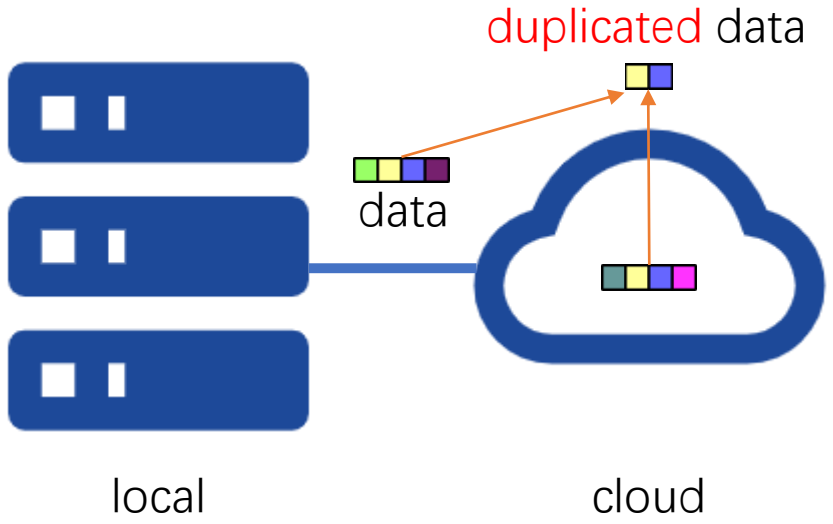
InftyDedup: Scalable and Cost-Effective Cloud Tiering with Deduplication

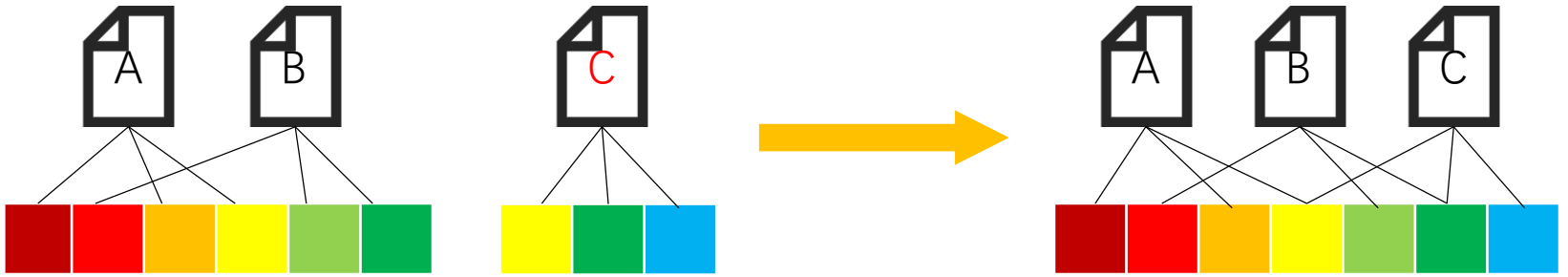
Iwona Kotlarska, Andrzej Jackowski, Krzysztof Lichota, Michal Welnicki, Cezary Dubnicki, Konrad Iwanicki

FAST'23 2023/04/05

Background

- Cloud-based backups
  - use tiering techniques to move colder data from the local to the cloud
  - backups contain a lot of duplicated data
- Deduplication
  - avoid writing the same data twice
  - reduce storage capacity





Add new file C

deduplication

need chunks
9 → 7

Problem & Challenge

- Problem
- Scalable and Cost-Effective Cloud Tiering with Deduplication

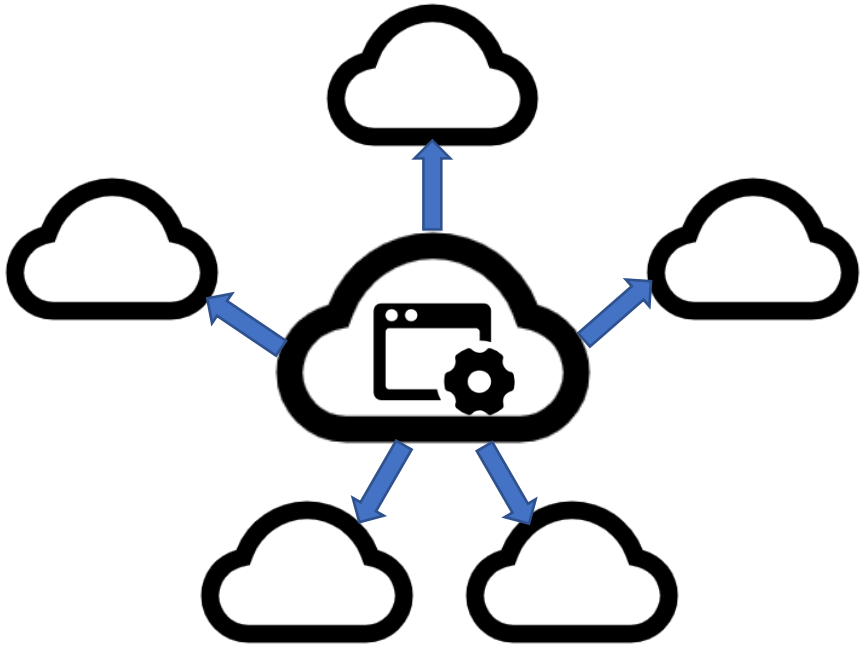
- Chanllenges :

1. deduplication is mainly implemented at the local teir and lacks scalability due to resource limitations at the local teir

