TiDB Lightning

huachao@pingcap.com

About me

- TiKV Engineer
- Wechat / Nice / PingCAP
- Work on distributed database, but ...

What is TiDB?

TiDB is an open source distributed NewSQL hybrid transactional and analytical processing (HTAP) database built by PingCAP.

What is TiDB Lightning?

A tool to import data into TiDB:

- 10x speed up
- more predictable performance
- support mydumper input format now, and other formats like CSV in the future

How we import data into TiDB before?

Mydumper / Myloader

Mydumper

=>

INSERT INTO `table` VALUES (...)

=>

Myloader / Loader

=>

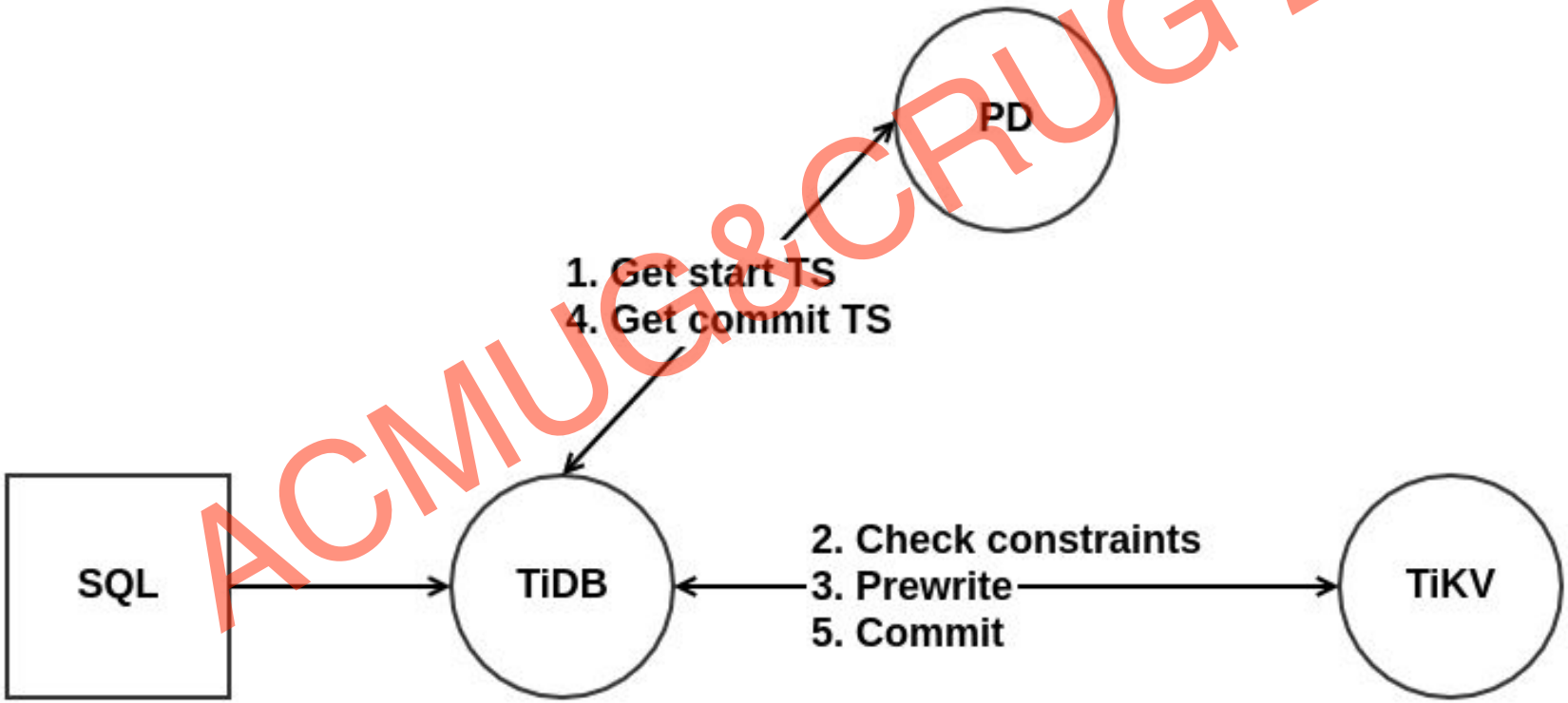
TiDB

How does TiDB execute INSERT?

