STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers
Virtual Conference
September 28-29, 2021
A SNIA Event

Replication of Object Storage System

Gowtham Alluri (Staff Engineer),
Sarthak Moorjani (Member of Technical Staff)
Nutanix Inc

Agenda

- Object Storage: Overview & Building blocks
- Object Storage Replication: Use Cases
- Design Challenges and Solutions
- Performance

What is Object Storage

Block storage

- Blocks over volumes
- SCSI / iSCSI
- Structured Data

File Storage

- Hierarchical: Files & Directories
- NFS / SMB
- Structued & Semi Structured data

Object Storage

- Flat namespace: Buckets & Objects
- S3 - HTTP REST
- Semi structured & Unstructured data

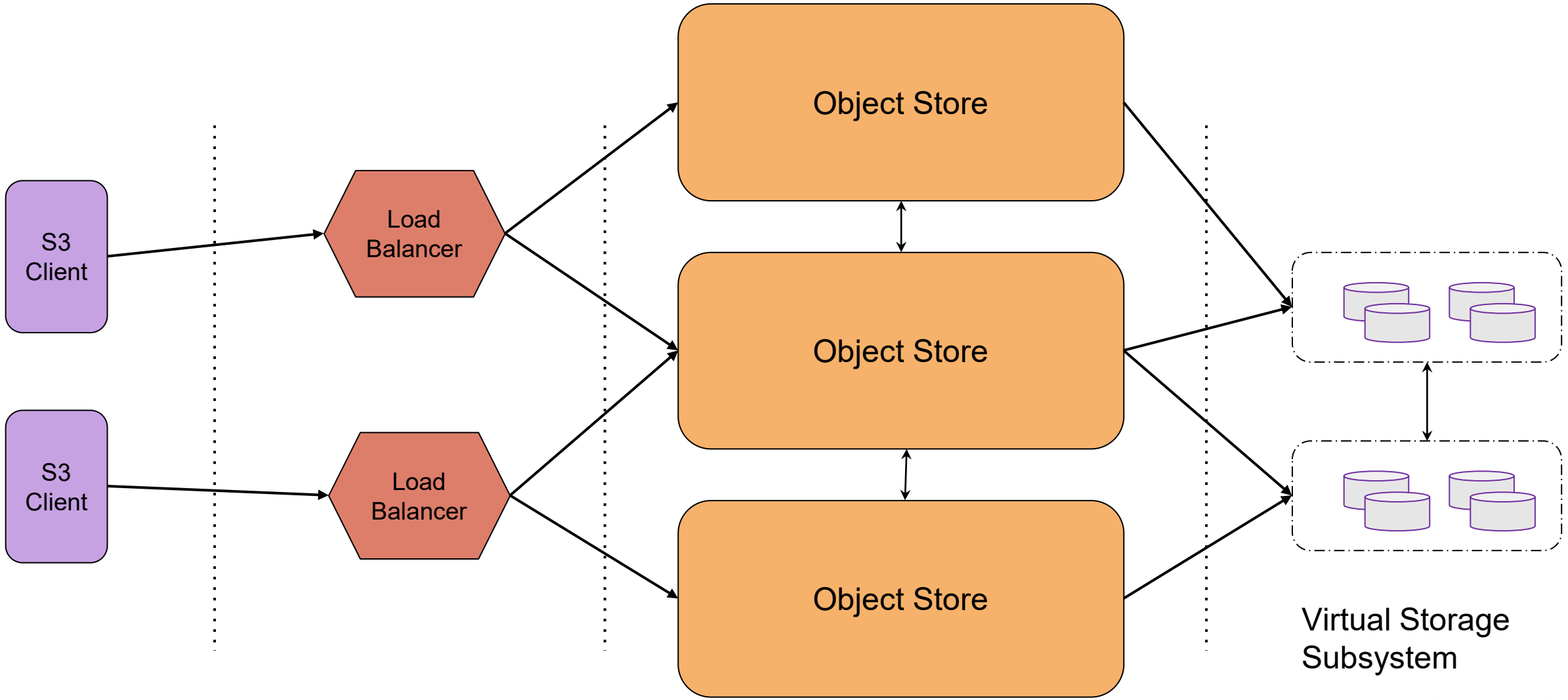
Object storage features

Object Versioning, Multipart uploads, Rich user metadata, Object tags, WORM, LegalHold etc...

