hosted by Alibaba Group 阿里巴巴集团 APACHE HBASE

Improving HBase reliability at Pinterest with Geo-replication and Efficient Backup

Chenji Pan
Lianghong Xu
Storage & Caching, Pinterest

August 17, 2018

Content

01 HBase in Pinterest

HBase in Pinterest

- Use HBase for online service since 2013
- Back data abstraction layer like Zen, UMS
- ~50 Hbase 1.2 clusters
- Internal repo with ZSTD, CCSMAP, Bucket cache, timestamp, etc

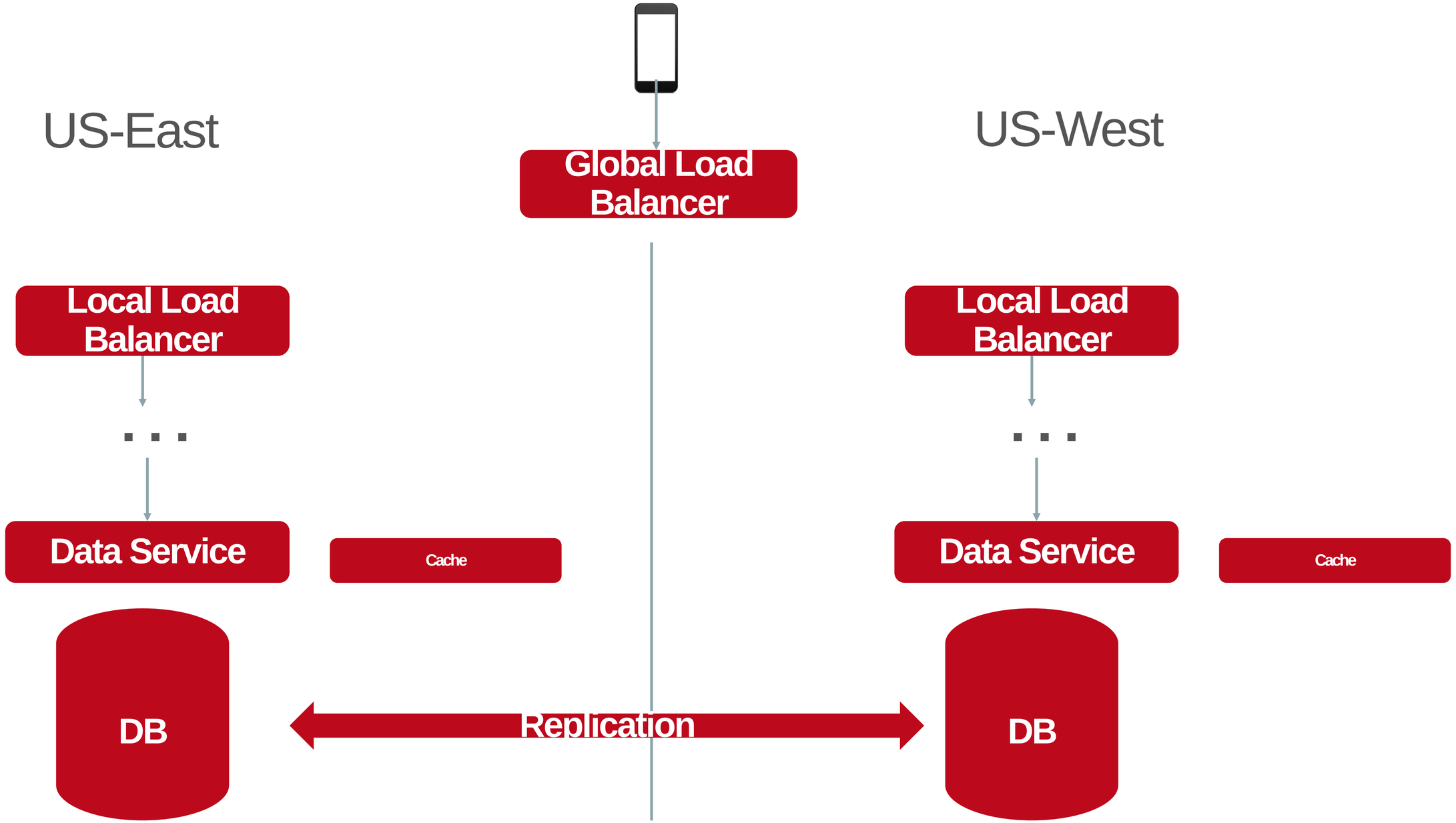
02 Multicell

Why Multicell?

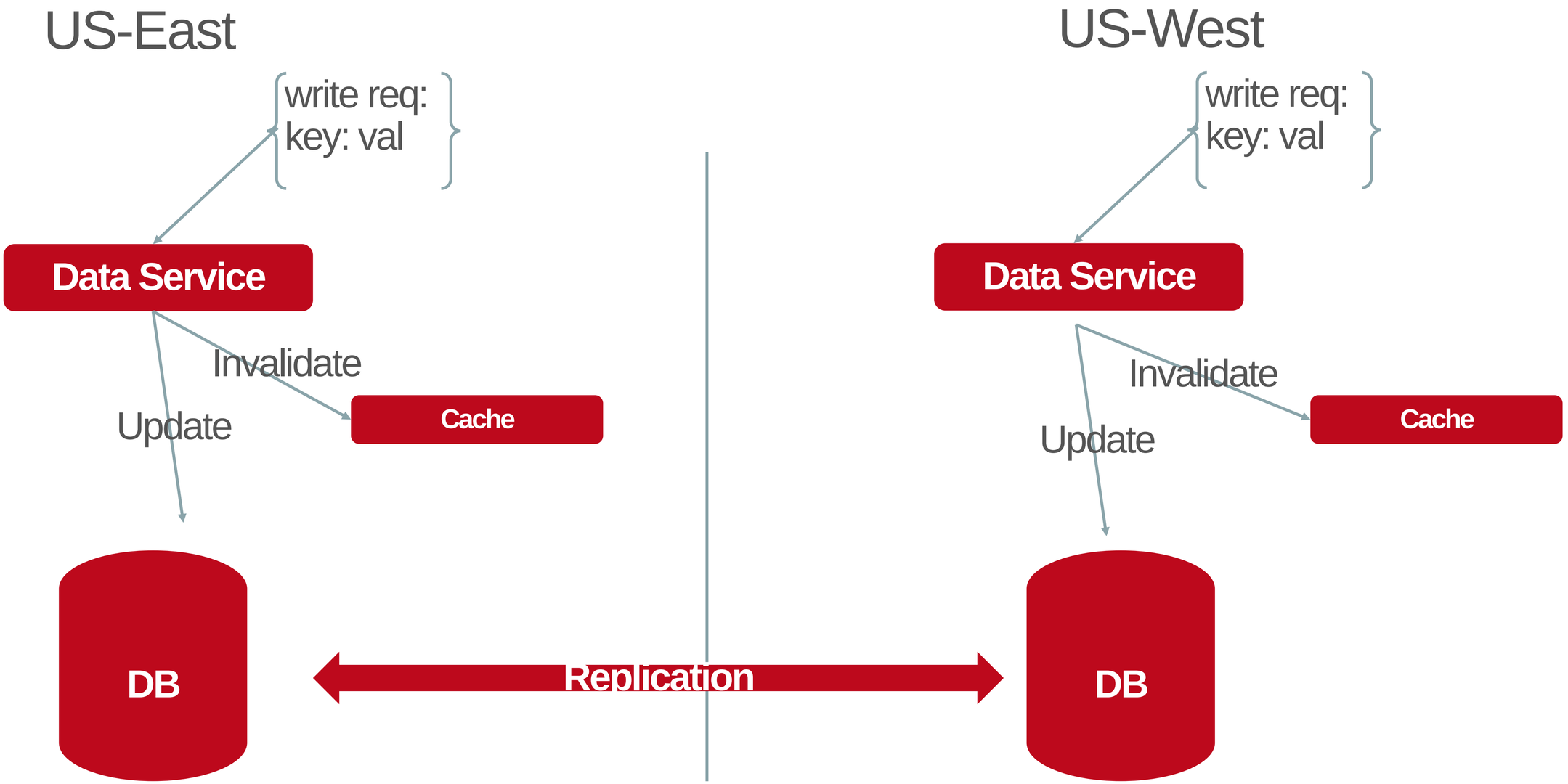
- Internationalization
- Reliability
- Latency

