RocksDB Festival

RF5_Team_LayOut

Supported by IITP, StarLab.

August 09, 2021

Minguk Choi, Jungwon Lee, Guangxun shin
koreachoi96@gmail.com, gardenlee960828@gmail.com, guangxun0621@naver.com

Docks

Before Start…

History / BlobDB

Revisions

- Updated BlobDB (markdown)
  Itamasi committed 18 days ago
- Updated BlobDB (markdown)
  Itamasi committed on 27 May
- Updated Blob DB (markdown)
  Itamasi committed on 7 Apr 2020

RocksDB features, and required users to adopt a custom API. In 2020, we decided to rearchitect BlobDB from the ground up, taking the lessons learned from WiscKey and the original BlobDB but also drawing inspiration and incorporating ideas from other similar systems. Our goals were to eliminate the above

- BlobDB is
  - ✓ updated at May 27
  - ✓ Similar to Wisckey(2016), but difference exists
  - ✓ Insufficient explanation, studied with codes and comments
- We are not sure…

RocksDB Festival

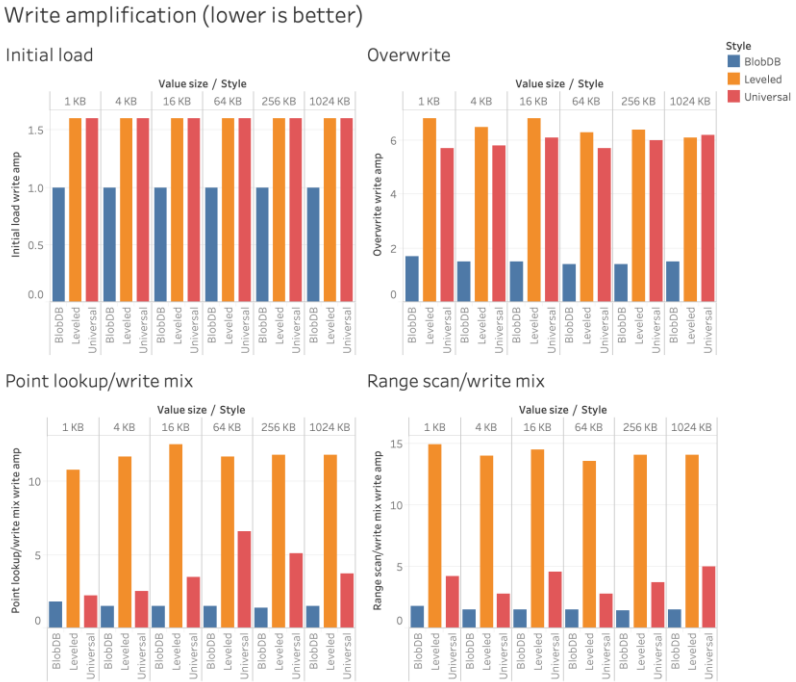
- 1. Performance (vs RocksDB)
  - ✓ Write amplification
  - ✓ Write – Inital load / Overwrite
  - ✓ Read – Point lookups / Range Scan
- 2. Trade-off
  - ✓ Write/Read/Space
- 3. Garbage Collection
  - ✓ RocksDB
  - ✓ Titan
  - ✓ DiffKV
- 4. Future Work

Performance

- 0. Workload
  - ✓ BlobDB vs RocksDB (leveled/universal compaction)
  - ✓ Value Size
    - 1KB/4KB/16KB/64KB/256KB/1MB
  - ✓ Workload
    - Initial load
    - Overwrite
    - Point lookup/write mix
    - Range scan/write mix
    - Point lookups (read-only)
    - Range scans (read-only)

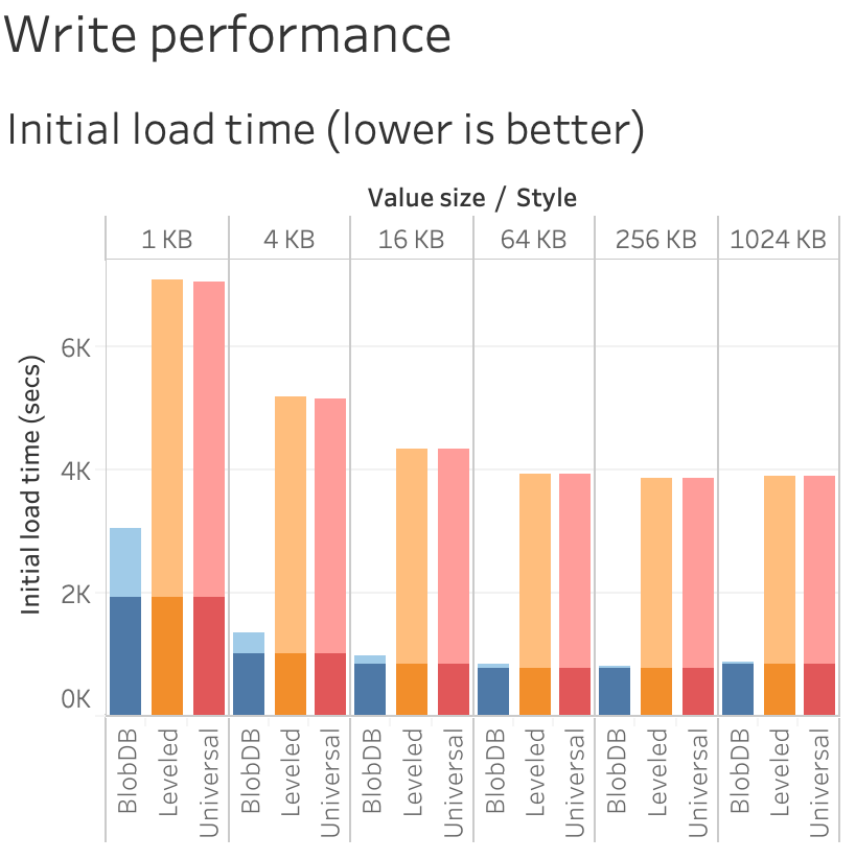
Performance

- 1. Write Amplification
  - ✓ Write Amplification = $\frac{\textit{total amount of data written by flushes and compactions}}{\textit{the amount of data written by flushes}}$
  - ✓ Way better than vanilla RocksDB
    - by avoid copying the values over and over again during compaction



Performance

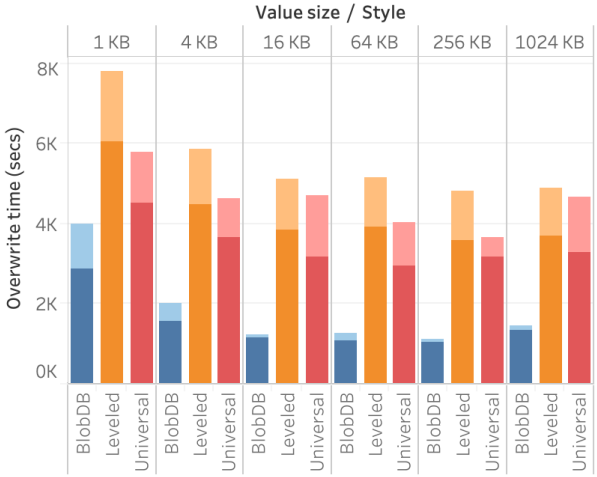
- 2. Write - Initial load
  - ✓ Way better than RocksDB



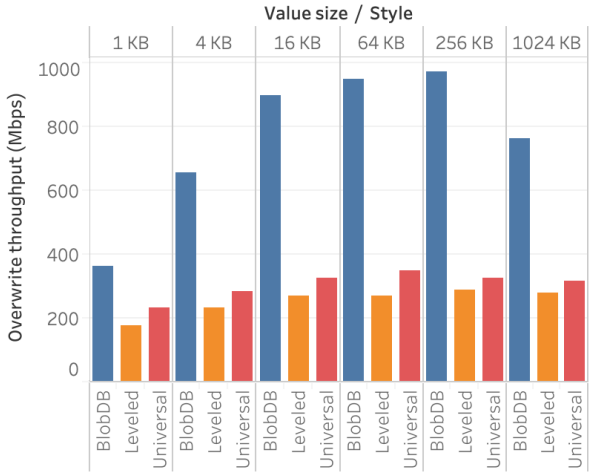
Performance

- 2. Write – Overwrite
  - ✓ Way better than RocksDB
    - throughput 2.1x ~ 3.5x higher than leveled compation
      1.6x ~ 3.0x higher than universal
  - ✓ Write stall
    - When: *Estimated pending compaction bytes > soft/hard_pending_compaction_bytes*
    - compactions can't keep up with the write rate