2. the financial cost of storing deduplication data in cloud teir

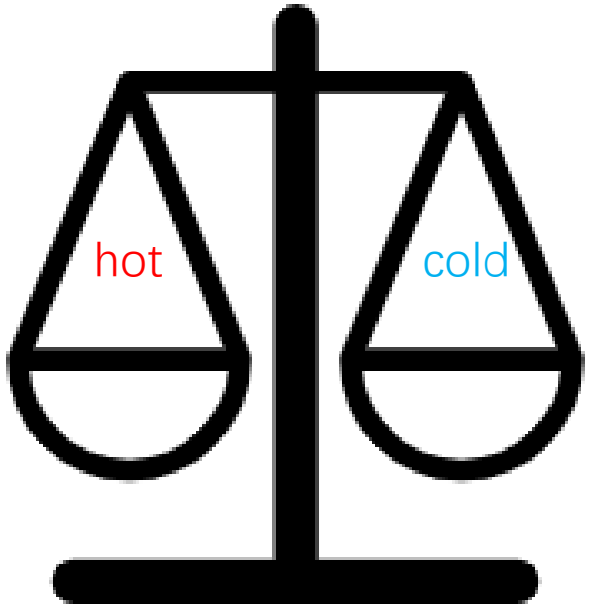
Approach

1. InftyDedup, a novel system for cloud tiering with deduplication



batch operation in the cloud

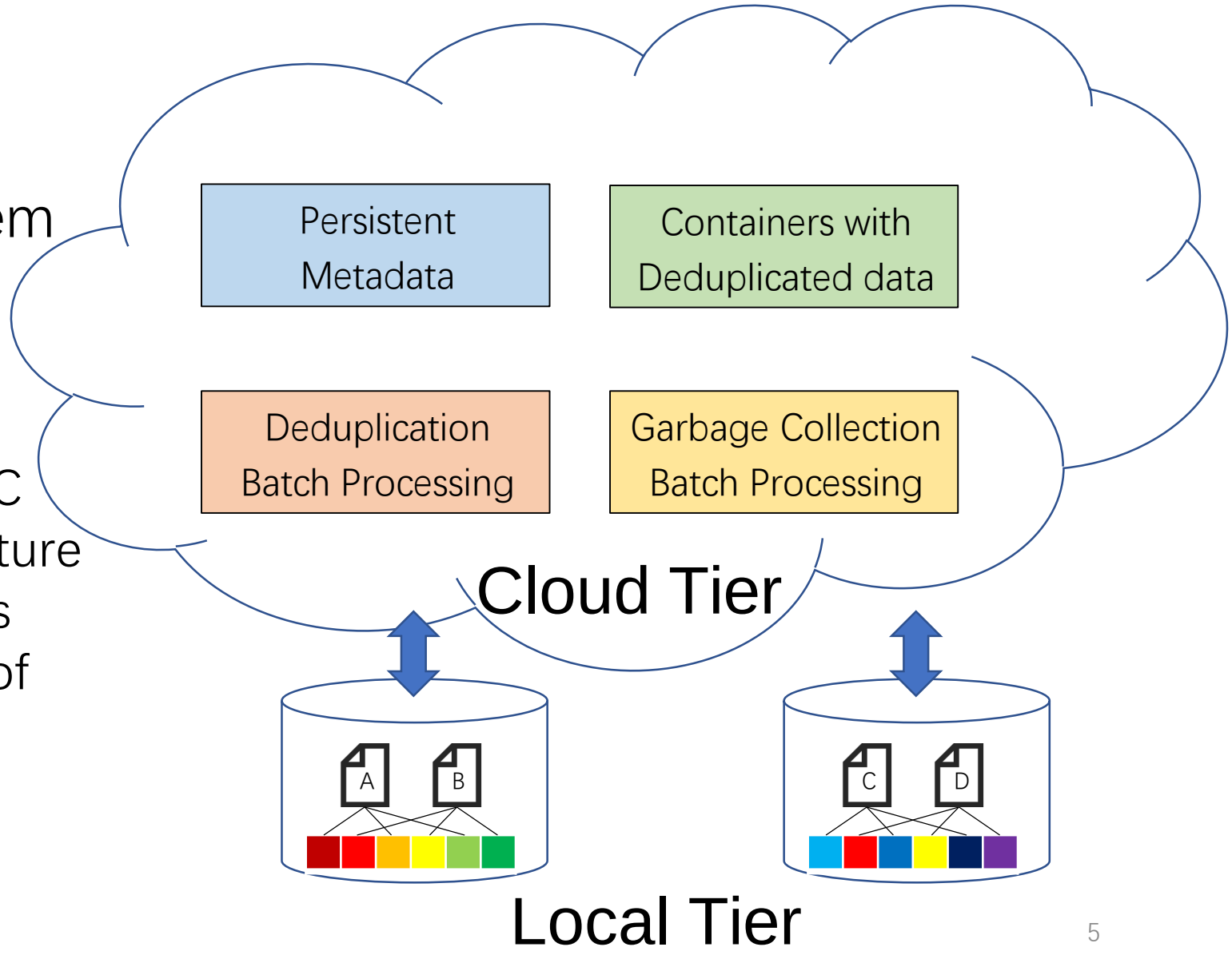
2. An algorithm for decreasing the financial cost of storing deduplicated data in the cloud tier