INSERT INTO `table` VALUES (...)

1. AST / Logical Plan / Physical Plan / Executor
2. Start a transaction (with a start timestamp)
3. Check constraints (PRIMARY KEY, UNIQUE INDEX …)
4. Encode record/index KVs
5. Prewrite
6. Commit (with a commit timestamp)

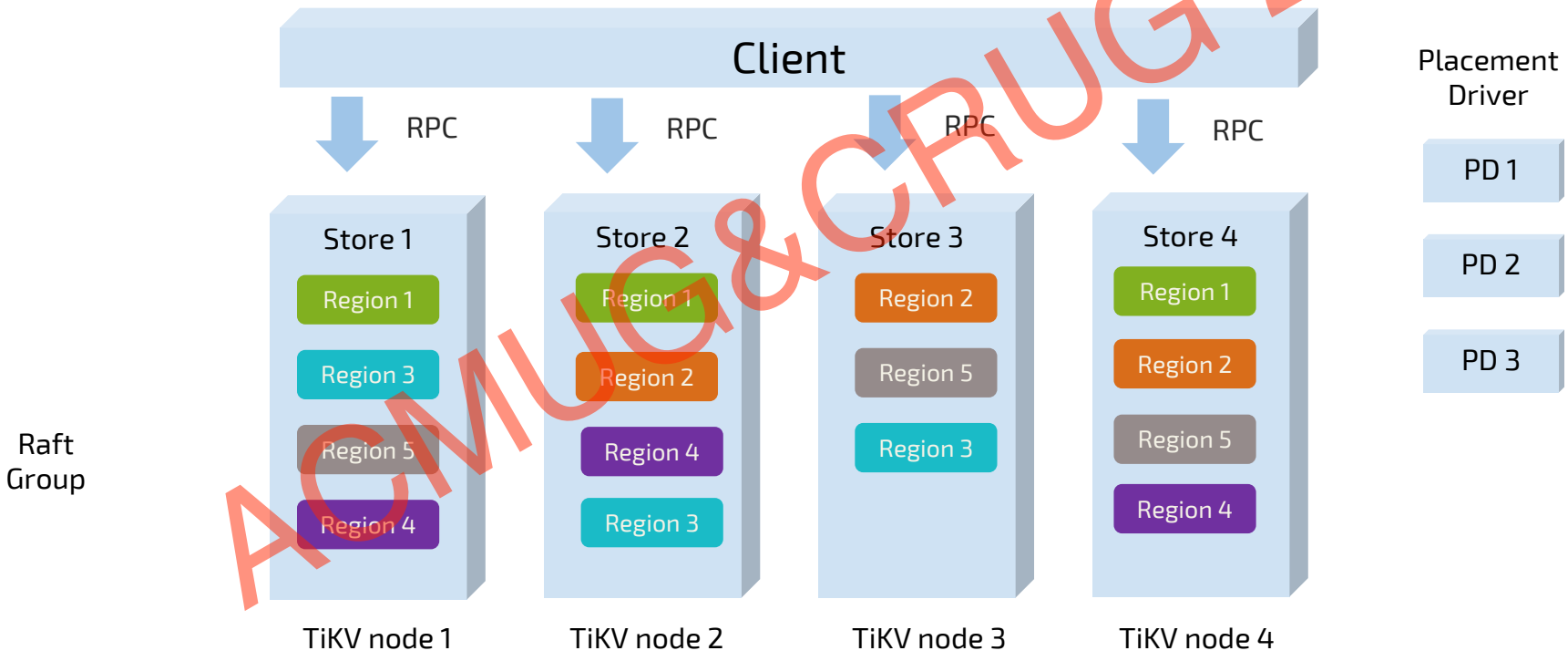
How does TiDB execute INSERT?



What's the bottleneck here?

That’s all? Not even close …

TiKV Architecture



What should we consider here?

