Western Digital.

HDFS Tiered Storage: Mounting Object Stores in HDFS

Ewan Higgs
Thomas Demoor

Thomas Demoor

- Product Owner Object Storage Access
  - S3-compatible features (versioning, lifecycle management, async. replication)
  - Hadoop integration
- Apache Hadoop contributions:
  - S3a filesystem improvements (Hadoop 2.6–2.8+)
  - Object store optimizations: rename-free committer (HADOOP-13786)
  - HDFS Tiering: Provided Storage (HDFS-9806)
- Ex: Queueing Theory PhD @ Ghent Uni, Belgium
- Tweets @thodemoor

Ewan Higgs

- Software Architect
  - Hadoop integration
- Apache Hadoop contributions:
  - HDFS Tiering: Provided Storage
- Ex:
  - Embedded systems
  - Managing market data in finance
  - HPC Systems administrator

HDFS + Object Store

- External Tier for Hadoop: Object Store or HDFS or …
- Your trusted HDFS workflow
  - Hadoop Apps interact with HDFS
  - Data Tiering Async in background
  - Admin controls data placement through usual CLI
- HDFS Storage Policy
  - DISK, SSD, RAM, ARCHIVE, **PROVIDED**
- Read Path (HDFS-9806) merged in December 
  - Already being used in production at Microsoft.

Demo

- **AS AN** Administrator,

- **I CAN** configure HDFS with an S3a backend with an appropriate Storage Policy
  - hdfs storagepolicies -setStoragePolicy -policy PROVIDED -path /var/log
  - hdfs syncservice -create -backupOnly -name activescale /var/logs s3a://hadoop-logs/

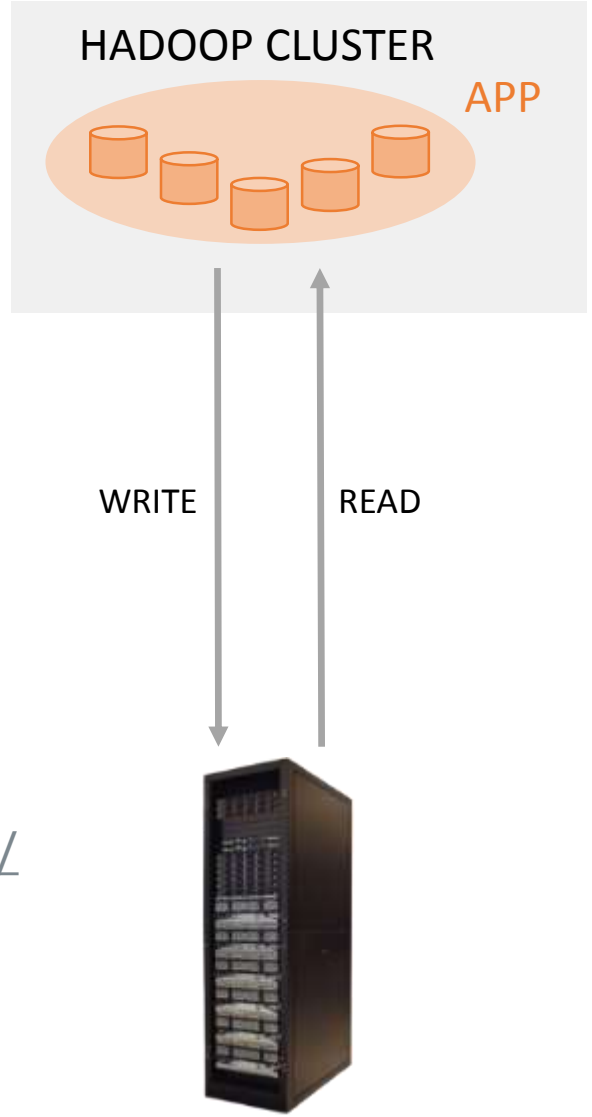
- **SO THAT** when a user copies files to HDFS they are asynchronously copied to to the synchronization endpoint

WIP – Mock Demo

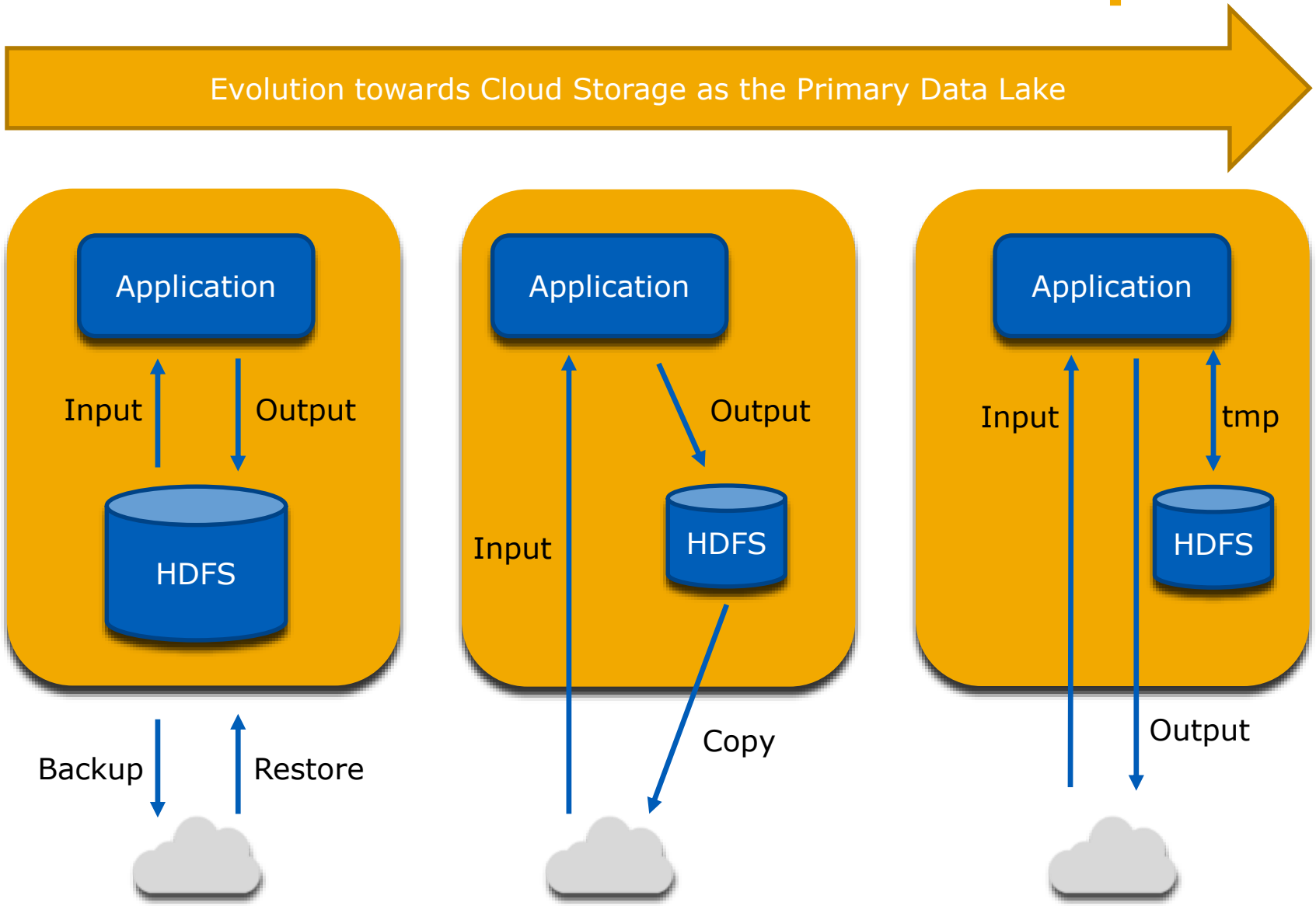
- **AS AN** Administrator
- **I CAN** set the Storage Policy to be PROVIDED_ONLY
  - hdfs –setStoragePolicy -policy PROVIDED_ONLY -path /var/log
- **SO THAT** data is no longer in the Datanode but is transparently read through from the synchronization endpoint on access.