balance the cost

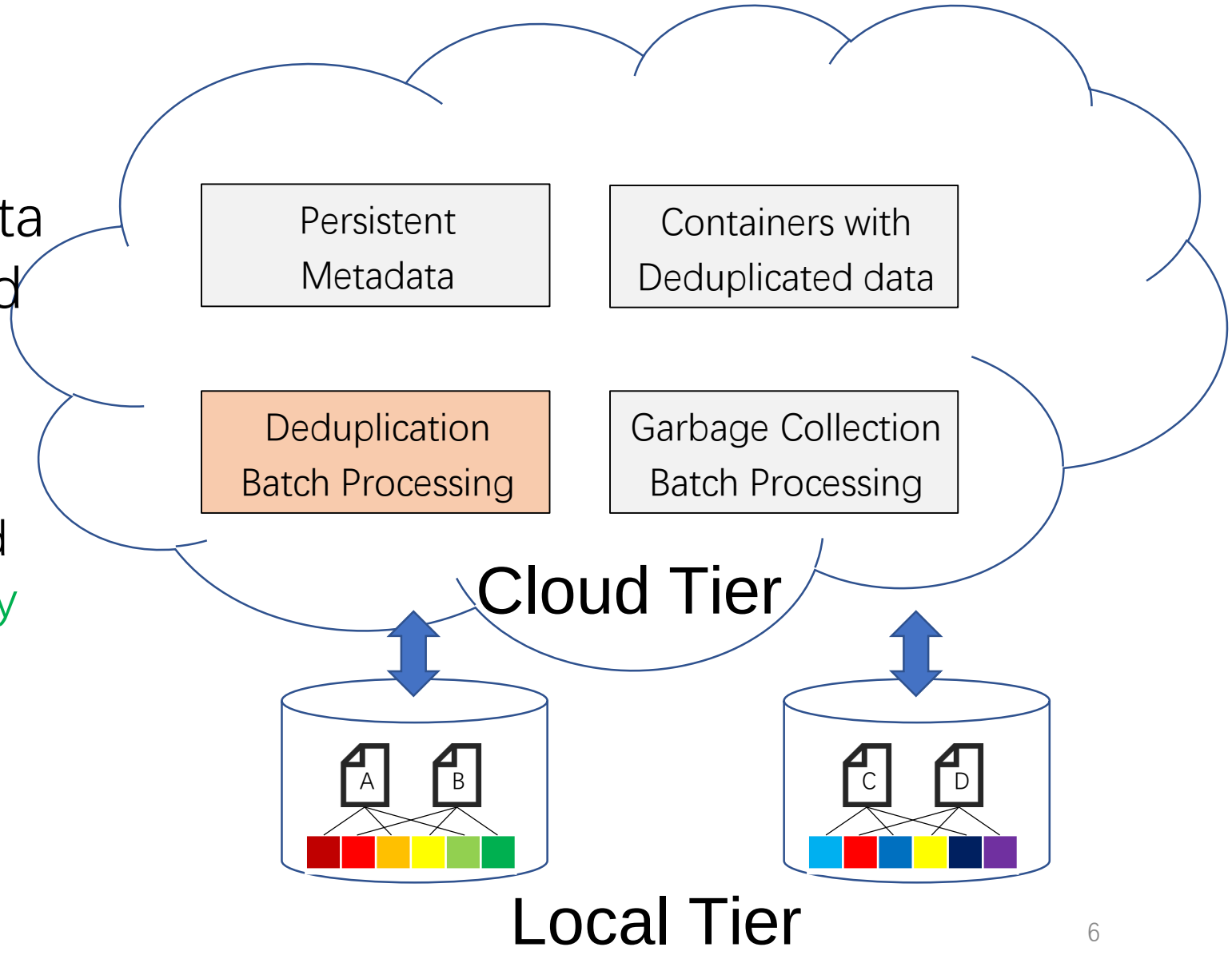
InftyDedup Architecture

1. InftyDedup, a novel system for cloud tiering with deduplication
   - HOW : implement BatchDedup and BatchGC using the cloud infrastructure
   - WHY : performance scales linearly with the number of cloud instances deployed