2011 2012 2013 2014 2015 2016

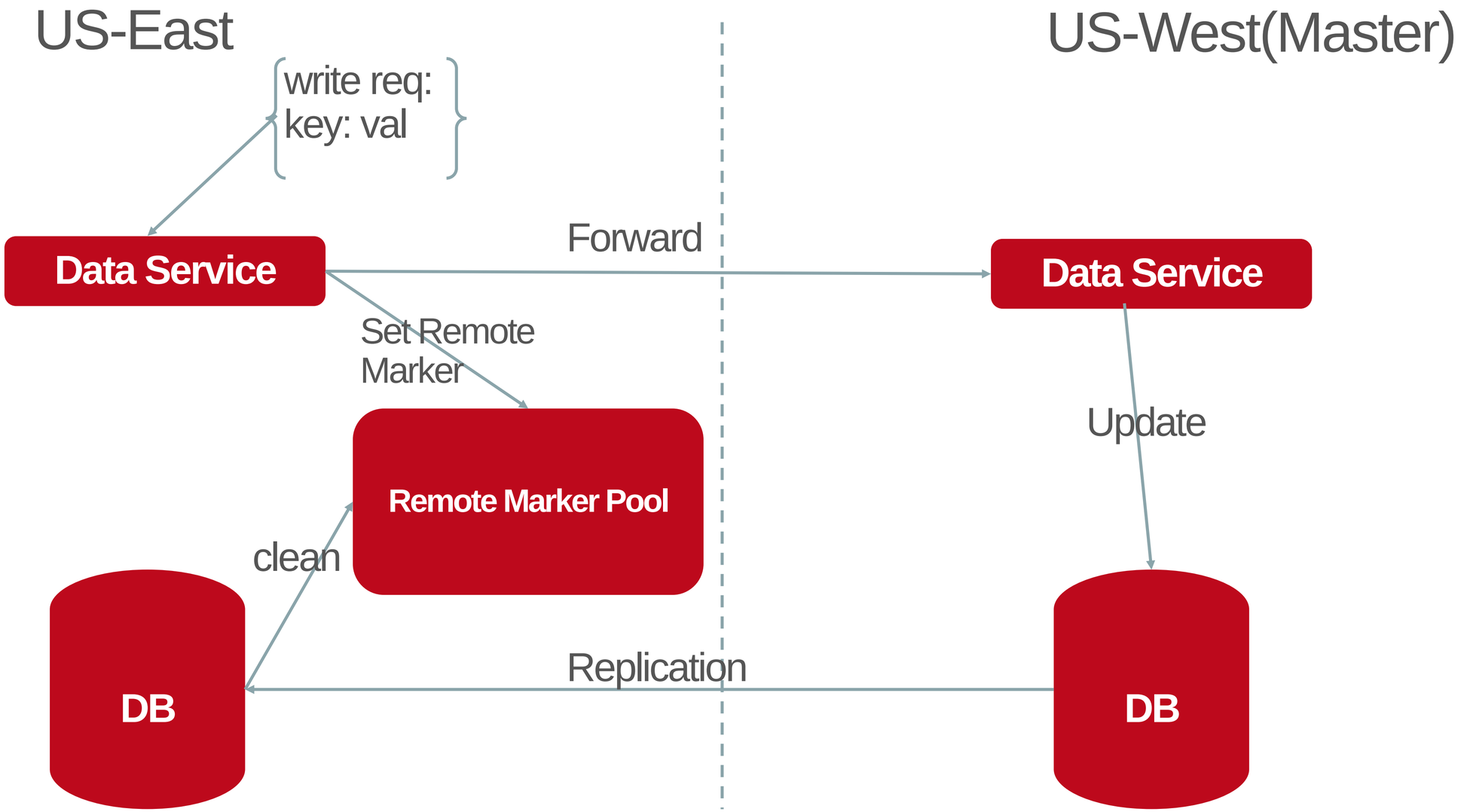
Architecture



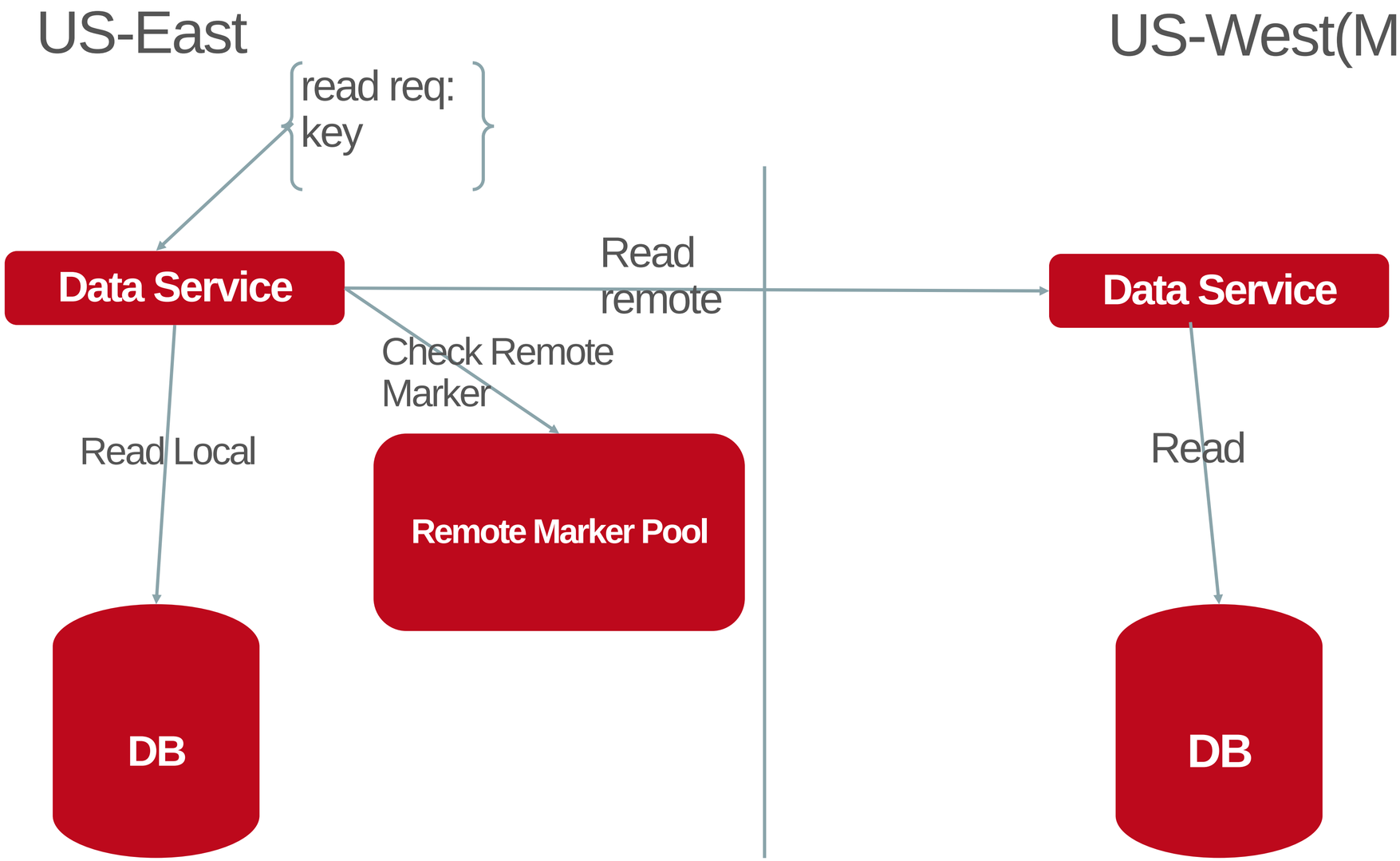
Master-Master



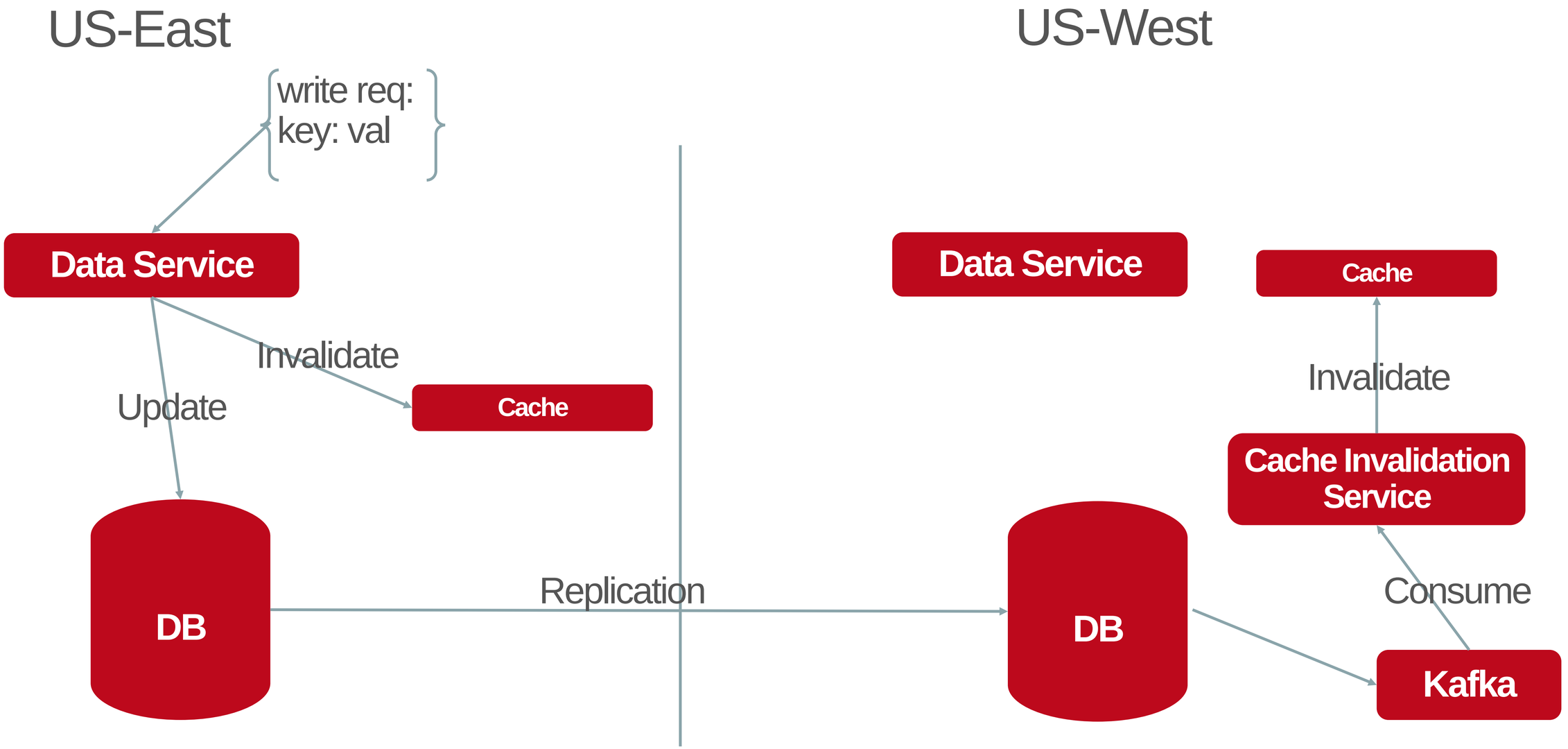
Master-Slave Write



Master-Slave Read



Cache Invalidation Service



DB