S3a: quick recap

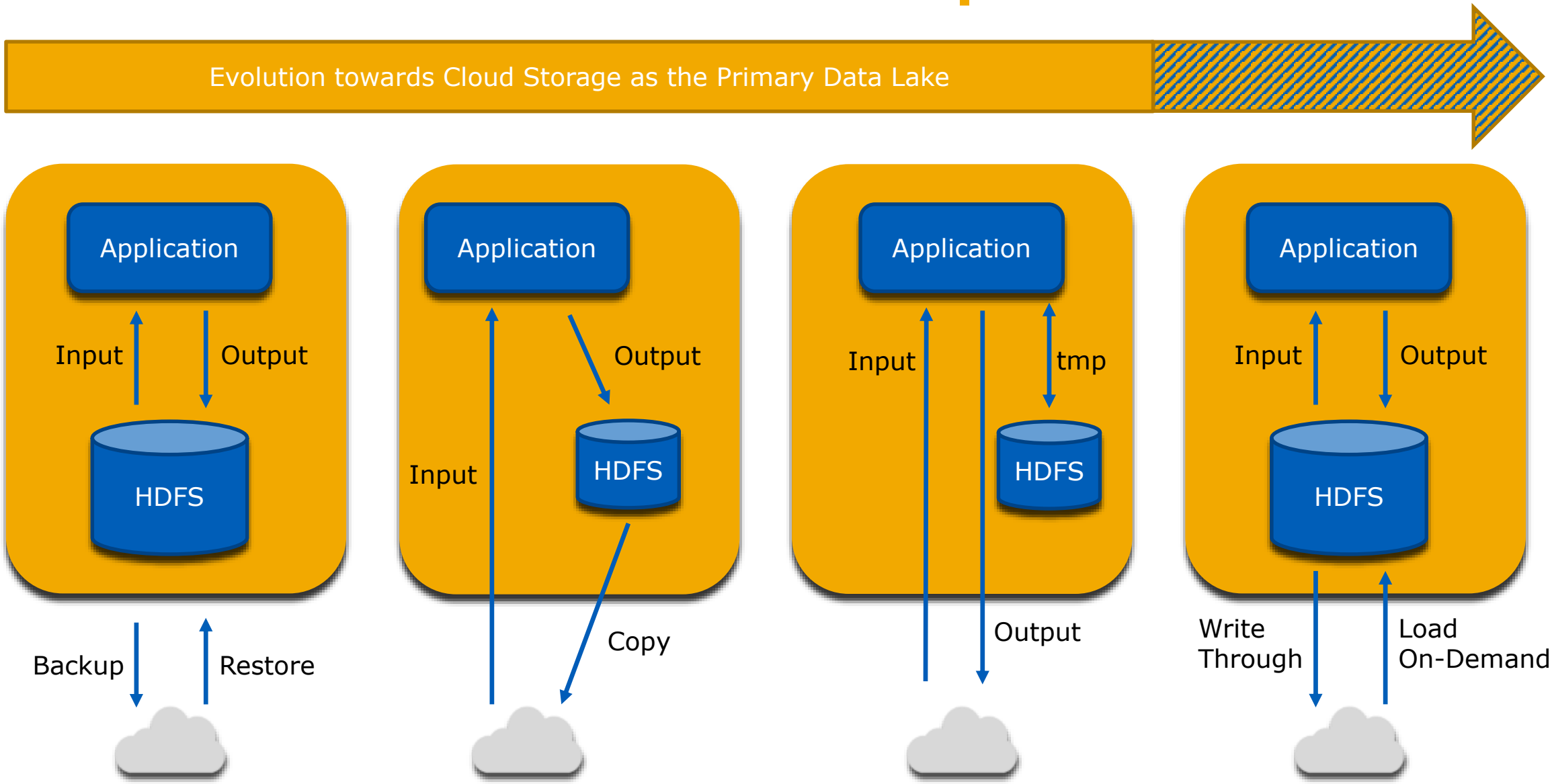
- S3a = Hadoop Compatible FileSystem
  - (cfr. wasb://, adl://, gcs://, ceph://, …)
  - Direct IO between Hadoop apps and object store: s3a://bucket/object
  - Scalable & Resilient: NameNode functions taken up by object store
  - Functional since Hadoop 2.7, improvements in 2.8 & ongoing (2x write)
- Our contributions:
  - Non-AWS endpoints
  - YARN/Hadoop2
  - Streaming upload
  - …
- S3a in action:
  - Hortonworks Data Cloud: https://hortonworks.com/products/cloud/aws/
  - Databricks Cloud: https://databricks.com/product/databricks
  - Backup entire HDFS clusters
  - Archive cold data from HDFS clusters



Hortonworks evolution of Hadoop+Cloud



Hortonworks evolution of Hadoop+Cloud: Next

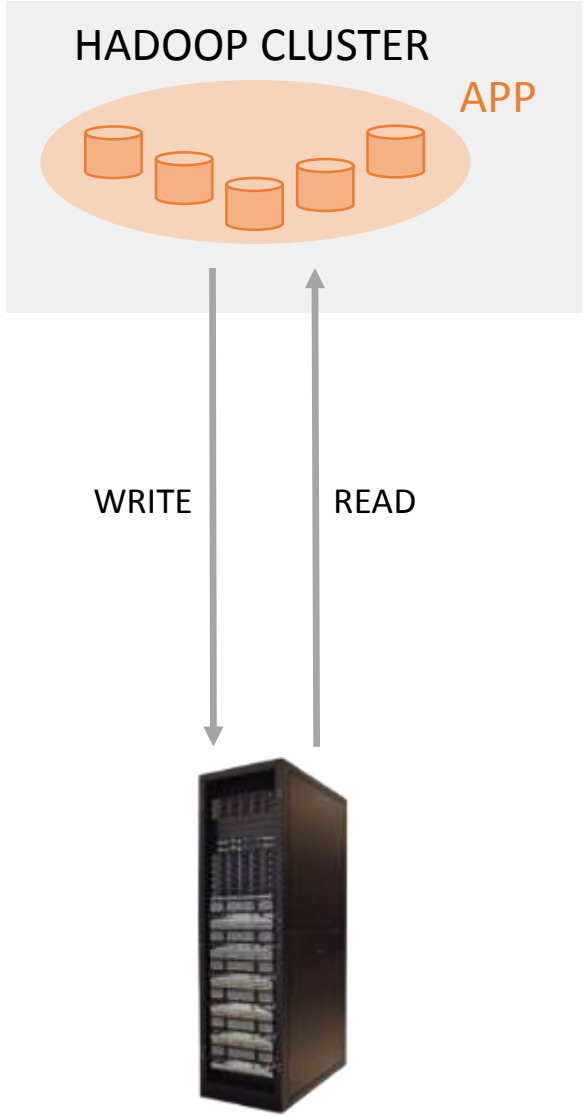


WD Active Archive Object Storage

- Western Digital moving up the stack
- Scale-out object storage system for Private & Public Cloud
- Key features:
  - Compatible with Amazon S3 API
  - Strong consistency (not eventual!)
  - Erasure coding for efficient storage
- Scale:
  - Petabytes per rack
  - Billions of objects per rack
  - Linear scalability in # of racks
- More info at http://www.hgst.com/products/systems