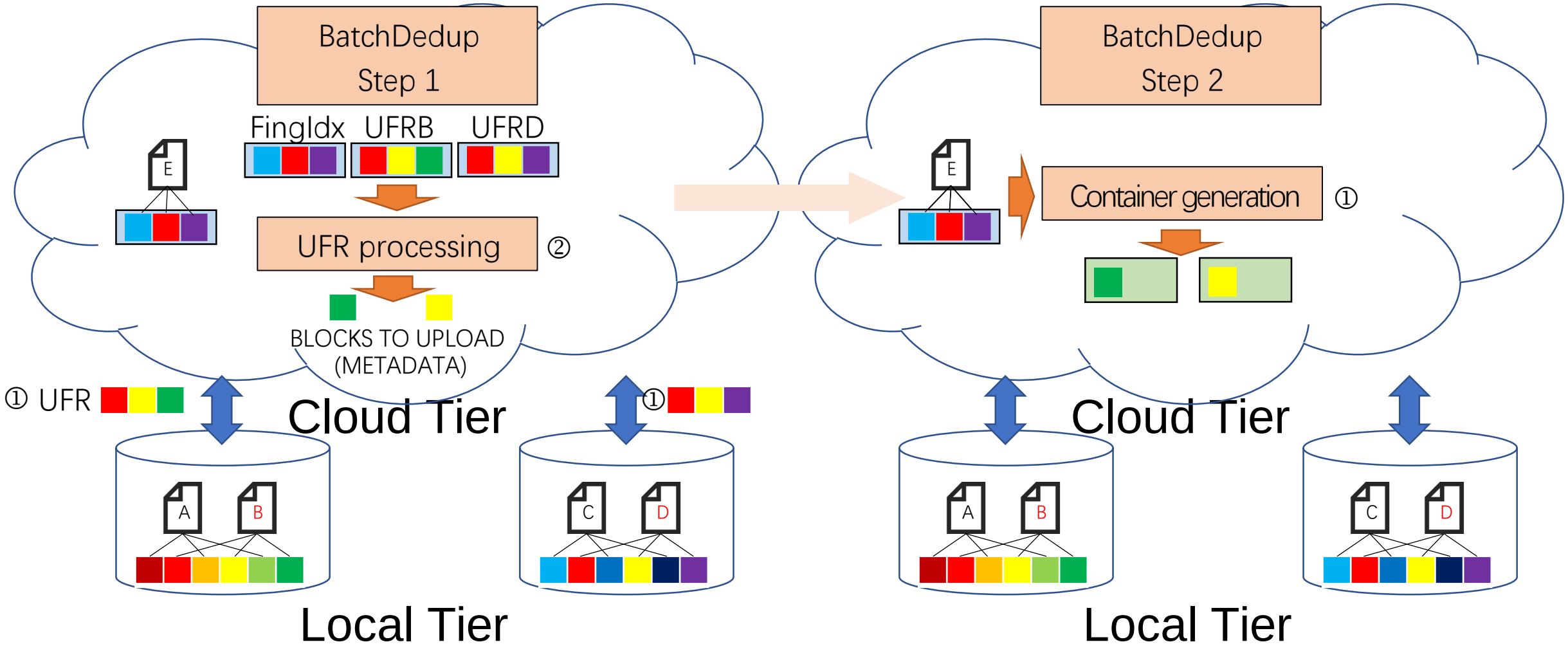


InftyDedup Architecture: Batch Deduplication

- Batch Deduplication is a distributed method of data deduplication in the cloud
  - creatively deploy multiple cloud instances and
  - each instance processes fingerprints in a distributed manner for linear scalability



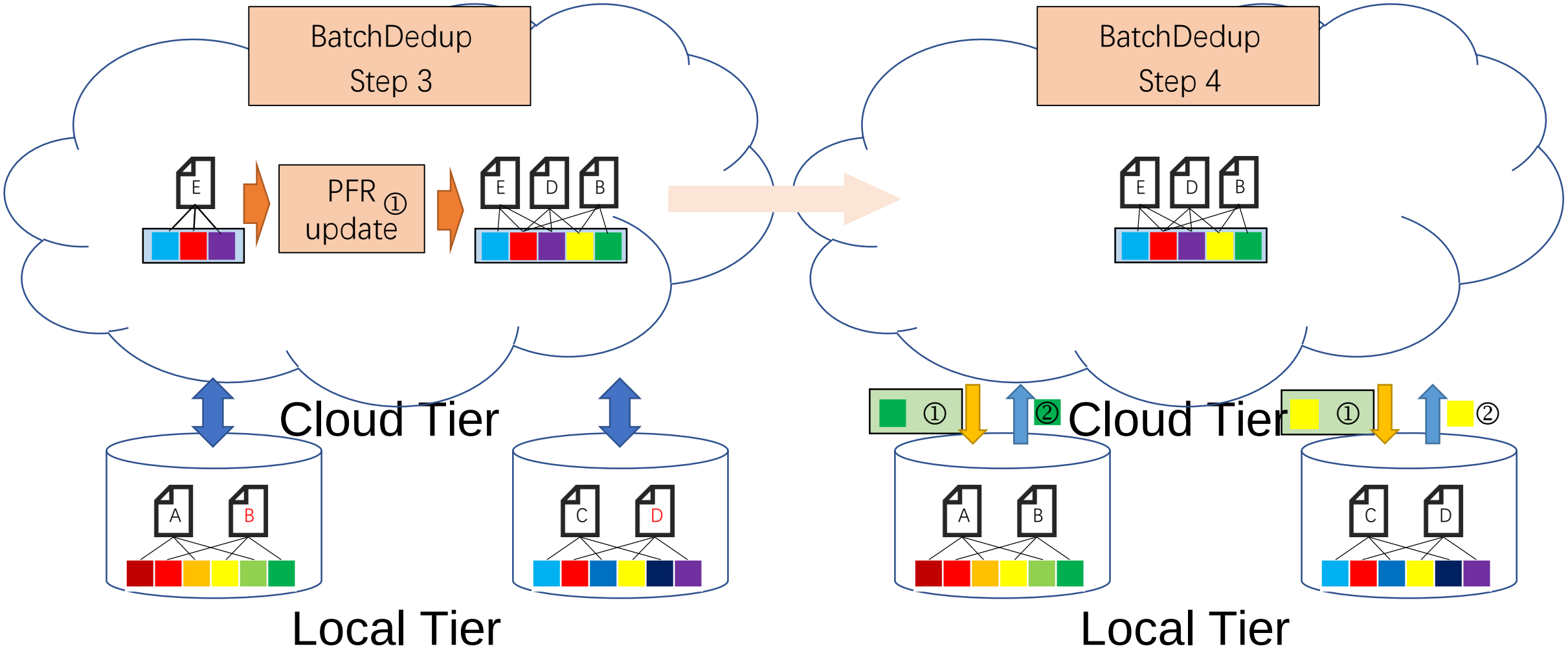
InftyDedup Architecture: Batch Deduplication Steps



Step #1: UFR processing
①local : upload UFR(Unprocessed File Recipe)
②cloud : find non-duplicate blocks