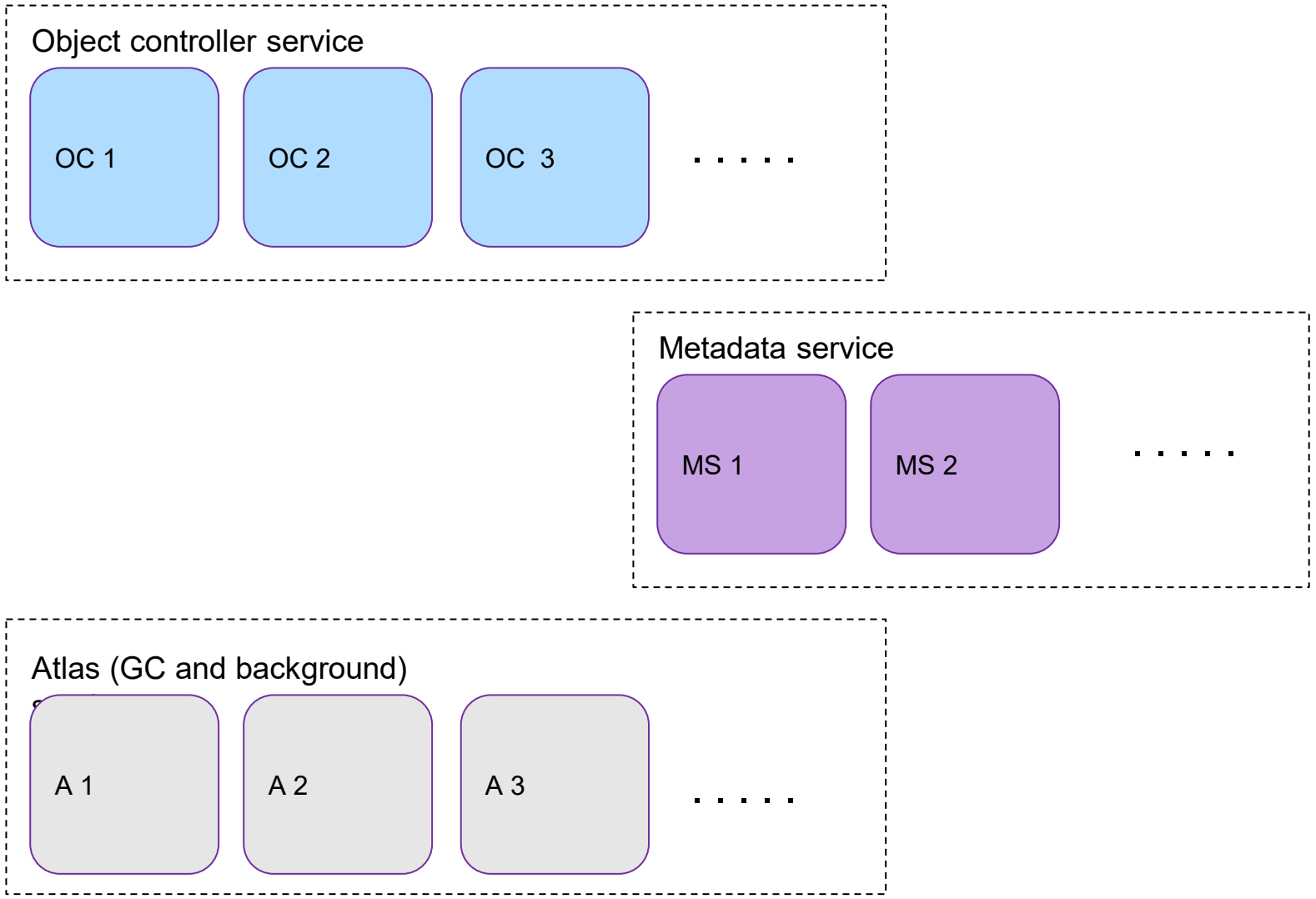
Why Object Storage

- Enterprise applications are increasingly supporting object storage.
- Cloud native apps are moving to on premises.
- Scale and performance.
- Evolving feature set - Website Hosting, Notifications, Life Cycle Policies etc...