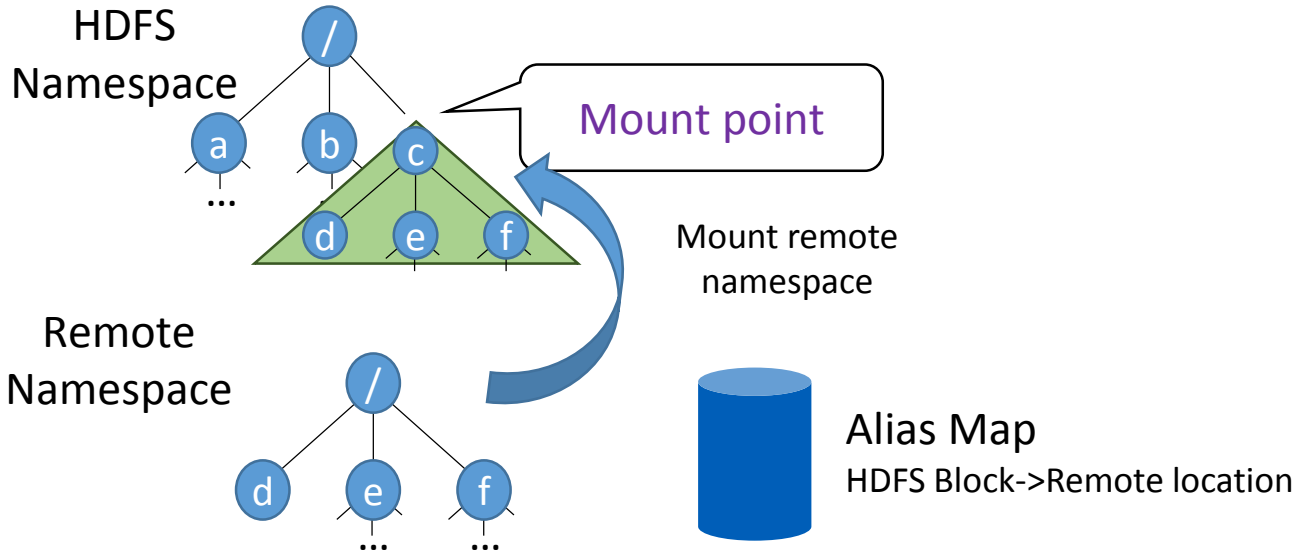
S3a doesn’t fit all use cases

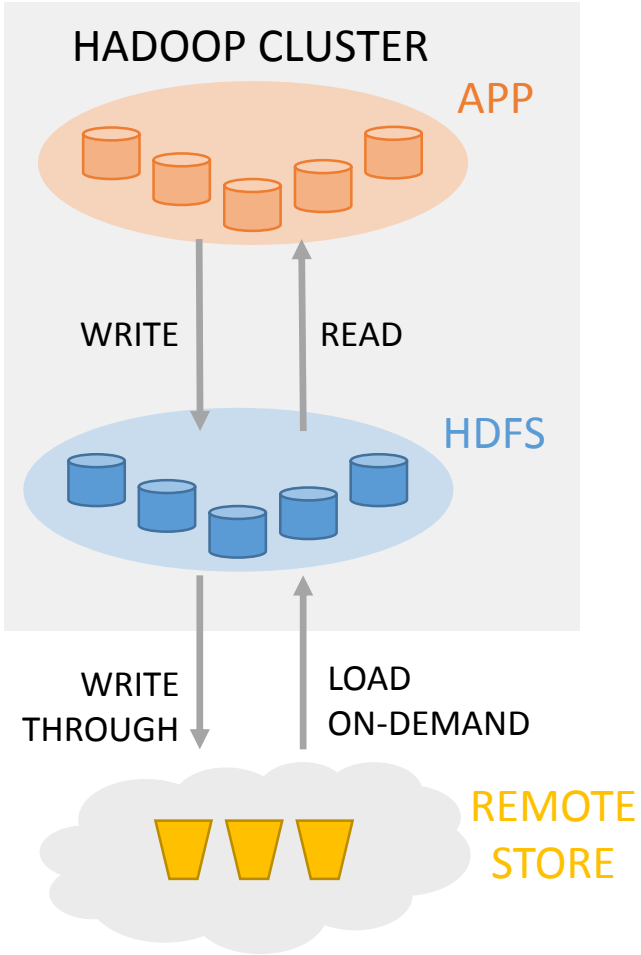
- Hadoop-compatible does not mean identical to HDFS
  - S3a is not a FileSystem
    - no notion of directories
    - no support for flush/ append (breaks Apache HBase: KV-store)
  - No Data Locality:
    - less performant for hot/real-time data
  - Hadoop admin tools require HDFS:
    - permissions/quota/security/…
- Goal: integrate better with HDFS
  - Data-locality for hot data + object storage for cold data
  - Offer familiar HDFS abstractions, admin tools, …



Solution: "Mount" remote storage in HDFS

- Use HDFS to manage remote storage
  - HDFS coordinates reads/writes to remote store
  - Mount remote store as a PROVIDED tier in HDFS
    - Details later in the talk
  - Set StoragePolicy to move data between the tiers



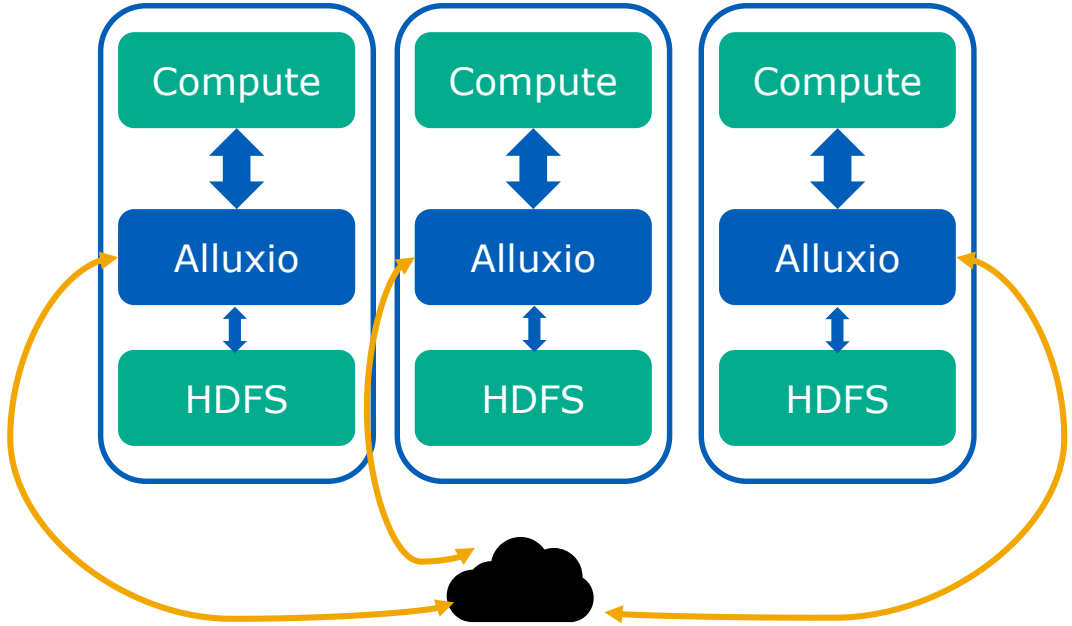


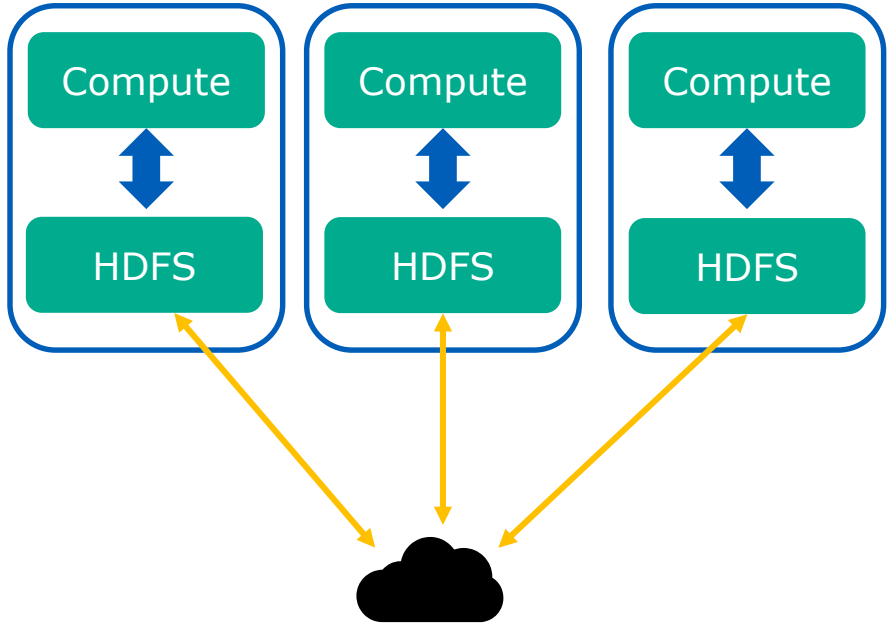
Best of Both Worlds

- HDFS + S3A/WASB/…
- HDFS for streaming append workloads and data locality
- Object Storage for dense resilient storage of arbitrarily large scale
- Can also do HDFS on HDFS

Caching

- This design means HDFS could be used as a cache for Object Storage
- Similar to Alluxio, Ignite, Rubix?
  - These are all caching systems that dispatch between storage systems horizontally
  - We want to tier the storage systems vertically
  - We want HDFS to remain source of truth for Hadoop