Mysql

HBase

Mysql

Maxwell

Mysql Comment

Maxwell's Daemon

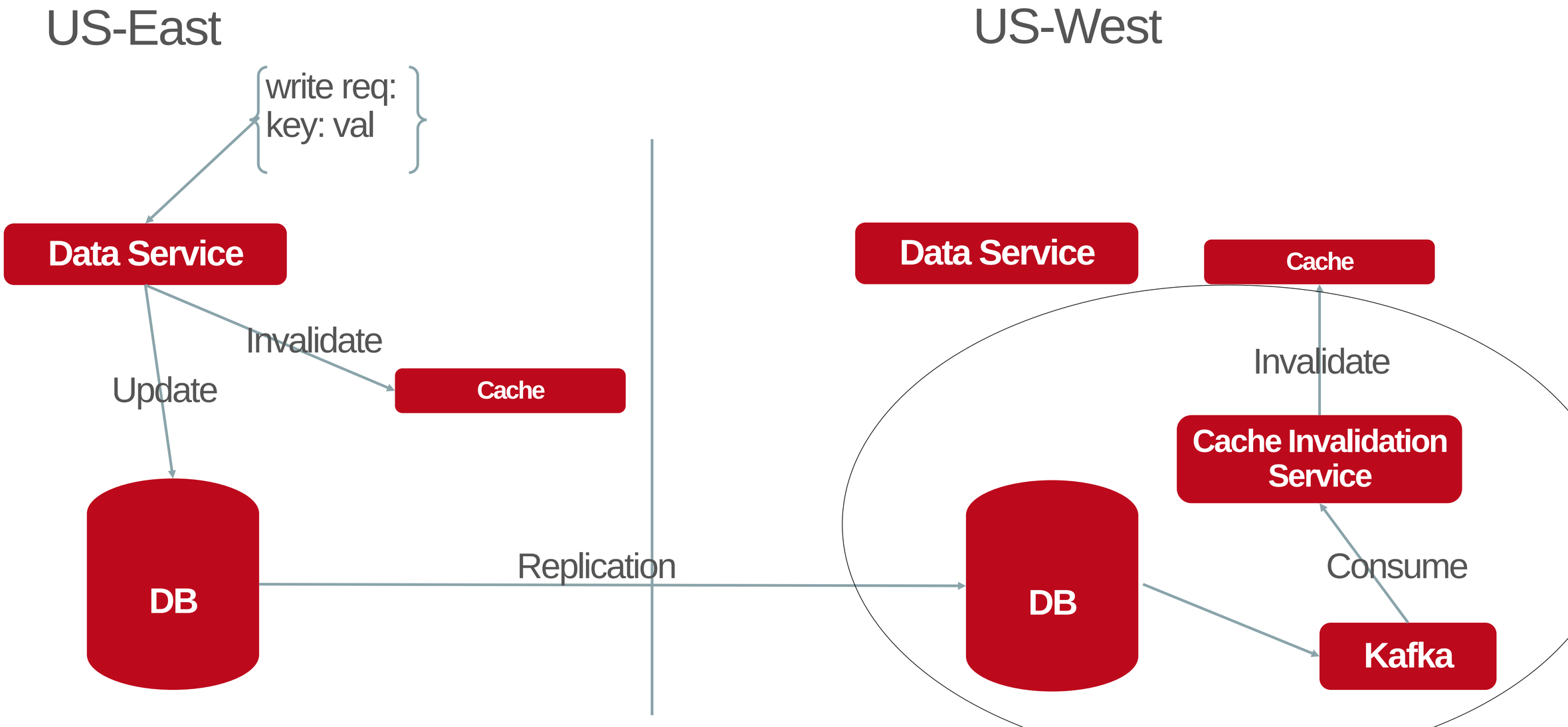
- Overview
- Quick Start
- Configuration
- Producers
- Data Format
- Encryption
- Bootstrapping
- Monitoring
- Embedding
- Schemas
- Compat / Caveats
- Changelog

