Async Inserts and Deduplication

Fei.Han@clickhouse.com

Inserts in ClickHouse

- 1 insert will generate at least 1 part.
- We recommend a insert contains at least 1k rows
  - Ideally 10k - 100k
- If inserts are too small, it’s easy to get ‘TOO_MANY_PARTS’ errors.

Work around for small inserts

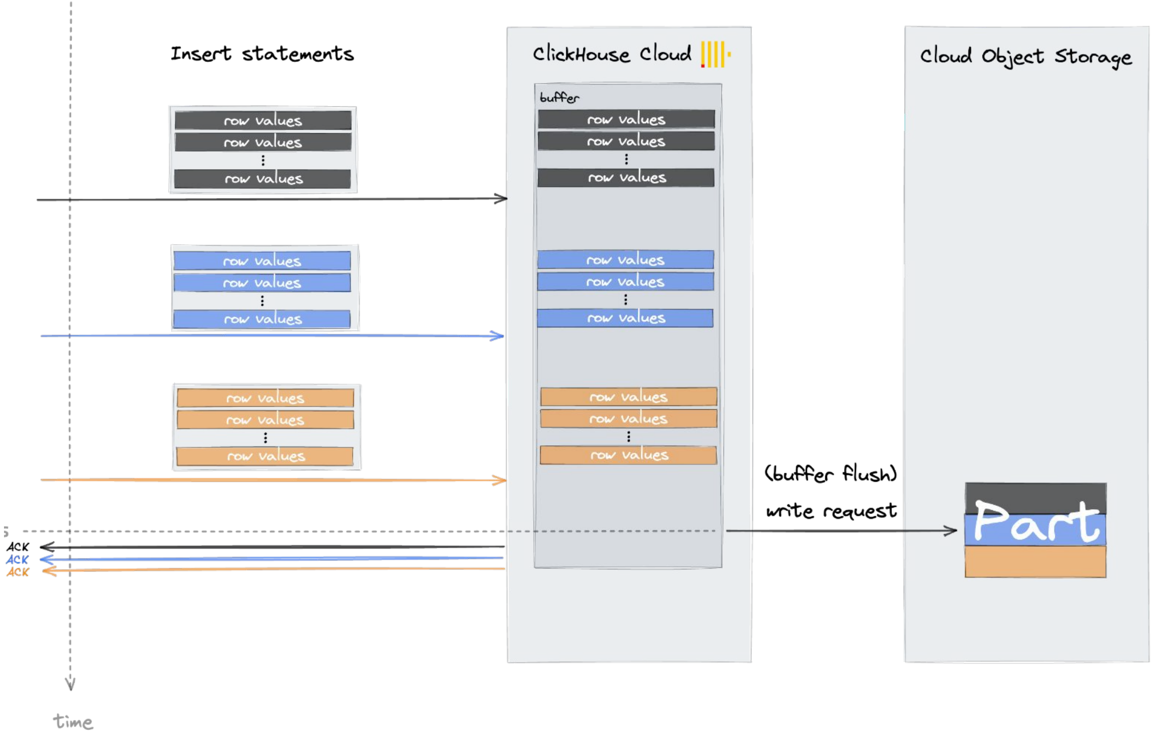
https://clickhouse.com/docs/en/optimize/bulk-inserts

- Use buffer table linking to another table
  - It’s possible to lose data if the buffer is not flushed and server crashes
- Batch data on client side
  - Need extra user code
  - Retries costs more

What is async insert ?

We launched “Async Insert” in ClickHouse v2021.11

- “Async insert”s *buffers* inserts inside server to make a larger block.
- Return ACK if and only if the insert is flushed to the storage.
- Support deduplicate since ClickHouse v2023.01



ClickHouse

How do we guarantee Idempotence?