Scale Out Object Storage Building Blocks



Object Storage System Components



Replication of Object Storage System

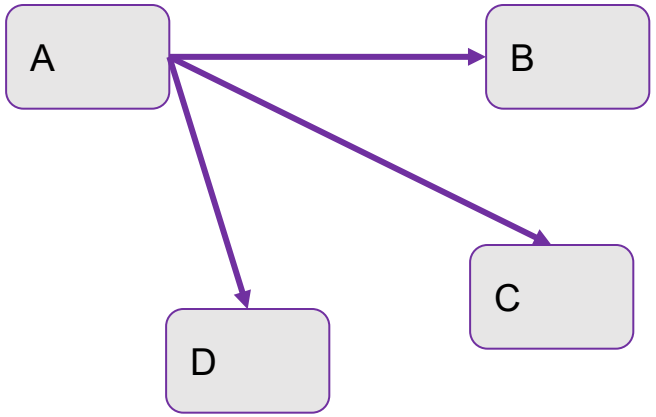
Replication Use Cases

- Disaster Recovery
  - Backups
- Data Compliance
  - Health records
- Low latency data access across geo-locations.
  - Data Analytics
- Sync and share to multiple destinations
  - Content distribution
- Migration
  - Migrate data between clusters

Replication Topologies

1. One way

2. Bi Directional



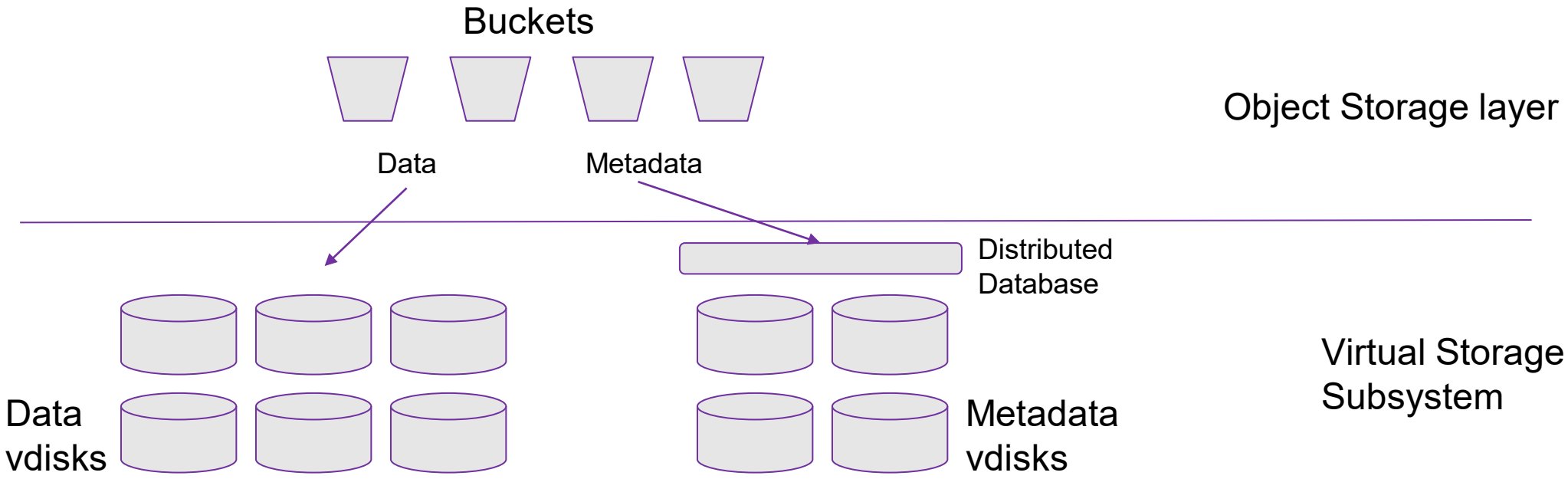
3. 1 to Many

4. Chain

Design Approach 1

Replication done at Virtual storage sub system

Challenges: Data and Metadata consistency
No granular ability for replication configuration.
Not scalable for larger capacity or multiple cluster systems.