Thi
Kir
lov
eas
co
bui

→
→

MySQL®

Cache Invalidation Service

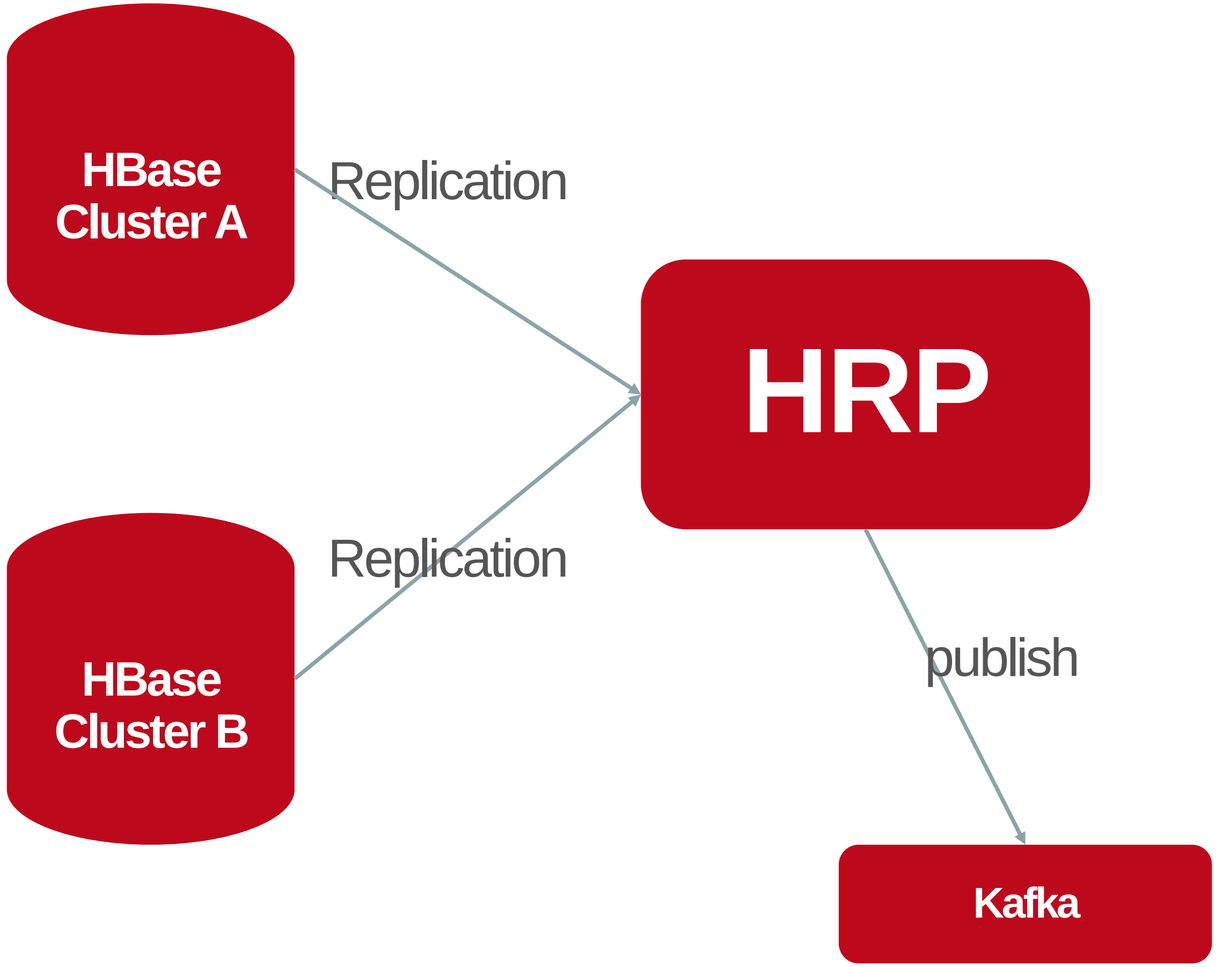


Mysql and HBase

DB	Kafka	Comment
Mysql	Maxwell	Mysql Comment
HBase	HBase replication proxy	Hbase Annotations

HBase Replication Proxy

- Expose HBase replicate API
- Customized Kafka Topic
- Event corresponding to mutation
- Multiple HBase clusters share one HRP



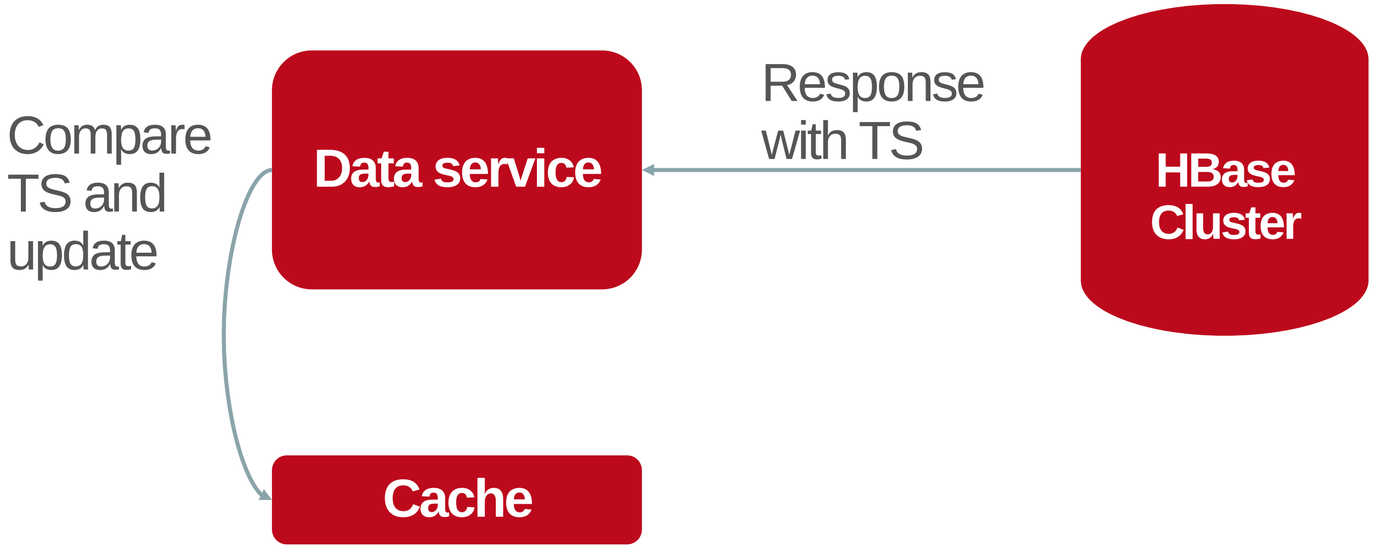
HBase Annotations

- Part of Mutate
- Written in WAL log, but not Memstore

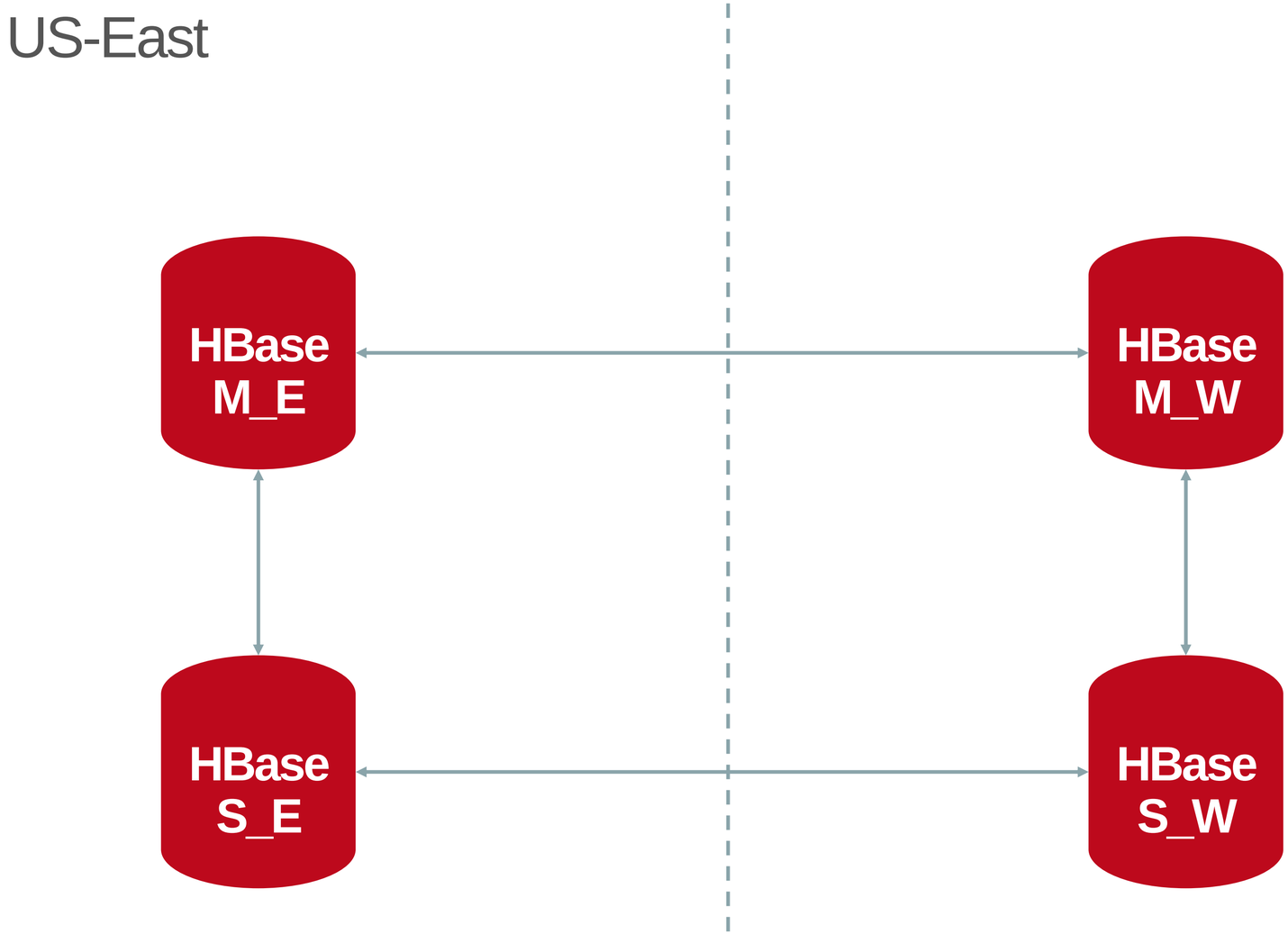

```
{
    "data": {
        "rowKey": "gAAAAAAAAAA=",
        "table": "test",
        "operation": "put",
        "delta": {
            "ZA==": {
                "Xw==": "AAAnEQ==",
                "Kg==": "AAAAAQ==",
                "bm9uX3VuaXF1ZQ==": "QUFBQQ==",
                "dW5pcXVl": "QUFBQQ==",
                "aW5jb21pbmc=": "QUFBQQ==",
                "b3V0Z29pbmc=": "QUFBQQ=="
            }
        },
        "type": "AAAnEQ==",
        "forwardRequestId": "MTUwNjM3NDE5NDc1MQ==",
        "ts": 1526676858593
    }
}
```