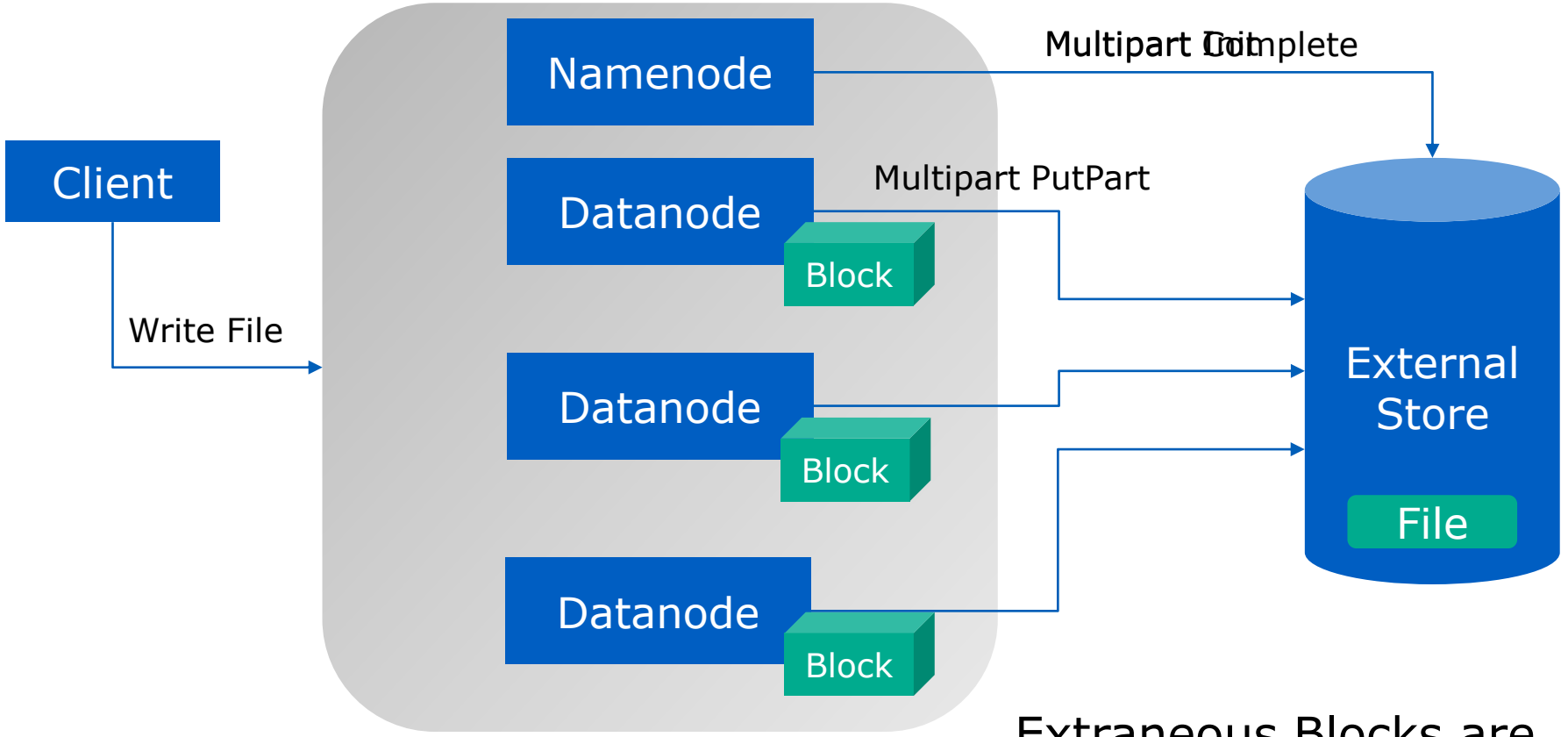
Requirements

- Preserve file-object mapping
  - AliasMap (last year's talk): used to synchronize namespaces
  - Multiple Datanodes collaborate to move blocks which together form a file/object in destination system
- Minimize impact on frontend traffic / Efficient data transfer
  - Obvious: Read all blocks into a single Datanode to reconstruct a file before transferring
  - Efficient: Transfer directly copies block per block outside of cluster using
    - S3: multipart upload
    - WASB: append blobs
    - HDFS: tmpdir + concat
- Deployment model
  - In Namenode is easy to deploy but adds resource pressure
  - External service is more difficult to deploy for some sites but reduces resource pressure
    - See AliasMap from HDFS-9806 and SPS service from HDFS-10825
    - Would like to re-use SPS protocols

Tech Details

Tech Details

- Applications write to HDFS
  - First to DISK, then SyncService asynchronously copies to synchronization endpoint
  - When files have been copied, the extraneous disk replicas can be removed

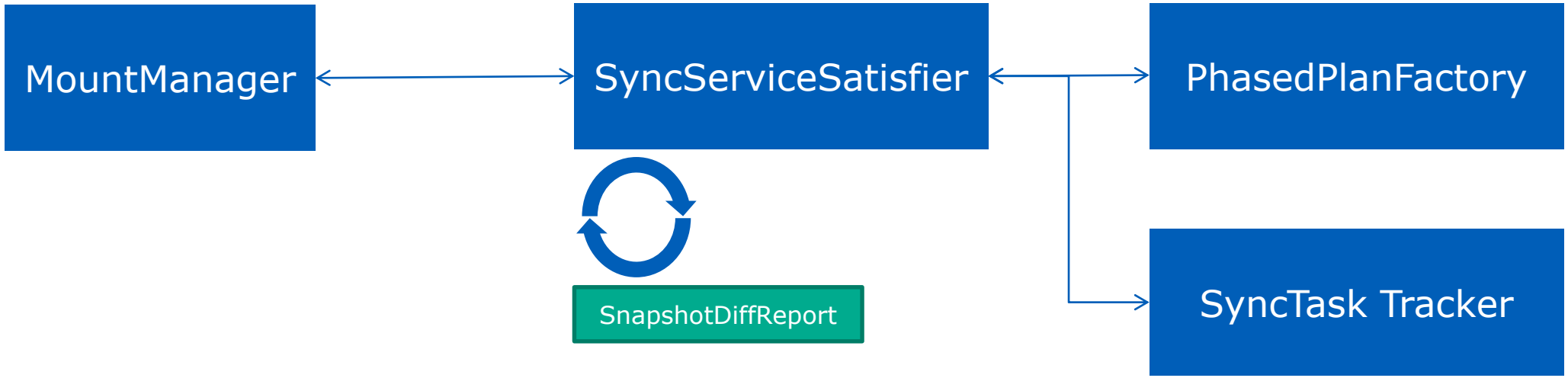


Extraneous Blocks are removed.

Satisfy Synchronization: How it works

- MountManager
  - Manages all the local mount points
- SyncServiceSatisfier
  - Service in Namenode that periodically creates a diff by comparing Snapshots
- PhasedPlanFactory
  - Generates the plan for ordering the operations in the diff
    - E.g. Create, Modify, Rename, Delete of Files and Directories
- SyncTask Tracker
  - Runs the work and tracks it
    - Namespace operations (MetadataSyncTask) run from Namenode
    - Data operations (BlockSyncTask) run in Datanode (❤-beat mechanism)

Tech Details – Namenode Components



Example Diff – Simple Case

Simple Case - New dirs; new files

Commands

```
#given /basic-test
mkdir -p /basic-test/a/b/c


touch /basic-test/a/b/c/d/f1.bin


touch /basic-test/f1.bin
```

SnapshotDiffReport

```
M d .
+ d ./a
+ f ./f1.bin
```

Example Diff – Harder Case

Harder Case - Cycle

Commands

```
#given /swap-test/a.bin
#given /swap-test/b.bin
mv /swap-test/a.bin /swap-test/tmp

mv /swap-test/b.bin /swap-test/a.bin

mv /swap-test/tmp /swap-test/b.bin
```

SnapshotDiffReport

```
M d .
R f ./a.bin -> ./b.bin
R f ./b.bin -> ./a.bin
```

Use solution in DistCpSync for PhasedPlan

- **DistCpSync** Algorithm:
  1. Rewrite Target Destinations based on other Renames
  2. Renames and Deletes to temp directory
  3. Renames to final location (mkdirs if directory doesn’t exist yet)
  4. Drop temp directory (and the deletes with them)
  5. Creates and Modifies

- **PhasedPlan** Algorithm:
  1. Rewrite Target Destinations based on other Renames
  2. Renames and temp directory
  3. Delete Files and Directories
  4. Renames to final location (mkdirs if directory doesn’t exist yet)
  5. Drop temp directory
  6. Create Directories on end point
  7. Create Files on end point

Example Diff – Harder Case

Harder Case - Cycle

Commands

```
#given /swap-test/a.bin
#given /swap-test/b.bin
mv /swap-test/a.bin /swap-test/tmp

mv /swap-test/b.bin /swap-test/a.bin

mv /swap-test/tmp /swap-test/b.bin
```