1. Consider a table with AUTO_INCREMENT PK
2. Raft replication
3. Region split
4. Region balance
5. RocksDB compaction
6. RocksDB rate limit / stall

How we import data with Lightning?

Skip Transaction (Convert SQL to KV)

INSERT INTO `table` VALUES (...)

- Record:
- t_{table_id}_r_{handle_id} => {column*}
- Unique Index:
- t_{table_id}_i_{index_id}_{column*} => handle_id
- Non Unique Index:
- t_{table_id}_i_{index_id}_{column*}_{handle_id} => {}

Split Region / Scatter Region

- Pre-split region to avoid data scan
- Pre-scatter region to avoid snapshot, data cleanup and hotspot

Ingest SST file into RocksDB

- No overlap between different SST files in a batch
- Avoid unnecessary compaction and write amplification
- Reduce Raft CPU consumption

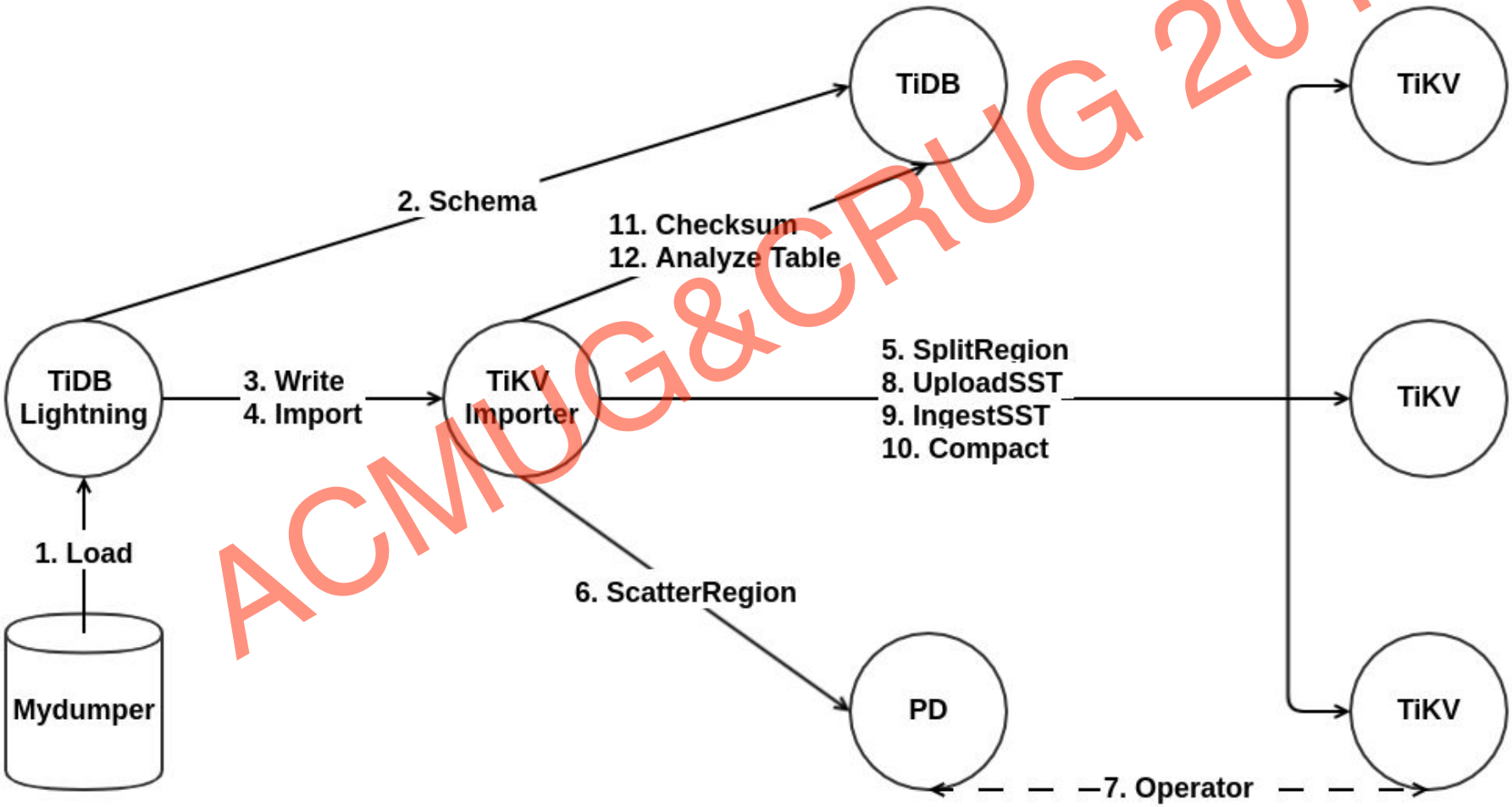
How to Import an SST file into a Region?

