HyPer: A Hybrid OLTP&OLAP Main Memory Database System

Presenter: Lavanya Subramanian
Need for Online Analytics

• Business intelligence today demands fresh data

• Business analytics of yesterday
  – Transactions are run on an OLTP database
  – OLTP database state extracted periodically
  – Analytics performed on the extracted state

• The “perform analytics offline” model too stale and slow for today’s business intelligence
How To Perform Online Analytics?

• Run transactions (OLTP queries) and analytics (OLAP queries) on the same machines

• Problem: *Long running analytics queries interfere with transactions*
HyPer: Key Idea

• In-memory database runs transactions & analytics

• Transactions are run on the main database

• Snapshots are created for analytics
  – by forking the OLTP process

• Properties of snapshots created on a fork()
  – Data is not duplicated rightaway
  – A page is duplicated only when modified (copy-on-write)
Basic Transaction Processing Model in HyPer

• Builds on prior work on in-memory transaction processing

• Single-threaded execution is effective enough
  – No IO wait times

• Short transactions
  – No interactive transactions
Analytical Processing in HyPer

Image Credit: Alfons Kemper
How Does Copy on Write Work?

1) High latency
2) High bandwidth utilization
3) Cache pollution
4) Unwanted data movement

Image Credit: Vivek Seshadri
Hardware Support For Fast Copy-On-Write

1) Low latency

2) Low bandwidth utilization

3) No cache pollution

Image Credit: Vivek Seshadri
Parallelizing Analytics and Transactions
Multiple OLAP Sessions

• Snapshots for OLAP
  – Do not consume much space
  – Can be created easily using fork()

• Parallelize OLAP query execution
  – Using multiple snapshots
  – Executing on idle CPU cores

• Snapshot deleted after last query of a session
Multi-Threaded Transaction Processing

- Execute multiple read-only queries in parallel

- Execute read-write queries in parallel
  - Scenarios where data can be partitioned
  - Transactions confined to partitions

- Only one transaction per partition

- Cross-partition transactions run single threaded
More Discussion on Transactions

- Snapshot Isolation
- Durability
- Transaction Consistency
Snapshot Isolation

• Roll-back
  – Roll back when an older query needs older data

• Versioning
  – Create a new object version on every update
  – Retrieve youngest version before query start time

• Shadowing
  – Write updates to a shadow copy
  – Update main copy upon commit

• Virtual memory snapshots
Durability

• On failure recovery, all effects of committed transactions should be restored

• Solution: Logical redo logging
  – Apply log to database after failure recovery

• Redo log can be used to feed a secondary server
  – Potential uses: standby, analytics processing
Transaction Consistency

• Perform Undo logging to obtain a transaction consistent snapshot

• Applied to a snapshot created from a fork()
  – To undo effects of current transactions
Methodology

• Benchmark
  – TPC-C scheme
  – Additional three relations from TPC-H

• Hardware
  – Intel X5570 – Quad Core CPU
  – 64 GB DRAM

• Comparison Points
  – MonetDB (for analytics)
  – VoltDB (for transactions)
Results - Performance and Memory Consumption