Design Challenges

Replication Granularity:

Bucket level : all objects within a bucket

Sub-bucket level : Objects filter criteria ex: prefix, tags

An Object is more than data

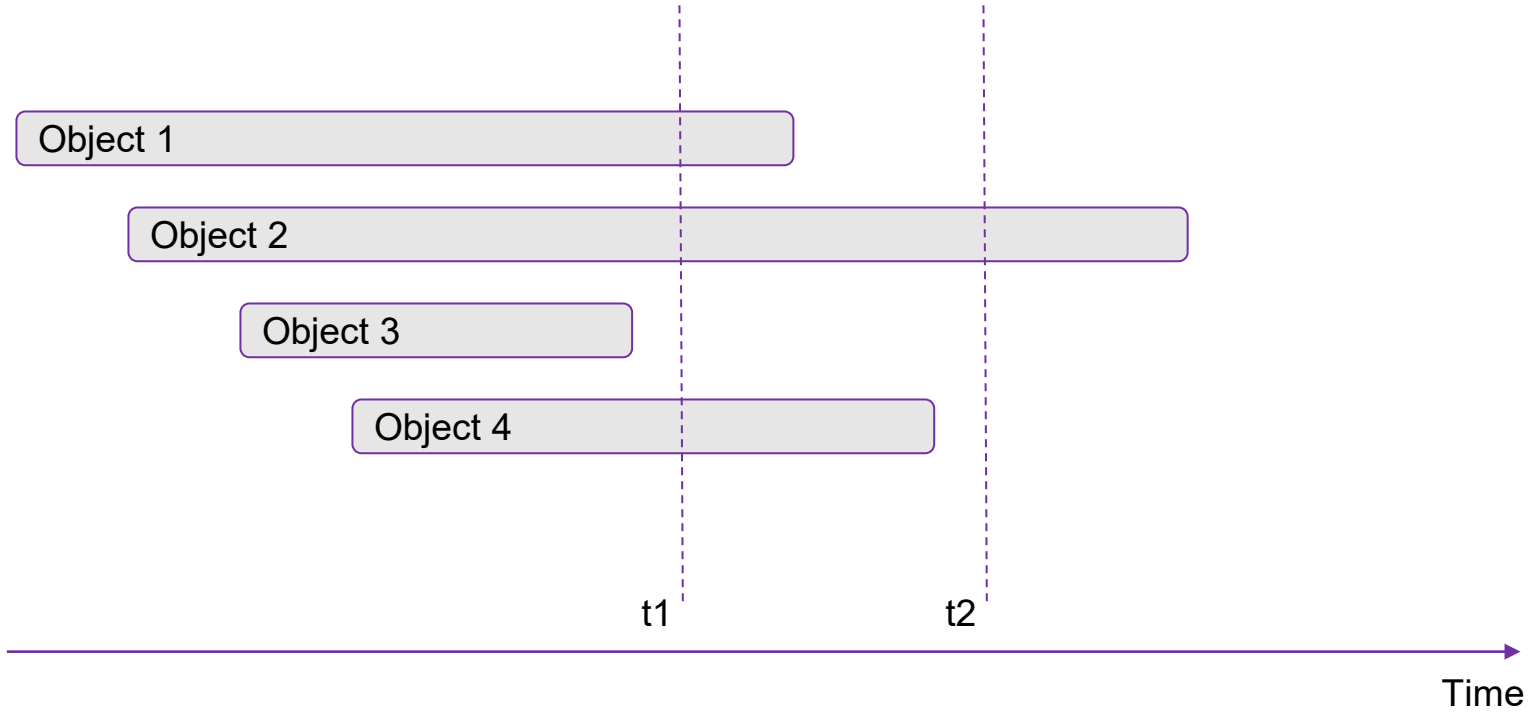
- System metadata (create/mod timestamps,version numbers etc...)
- User tags
- User metadata
- Properties (Expiration, Lock etc..)

Needs ability to track changes at each individual object level to replicate only the changes in order to achieve replication efficiency.

Design Challenges: Object size and ordering

Object sizes: Zero bytes to Terabytes.

Object upload finish order is different from start order.