- Send the SST to the region leader
- Rely on Raft log to replicate the SST to followers
- When the log entry has been committed, ingest the SST from the log to RocksDB
- But the SST can be large, which will block the Raft process

Import an SST file with two steps

- Upload the SST to all peers of the region, with the file checksum and the region epoch
- Issue an IngestSST Raft command with the SST metadata to the region
- When the command has been committed, ingest the uploaded SST file to RocksDB
- Rely on the checksum and the region epoch to guarantee data safety.
- The Raft log is small and the CPU consumption is low.

TiDB Lightning Architecture



What are the difficulties here?

The bottleneck

- Not IO, not network, it’s CPU
- KVEncoder in TiDB Lightning
- Produce KV pairs in tens of MB/s with 32 cores

The workaround

- Optimize KVEncoder
- Split import task into batches, to cut the cost of TiKV Importer

The bottleneck-2

- KV pairs in a batch can be sorted, so the generated SST files will not overlap with each other.
- However, SST files in different batches can overlap, which can result in RocksDB compaction and rate limit.

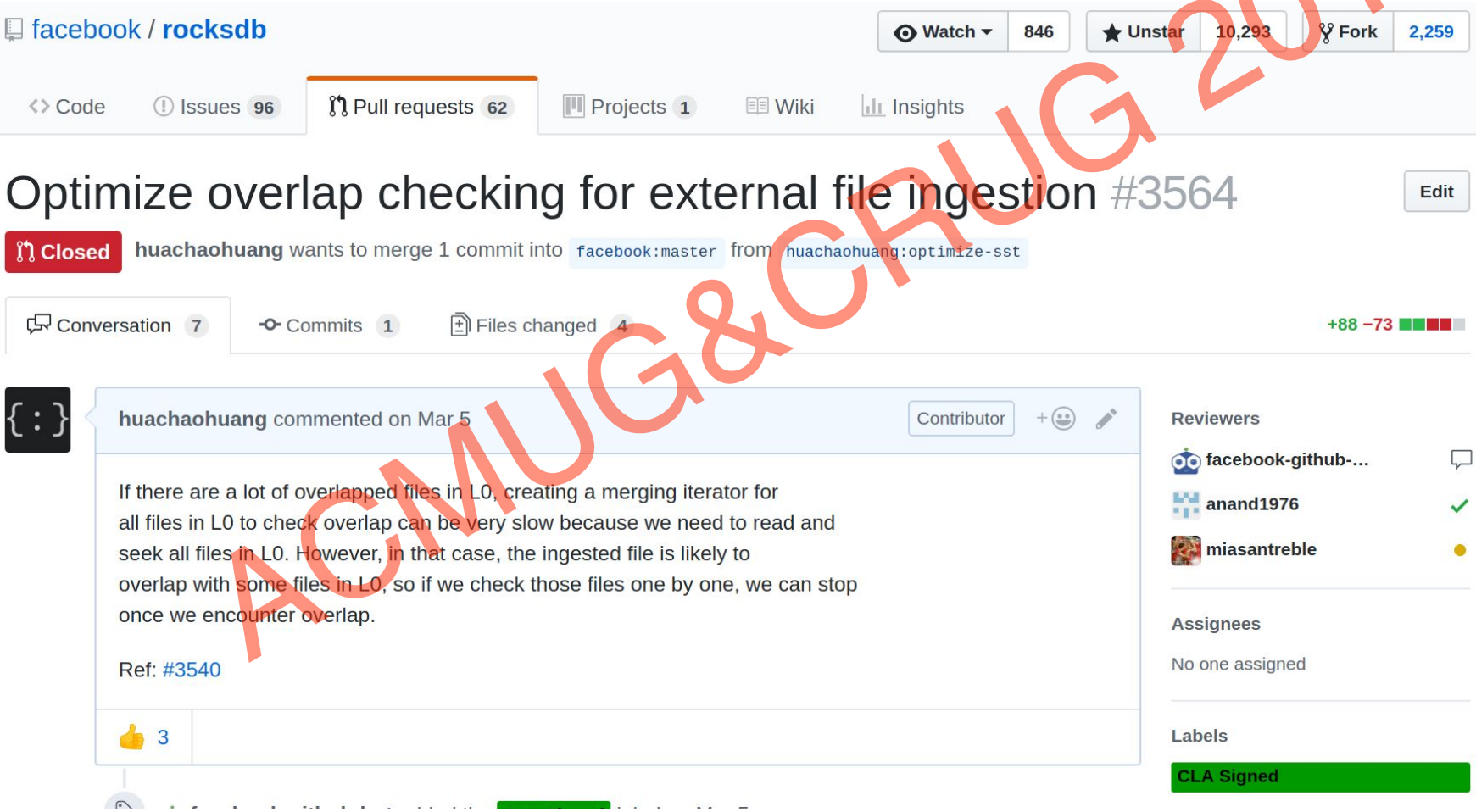
The workaround-2

- Since we can not avoid the overlap, let’s face it
- Just ingest all SST files in level-0, and do a full compaction in the end

The bottleneck-3

If there are a lot of overlapped files in L0, creating a merging iterator for all files in L0 to check overlap can be very slow because we need to read and seek all files in L0.

The workaround-3



facebook / rocksdb

Watch 846 | Unstar 10,293 | Fork 2,259

Code | Issues 96 | Pull requests 62 | Projects 1 | Wiki | Insights

