Microsoft SQL Hekaton – Towards Large Scale Use of PM for In-memory Databases
What this talk is about

 Trying to answer the question: What does it take for an in-memory database to out-compete a disk based database on all storage-related metrics?

 Our experience with Persistent Main Memory for In-Memory Databases

 Talk Outline:
  - Definitions and terms
  - Transaction log in persistent main memory
  - Impact on database high availability
  - Checkpoint, recovery and very large databases
In-memory databases

- Database engines that take advantage of large memory
- Store all/most of the data in DRAM – basis for perf gains

- In-memory data representation != on disk representation
- Fully integrated with Sql Server (transactions, **logging**)
- No buffer pool, no dirty writes, no locks/latches/blocking
- Does not compromise on ACID properties:
  - Atomicity
  - Consistency
  - Isolation
  - **Durability (logging and checkpoint)**
- **Up to 30X performance gains on important workloads**
Memory-Optimized Tables

- Row can be part of multiple indexes, but there is only a single copy
- Each row version has a valid time range indicated by two timestamps
- A version is visible if transaction read time falls within the version’s valid time
- Garbage collection of versions: incremental, parallel, non-blocking, cooperative
Direct Access, Multi-Version, Lock-Free Transactions

Transaction 99: Running compiled query
SELECT City WHERE Name = 'John'
Simple hash lookup returns direct pointer to ‘John’ row

Background operation will unlink and deallocate the old ‘John’ row after transaction 99 completes.

Transaction 100:
UPDATE City = ‘Prague’ where Name = ‘John’
No locks of any kind, no interference with transaction 99
What this talk is about

- Trying to answer the question: What does it take for an in-memory database to out-compete a disk based database on all storage-related metrics?
- Experience with Persistent Main Memory for In-Memory Databases

**Talk Outline:**
- Definitions and terms
- Transaction log in persistent main memory
- Impact on database high availability
- Checkpoint, recovery and very large databases
PMM Log

- Use byte-addressable log implementation
- Using Windows DirectAccess filesystem capability
- Compared with the ideal block mode access device
PMM Log – *Not* TPCC, 2S

<table>
<thead>
<tr>
<th>Storage</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD</td>
<td>601K</td>
</tr>
<tr>
<td>HDD-PMM</td>
<td>1,555K</td>
</tr>
<tr>
<td>SDD-PMM</td>
<td>3,321K (109.6%)</td>
</tr>
<tr>
<td>SDD</td>
<td>3,214K (100%)</td>
</tr>
<tr>
<td>HDD-Lazy</td>
<td>1,720K</td>
</tr>
</tbody>
</table>
PMM Log - Implementation

- Add-on over the Sql Log Manager
- 1:1 relationship between existing log staging area (LC) and new PMM log store memory
- Adding a log record (algorithm outline):
  - Obtain LSN
  - Copy to existing LC slot and then copy to PMM
  - Atomically attach PMM log record to PMM log store
  - If log record is a transaction commit, cl-flush PMM log records
- Surprise: Increased throughput with longer code path!
  - IO path is shorter; IO blocks are larger; lazy commit behavior
- No free lunch: Recovery needs to account for “holes”
  - Still very simple – just move content from PMM to regular log
What this talk is about

- Trying to answer the question: What does it take for an in-memory database to out-compete a disk based database on all storage-related metrics?
- Experience with Persistent Main Memory for In-Memory Databases

**Talk Outline:**
- Definitions and terms
- Transaction log in persistent main memory
- Impact on database high availability
- Checkpoint, recovery and very large databases
Impact on High Availability

- Sql Server HA: send log blocks to secondary replicas
- PMM log side-effect: larger (and slower) block creation
- Solution: Send content on the tx commit path as well

- Consequences: double the network traffic
- Add roughly 25us of latency to transaction commit
- Which does not translate into loss of throughput (context-free work is valuable)
What this talk is about

- Trying to answer the question: What does it take for an in-memory database to out-compete a disk based database on all storage-related metrics?
- Experience with Persistent Main Memory for In-Memory Databases

**Talk Outline:**
- Definitions and terms
- Transaction log in persistent main memory
- Impact on database high availability
- Checkpoint, recovery and very large databases
Hekaton checkpoint and recovery

**Checkpoint**
- Memory-optimized Filegroup (filestreams)
- Checkpoint
- Log

**Recovery**
- Memory
- Redo
- Memory-optimized Filegroup
- Filter
- Filter
- Filter

**Key**
- Data file with rows inserted in timestamp range a-b
- Delta file with IDs of deleted rows
Hekaton Checkpoint

- Fully parallel at both creation and recovery time
- Scalable and high throughput: 1G/second, external limit
- Fully integrated with Sql: encryption, backup/restore, IO resource governance, space management, etc.
- Can be produced from log stream alone (remote-able)

- Loading 1TB of data is slow, regardless of parallelism
  - Assume IO at 1G/s leads to 1000s (~17min) recovery time
- Many indices and fast IO can make it appear slower
- This is where in-memory DB has had an inherent disadvantage!
What to do? O(N) attempt...

- Sort rows before loading them.

- Results in significant time reduction, but still O(N).
- We want constant time recovery instead.
What to do? O(1) attempt…

- Introduce GenerationEra – incremental value which tracks new instances of the host process lifetime.
- Every object is marked with its GenerationEra.
- Differentiate between ‘visibility’ and ‘reachability’.
- Analyze each container: heap, bw-tree index, hash index, free lists to identify object lifetime.
- Use a mark and sweep approach to move eligible old-era items to the current era’s freelist.
- Sweep happens in parallel with DB becoming available.
- Result: DB is available in small constant time.
- Insight: Requires hardware support for ‘reachability’.
Credits

- Paul Larson, Bob Fitzgerald, Ildar Absalyamov – MSR
- Sridharan Sakthivelu – Intel
- Hekaton Engine Team – Microsoft Sql Server