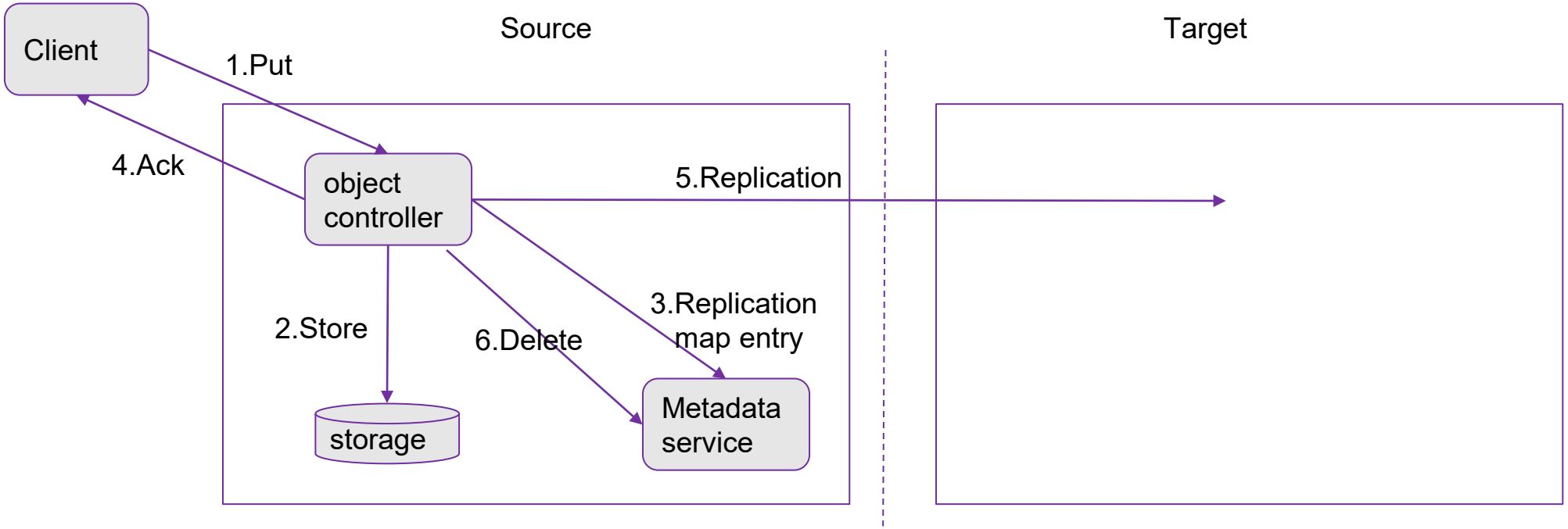


Replication: Change tracking

- Map based approach to track the changes on objects.
- Replication Map: Entries in the map identify the objects pending replication.
- Map entry lives as long as replication is pending.
- Scalable
  - Size of replication map is proportional to pending replications.
  - Multiple updates are absorbed in the same map entry.

Replication: Near Sync

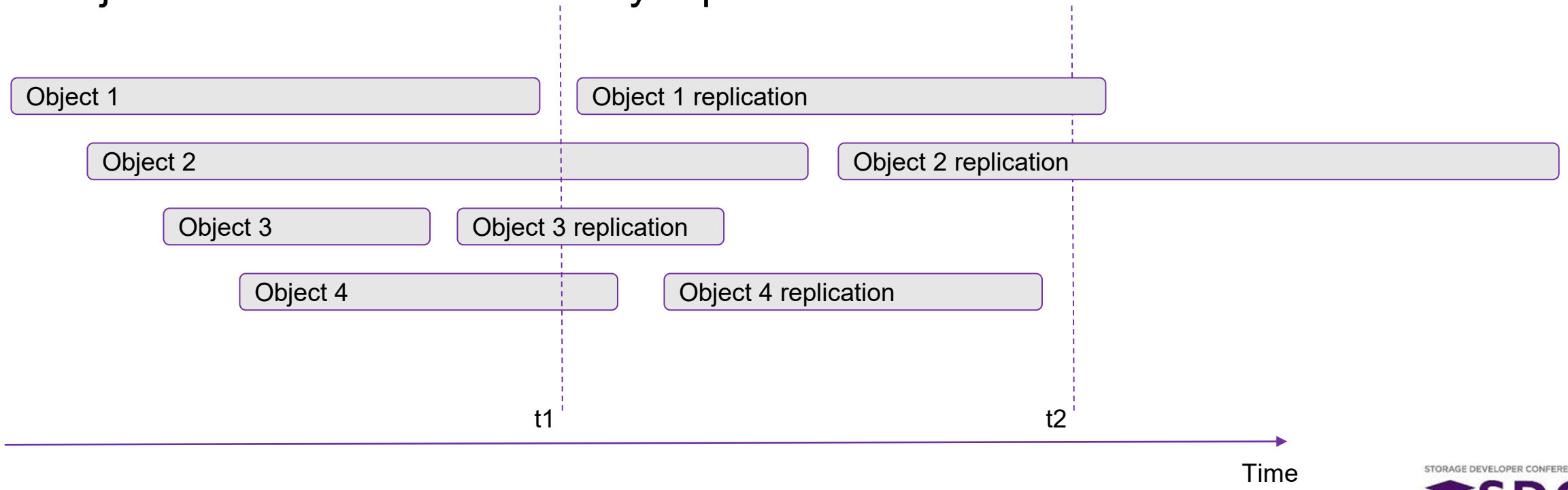
- Object put adds an entry in replication map.
- Replication starts after the put is acknowledged to client.
- The map captures multiple updates to the object while the replication is pending.
- If the foreground replication fails due to some error, we rely on background replication to achieve eventual consistency.