Optimize overlap checking for external file ingestion #3564

Edit

Closed **huachaohuang** wants to merge 1 commit into `facebook:master` from `huachaohuang:optimize-sst`

Conversation 7 | Commits 1 | Files changed 4 | +88 −73

huachaohuang commented on Mar 5 — Contributor

If there are a lot of overlapped files in L0, creating a merging iterator for all files in L0 to check overlap can be very slow because we need to read and seek all files in L0. However, in that case, the ingested file is likely to overlap with some files in L0, so if we check those files one by one, we can stop once we encounter overlap.

Ref: #3540

3

Reviewers

facebook-github-...

anand1976

miasantreble

Assignees

No one assigned

Labels

CLA Signed

Another interesting optimization

- How to process a large sorted dataset concurrently?
- We need to cut the dataset into approximately equal ranges.
- We can scan the whole dataset first with a single thread to collect the split points.
- But that's slow, and can be even slower if the dataset is compressed, because the CPU will be the bottleneck.

The size table properties

- We can add a table properties collector to RocksDB to collect sample points of the dataset
- When we need to split regions into approximately equal ranges, we can just calculate the boundaries from the size properties, which is very lightweight.

So, that's all? Almost, until ...

A dinner conversation

A => Me, B => Business team

B: Hey, how does the lightning go?

A: It’s almost done, we can use it soon.

B: Awesome, how does it work?

A: ^%^#$%#$%&&(&(*^%^

B: All right, I just care about how to guarantee that it won’t miss any data?

A: Orz, let’s finish the dinner first.

How to guarantee no data loss?