HBase Timestamp

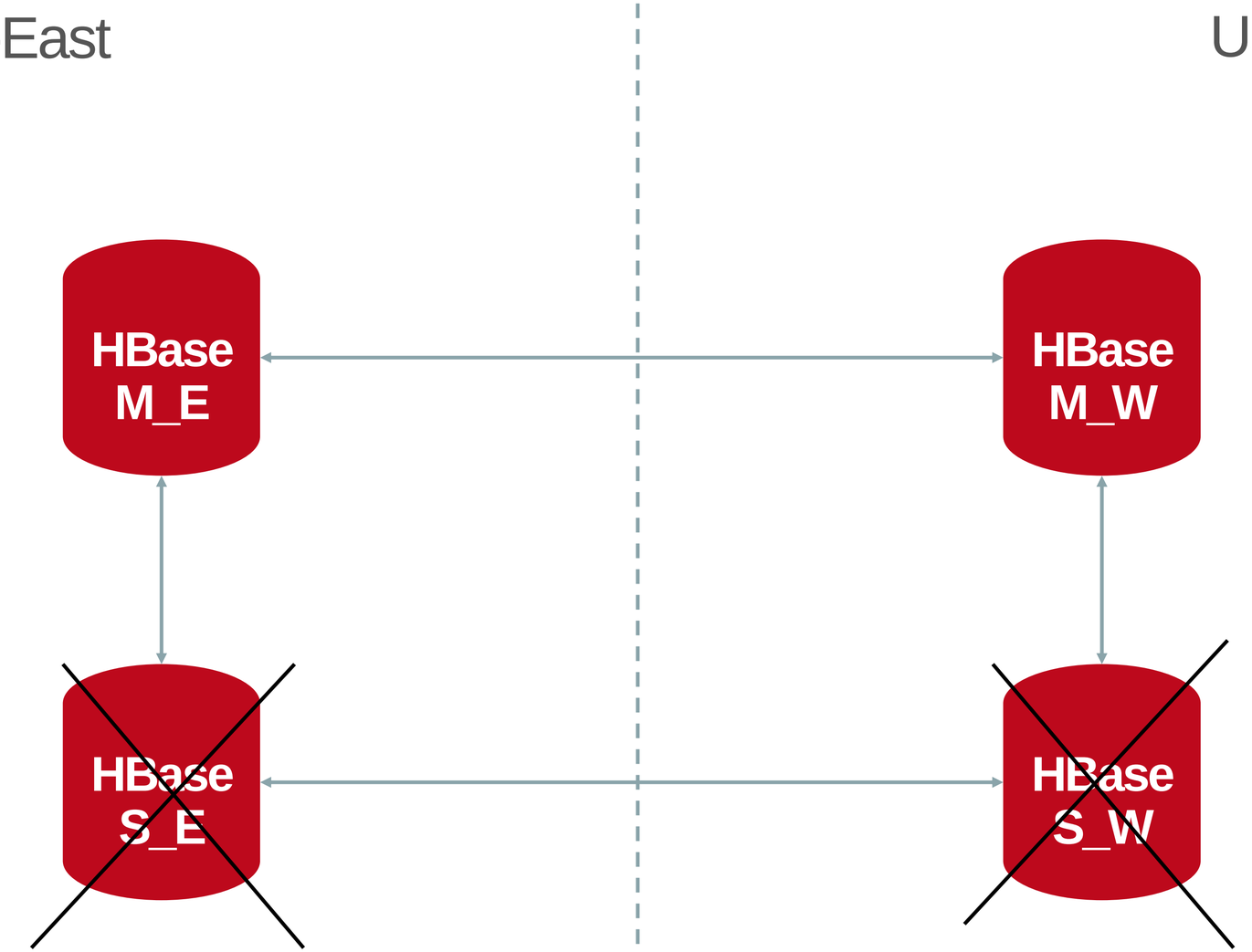
- Avoid race condition



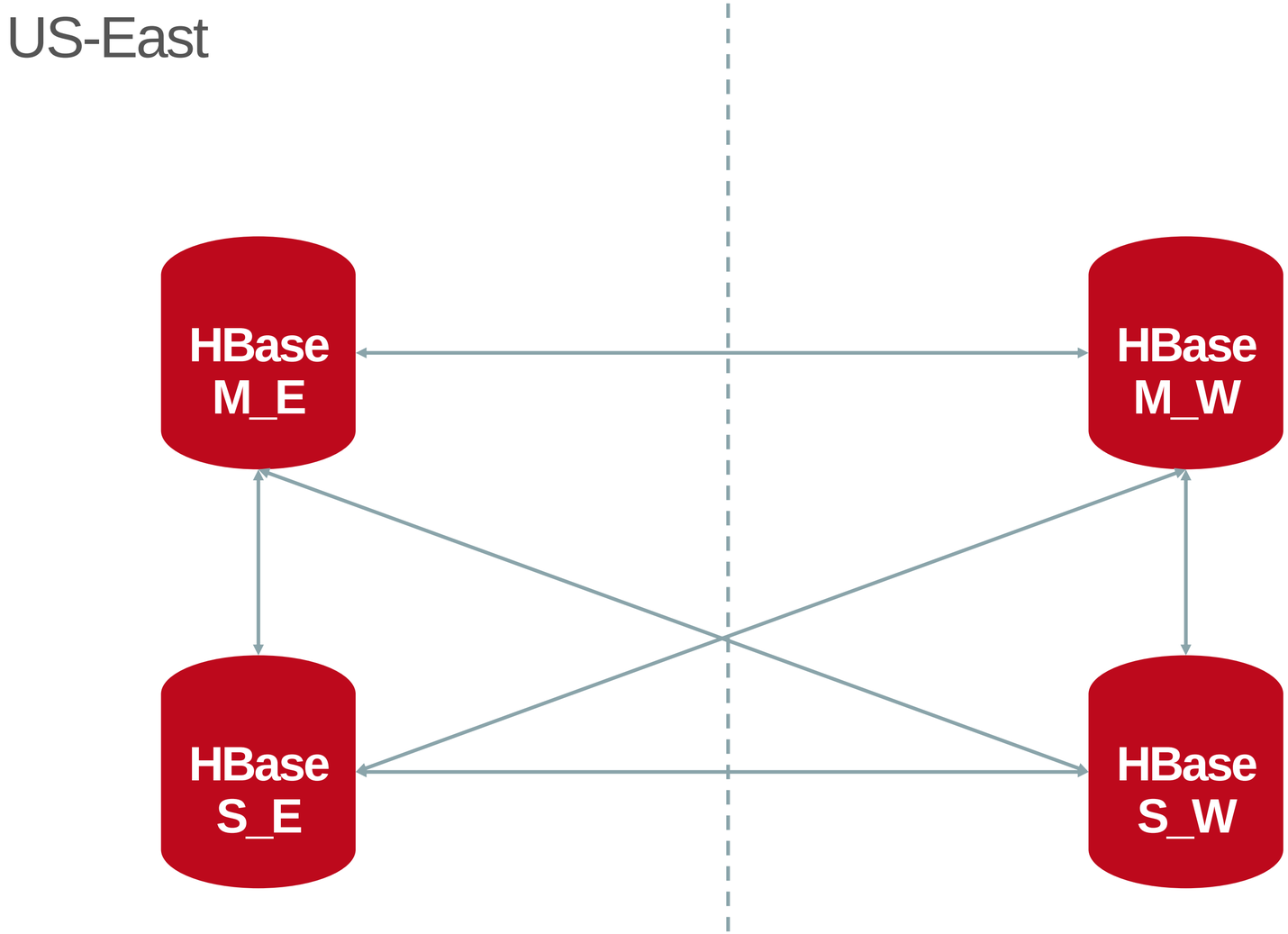
Replication Topology Issue



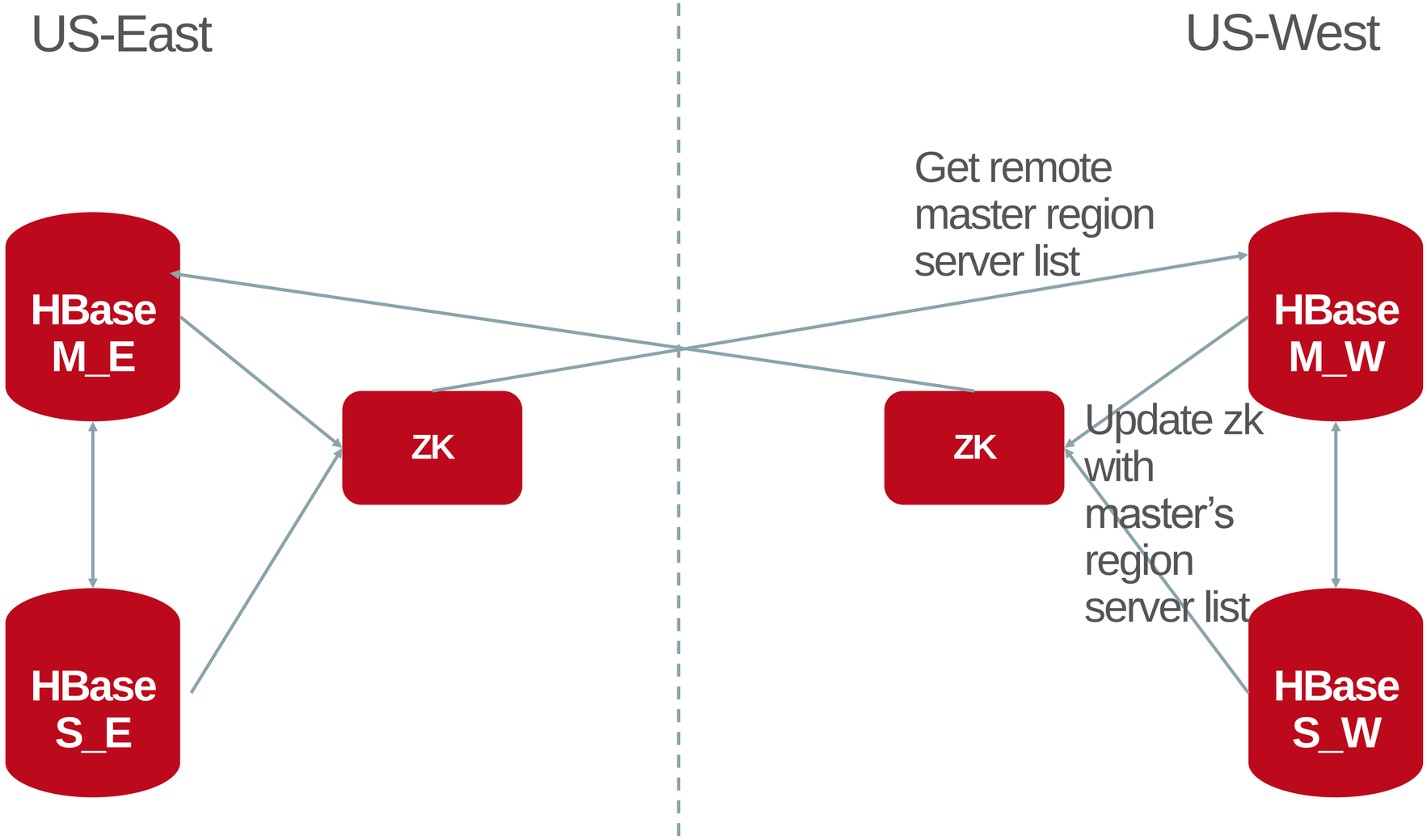
Replication Topology Issue



Replication Topology Issue



Replication Topology Issue



Improving backup efficiency

