TG Freeing & Compaction + Garbage Collection Fixes

Matt Butrovich
David Gershuni

Objectives

75%: TG Freeing

- Free empty tile groups
- Track tile group utilization

100%: TG Compaction

- Compact TileGroups: moving data using no-op update which maintains the version chain. Prune now-empty TileGroup

125%:

- Expose Compaction settings to the Brain
- Garbage Collection Fixes

Garbage Collection Fixes

- Modified TOTransactionManager to pass tombstones created by deletes to the GCManager.
- GCManager::RecycleTupleSlot allows unused ItemPointers to be returned without going through the entire Unlink and Reclaim process.
- Modified DataTable's Insert to return the ItemPointer to the GCManager in the case of a failed insert.
- Modified DataTable's InsertIntoIndexes to iterate through indexes and remove inserted keys in the event of a failure.
- Modified GCManager to clean indexes from garbage created by COMMIT_DELETE, COMMIT_UPDATE, and ABORT_UPDATE.

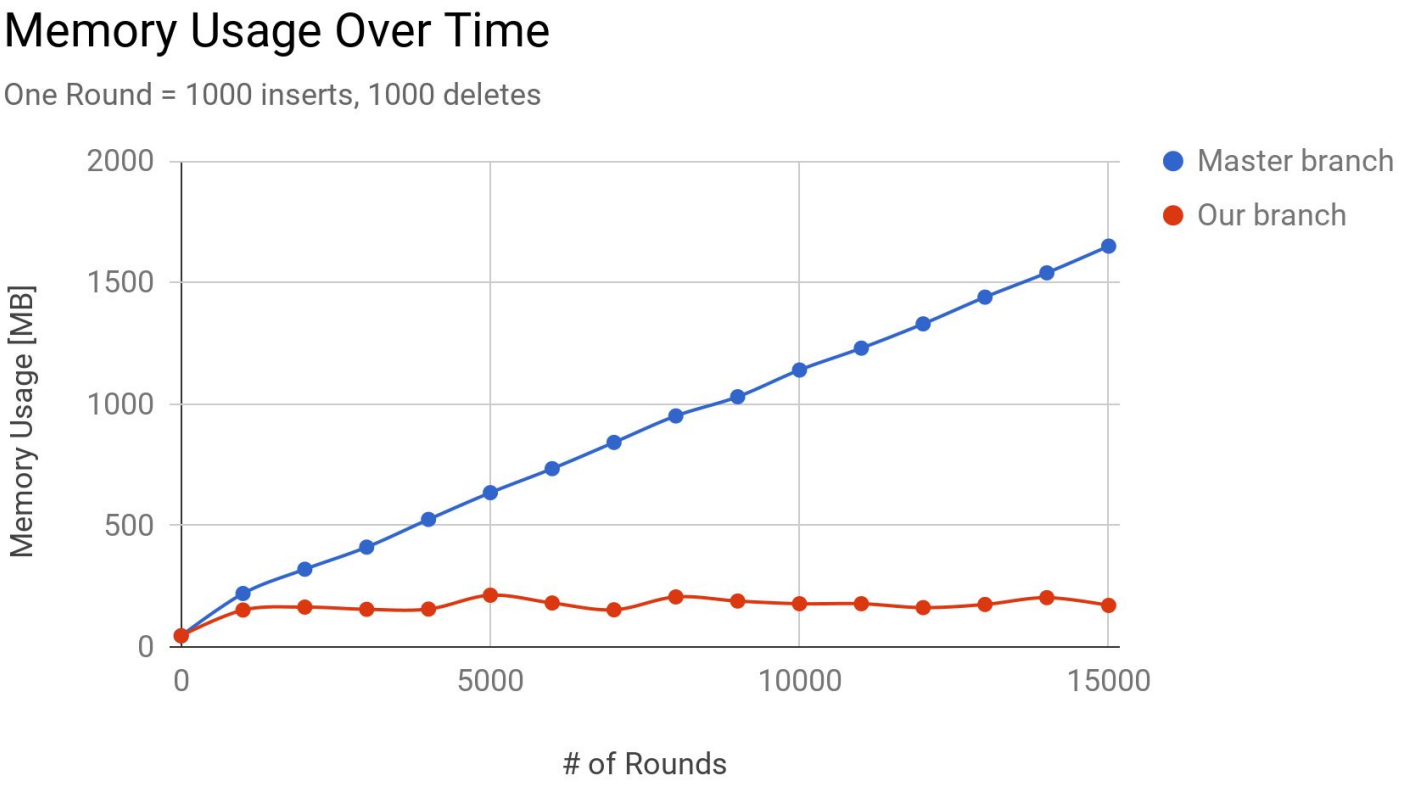
TileGroup Freeing

- Enhanced the Garbage Collector to free empty TileGroups when all of their tuple slots have been recycled.
- RecycleStack replaces the Garbage Collector’s recycle queues.
- Scanned the codebase for all points (outside of codegen) that fetch tile groups from the catalog without checking for a nullptr. TileGroups can no longer be assumed to live forever, so these checks must be done.

TileGroup Compaction

- Created TileGroupCompactor that performs compaction of tile groups.
- Compaction is triggered by the GarbageCollector, which submits a CompactTileGroup() task to the MonoQueuePool when the majority of a TileGroup is recycled garbage.
- The fraction of garbage required to trigger compaction is determined by a global setting, settings::SettingId::compaction_threshold.
- Compaction can also be enabled/disabled via another setting, global setting settings::SettingId::compaction_enabled.

Performance Results



Testing & Code Quality

- Code was designed to reflect separation of responsibilities between GCManager, RecycleStack, and TileGroupCompactor
- We expanded the existing garbage collection test suite, adding 14 new tests to identify leaked ItemPointers and index entries
- We added tests to verify TileGroup freeing functionality, and maintain compatibility with current tests
- We added tests to verify TileGroupCompactor functionality, including correct behavior on aborts/conflicts with other transactions
- High emphasis on readability via clear comments and naming

Future Work

- Brain integration
  - Use hot/cold metrics instead of fixed thresholds
  - Be less eager to compact and free tile groups
- Evaluation of multi-threaded garbage collection
- Codegen nullptr checks

Thanks!