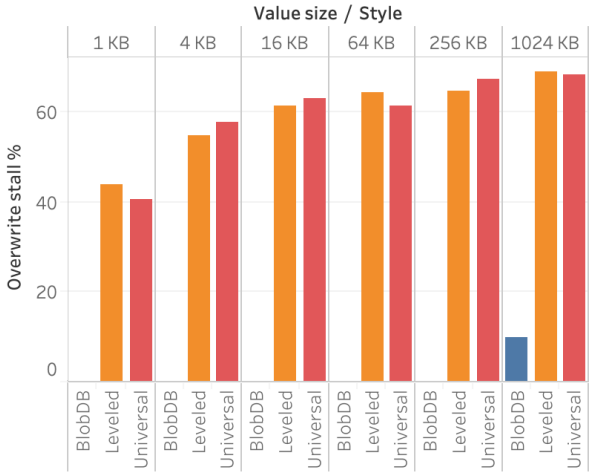
Overwrite time (lower is better)



Overwrite throughput (higher is better)

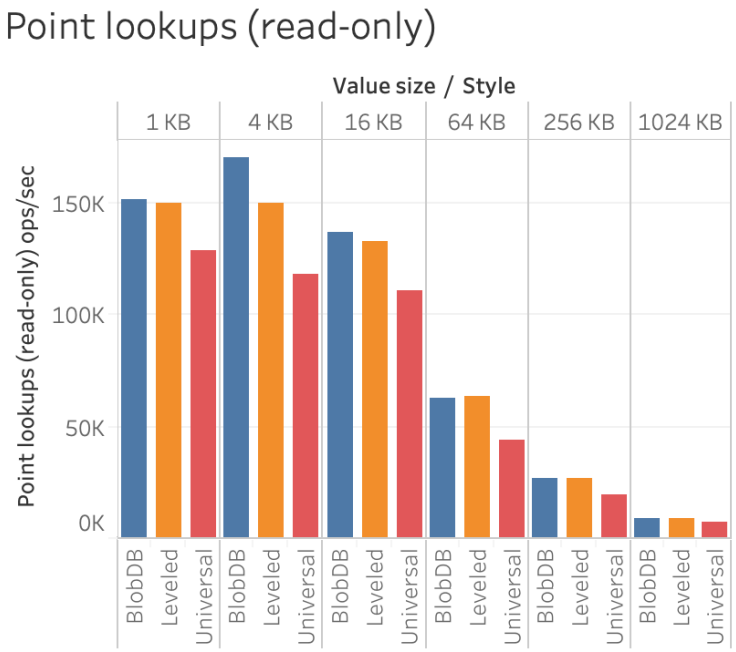


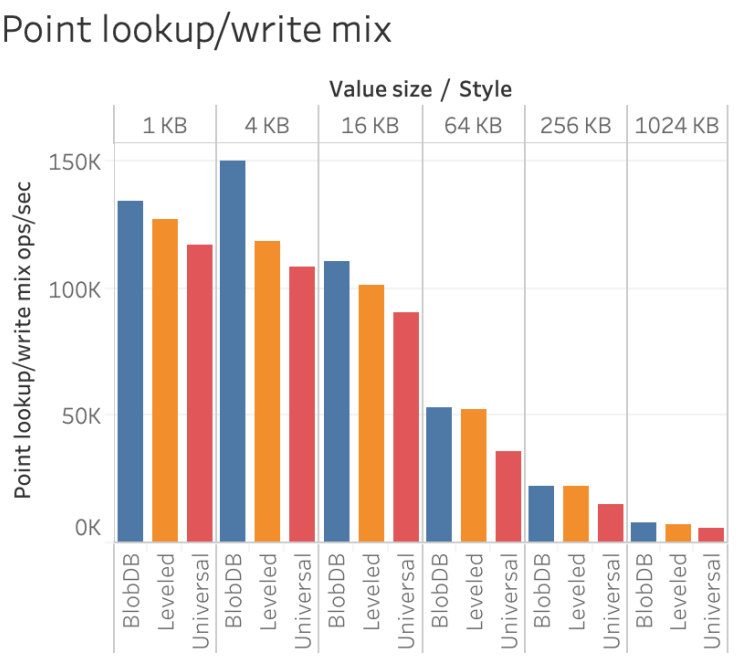
Overwrite stall % (lower is better)



Performance

- 3. Read - Point lookups
  - ✓ Meets/Exceeds RocksDB

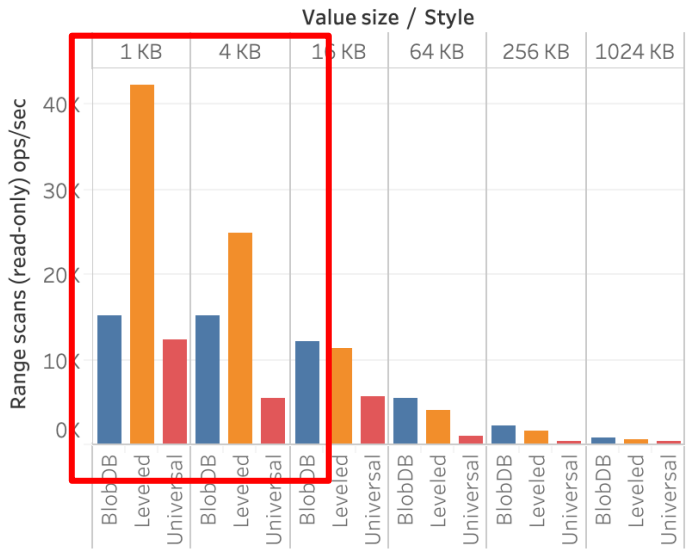




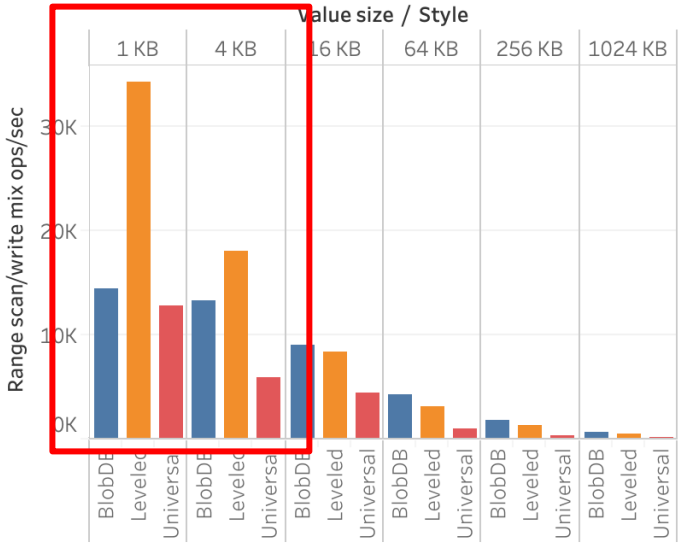
Performance

- 3. Read - Range scan
  - ✓ Way worse in small values (1KB / 4KB)
    - Values in blob file stored in random, not sequential -> low space locality
  - ✓ Meets/Exceeds in large values (16KB/64KB/256KB/1MB)
    - Space locality doesn't matter anymore

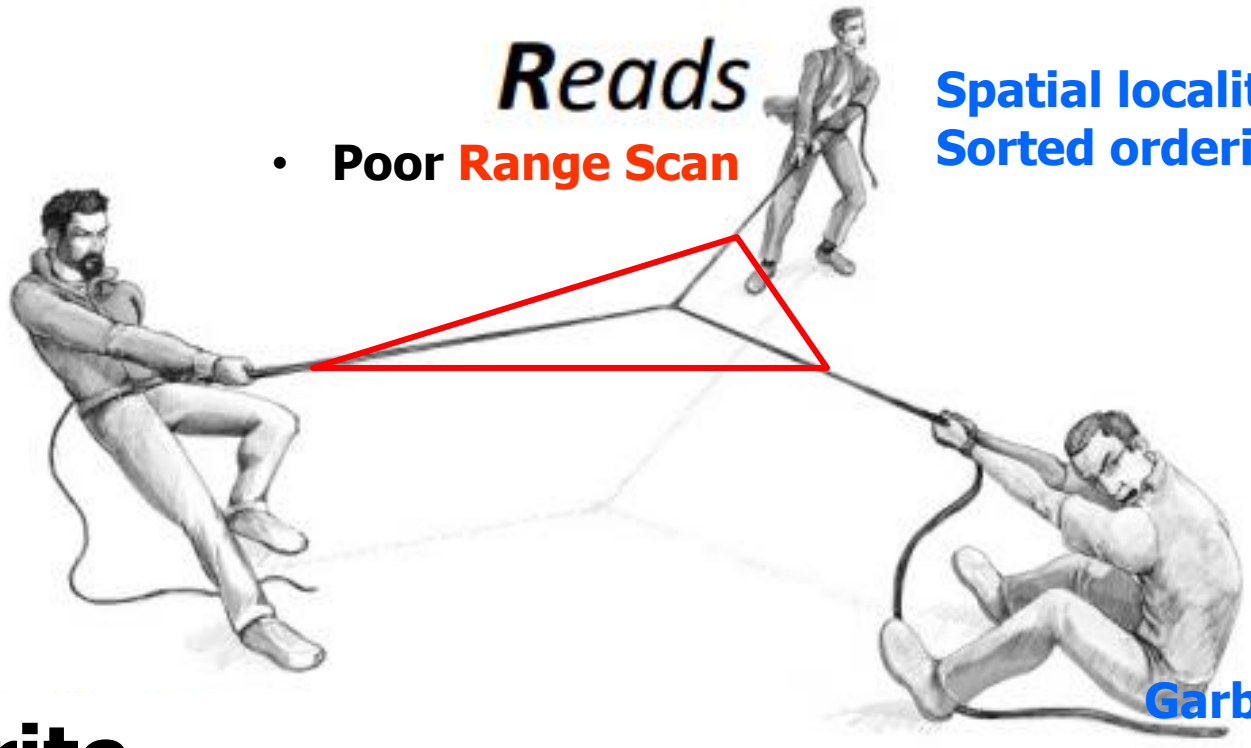
Range scans (read-only)