01 HBase backup at Pinterest

02 Simplifying backup pipeline

03 Offline Deduplication

HBase Backup at Pinterest

HBase serves highly critical data

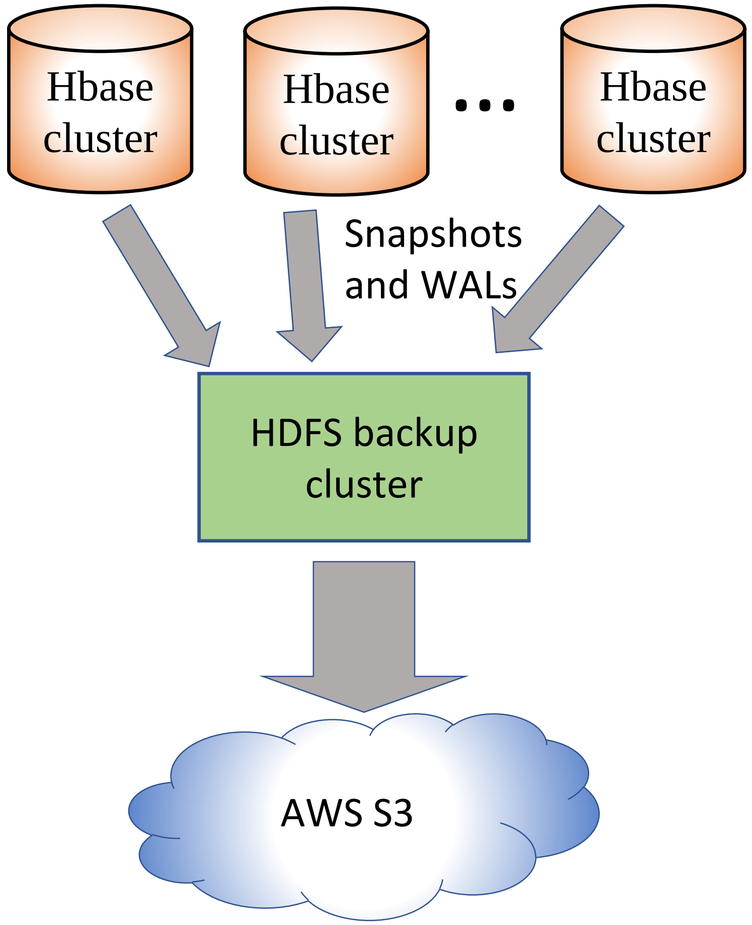
- Requires very high availability
- 10s of clusters with 10s of PB of data
- All needs backup to S3