Step #2: Container generation
①cloud : generate descriptions for the local tier

InftyDedup Architecture: Batch Deduplication Steps

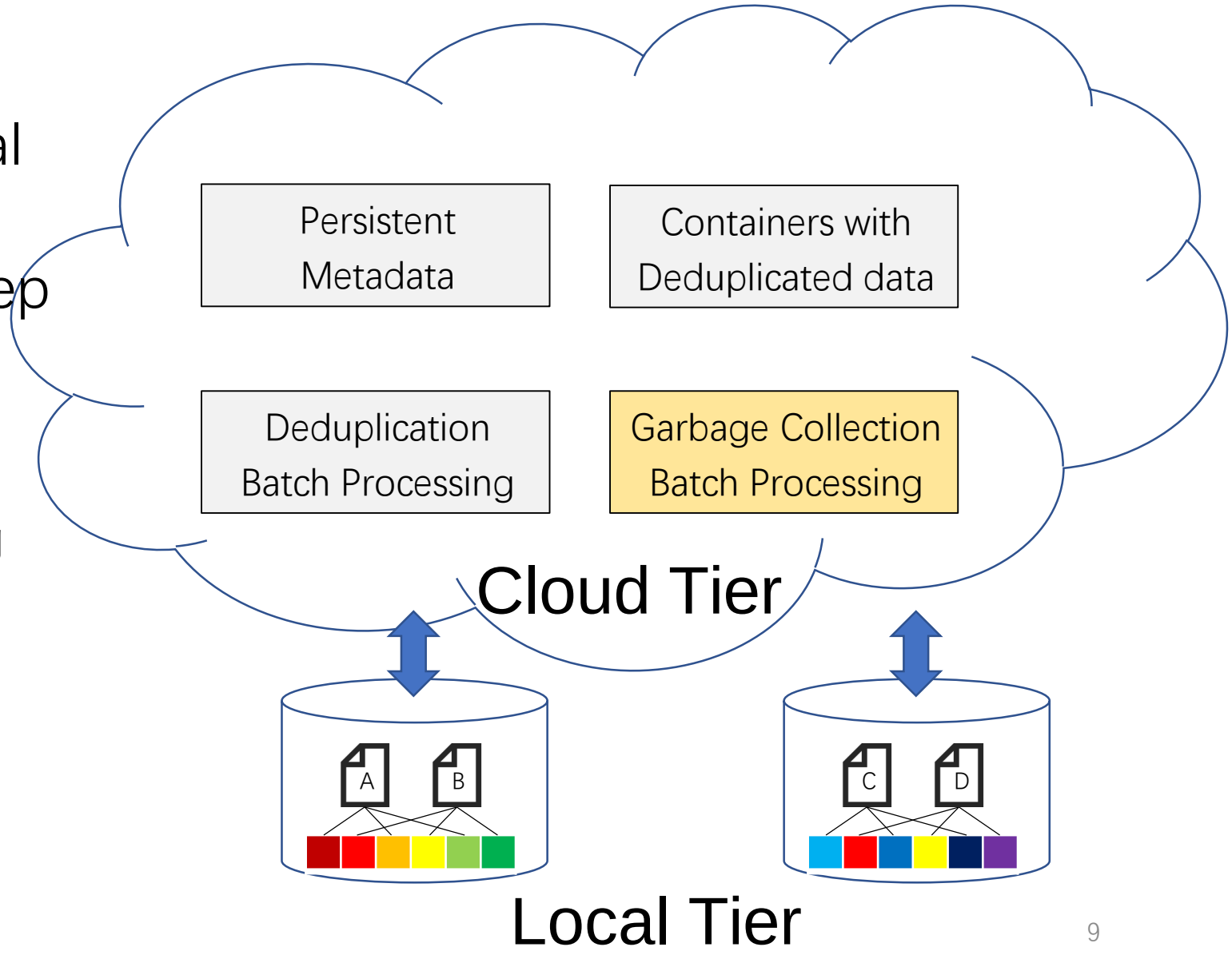


Step #3: PFR update
①local : cloud : update PFR(Processed File Recipe)

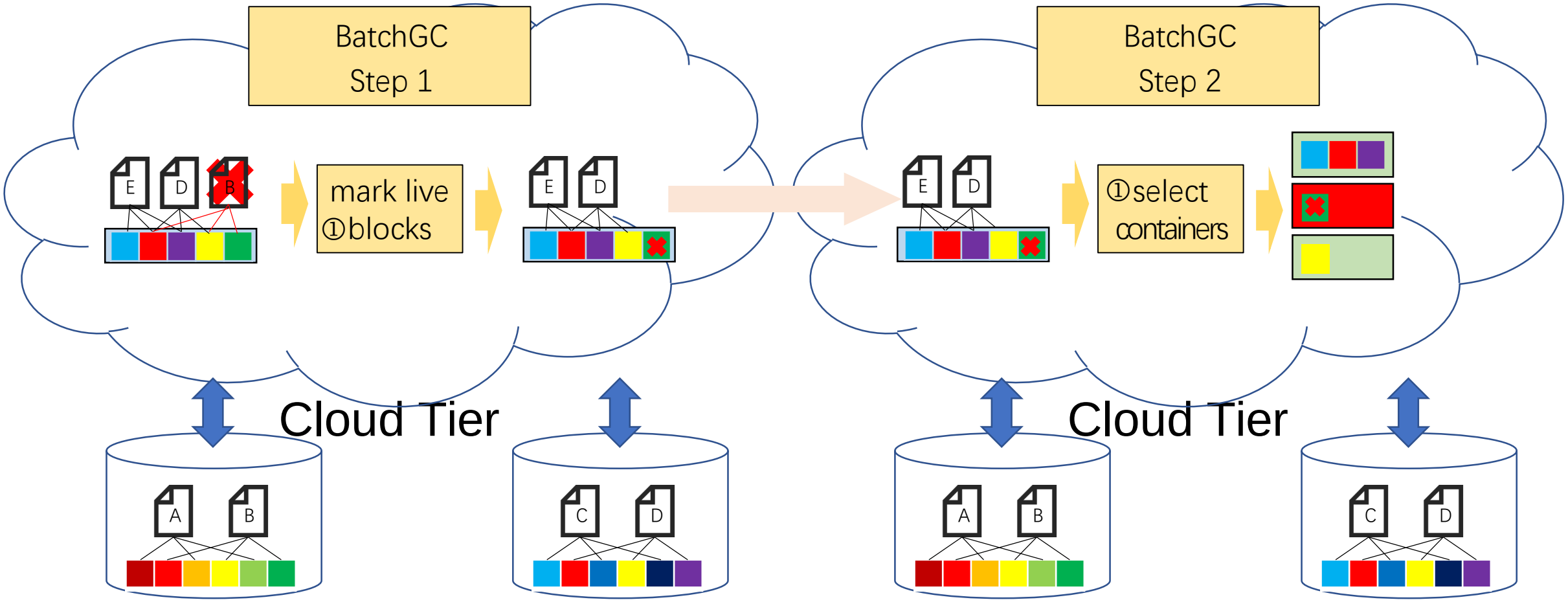
Step #4: Blocks upload
①cloud : send description
②local : upload the actual data