Range scan/write mix



Trade-Off



Write

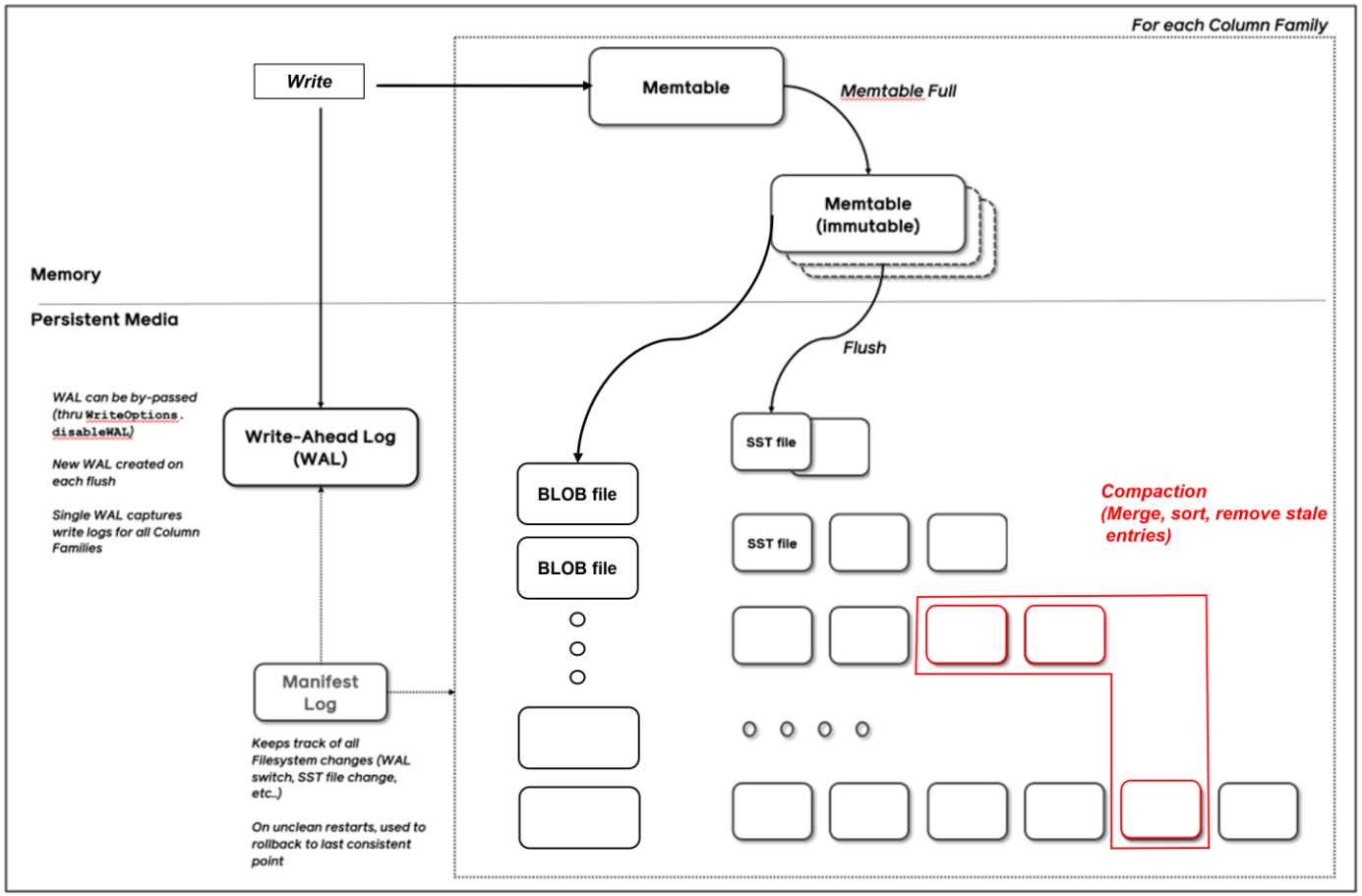
- High Write performance
- Low write amplification

Storage

- High space Amplification

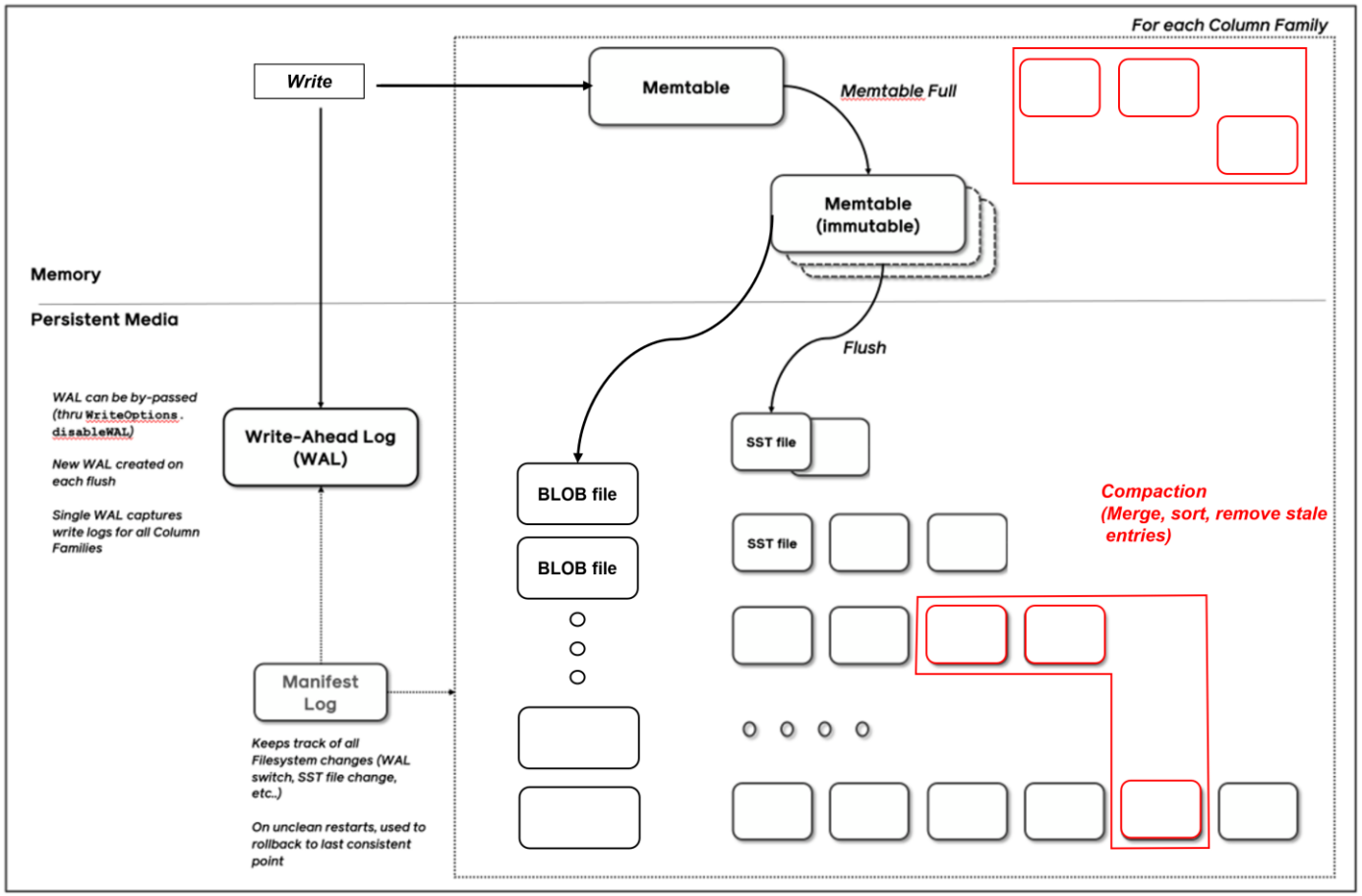
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