Some possible solutions in other systems

- Streaming services
  - Assign unique id for network packets
- RDBMS
  - Primary keys / Unique Keys

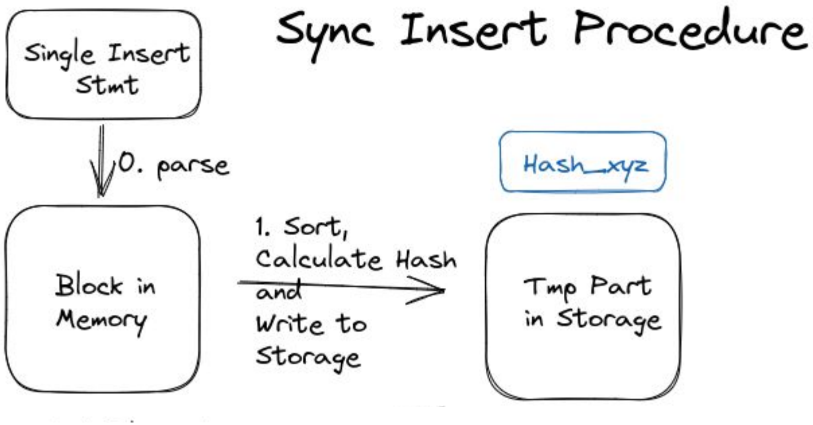
How do we guarantee Idempotence?

Some possible solutions in other systems

- Streaming services
  - Assign unique id for network packets
- RDBMS
  - Primary keys / Unique Keys

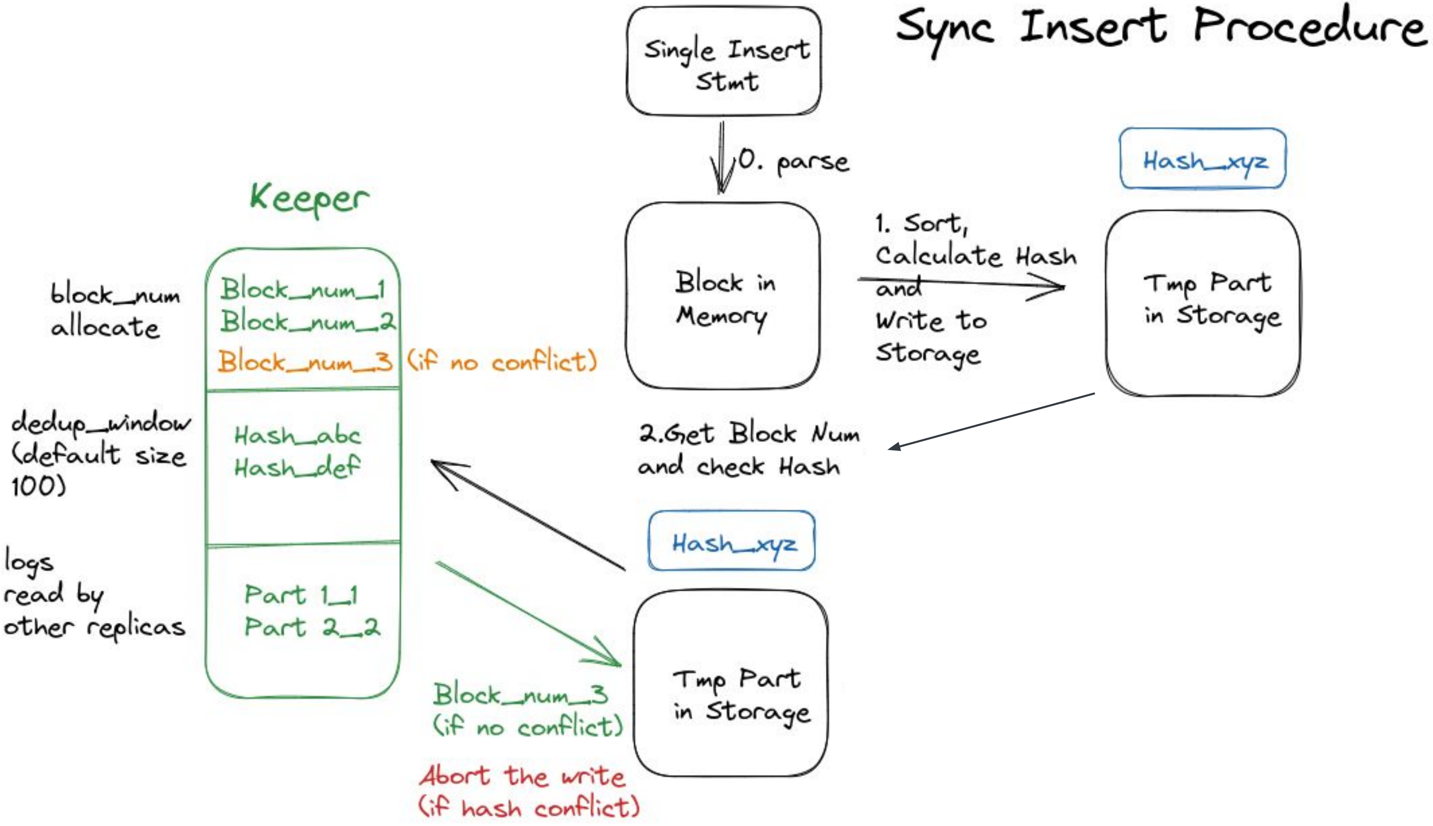
ClickHouse resolves the problem by:

- Calculating hash id for an insert
- Check if it conflicts with hash ids in the “dedup_window”
  - Commit write and write hash id to “dedup_window”, or
  - Abort write if conflicts

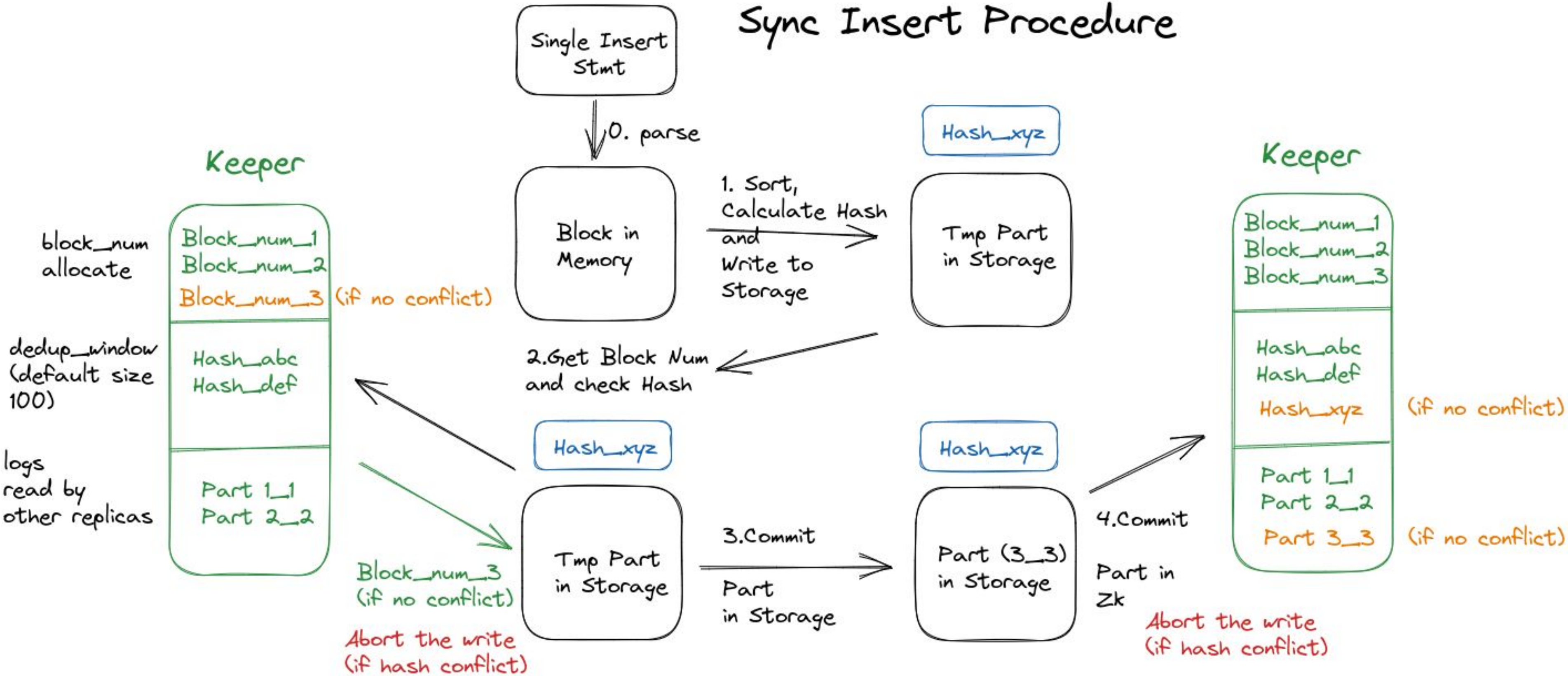


Sync Insert Procedure

Sync Insert Procedure


Single Insert Stmt
0. parse
Block in Memory
1. Sort, Calculate Hash ~~and~~ and Write to Storage
Hash_xyz
Tmp Part in Storage
2.Get Block Num and check Hash
Keeper
block_num allocate
Block_num_1
Block_num_2
Block_num_3 (if no conflict)
dedup_window (default size 100)
Hash_abc
Hash_def
logs read by other replicas
Part 1_1
Part 2_2
Hash_xyz
Tmp Part in Storage
Block_num_3 (if no conflict)
Abort the write (if hash conflict)

Sync Insert Procedure



Single Insert Stmt

0. parse

Block in Memory

1. Sort, Calculate Hash and Write to Storage

Hash_xyz

Tmp Part in Storage

2.Get Block Num and check Hash

Keeper

block_num allocate

Block_num_1
Block_num_2
Block_num_3 (if no conflict)

dedup_window (default size 100)

Hash_abc
Hash_def

logs read by other replicas

Part 1_1
Part 2_2

Block_num_3 (if no conflict)

Abort the write (if hash conflict)

Hash_xyz

Tmp Part in Storage

3.Commit

Part in Storage

Hash_xyz

Part (3_3) in Storage

4.Commit

Part in Zk

Abort the write (if hash conflict)

Keeper

Block_num_1
Block_num_2
Block_num_3

Hash_abc
Hash_def
Hash_xyz (if no conflict)