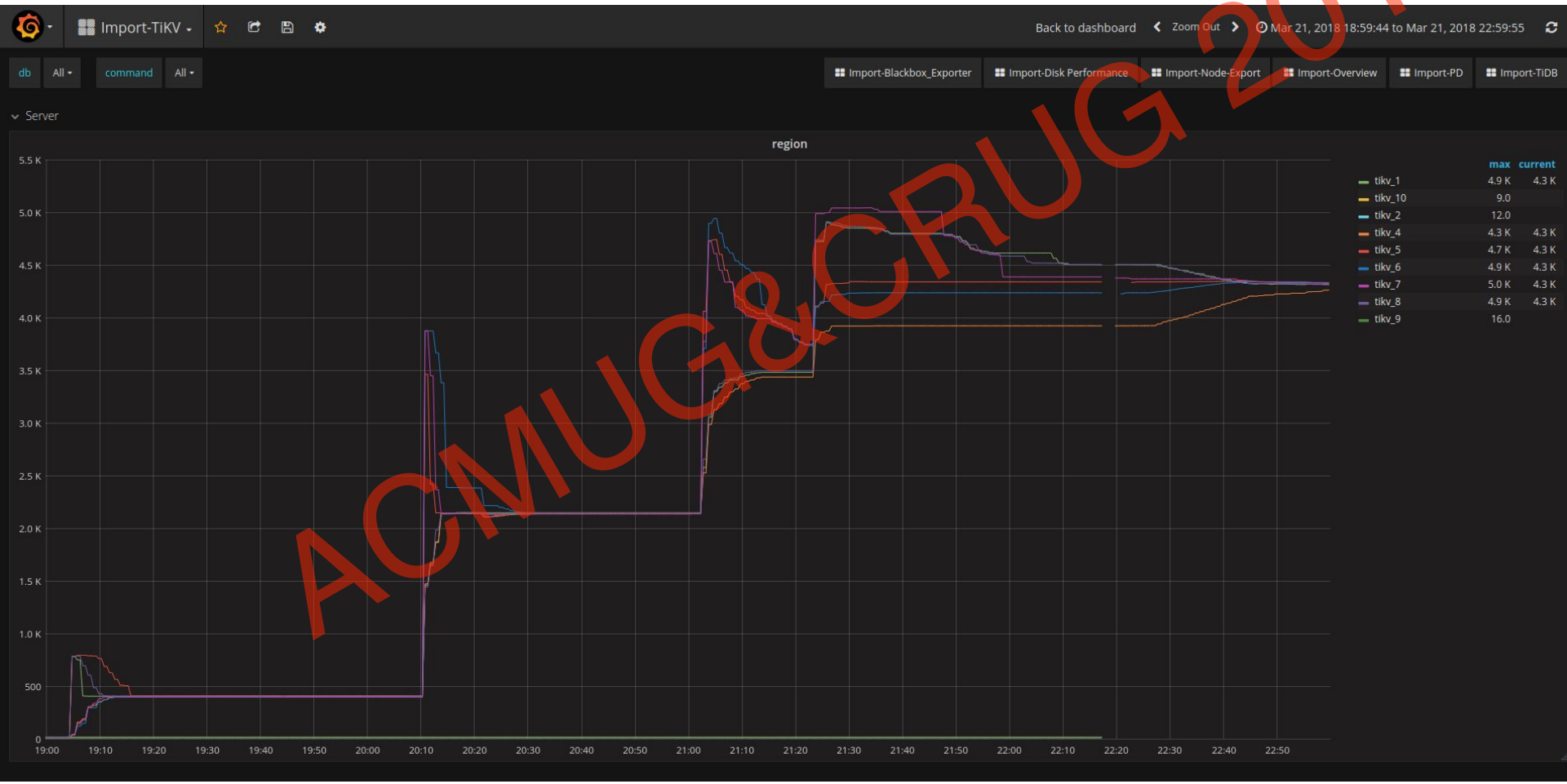
- We have ADMIN CHECK TABLE, but that can only check data consistency, and it's a bit slow.
- We have SELECT count(*), but that can only check the number of rows.
- We can scan rows from the source and check them with the target one by one, but that speed is unacceptable with a large dataset.