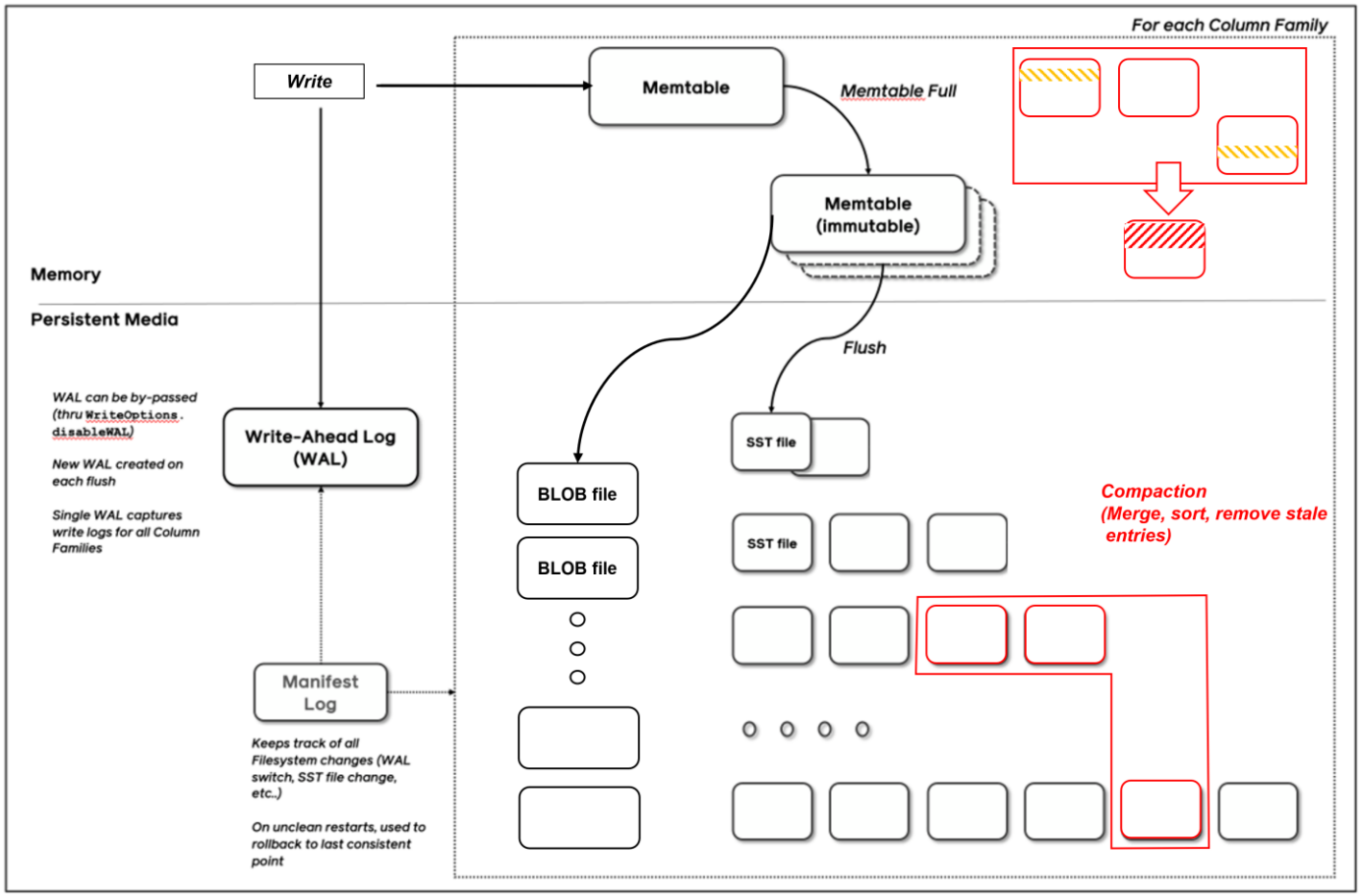
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



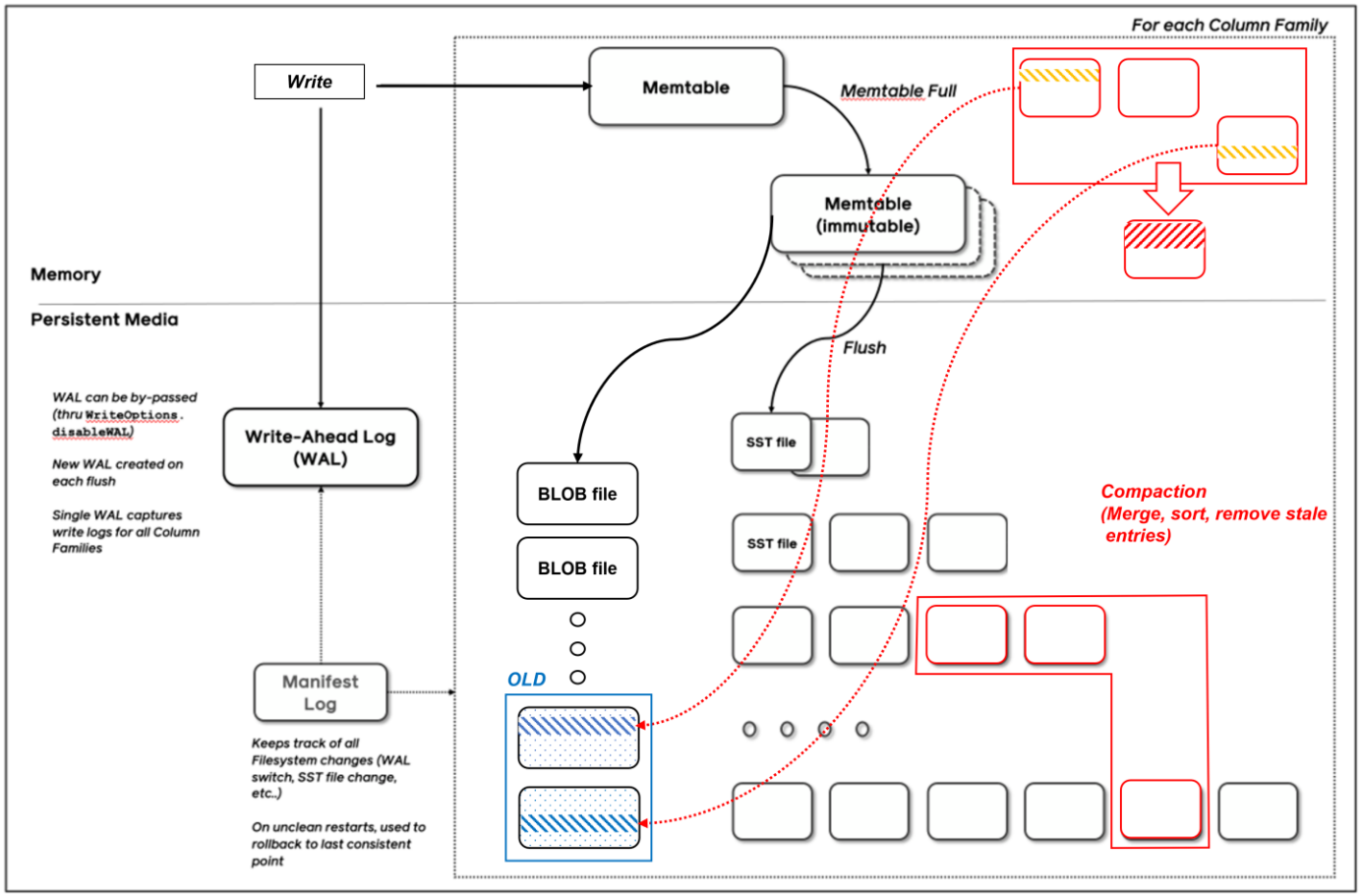
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