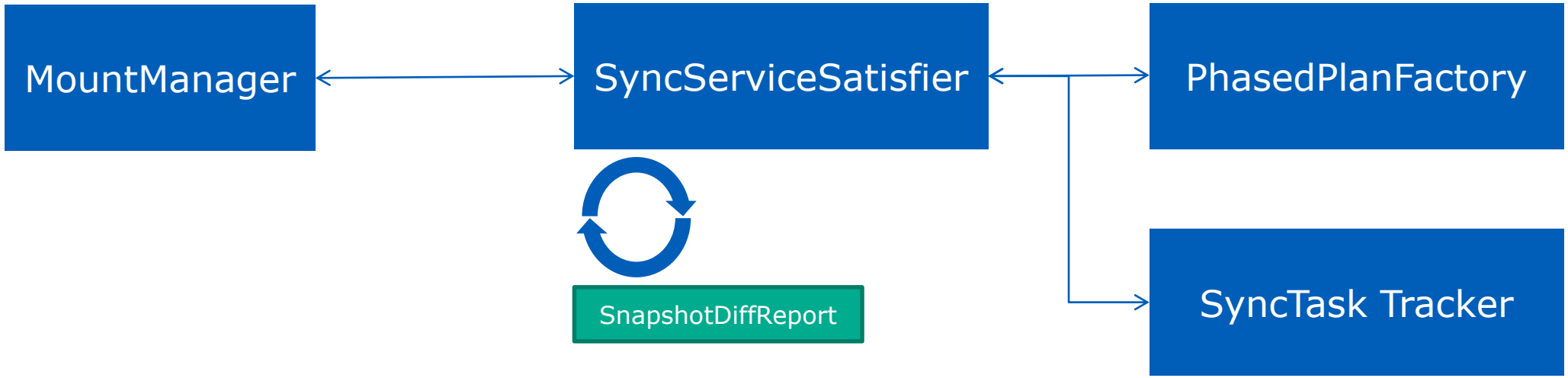
SnapshotDiffReport

```
M d .
R f ./a.bin -> ./b.bin
R f ./b.bin -> ./a.bin
```

PhasedPlan

```
M d .
R f ./a.bin -> ./tmp/b.bin
R f ./b.bin -> ./tmp/a.bin
R f ./tmp/b.bin -> b.bin
R f ./tmp/a.bin -> a.bin
```

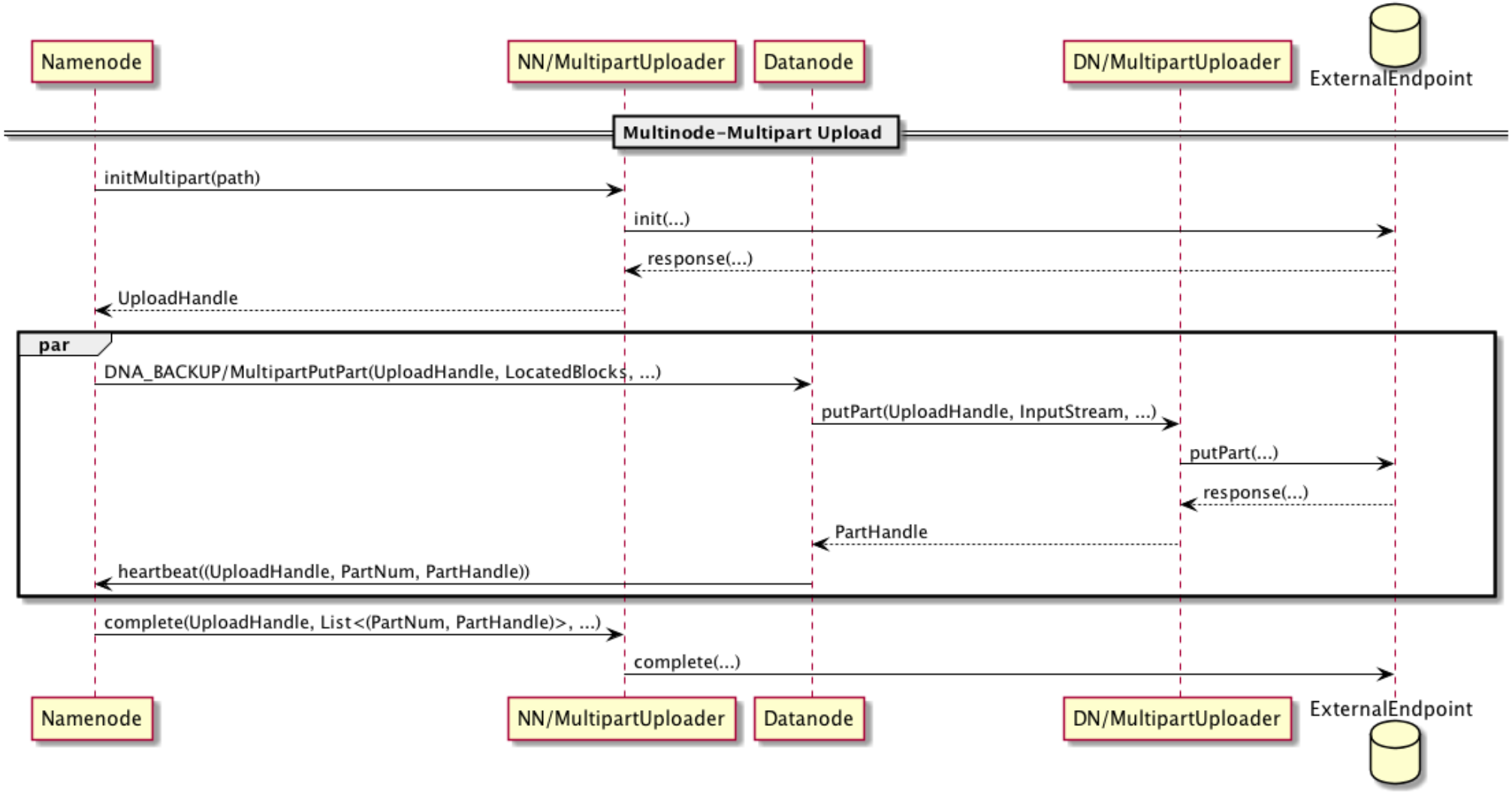
Tech Details – Namenode Components



MultipartUploader

- Common concept in Object Storage
  - Supported by S3, WASB
- Usage in Hadoop
  - S3A uses it – see Steve Loughran's talk
  - New to HDFS – HDFS-13186
- Three phases
  - `UploadHandle initMultipart(Path filePath)`
  - `PartHandle putPart(Path filePath, InputStream inputStream, int partNumber, UploadHandle uploadId, long lengthInBytes)`
  - `void complete(Path filePath, List<Pair<Integer, PartHandle>> handles, UploadHandle multipartUploadId)`
- Benefits:
  - Object/File Isolation – you only see the results when it's done
  - Can be written in parallel across multiple nodes