Replication: Eventual Consistency

Replication Ordering:

- Out of order replication across objects.
- Object versions are replicated out of order
- All object / versions are eventually replicated.



Design challenge: Track progress

Last Replication Point

- Point in time up to which all objects on the source are replicated.
- Serves as RPO indicator.
- Scan replication metadata map to determine the entry with lowest create time.

Design Challenges: Racing updates

Change Versioning:

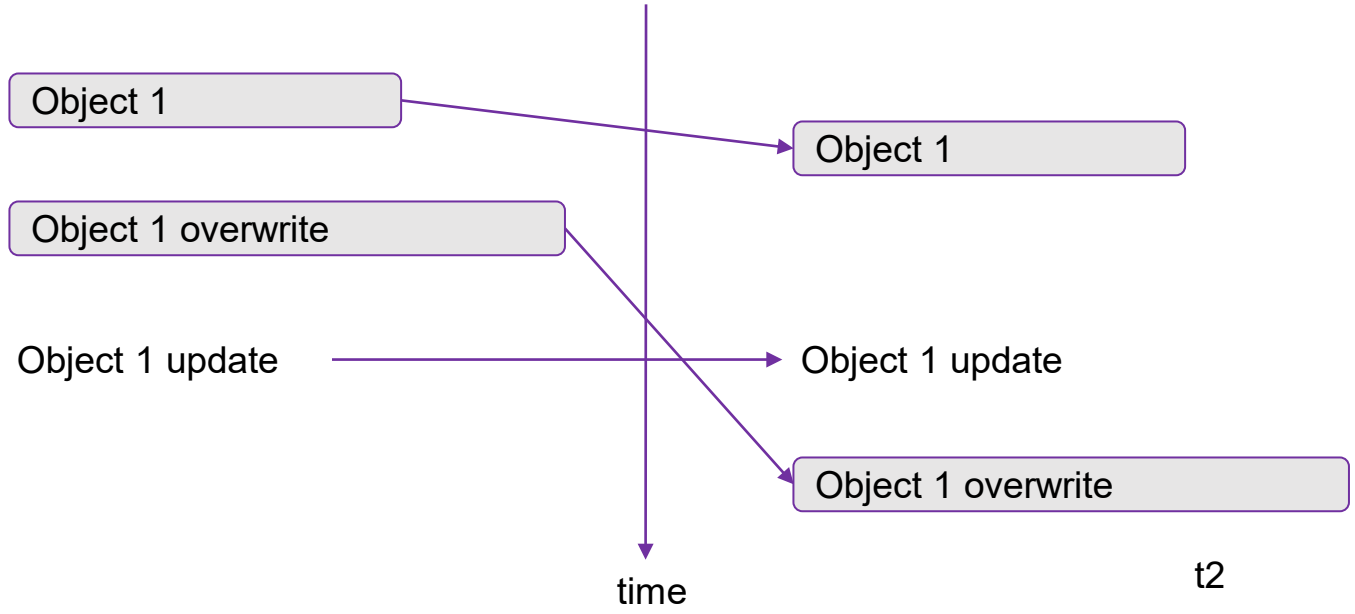
- Objects may get overwrites and updates while the replication is pending.
- Replication metadata needs versioning through a sequence number.
- Every update bumps the sequence number.
- Sequence number reflect the updates agnostic to the knowledge of update
- Multiple updates are absorbed through sequence number.

Replication Data Path

- It is possible that asynchronous foreground replication fails due to some errors. One common error can be the unavailability of the remote.
- We rely on background replication to achieve eventual consistency.
- A background process scans the replication map and issues the replication of the objects whose entry is present in the map.