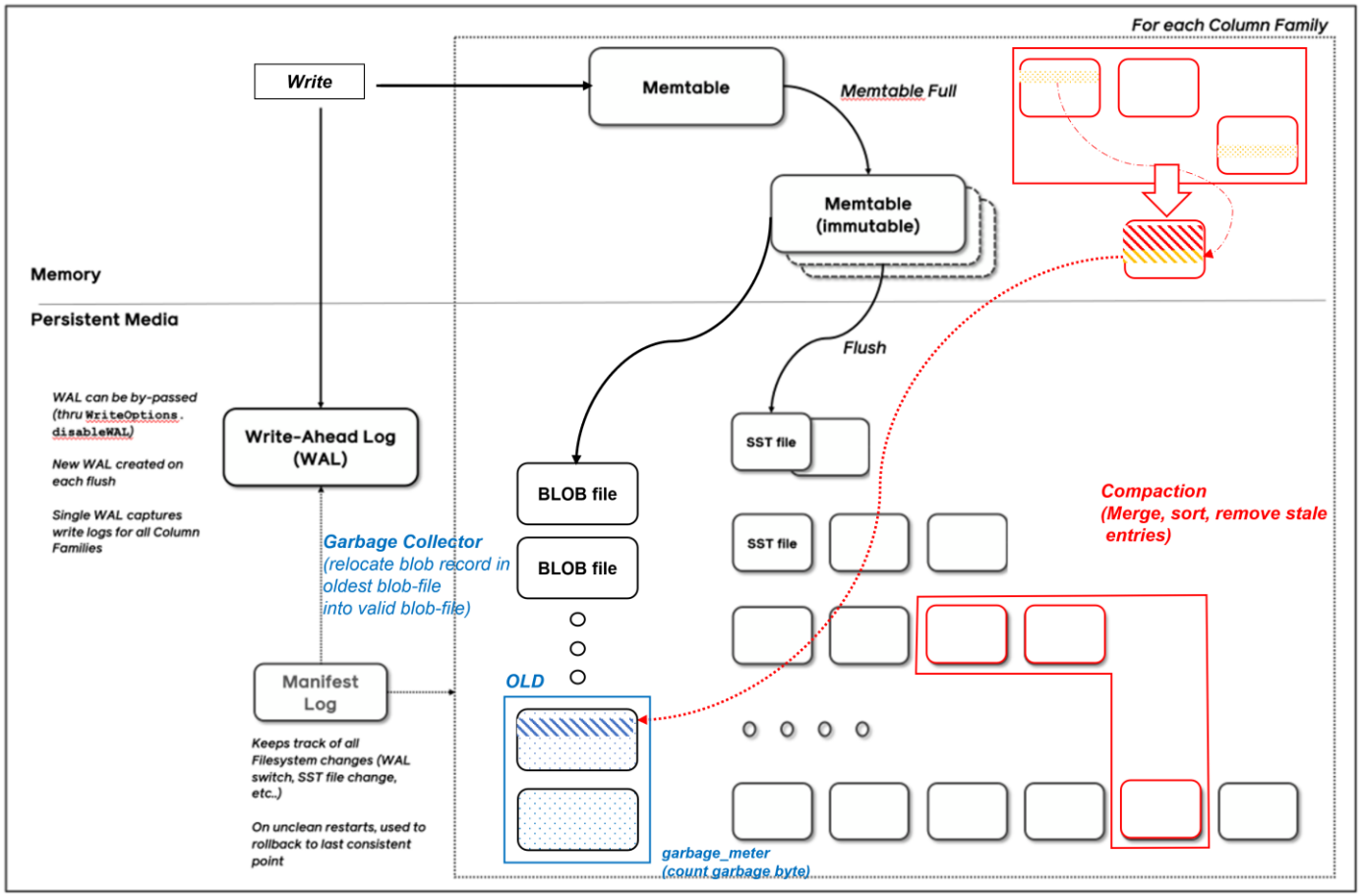
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



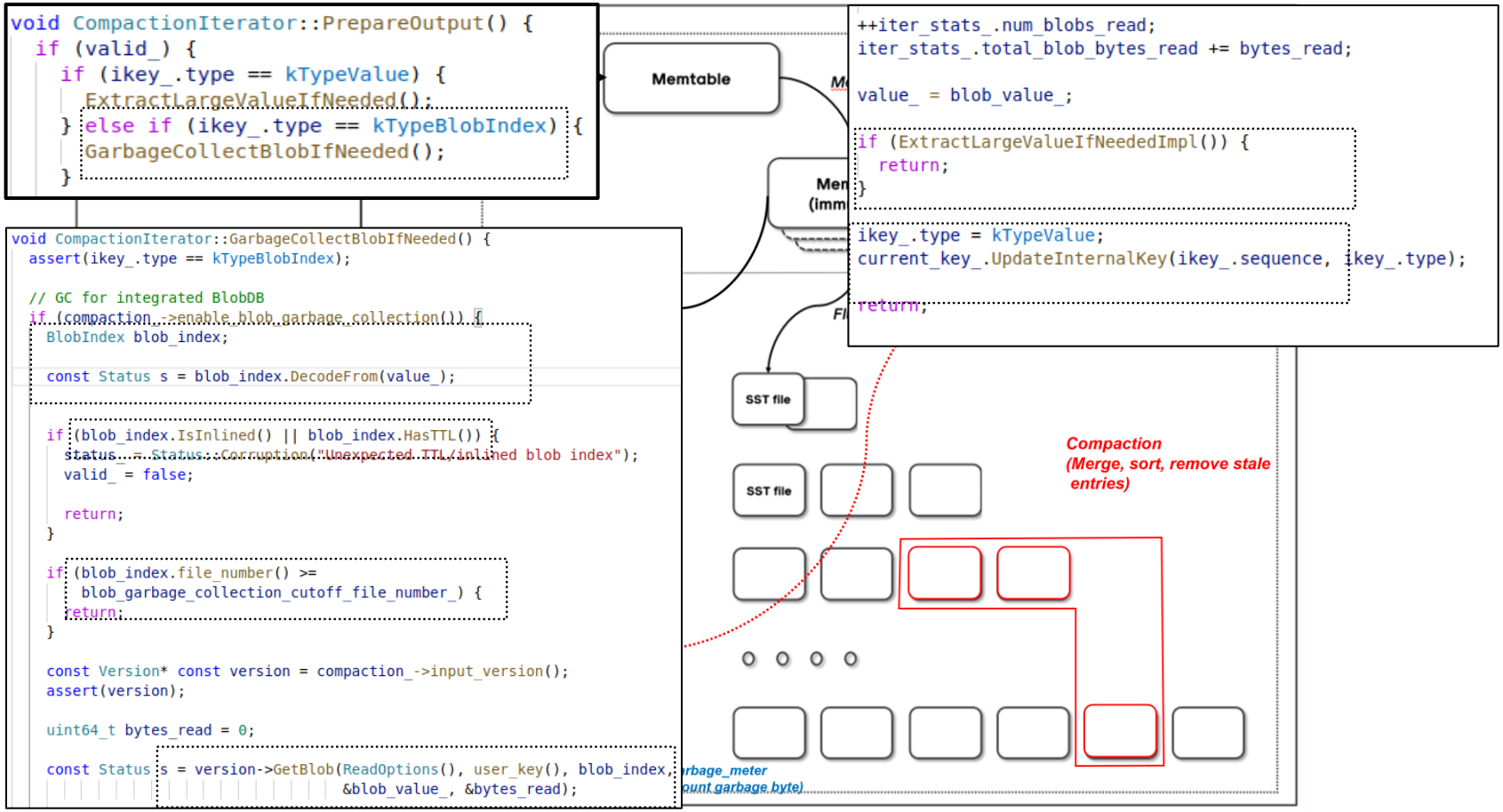
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files

```
void CompactionIterator::PrepareOutput() {
  if (valid_) {
    if (ikey_.type == kTypeValue) {
      ExtractLargeValueIfNeeded();
    } else if (ikey_.type == kTypeBlobIndex) {
      GarbageCollectBlobIfNeeded();
    }
```

