Interval GC

Huzaifa Abbasi
Pulkit Agarwal
Utkarsh Agarwal
Overview

- Project Goals
- Tests & Benchmarks
- Quality of Code
Project Goals

- **75%**
  - Implement single threaded Interval GC
    - HANA has append-only storage
    - <Terrier> uses Delta Records
    - Implement two pass undo record compaction, which reduces the version chain length

- **100% + 125%**
  - Multi-threaded GC + Group Garbage Collection

- **New Goal**
  - Implement Two Threaded Logging
    - Serializer Thread (or Main Thread) : Serializes the data and puts it into a buffer
    - Log Writer Thread : Flushes the serialized buffer to disk
Correctness Tests

- Interval GC
  - Existing Small + Large Randomized Unit Tests
  - New Small Tests to check compaction and versions collected
  - New Large Tests for Long Running OLAP Query with OLTP Queries
  - 97% code coverage for GC (+ 6%)

- Two Threaded Logging
  - Existing Tests
  - 83.7% code coverage
## Benchmarks

- **Interval GC**
  - New benchmark: Long Running OLAP Query with OLTP Queries
  - Performance Change

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Change (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlink (735k items/s)</td>
<td>- 25</td>
</tr>
<tr>
<td>Reclaim (3.2 million items/s)</td>
<td>No change</td>
</tr>
</tbody>
</table>

- **Two Threaded Logging**
  - Existing Benchmarks
GC Benchmark: Long Running OLAP

- Table: 100k tuples, 1k tuple hotspot
- 1000 updates per iteration per hotspot tuple
- On Local Machine [2.2 GHz Intel Core i7, 6 cores]

Average version chain length traversed in OLAP scan

Total time for OLAP scan
Quality of Code

- Garbage Collector
  - Strong: The Interval GC algorithm
  - Weak: The Interval GC algorithm

- Logging
  - Producer Consumer Queue for adapting the number of buffers in use