Data transfer: One Op Outstanding

- Replications issued for different versions cause inconsistency.
- Prevent races by limiting replication to one request outstanding per object.
- Use version number in replication metadata to serialize the replication operations within an object.



Replication Data Path

- There can be a network failure at any time when replication is being done.
- For a large object, we do not want to restart the transfer again.
- So, we use multipart uploads for transferring large objects.
- This helps us to restart replication from the point the connection was lost.
- The part sizes are calculated dynamically and depend on the size of the original object.

The multipart upload protocol

The multipart upload protocol has 3 phases

- Multipart Upload Start
- Multipart Upload Part
- Multipart Upload Finalize

Multipart replication

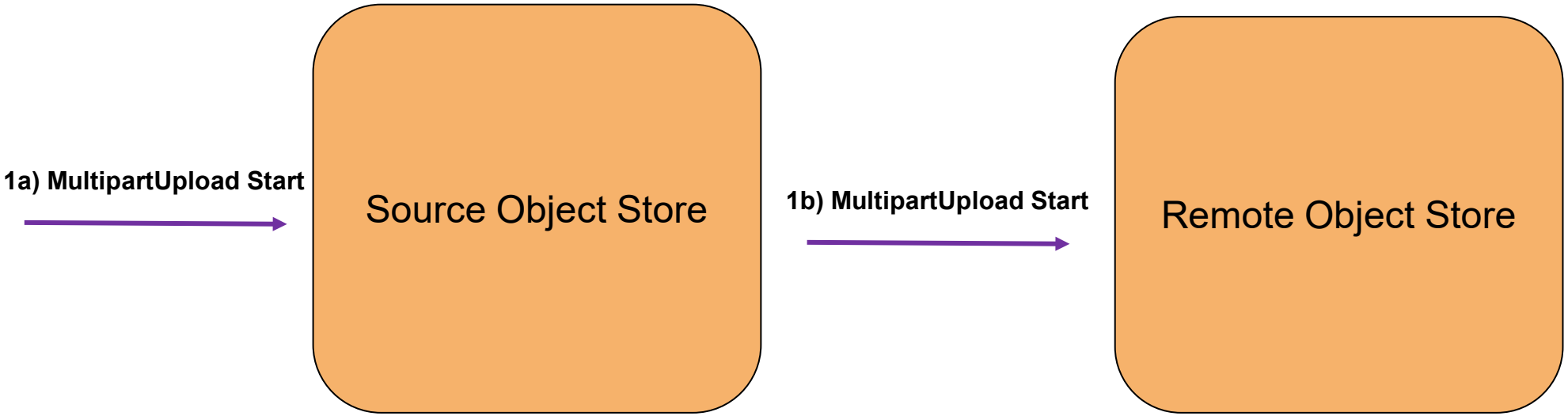
- Replication of multipart objects had high latency because replication started only when the parts were finalized.

Multipart replication Challenges

- Replication of multipart objects had high latency because replication started only when the parts were finalized.
- Consider replication for a VM backup where the backup size can go in terabytes.
- If we start replication after the entire object is uploaded on source, we compromise on the near sync behaviour.
- If the upload took X hours to complete, then replication would take another X hours (since it will start only when the object is finalized).