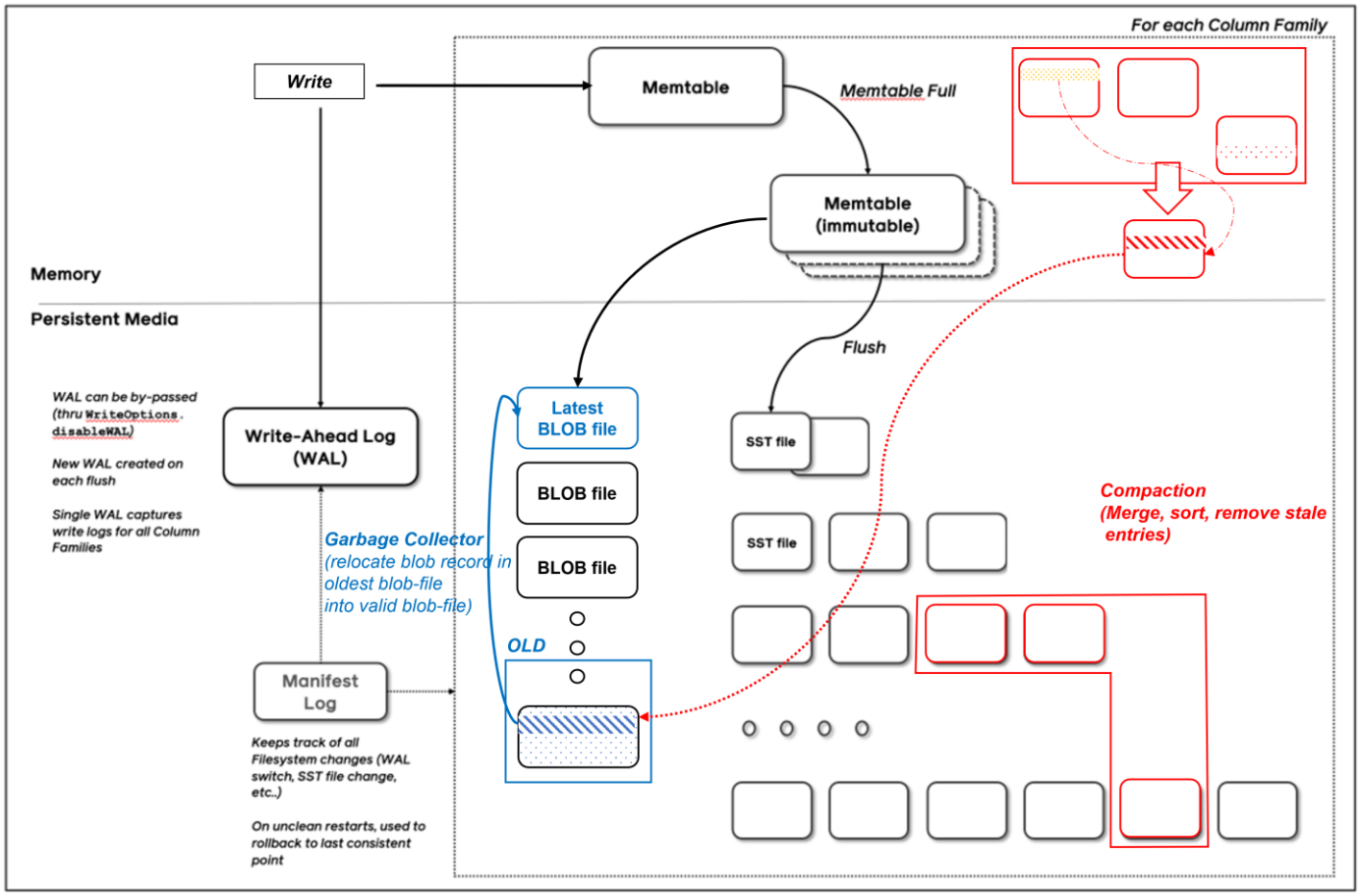
```
void CompactionIterator::GarbageCollectBlobIfNeeded() {
  assert(ikey_.type == kTypeBlobIndex);

  // GC for integrated BlobDB
  if (compaction_->enable_blob_garbage_collection()) {
    BlobIndex blob_index;

    const Status s = blob_index.DecodeFrom(value_);

    if (blob_index.IsInlined() || blob_index.HasTTL()) {
      status_ = Status::Corruption("Unexpected TTL/inlined blob index");
      valid_ = false;

      return;
    }

    if (blob_index.file_number() >=
        blob_garbage_collection_cutoff_file_number_) {
      return;
    }

    const Version* const version = compaction_->input_version();
    assert(version);

    uint64_t bytes_read = 0;

    const Status s = version->GetBlob(ReadOptions(), user_key(), blob_index,
                                      &blob_value_, &bytes_read);
```

```
++iter_stats_.num_blobs_read;
iter_stats_.total_blob_bytes_read += bytes_read;

value_ = blob_value_;

if (ExtractLargeValueIfNeededImpl()) {
  return;
}

ikey_.type = kTypeValue;
current_key_.UpdateInternalKey(ikey_.sequence, ikey_.type);

return;
```



4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



4. Garbage Collector

- Relocate where?
  - ✓ New blob file? Latest blob file?
    - May be… Latest blob file.

```
bool CompactionIterator::ExtractLargeValueIfNeededImpl() {
  if (!blob_file_builder_) {
    return false;
  }

  blob_index_.clear();
  const Status s = blob_file_builder_->Add(user_key(), value_, &blob_index_);
```

```
Status BlobFileBuilder::Add(const Slice& key, const Slice& value,
                            std::string* blob_index) {
  assert(blob_index);
  assert(blob_index->empty());

  if (value.size() < min_blob_size_) {
    return Status::OK();
  }

  {
    const Status s = OpenBlobFileIfNeeded();
    if (!s.ok()) {
      return s;
    }
  }

  *blob_file_number = writer_->get_log_number();
```

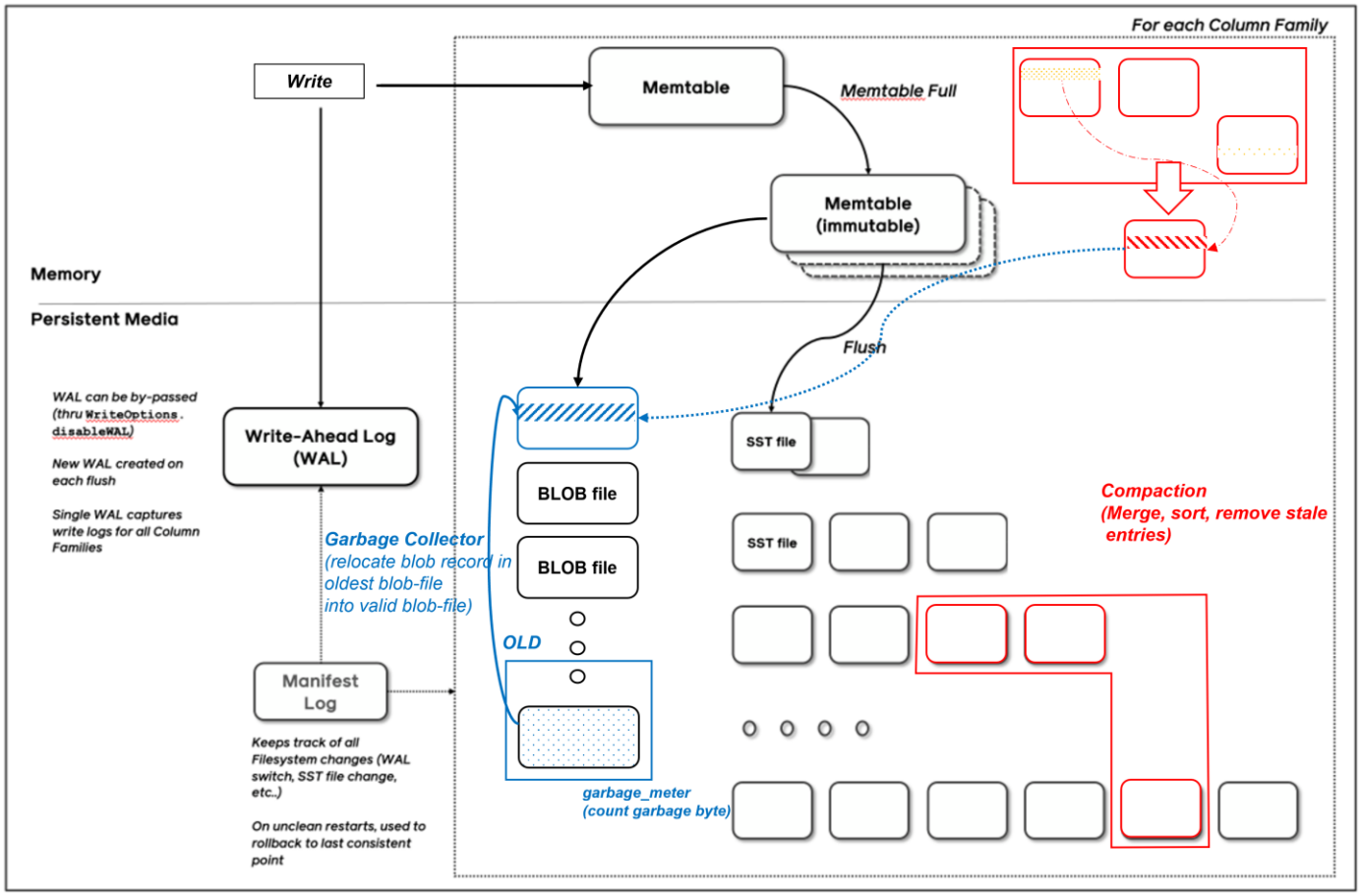
```
Status BlobFileBuilder::OpenBlobFileIfNeeded() {
  if (IsBlobFileOpen()) {
    return Status::OK();
  }

  assert(!blob_count_);
  assert(!blob_bytes_);

  assert(file_number_generator_);
  const uint64_t blob_file_number = file_number_generator_();

  assert(immutable_options_);
  assert(!immutable_options_->cf_paths.empty());
  std::string blob_file_path =
      BlobFileName(immutable_options_->cf_paths.front().path, blob_file_number);

  std::unique_ptr<FSWritableFile> file;

  {
    assert(file_options_);
    Status s = NewWritableFile(fs_, blob_file_path, &file, *file_options_);
```

```
std::function<uint64_t()> file_number_generator_;
```

```
BlobFileBuilder::BlobFileBuilder(
    std::function<uint64_t()> file_number_generator,
```