MultipartUploader in SyncService



Namenode work reduction

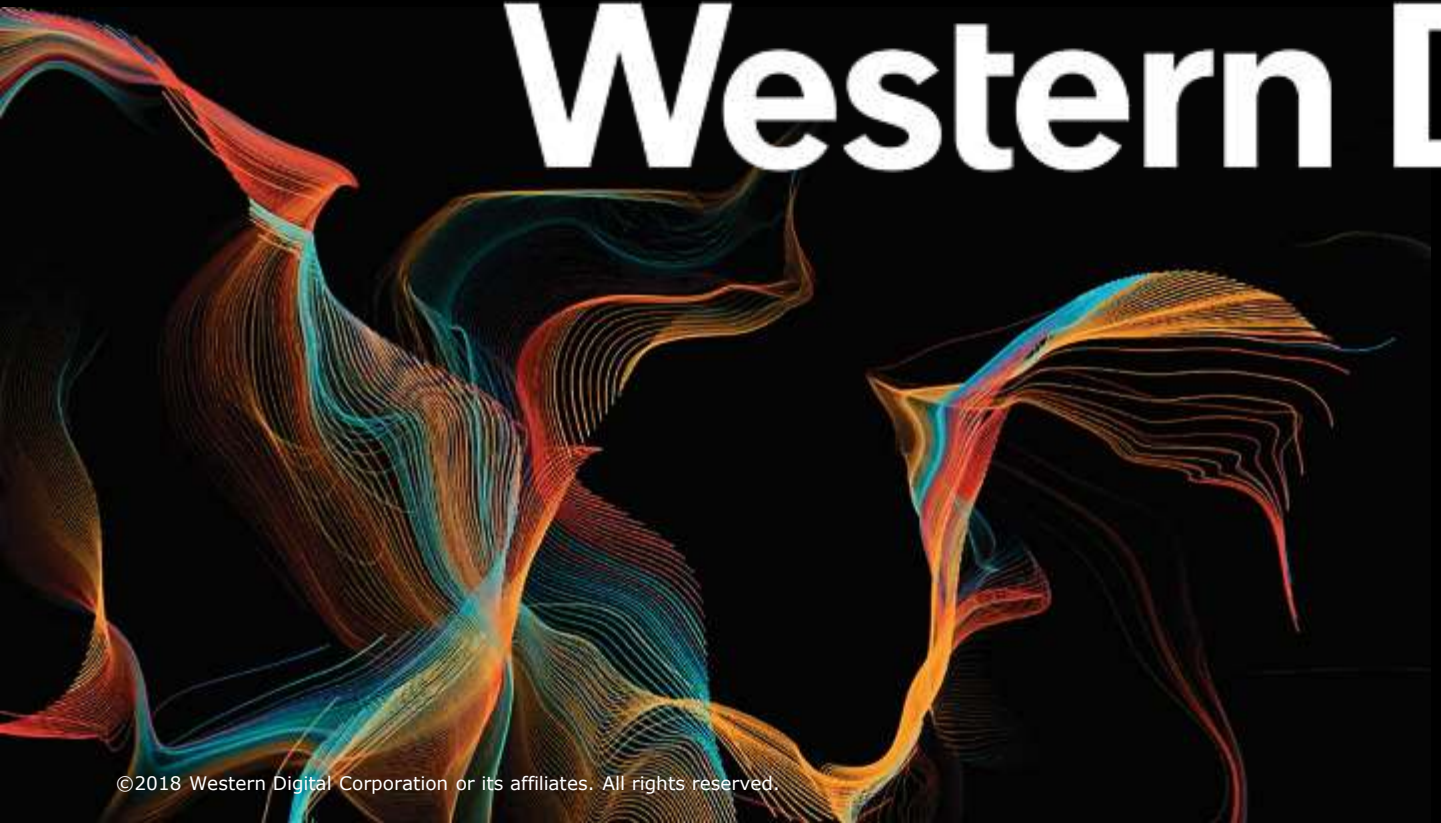
- All this work requires a synchronization point
- Namenode is the obvious place
- Namenode is too busy in large systems
- Optionally push SyncService to external service
  - Precedents: AliasMap, StoragePolicySatisfier

Resources

- Tiered Storage **HDFS-12090** [issues.apache.org]
  - Design documentation
  - List of subtasks, lots of linked tickets – take one!
  - Discussion of scope, implementation, and feedback
- Joint work Microsoft – Western Digital
  - {thomas.demoor, ewan.higgs}@wdc.om
  - {cdoug,vijala}@microsoft.com

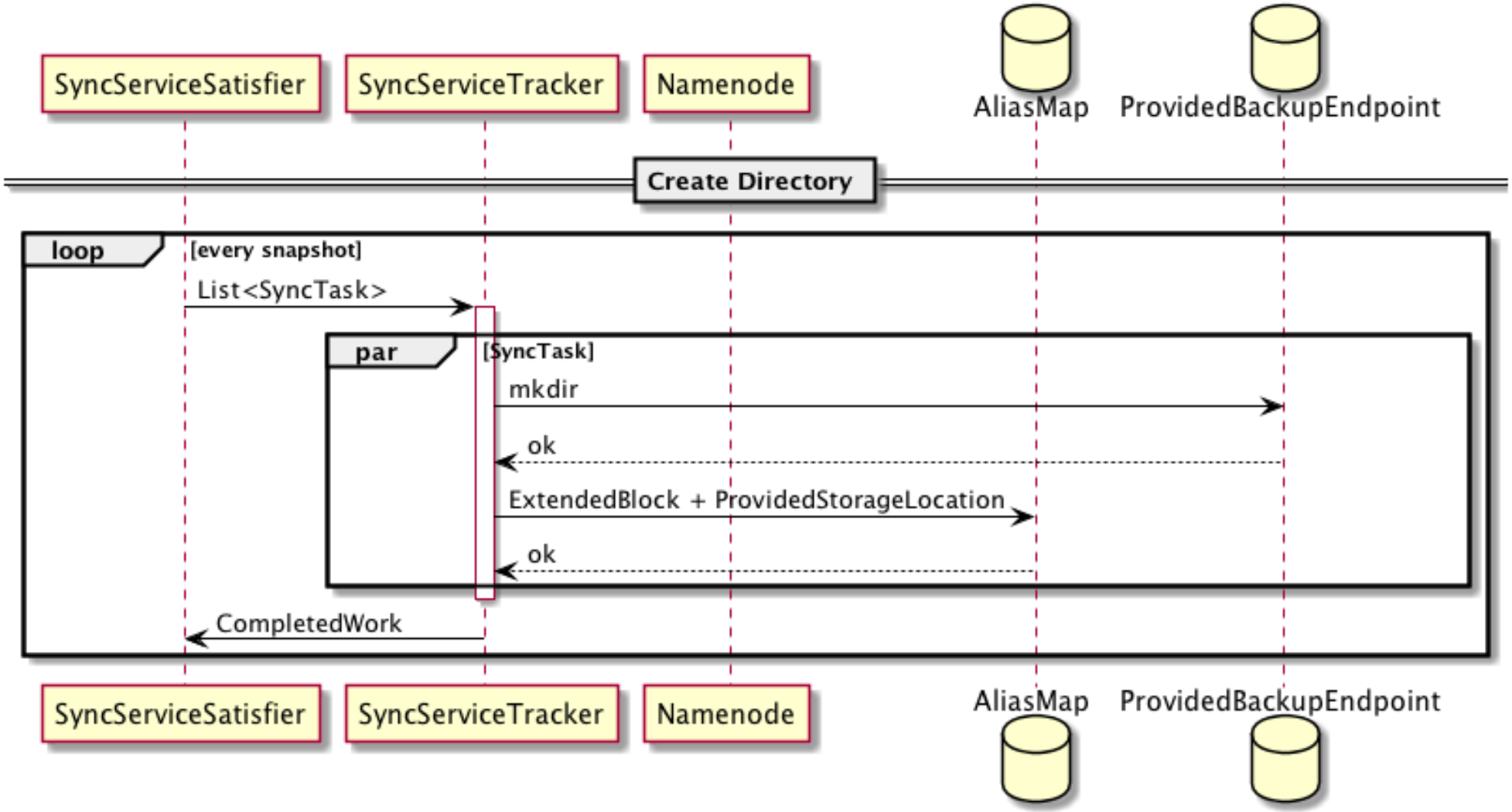
Thanks!

Virajith Jalaparthi, Chris Douglas, Ewan Higgs, Kasper Janssens, Bert Verslyppe, Hendrik Depauw, Rakesh R, Daryn Sharp, Steve Loughran, Thomas Demoor, Allen Wittenauer, and more!