Daily backup to S3 for disaster recovery

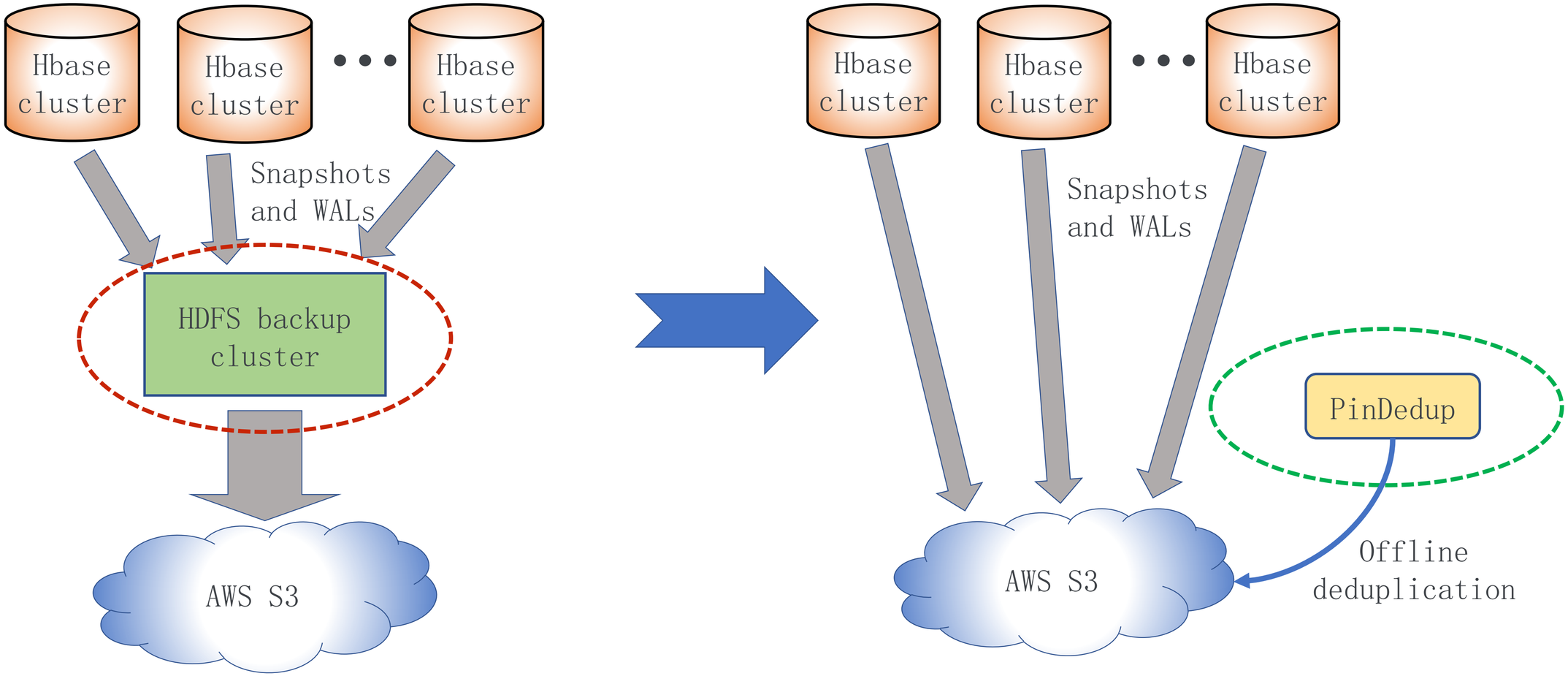
- Snapshot + WAL for point-in-time recovery
- Maintain weekly/monthly backups according to retention policy
- Also used for offline data analysis

Legacy Backup Problem

- HBase 0.94 does not support S3 export
- Two-step backup pipeline
  - HBase -> HDFS backup cluster
  - HDFS -> S3
- Problem with the HDFS backup cluster
  - Infra cost as data volume increases
  - Operational pain on failure



Upgrade Backup Pipeline



HBase 0.94

HBase 1.2

Challenge and Approach

Directly export HBase backup to S3

- Table export done using a variant of distcp
- Use S3A client with the fast upload option

Direct S3 upload is very CPU intensive

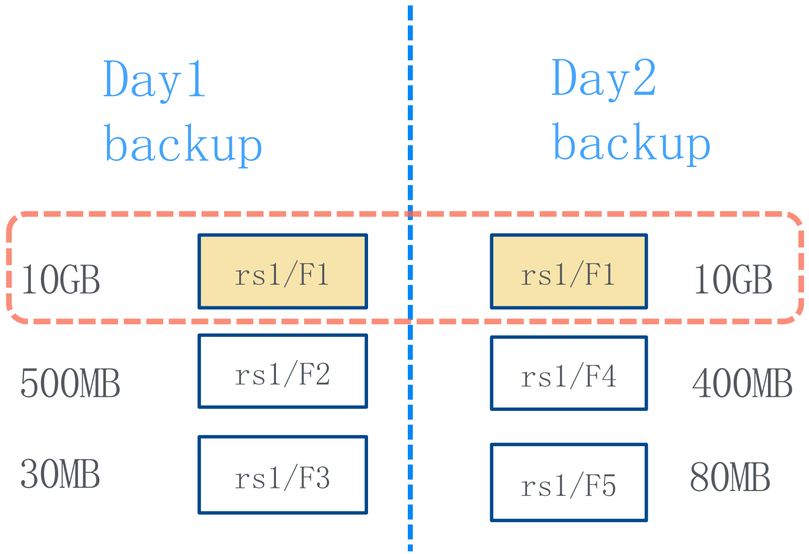
- Large HFiles broken down into smaller chunks
- Each chunk needs to be hashed and signed before upload

Minimize impact on prod HBase clusters

- Constrain max number of threads and Yarn contains per host
- Max CPU Overhead during backup < 30%

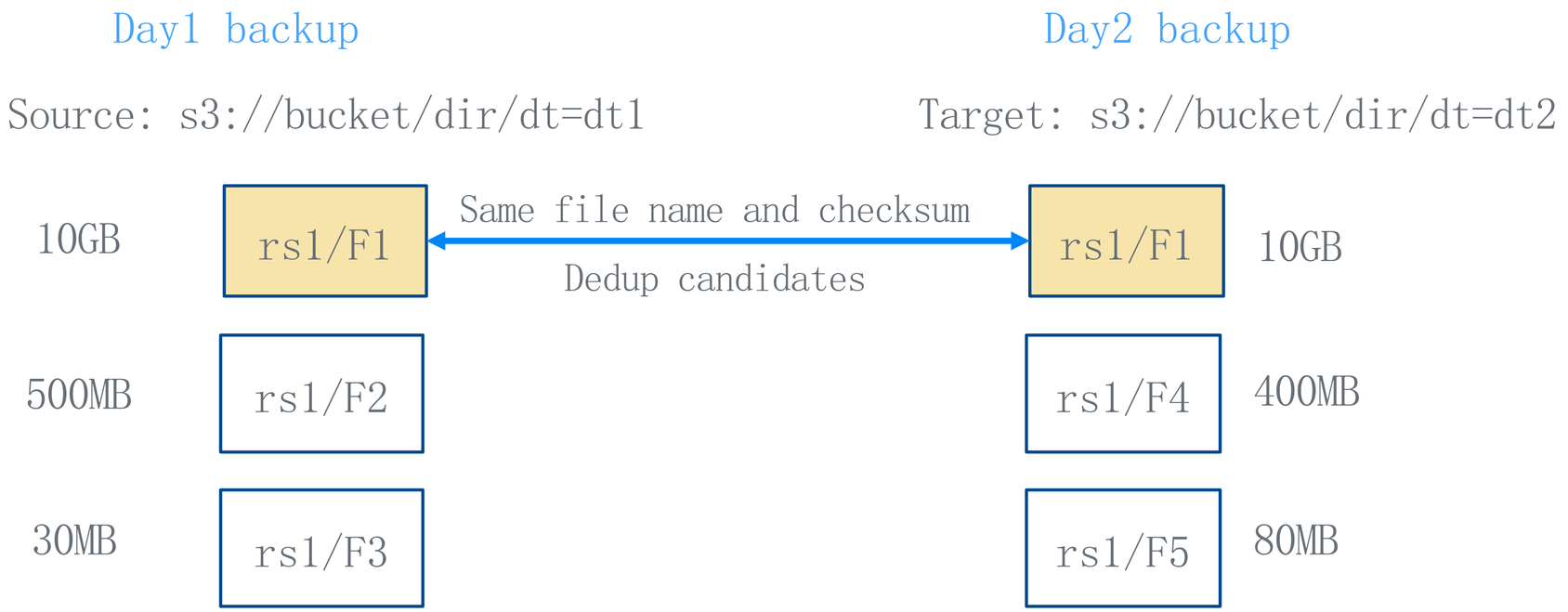
Offline HFile Deduplication

- HBase backup contains many duplicates
- Observation: large HFiles rarely change
  - Account for most storage usage
  - Only merged during major compaction
  - For read-heavy clusters, much redundancy across backup cycles
- PinDedup: offline S3 deduplication tool
  - Asynchronously checks for duplicate S3 files
  - Replace old files with references to new ones



Largest file usually unchanged

PinDedup Approach



- Only checks HFiles in the same regions in adjacent dates
- Declare duplicates when both filename and md5sum match
- No need for large on-disk dedup index, very fast lookup