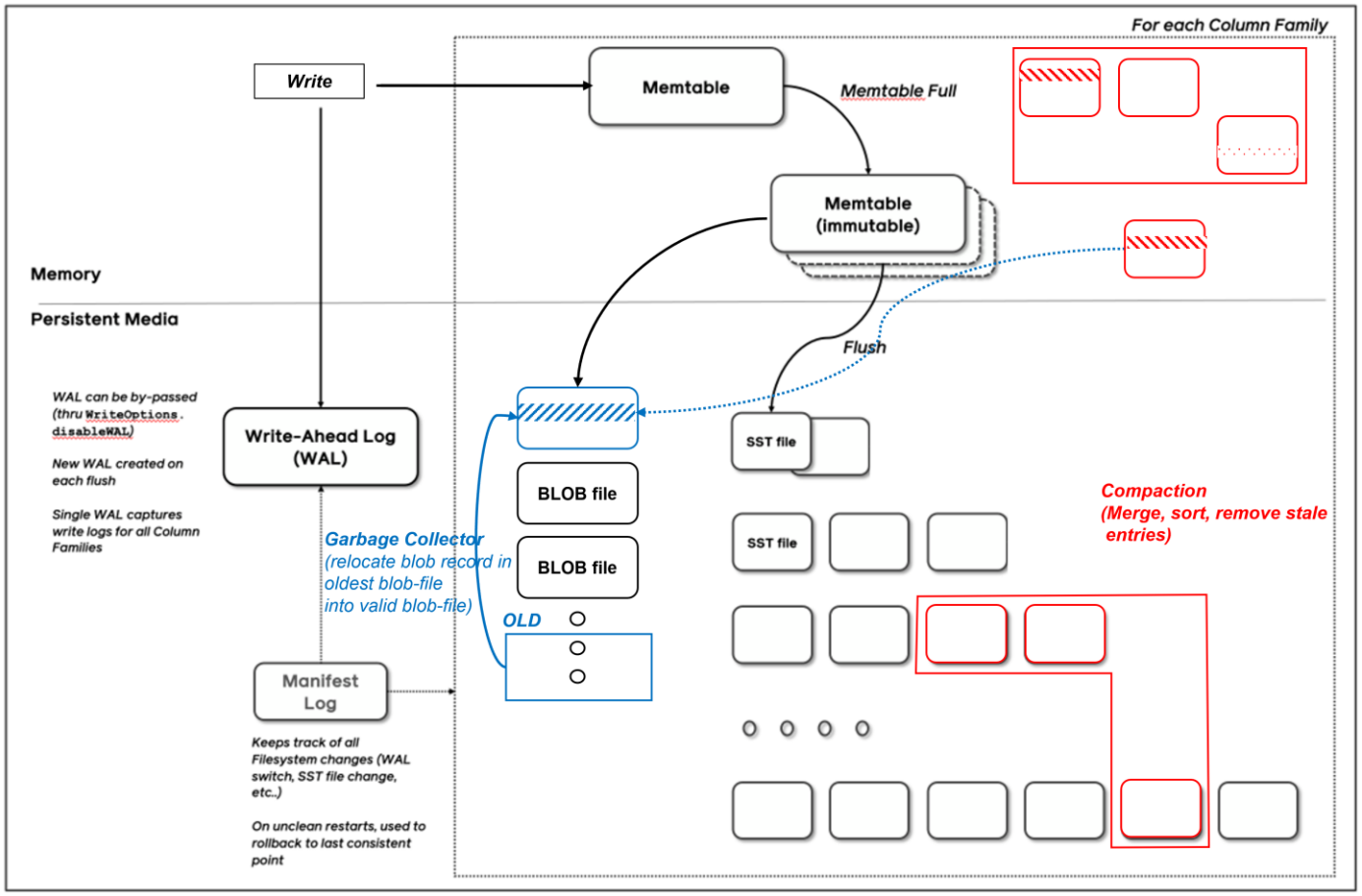
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