ADMIN CHECKSUM TABLE

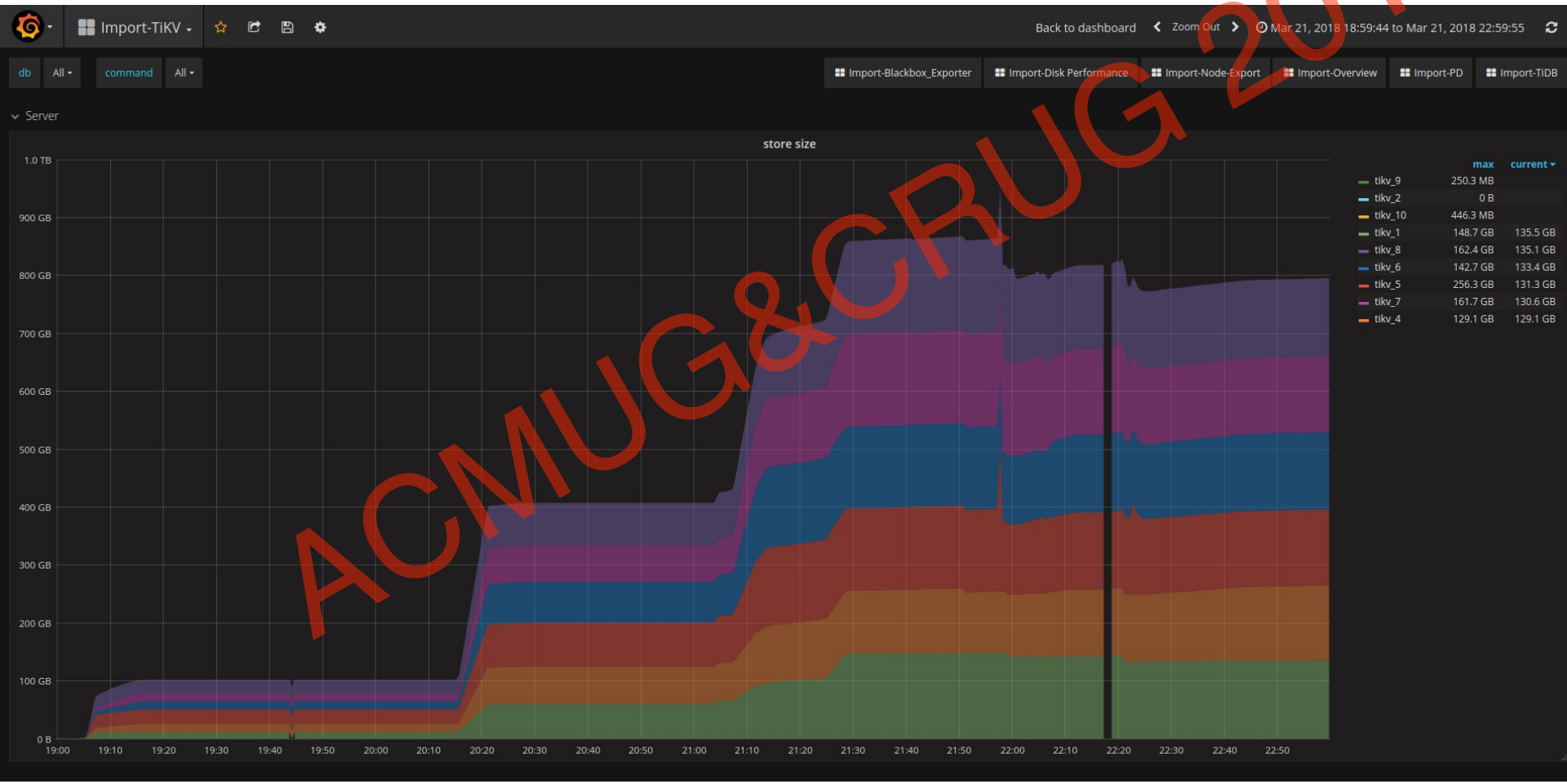
- Calculate a source checksum from the KV pairs converted from the source data.
- Calculate a target checksum from the KV pairs from the target data.
- Utilize the coprocessor framework to make it as fast as SELECT count(*).
- Use an order independent checksum algorithm for concurrency.
- The current algorithm: checksum = checksum ^ crc64(key + value)

All right, let's import 655GB into TiDB

Split Region / Scatter Region



Import SST files into TiKV



That’s all? Yep, for now :)