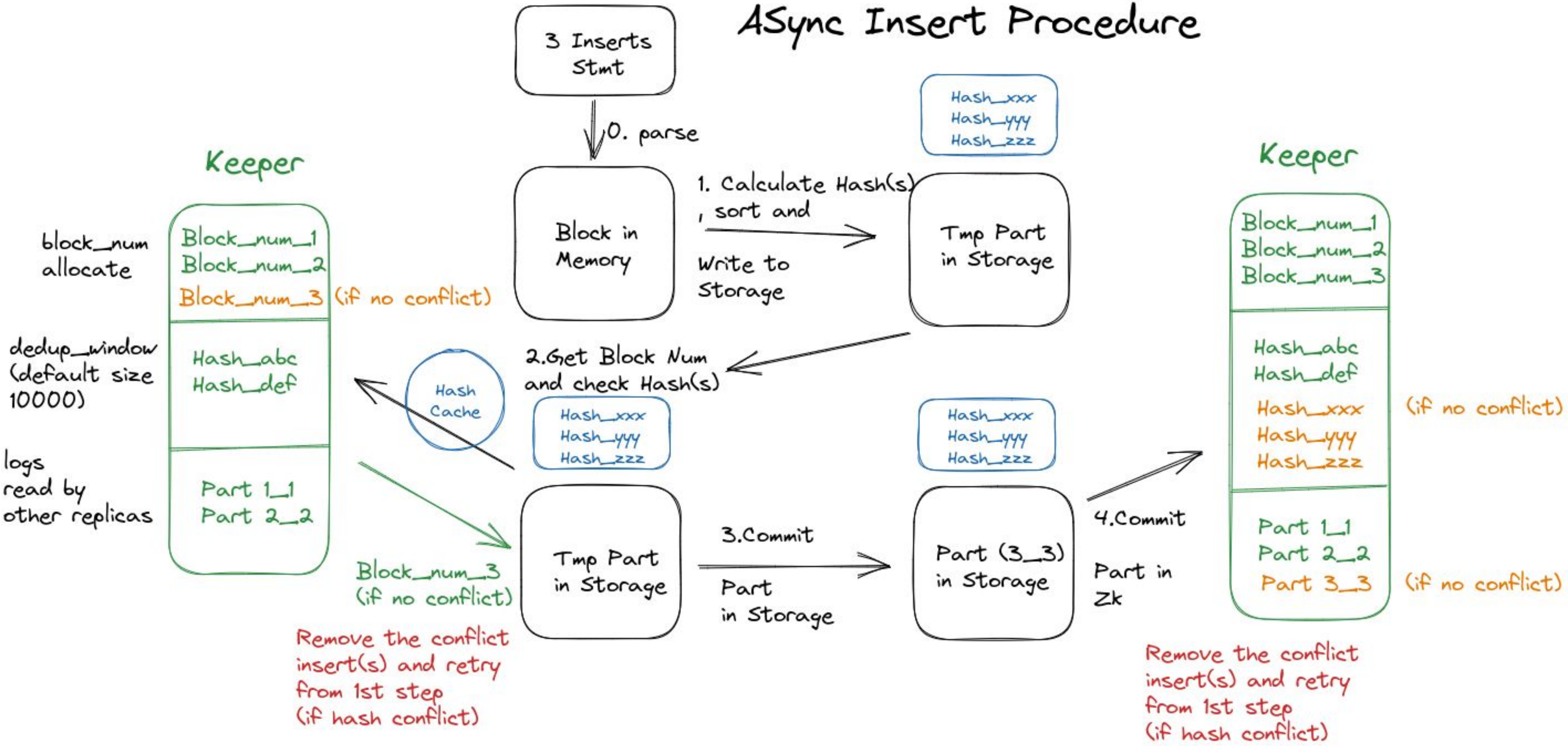
Part 1_1
Part 2_2
Part 3_3 (if no conflict)

ClickHouse

Problems with async inserts’ deduplication

- Multiple inserts in one block
    - And we need larger duplication window
- Should retry (partially) when hash id(s) conflict

ASync Insert Procedure



3 Inserts Stmt

0. parse

Block in Memory

1. Calculate Hash(s), sort and Write to Storage

Hash_xxx
Hash_yyy
Hash_zzz

Tmp Part in Storage

2.Get Block Num and check Hash(s)

Hash Cache

Hash_xxx
Hash_yyy
Hash_zzz

Keeper

block_num allocate

Block_num_1
Block_num_2
Block_num_3 (if no conflict)

dedup_window (default size 10000)

Hash_abc
Hash_def

logs read by other replicas

Part 1_1
Part 2_2

Block_num_3 (if no conflict)

Remove the conflict insert(s) and retry from 1st step (if hash conflict)

Tmp Part in Storage

3.Commit Part in Storage

Hash_xxx
Hash_yyy
Hash_zzz

Part (3_3) in Storage

4.Commit Part in Zk

Keeper

Block_num_1
Block_num_2
Block_num_3

Hash_abc
Hash_def
Hash_xxx
Hash_yyy
Hash_zzz
(if no conflict)

Part 1_1
Part 2_2
Part 3_3
(if no conflict)

Remove the conflict insert(s) and retry from 1st step (if hash conflict)

Simple Benchmark

Insert 1,000,000 integers, 200 per insert, 100 threads

benchcode

Percentage of deduplication	Async (deduplicate=0)	Async (deduplicate=1)	Async (wait_for_async_insert=0 deduplicate=1)
0%	20.070	20.116	8.780
5%		21.221	9.359
10%		22.858	9.790
20%		24.092	10.614
40%		27.972	12.435
80%		36.483	15.918

The End

Thank you!

ClickHouse