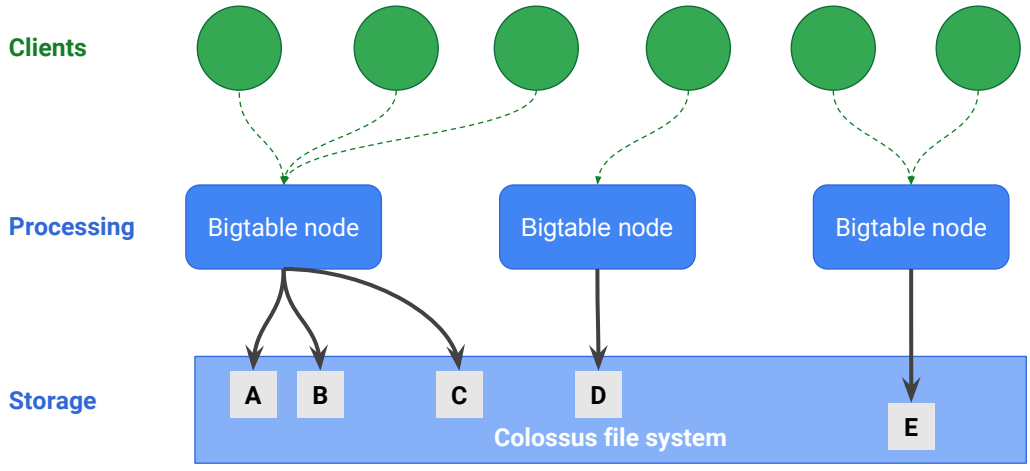
Now, if you want to see approximately how it works, it's a basic key value schema. You can compare this a lot, like DynamoDB from AWS. So it is NoSQL, you have a single index which is going to be your row key. There is additional information you can kind of put in there, but what you'll notice is, when you make changes to the data inside a big table, you just simply append a new version. That's how to get around or that's how to maintain low latency and high throughput. You do not delete the row; you can truncate data but you just simply append new updates. So if you're scanning across a bunch of rows, you're just going to simply take the latest value for what you're looking for.