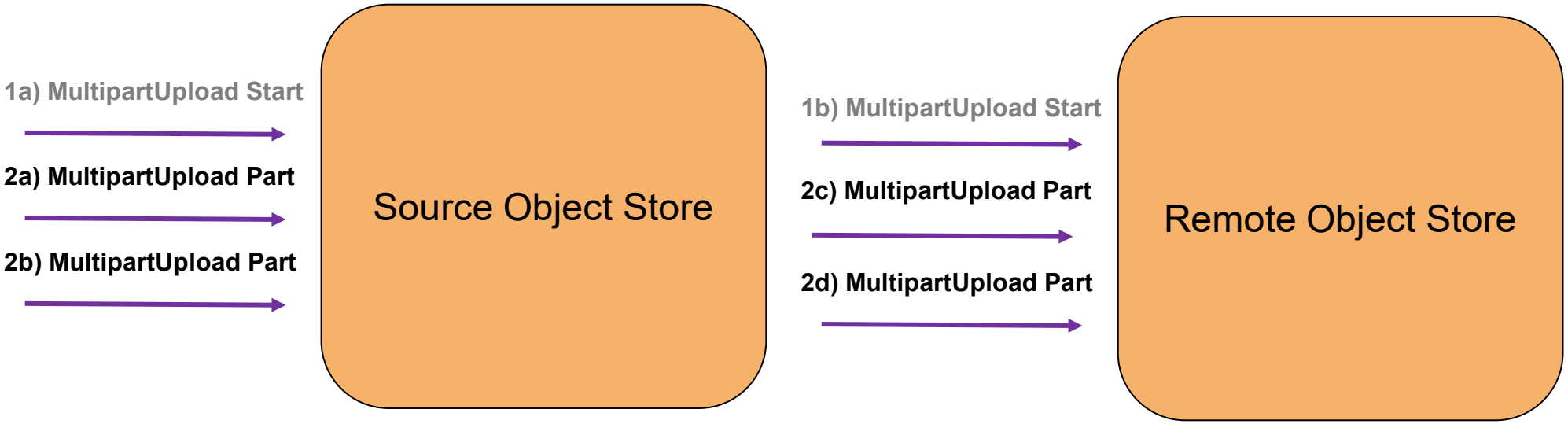
Multipart replication - Optimisation

- The protocol was optimized by streaming the replication of the multipart actions i.e. issue replication actions as soon as they are received on source.



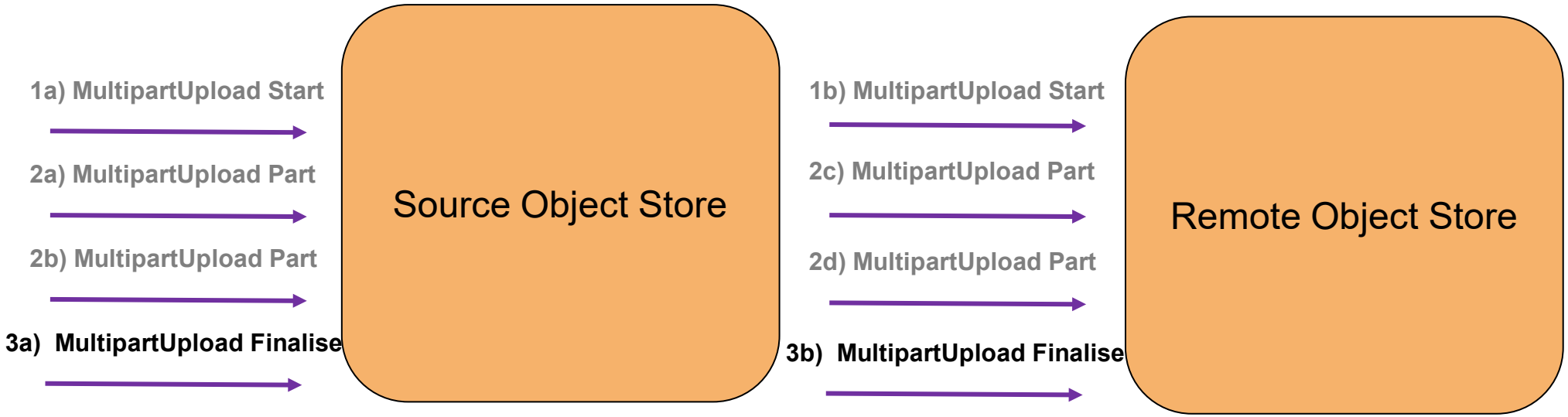
Multipart replication - Optimisation

- Once start is replicated, replicate the parts as soon as they arrive on the source.



Multipart replication - Optimisation

- Then issue finalise when it lands on the source.



Multipart replication optimisation - Challenges

- Out of order replications.
- Replication of parts in different order is fine.
- However, upload part cannot land on remote till upload start is completed.
- Similarly finalize cannot land on remote till all parts have been replicated successfully.

Performance Numbers

Small objects got replicated almost instantaneously.

Object Sizes	Time Taken Pre Multipart replication	Time Taken post Multipart replication	Improvement (x times)
5GiB	~ 5 minutes	Instantaneous	
1TiB	~ 24.5 hours	~ 5 hours	5x
5TiB	~ 154 hours (~6.4 Days)	~ 17 hours (~ 0.7 Days)	9x

Please take a moment to rate this session.

Your feedback is important to us.