Design Choices

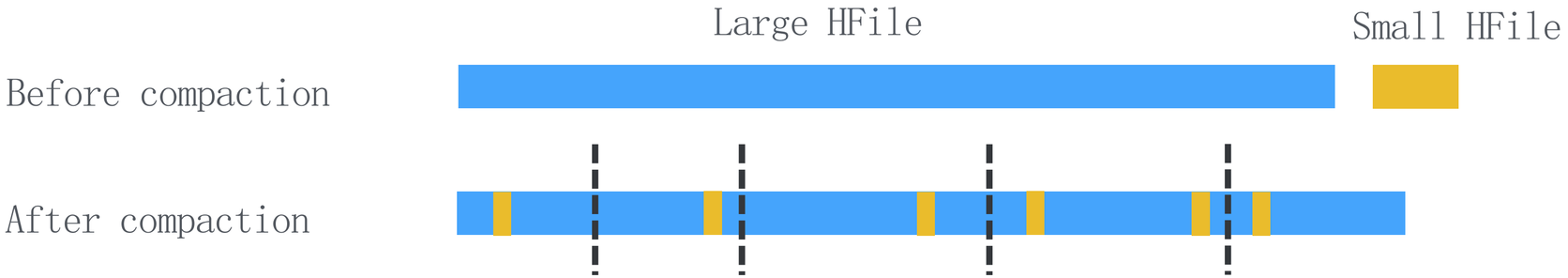
- File- vs. chunk-level deduplication
- Online vs. offline deduplication
- File encoding

File- vs. Chunk-level Dedup

More fine-grained duplication detection? -> Chunk-level dedup

Only marginal benefits

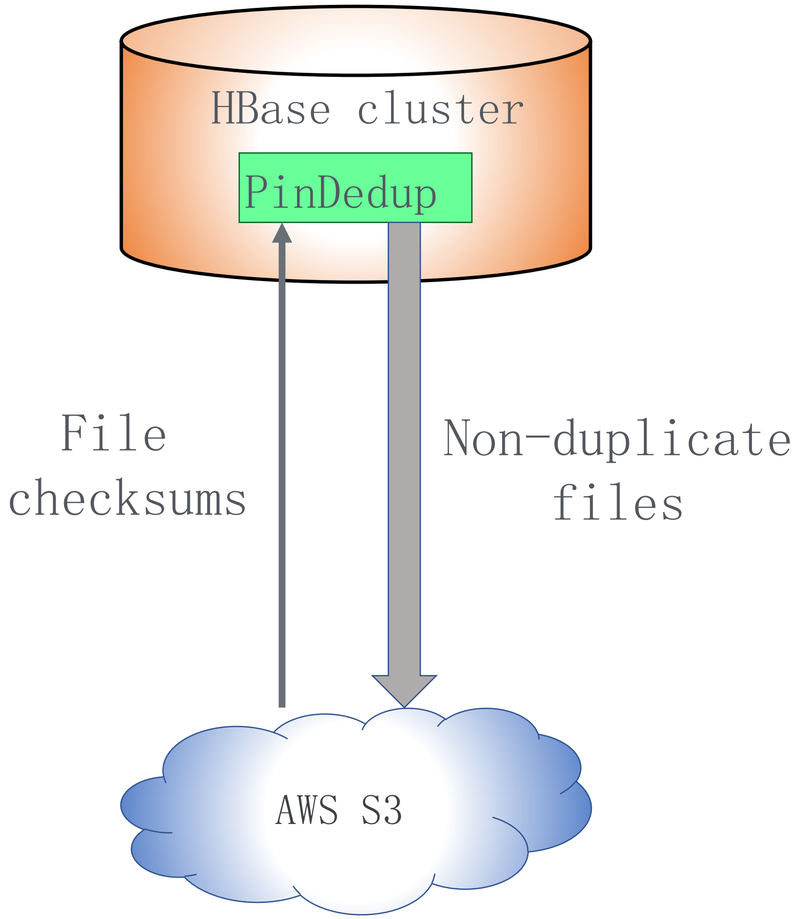
- Rabin fingerprint chunking, 4K average chunk size
- Increased complexity for implementation
- During compaction, merged changes are spread across entire file

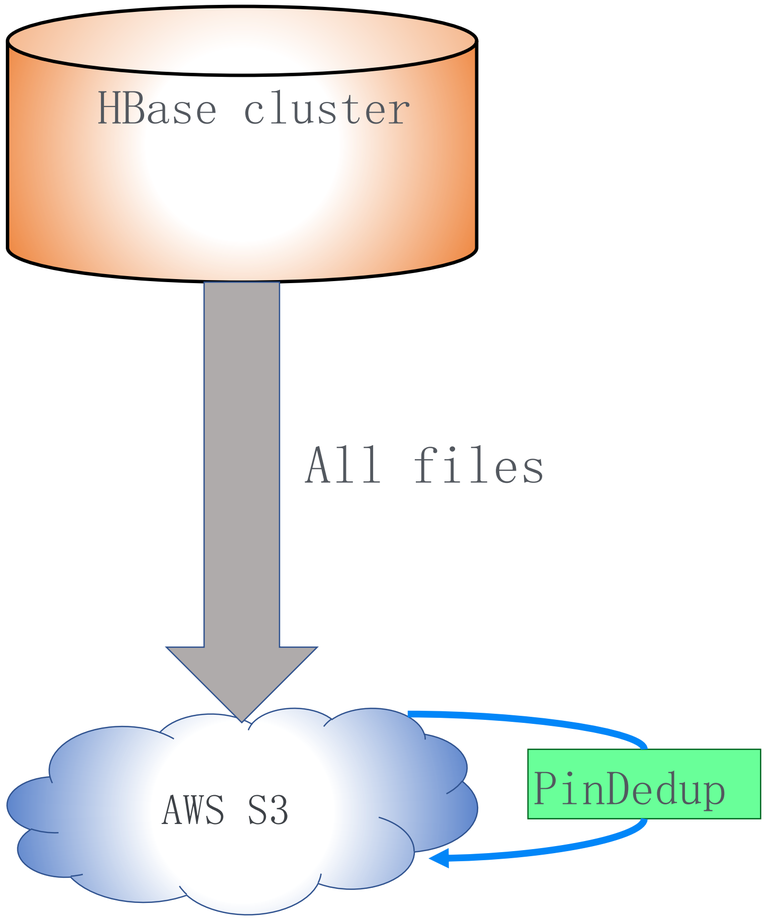


Lessons

- File-level dedup is good enough
- Less aggressive major compaction to keep the largest files unchanged

Online vs. Offline Dedup





Online dedup:

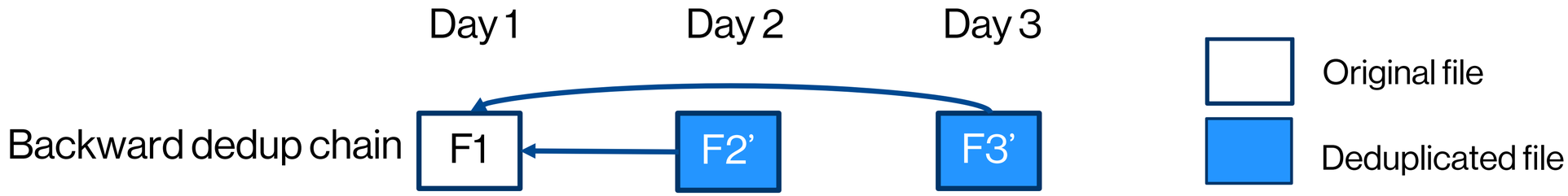
- reduces data transfer to S3

Offline dedup:

- More control on when dedup occurs
- Isolate backup and dedup failures

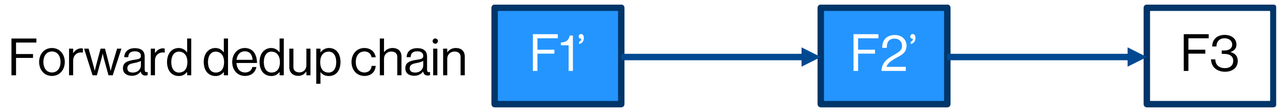
File encoding

- Dedup the old or new file?
- Intuition: keep the old file, dedup the new one



  - Pros: one-step decoding
  - Cons: dangling file pointers when old files are deleted. E.g, when F1 is garbage collected, F2' and F3' become unaccessible.

- Design choice: keep the new file, dedup the old one



  - No overhead accessing the latest copy (most use cases)
  - Avoids the dangling pointer problem

Results

- Reduced backup end-to-end time by 50%
- 3-137X compression on S3 storage usage
- Significantly reduced infra cost
- Lower operational overhead

Thanks