InftyDedup Architecture: Batch Garbage Collection

- BatchGC is an incremental contribution over the traditional mark and sweep
  - creatively take the financial cost of container rewriting into account and
  - rewrite only if the rewriting cost is lower than not rewriting



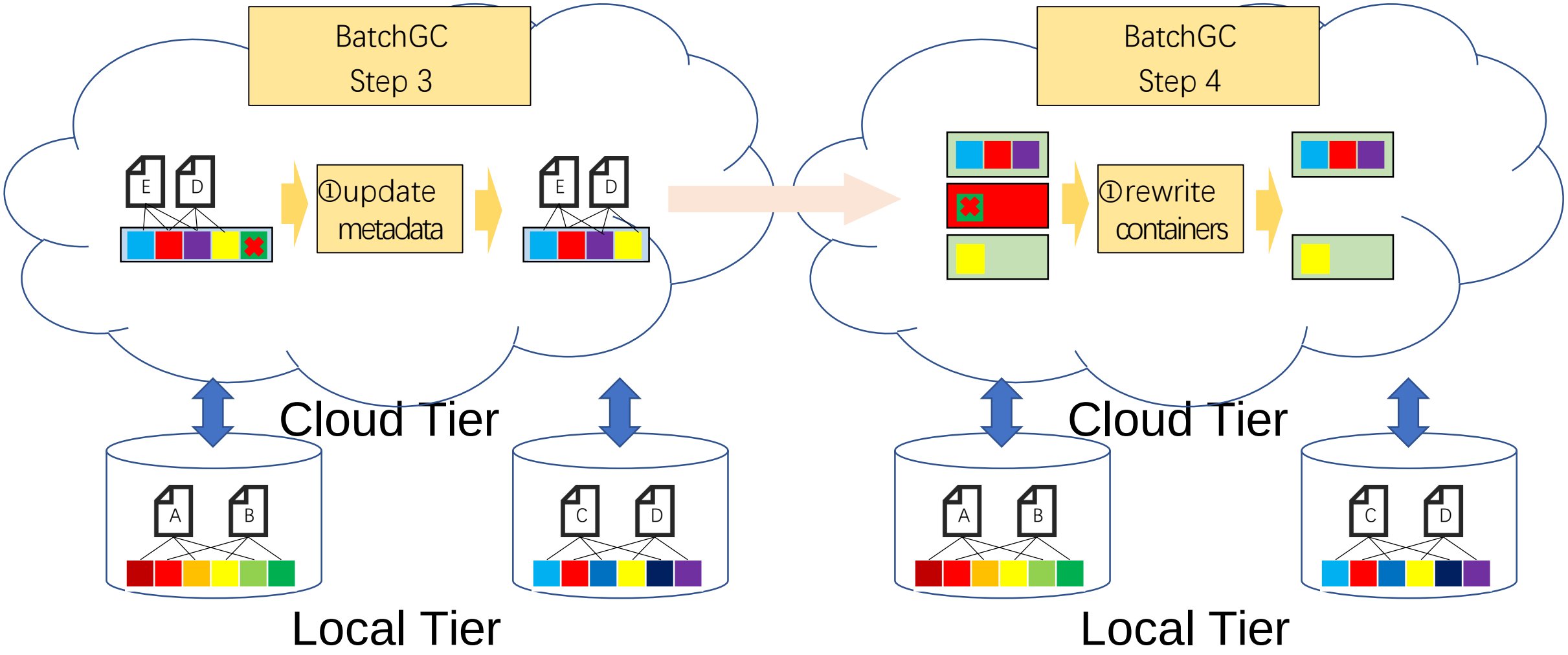
InftyDedup Architecture: Batch Garbage Collection Steps



Step #1: File removal
①mark live blocks

Step #2: Container verification
①select containers

InftyDedup Architecture: Batch Garbage Collection Steps



Step #3: Metadata are updated
①update metadata

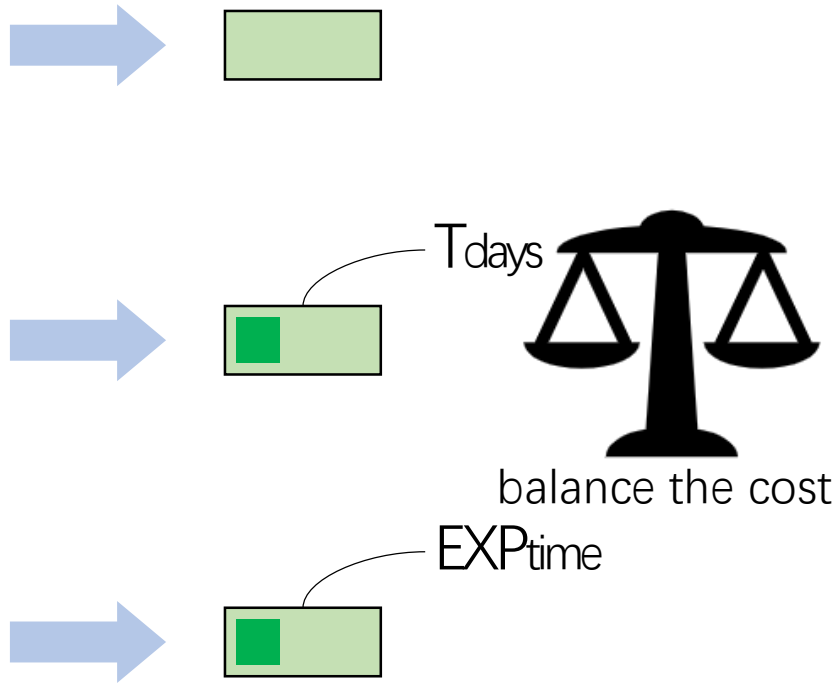
Step #4: Containers are rewritten
①rewrite containers