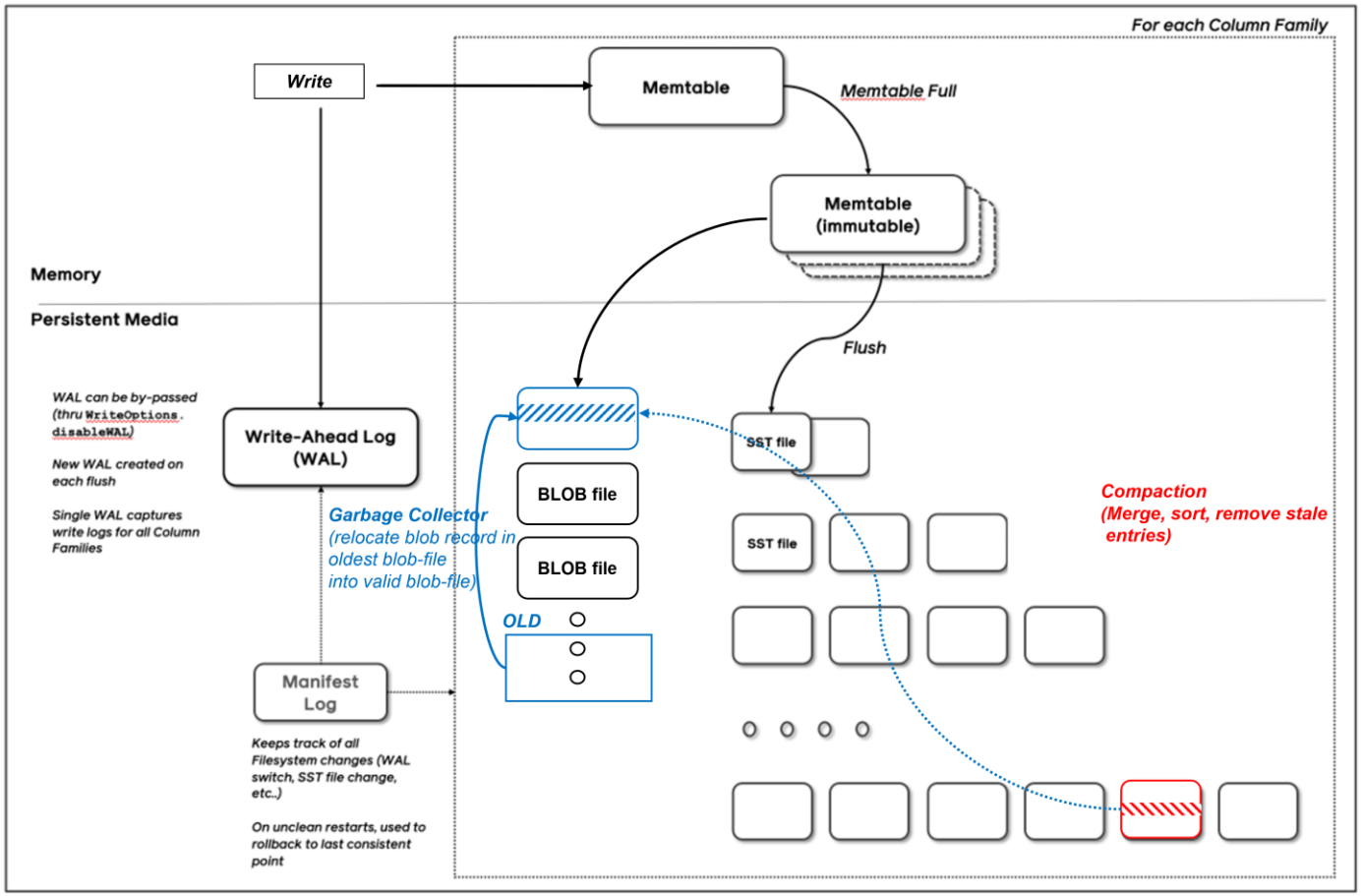
4. Garbage Collector

- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



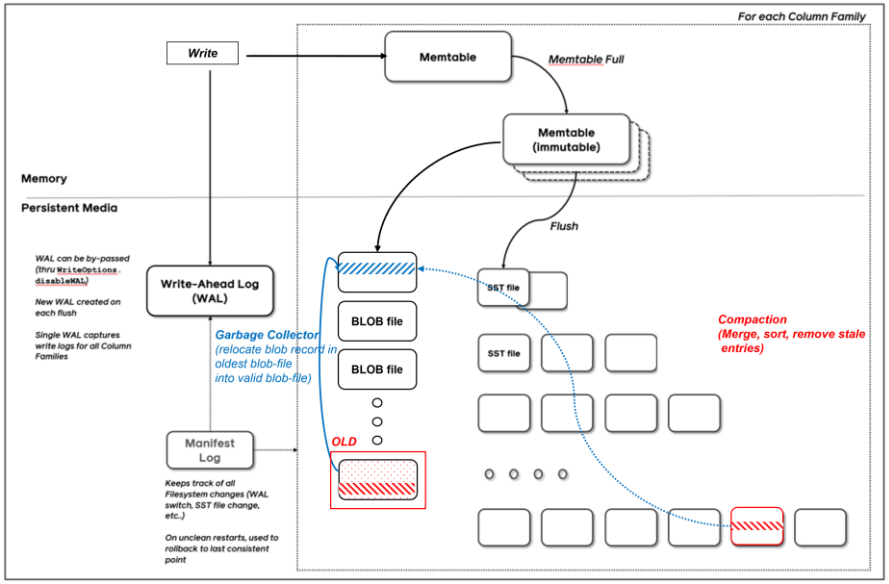
4. Garbage Collector

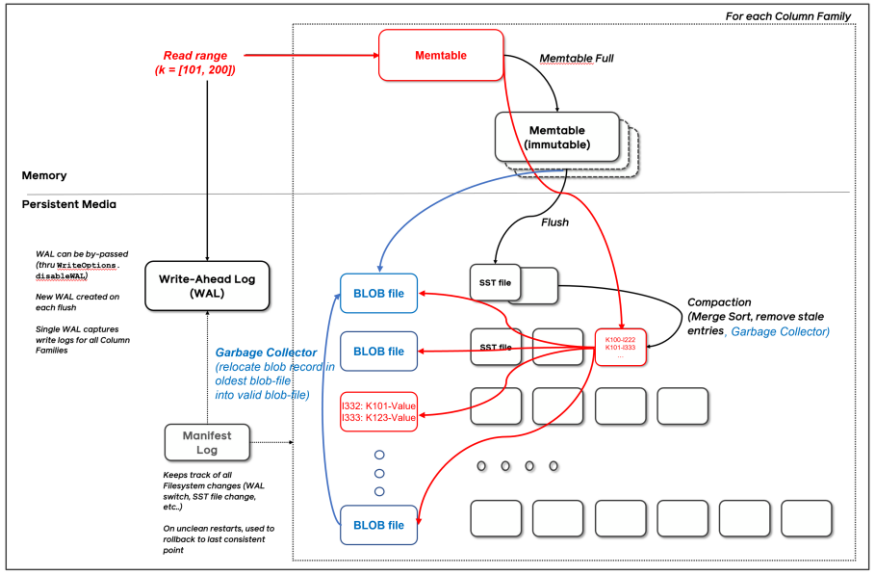
- RocksDB
  - ✓ During compaction, if encounter valid blobs in oldest blob files
  - ✓ Relocate valid blobs from the oldest file to valid files



4. Garbage Collector

- RocksDB
  - ✓ Problem?
    - Space Amplification/Leaks
      - What if compaction doesn't occur with oldest blob files
    - Weak spatial locality of values



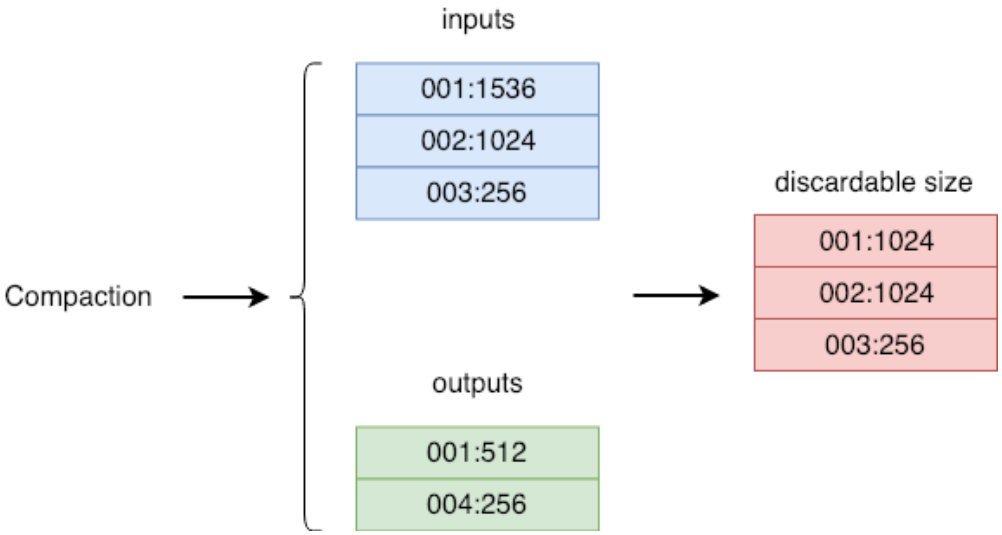


4. Garbage Collector

- Titan DB
  - ✓ RocksDB plugin for key-value separation since 2019
  - ✓ GC, Which/When?
    - the discardable file has reached a specific proportion in size.
  - ✓ BlobFileSizeCollector (BlobDB: garbage_meter)
    - Collect BlobFileSize of each SST files

4. Garbage Collector

- Titan DB
  - ✓ EventListener (BlobDB will be updated)
    - During compaction, collect/compare input/output of blob files size properties
      -> Determine which blob files require GC
    - Discardable size: Input – Output
      - Input: blob file size that participate compaction
      - Output: blob file size that generated in compaction



4. Garbage Collection

- DiffKV
  - ✓ Fork of Titan DB, 2021
  - ✓ Fine-grained KV separation
  - ✓ partially-sorted ordering of values by merge
    - Better Range Scan
  - ✓ Merge
    - Compaction triggered merge
    - Lazy merge / scan-optimized merge
    - Titan: leveled/range merge
  - ✓ Lazy GC
    - Marks GC tag
    - vTable with GC tag is involved in next merge

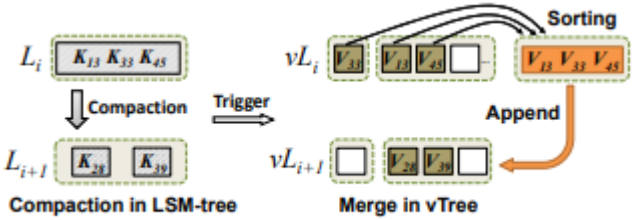


Figure 5: Compaction-triggered merge.

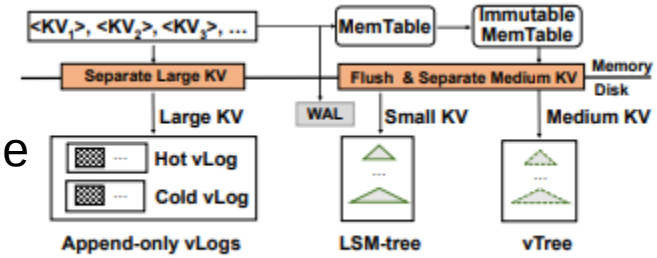


Figure 8: Fine-grained KV separation.

4. Future Work

- DiffKV
  - ✓ Better Range Scan by merge
- Compare Garbage Collection
  - ✓ Blobdb/Titan/DiffKV
- Study BlobDB Structure/API deeper
  - ✓ Iterator/MultiGet
  - ✓ Consistency/WAL
- Performance
  - ✓ garbage collection
  - ✓ dedicated cache for blobs
  - ✓ iterator and MultiGet
  - ✓ blob file format

Q & A