<table>
<thead>
<tr>
<th>Query No.</th>
<th>one query session (stream)</th>
<th>Query resp. times (ms)</th>
<th>HyPer configurations</th>
<th>Query resp. times (ms)</th>
<th>3 query sessions (streams)</th>
<th>Query resp. times (ms)</th>
<th>MonetDB no OLTP</th>
<th>Query resp. times (ms)</th>
<th>VoltDB no OLAP only OLTP results from [18]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>OLTP throughput</td>
<td>67</td>
<td>8 query sessions</td>
<td>71</td>
<td>3 query sessions</td>
<td>71</td>
<td>63</td>
<td>210</td>
<td>6003</td>
</tr>
<tr>
<td>Q2</td>
<td>OLTP throughput</td>
<td>163</td>
<td>OLTP throughput</td>
<td>233</td>
<td>OLTP throughput</td>
<td>212</td>
<td>210</td>
<td>75</td>
<td>5930</td>
</tr>
<tr>
<td>Q3</td>
<td>OLTP throughput</td>
<td>66</td>
<td>OLTP throughput</td>
<td>78</td>
<td>OLTP throughput</td>
<td>73</td>
<td>75</td>
<td>75</td>
<td>123</td>
</tr>
<tr>
<td>Q4</td>
<td>OLTP throughput</td>
<td>194</td>
<td>OLTP throughput</td>
<td>257</td>
<td>OLTP throughput</td>
<td>226</td>
<td>6003</td>
<td>6003</td>
<td>172</td>
</tr>
<tr>
<td>Q5</td>
<td>OLTP throughput</td>
<td>1276</td>
<td>OLTP throughput</td>
<td>1768</td>
<td>OLTP throughput</td>
<td>1564</td>
<td>5930</td>
<td>5930</td>
<td>172</td>
</tr>
<tr>
<td>Q6</td>
<td>OLTP throughput</td>
<td>9</td>
<td>OLTP throughput</td>
<td>19</td>
<td>OLTP throughput</td>
<td>17</td>
<td>123</td>
<td>123</td>
<td>208</td>
</tr>
<tr>
<td>Q7</td>
<td>OLTP throughput</td>
<td>1151</td>
<td>OLTP throughput</td>
<td>1611</td>
<td>OLTP throughput</td>
<td>1466</td>
<td>1713</td>
<td>1713</td>
<td>208</td>
</tr>
<tr>
<td>Q8</td>
<td>OLTP throughput</td>
<td>399</td>
<td>OLTP throughput</td>
<td>680</td>
<td>OLTP throughput</td>
<td>593</td>
<td>172</td>
<td>172</td>
<td>208</td>
</tr>
<tr>
<td>Q9</td>
<td>OLTP throughput</td>
<td>206</td>
<td>OLTP throughput</td>
<td>269</td>
<td>OLTP throughput</td>
<td>249</td>
<td>208</td>
<td>208</td>
<td>208</td>
</tr>
<tr>
<td>Q10</td>
<td>OLTP throughput</td>
<td>1871</td>
<td>OLTP throughput</td>
<td>2490</td>
<td>OLTP throughput</td>
<td>2260</td>
<td>6209</td>
<td>6209</td>
<td>6209</td>
</tr>
<tr>
<td>Q11</td>
<td>OLTP throughput</td>
<td>33</td>
<td>OLTP throughput</td>
<td>38</td>
<td>OLTP throughput</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>Q12</td>
<td>OLTP throughput</td>
<td>156</td>
<td>OLTP throughput</td>
<td>195</td>
<td>OLTP throughput</td>
<td>170</td>
<td>35</td>
<td>35</td>
<td>192</td>
</tr>
<tr>
<td>Q13</td>
<td>OLTP throughput</td>
<td>185</td>
<td>OLTP throughput</td>
<td>272</td>
<td>OLTP throughput</td>
<td>229</td>
<td>192</td>
<td>192</td>
<td>192</td>
</tr>
<tr>
<td>Q14</td>
<td>OLTP throughput</td>
<td>122</td>
<td>OLTP throughput</td>
<td>210</td>
<td>OLTP throughput</td>
<td>156</td>
<td>284</td>
<td>284</td>
<td>284</td>
</tr>
<tr>
<td>Q15</td>
<td>OLTP throughput</td>
<td>528</td>
<td>OLTP throughput</td>
<td>1002</td>
<td>OLTP throughput</td>
<td>792</td>
<td>722</td>
<td>722</td>
<td>722</td>
</tr>
<tr>
<td>Q16</td>
<td>OLTP throughput</td>
<td>1353</td>
<td>OLTP throughput</td>
<td>1584</td>
<td>OLTP throughput</td>
<td>1500</td>
<td>533</td>
<td>533</td>
<td>533</td>
</tr>
<tr>
<td>Q17</td>
<td>OLTP throughput</td>
<td>159</td>
<td>OLTP throughput</td>
<td>171</td>
<td>OLTP throughput</td>
<td>168</td>
<td>3562</td>
<td>3562</td>
<td>3562</td>
</tr>
<tr>
<td>Q18</td>
<td>OLTP throughput</td>
<td>108</td>
<td>OLTP throughput</td>
<td>133</td>
<td>OLTP throughput</td>
<td>119</td>
<td>342</td>
<td>342</td>
<td>342</td>
</tr>
<tr>
<td>Q19</td>
<td>OLTP throughput</td>
<td>103</td>
<td>OLTP throughput</td>
<td>219</td>
<td>OLTP throughput</td>
<td>183</td>
<td>2505</td>
<td>2505</td>
<td>2505</td>
</tr>
<tr>
<td>Q20</td>
<td>OLTP throughput</td>
<td>114</td>
<td>OLTP throughput</td>
<td>230</td>
<td>OLTP throughput</td>
<td>197</td>
<td>1698</td>
<td>1698</td>
<td>1698</td>
</tr>
<tr>
<td>Q21</td>
<td>OLTP throughput</td>
<td>46</td>
<td>OLTP throughput</td>
<td>50</td>
<td>OLTP throughput</td>
<td>50</td>
<td>750</td>
<td>750</td>
<td>750</td>
</tr>
<tr>
<td>Q22</td>
<td>OLTP throughput</td>
<td>7</td>
<td>OLTP throughput</td>
<td>9</td>
<td>OLTP throughput</td>
<td>9</td>
<td>141</td>
<td>141</td>
<td>141</td>
</tr>
</tbody>
</table>

Fig. 9. Performance Comparison: HyPer OLTP&OLAP, MonetDB only OLAP, VoltDB only OLTP
Memory Consumption
Discussion

• Simple mechanism that exploits an existing feature of virtual memory management
• How would memory consumption increase with multiple snapshots?
• Is their OLTP performance evaluation fair?