InftyDedup Architecture: Batch Garbage Collection Strategies

- Immediate removal of unreferenced data is not always optimal, as rewriting a container in the cloud has a significant cost.
  - GC-Strategy #1: Reclaim only empty containers
  - GC-Strategy #2: Reclaim containers if the rewrite pays for itself after T_{days}

$$x = \frac{COST_{rewrite}}{T_{days} * CAPACITY_{to\_be\_reclaimed} * COST_{byte\_per\_day}}$$

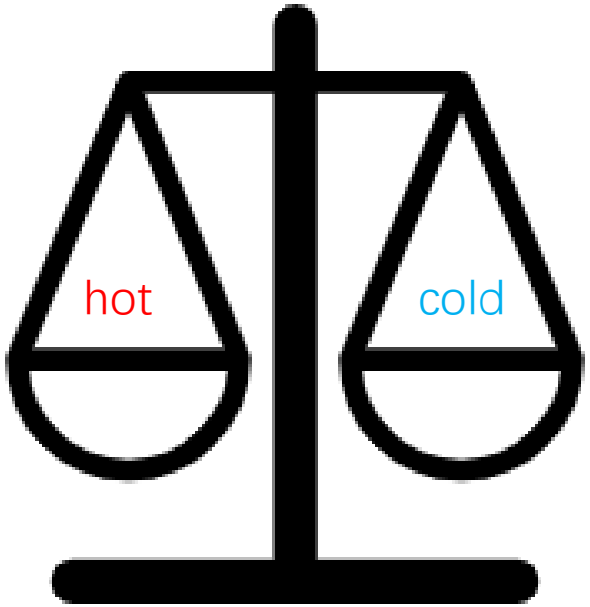
  - GC-Strategy #3: Reclaim containers based on file expiration dates
    - if UFR provides EXP_{time}, for each container, T_{days} can be calculated as the maximal EXP_{time} of its blocks



InftyDedup : Mixing Storage Classes

To reduce the cost of storing data in the cloud, InftyDedup can be extended with an algorithm that selects whether a block should be stored in hot or cold cloud storage.

+ Cheaper PUT/GET requests
+ No minimal storage period
+ No transfer fees
(other than egress traffic)
– Higher GB/month costs

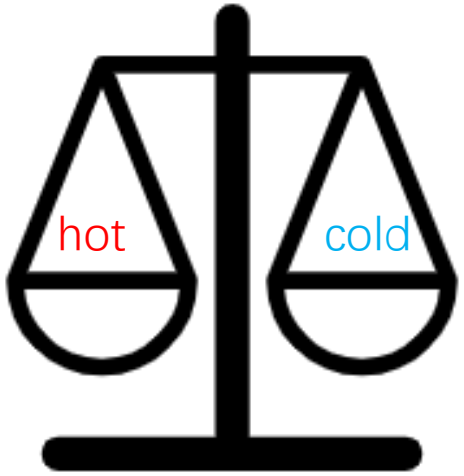


balance the cost

+ Lower GB/month costs
(e.g., 5.25 times)
– Minimal storage period
(e.g., 90-365 days)
– Additional transfer fees
– More expensive
requests (e.g., 25 times)

InftyDedup : Mixing Storage Classes

2. An algorithm for decreasing the financial cost of storing deduplicated data in the cloud tier
   - HOW : move deduplicated data chunks between cloud services dedicated to hot and cold storage according to $FREQ_{restore}$ and EXP_{time}
   - WHY :
     - recovery time : many cold storage services offer the same millisecond latency as hot storage
     - financial cost : mixing cloud storage types can reduce costs



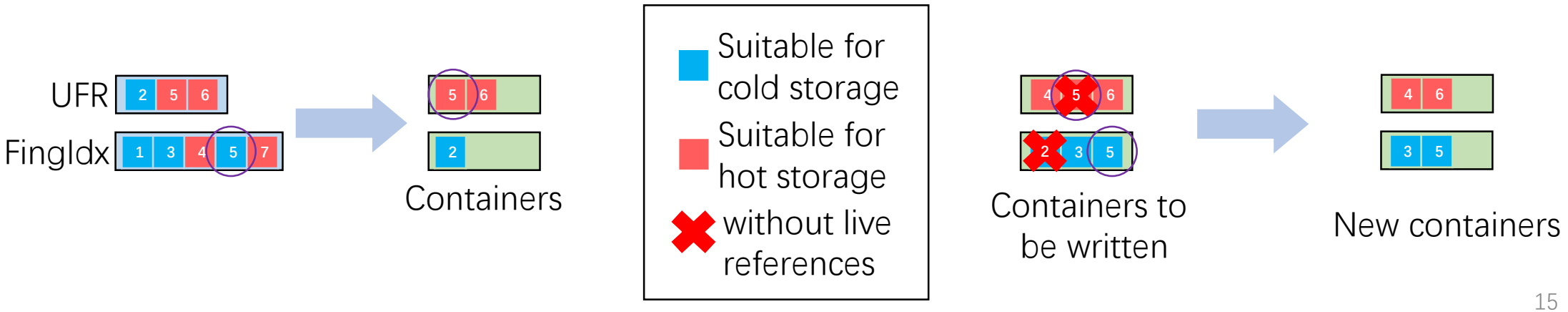
balance the cost

InftyDedup : Mixing Storage Classes

- Each block is stored in a storage type for which the following formula has lower value