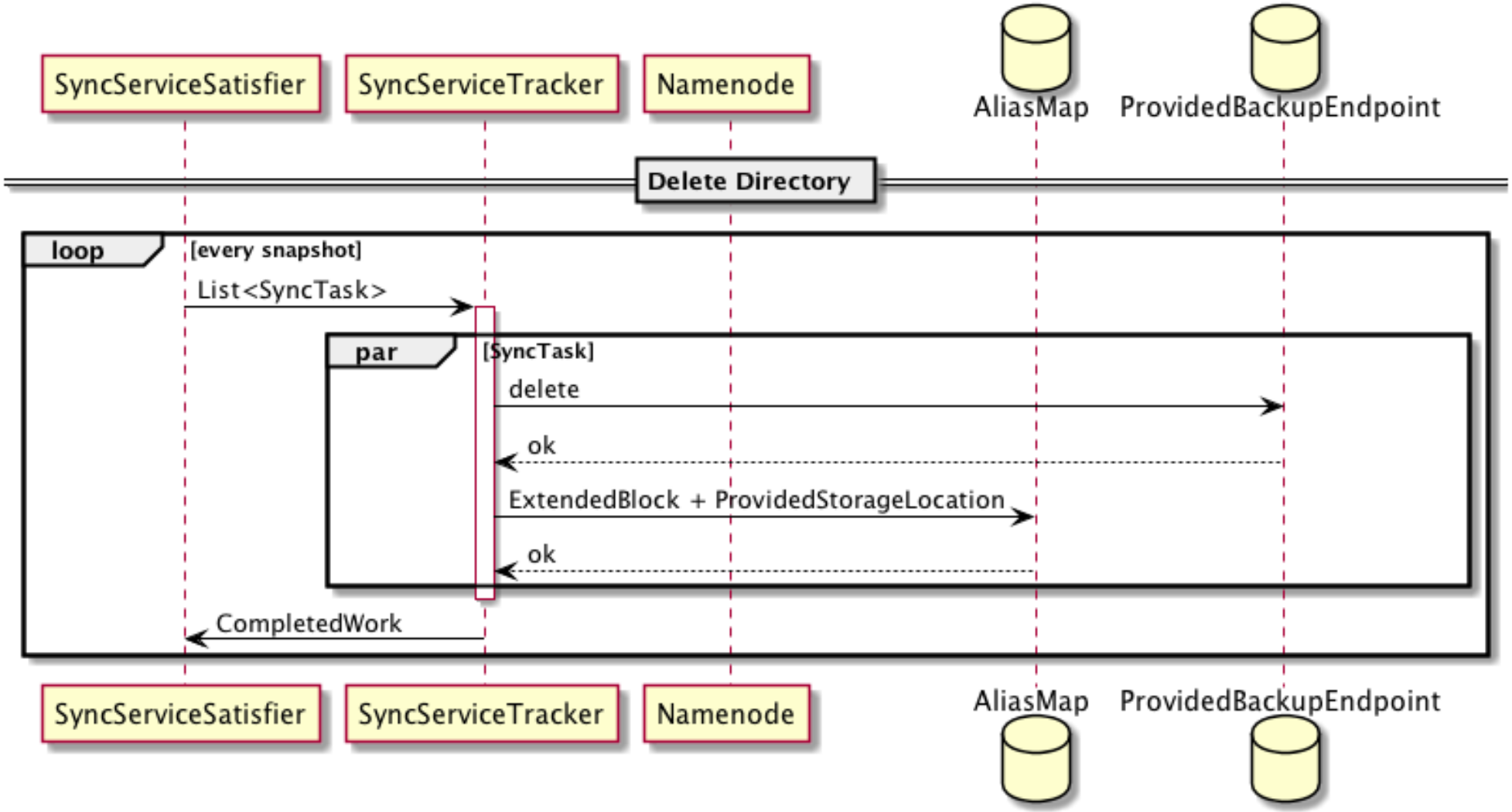


Extra Slides

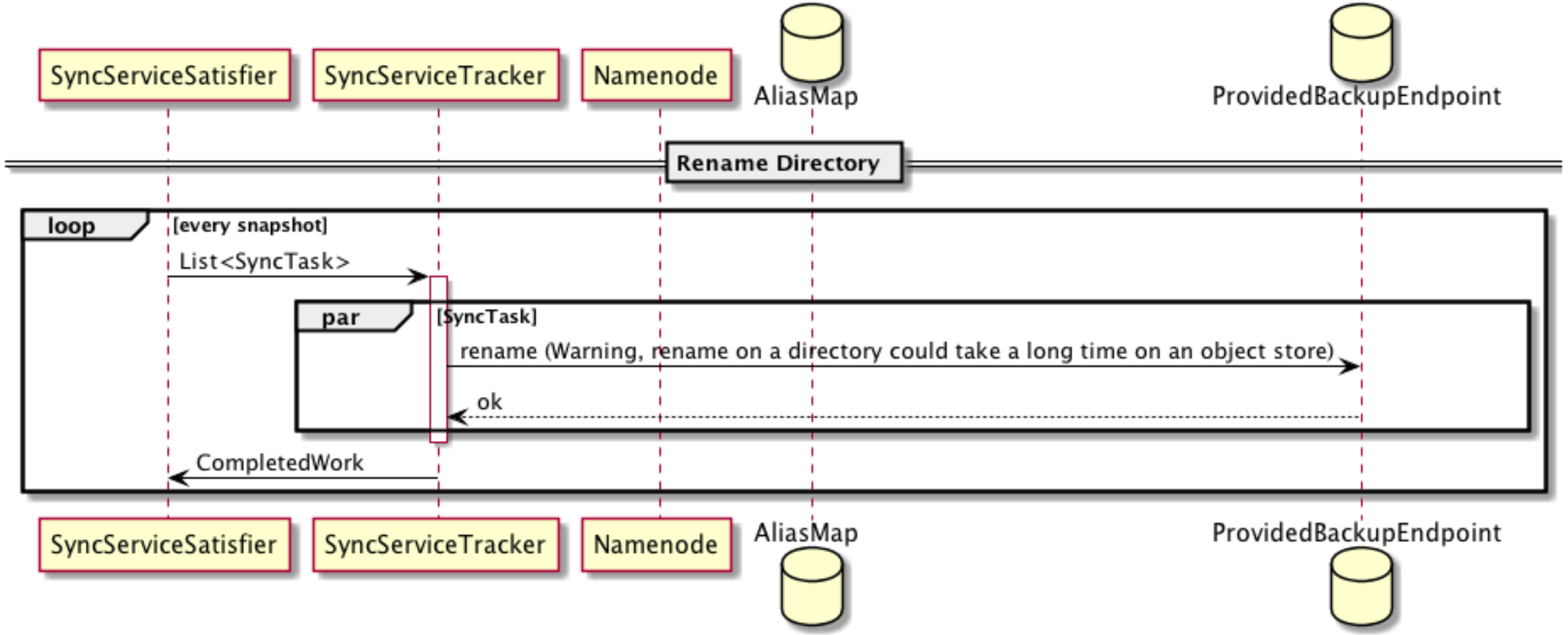
Create Directory



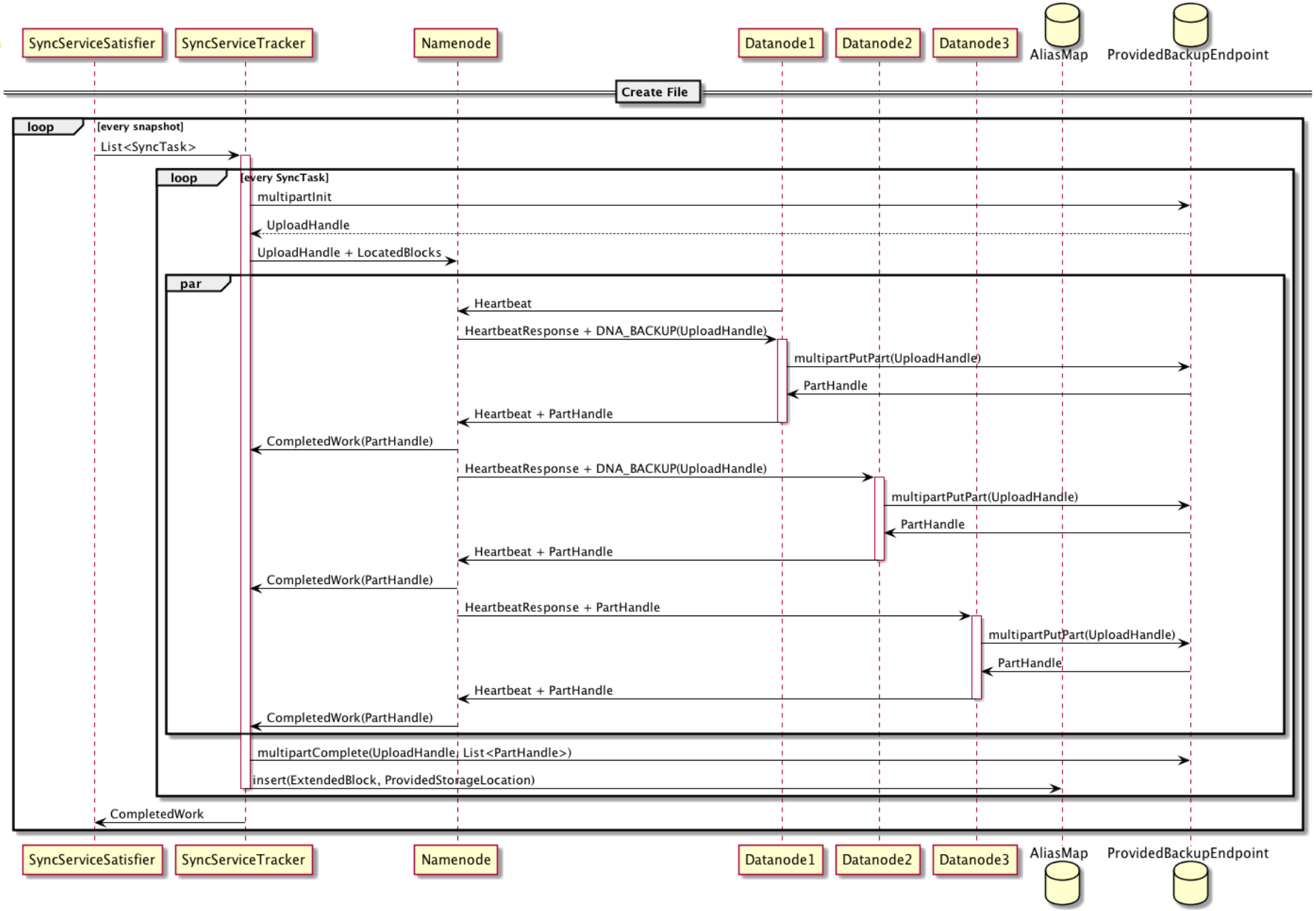