Processing is separated from storage



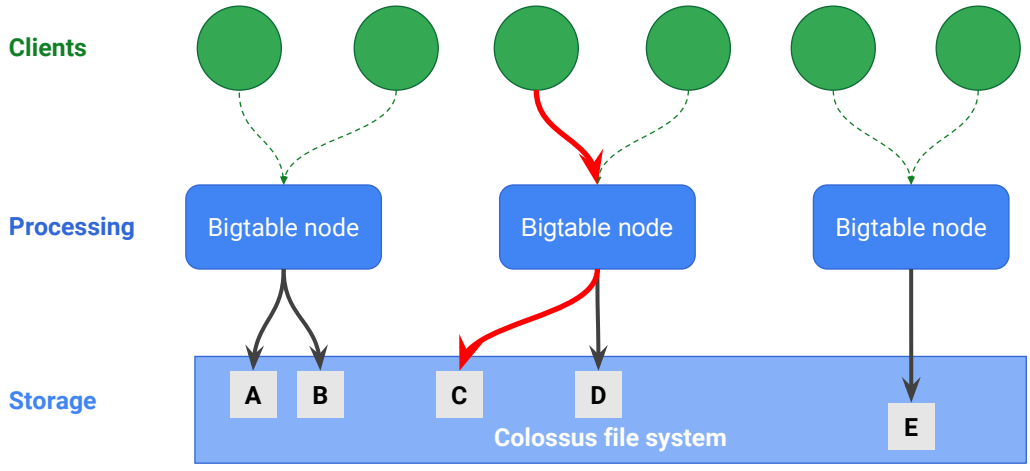
Google Cloud

Learns access patterns



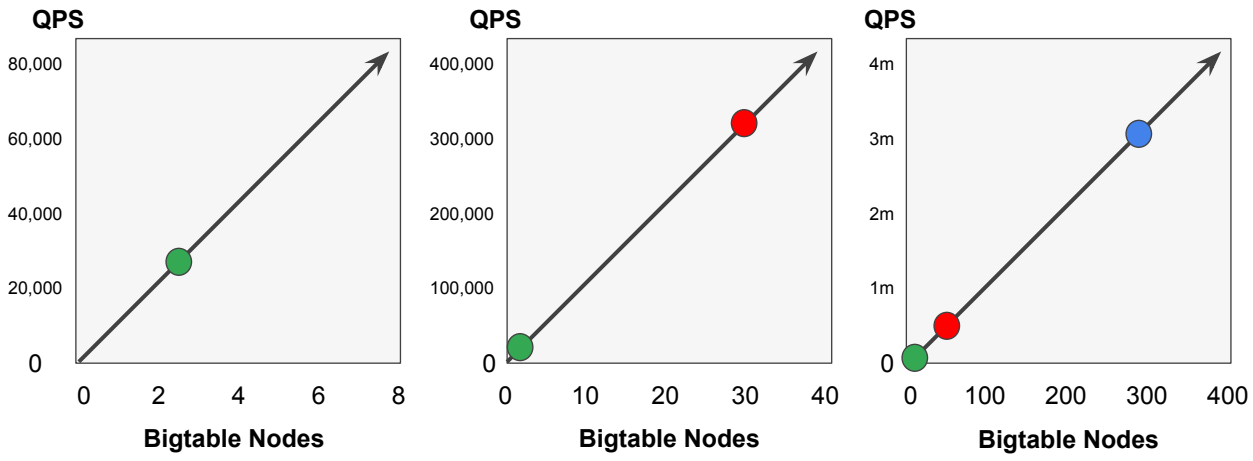
Google Cloud

Rebalances without moving data



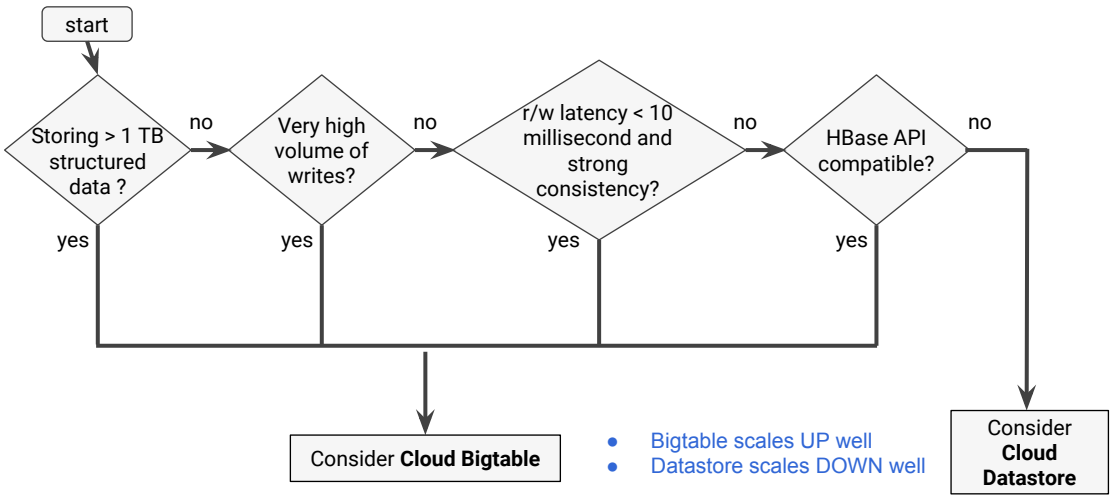
Google Cloud

Throughput scales linearly



Google Cloud

Choosing Cloud Bigtable



- Bigtable scales UP well
- Datastore scales DOWN well

Google Cloud

The smallest Cloud Bigtable cluster you can create has three nodes and can handle 30,000 operations per second. You pay for those nodes while they are operational, whether your application is using them or not. Compare this with Cloud Datastore, where you only pay for the resources that your application is using.

Agenda

- Google Cloud Storage
- Lab
- Cloud SQL
- Lab
- Cloud Spanner
- Cloud Datastore
- Lab
- Cloud Bigtable
- **Quiz**

Quiz

What data storage service might you select if you just needed to migrate a standard relational database running on a single machine in a data center to the cloud?

1. Cloud SQL
2. BigQuery
3. Persistent Disk
4. Cloud Storage

Quiz

Which GCP data storage service offers ACID transactions and can scale to thousands of nodes?

1. Cloud Storage
2. Cloud CDN
3. Cloud Spanner
4. Cloud SQL

Google Cloud

Quiz

Which data storage service provides data warehouse services for storing data but also offers an interactive SQL interface for querying the data?

1. BigQuery
2. Cloud Dataproc
3. Cloud Datalab
4. Cloud SQL

More resources

Cloud Storage
https://cloud.google.com/storage/docs/

Cloud SQL
https://cloud.google.com/sql/docs/

Cloud Spanner
https://cloud.google.com/spanner/docs/

Cloud Datastore
https://cloud.google.com/datastore/docs/

Cloud Bigtable
https://cloud.google.com/bigtable/docs/