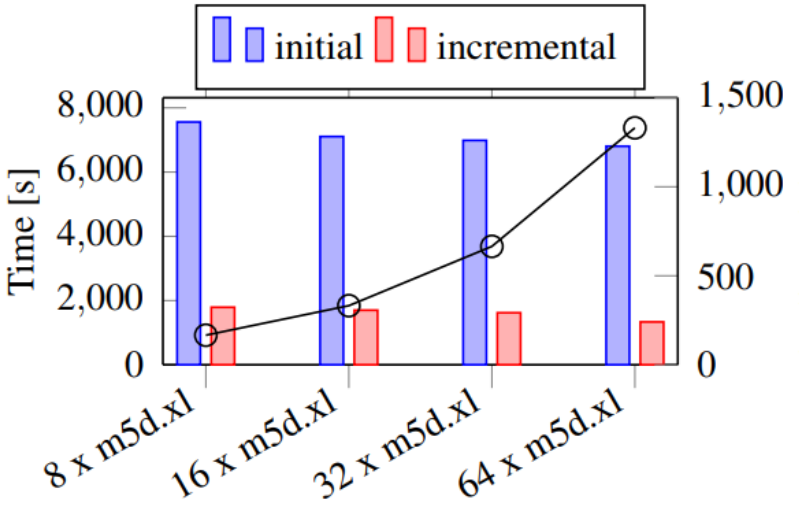
$$t = COST_{insert} + (COST_{B/day} + COST_{restore} * FREQ_{restore}) * EXP_{time}$$

- Adjustments to FREQrestore and EXPtime are required
  - e.g., a block can be initially stored in cold storage but later it receives more references
- BatchGC will eventually remove the unnecessary copies
  - e.g., when a reference with high restore frequency has been deleted



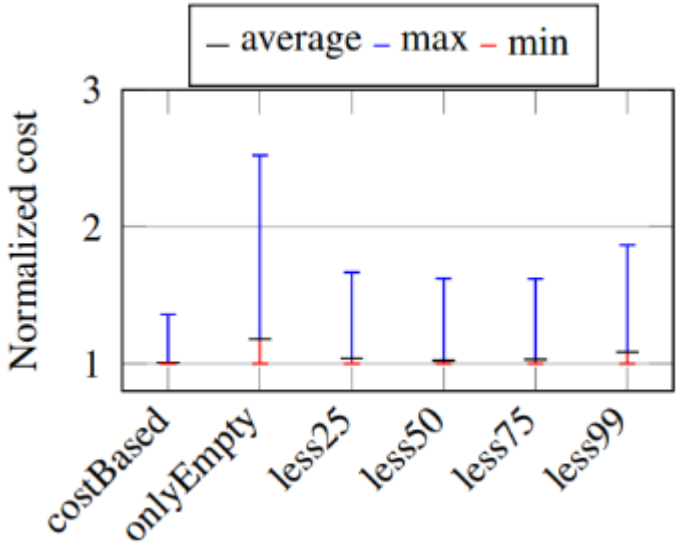
Evaluation : BatchDedup & BatchGC

- For BatchDedup, From deploying 8 instances to 64 instances, the amount of data processed increased linearly, but the processing time remained roughly the same, demonstrating the scalability of the system.



- initial step, files without duplicates
- incremental step, 3xsmaller, 90% are duplicates

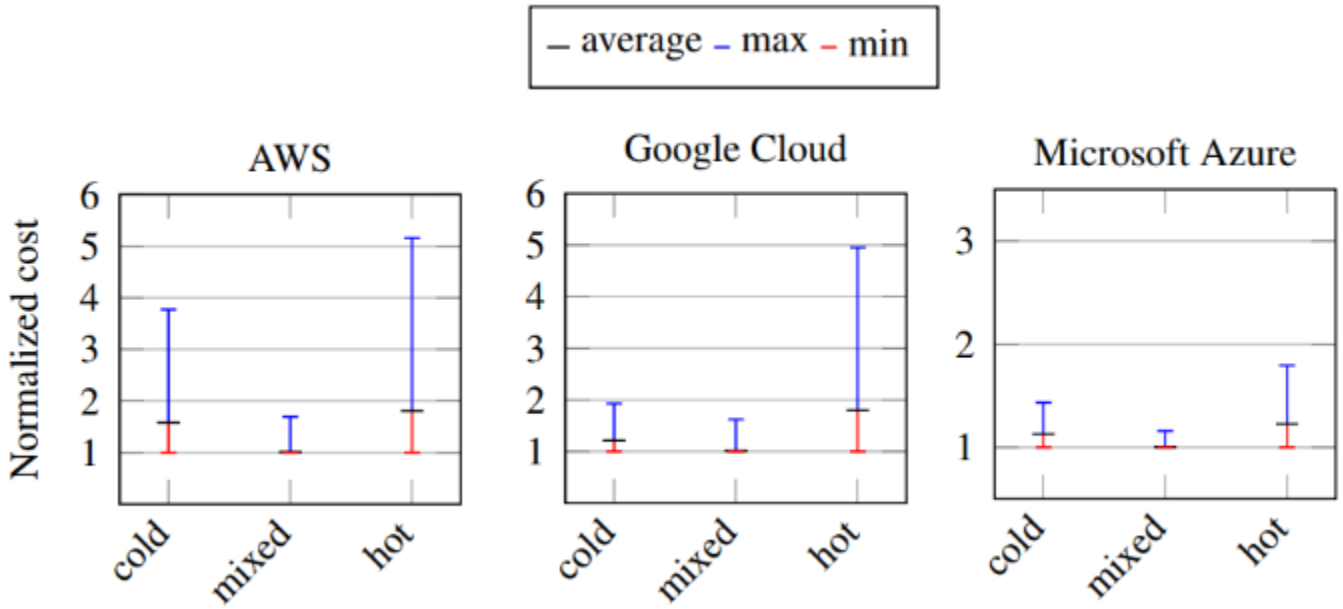
- For BatchGC, strategy 3(costBased) achieves the best result.



- Strategy 1 : onlyEmpty
- Strategy 2 : less than xx is live
- Strategy 3 : costBased

Evaluation : Mixing Storage Classes

- Mixing cold and hot storage reduces the costs for all three major providers



Conclusion

- InftyDedup, a novel system for cloud tiering with deduplication
  - solves the resource limitation of the local teir and insufficient scalability
- An algorithm for decreasing the financial cost of storing deduplicated data in the cloud tier
  - mixing cloud storage types can reduce costs