Delete Directory



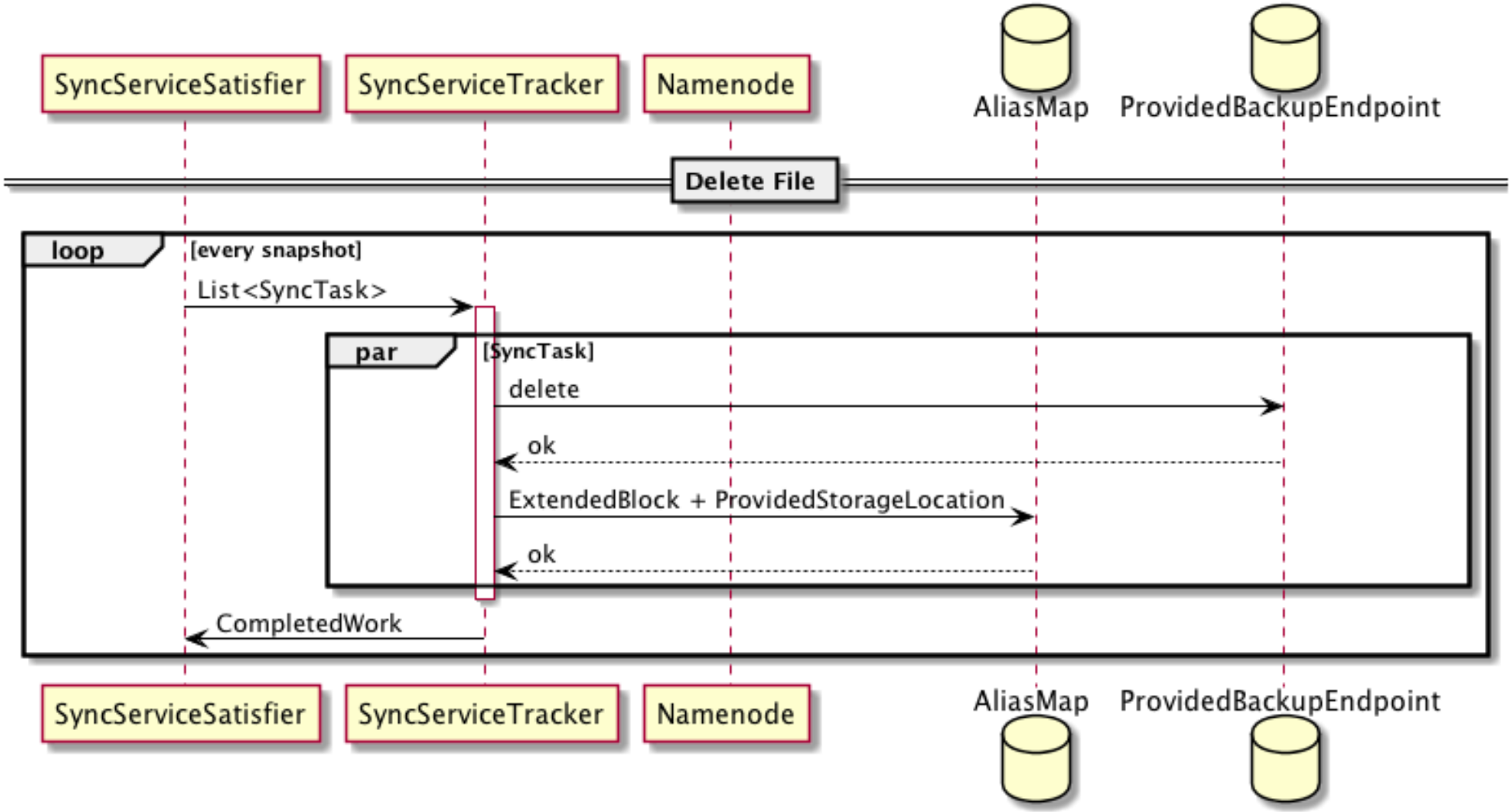
Rename Directory



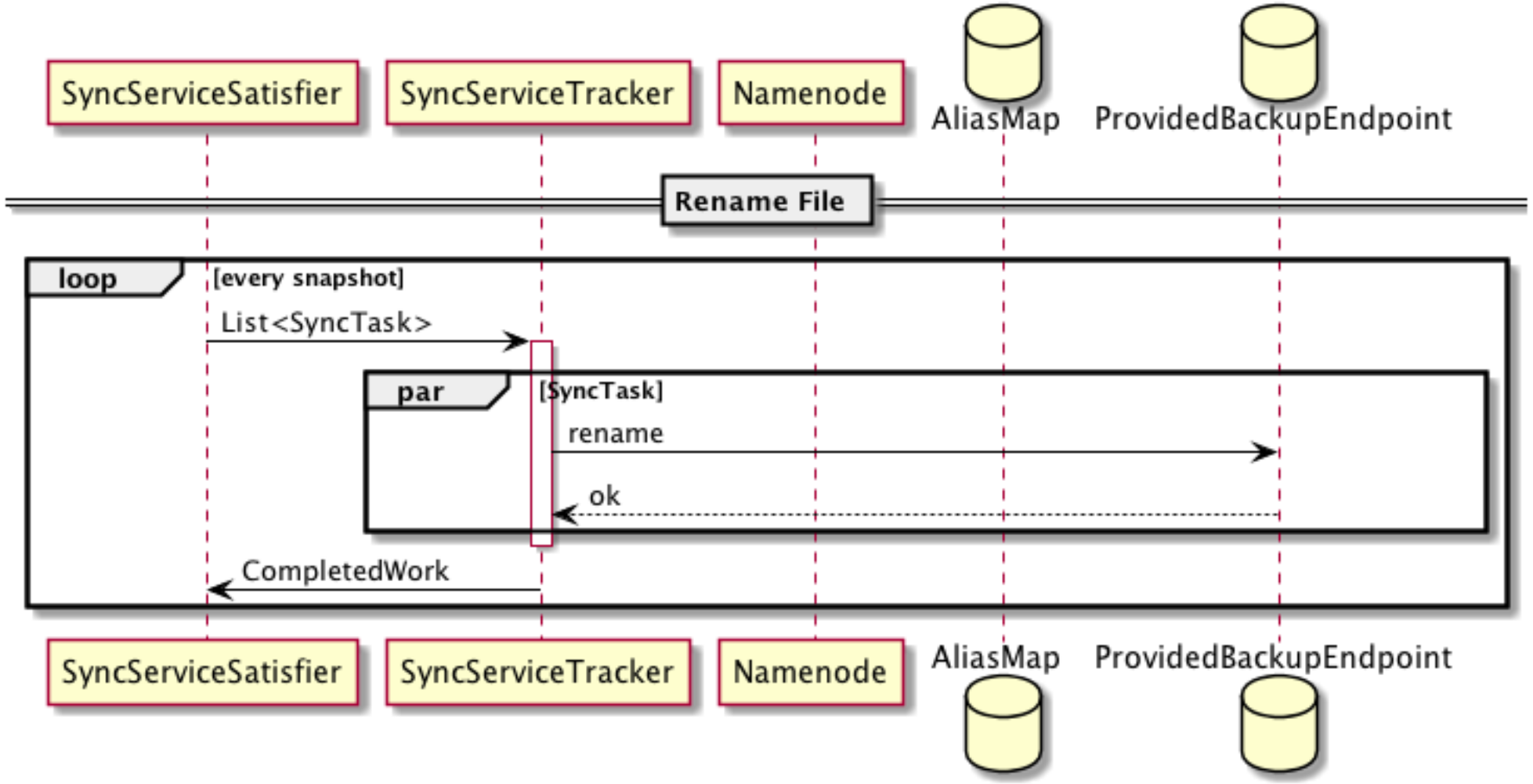
Create File



Delete File



Rename File



Modify File