Optimizing Redis for Locality and Capacity

Kevin C., Yoongu K. Lavanya S.
15-799 Project Presentation
12/4/2013
Goals of Our Project

• Leverage DRAM and dataset characteristics to improve performance of **in-memory database**

• **Locality**: Exploit DRAM internal buffers

• **Capacity**: Exploit redundancy in dataset
DRAM System Organization

CPU

Bus

DRAM
DRAM System Organization

Banks can be accessed in parallel
• **Row buffer** serves as a fast cache in a bank
  – **Row buffer miss** transfers an entire row of data to the row buffer
  – **Row buffer hit** for accesses in the same row (reduces latency by 1-2x)
RBL in In-Memory Databases

• **Idea**: Map hot data to a few DRAM rows

• **Hot data**: Data with high temporal correlation

• **Examples of temporally correlated data**:
  – Records touched around the same time
  – Query terms searched together often
Challenge

• How are data mapped to DRAM? Which bank? Which row?

Virtual Address
  Virtual Page Number  Offset

Physical Address
  Physical Page Number  Offset

Unexposed to the system: Determined by the HW (memory controller)
Task 1: Find the Mapping to DRAM

- **Approach**: Kernel module with assembly code to observe access latency to different addresses

<table>
<thead>
<tr>
<th>Input: addr1 &amp; addr2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Load addr1 // Fill TLB for addr1</td>
</tr>
<tr>
<td>2. Load addr2 // Fill TLB for addr2</td>
</tr>
<tr>
<td>3. Flush the cache lines of addr1 and addr2</td>
</tr>
<tr>
<td>4. Load addr1</td>
</tr>
<tr>
<td>5. Read CPU cycle counter // Tstart for addr2</td>
</tr>
<tr>
<td>6. Load addr2</td>
</tr>
<tr>
<td>7. Read CPU cycle counter // Tend of addr2</td>
</tr>
</tbody>
</table>

1. *Cache hit*
2. *Cache miss – Row Hit*
3. *Cache miss – Row Miss*

Courtesy: Backbone kernel module is obtained from Hyoseung Kim under Prof. Rajkumar
Task 1: Find the Mapping to DRAM

- Experimental setup: 3.4GHz Haswell CPU, 2GB DRAM DIMM (8 banks)
- With an exhaustive selection of addr1 and addr2, we discover the mapping to be:

<table>
<thead>
<tr>
<th>Physical Page Number</th>
<th>Offset</th>
<th>Physical Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>13 15 16 18</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>15 16 18</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>13 15 16 18</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>15 16 18</td>
</tr>
</tbody>
</table>

Byte offset within a row (8KB)

**XOR** bit [15:13] with bit [18:16] to select a bank
Task 1: Find the Mapping to DRAM

Physical Address Space

Rows

Bank 0

Bank 1

Bank 7

0xFFFF

0x0000

0x2000

0x4000

0xFFFF

P0

P9

P8

P7

Bank 0

Bank 1

Bank 7

Byte offset within a row (8KB)

XOR bit [15:13] with bit [18:16] to select a bank

Offset

Bank

Row

Row

0

18 16:15 13:12
Task 1: Find the Mapping to DRAM

• **Measurement:**

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Approximate Latency (CPU cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache hit</td>
<td>30</td>
</tr>
<tr>
<td>Row hit in the same bank</td>
<td>170</td>
</tr>
<tr>
<td>Row hit in a different bank</td>
<td>220</td>
</tr>
<tr>
<td>Row miss</td>
<td>270</td>
</tr>
</tbody>
</table>

- The cache hit latency includes the overhead of extra assembly instructions
- **Under investigation:** Why does row hit in a different bank incurs extra latency?
Task 2: Microbenchmark

- Kernel: Allocates 128KB of memory space (guaranteed to be contiguous physical pages)

- Test 1: Striding within a row
  - Results in row hits

- Test 2: Zigzag b/w 2 rows in the same bank
  - Results in row misses
Why Understand Mapping to DRAM?

• Enables mapping application data to exploit locality

• **Pages mapped to rows:**
  – Data accesses to the same row incur low latency
  – Colocate frequently accessed data in same row

• **Next cache line prefetched:**
  – Accessing next cache line incurs low latency
  – Map data accessed together to adjacent cache lines
Data Mapping Benefits in Redis

• Is memory access the bottleneck?

• Profiling using Performance API (PAPI)
  – An interface to hardware performance counters

• Profile set and get key functions
  – Determine what fraction of cycles are set and get
Data Mapping Benefits in Redis

Memory is not a significant bottleneck in Redis
Sensitivity to Payload Size

Memory still not a significant bottleneck in Redis
Next Steps

- **Row-hit vs. miss behavior on Redis:**
  - Memmap to allocate data contiguously in a page
  - Microbenchmarks to access same and different rows/pages
More Potential for Data Mapping?

• Single-node databases

• Mainframe transaction processing systems

• Data analytics systems
Dataset

- Could not find suitable in-memory dataset
- We constructed our own dataset based on the English Wikipedia corpus
  1. XML dump of current revisions for all English articles
     • 43GB (uncompressed)
     • 11/04/2013
     • http://dumps.wikimedia.org/enwiki/20131104/enwiki-20131104-pages-articles.xml.bz2
  2. Article hit-count log (one hour)
     • 307MB (uncompressed)
     • Last hour of 11/04/2013
Dataset (cont’d)

• Sanitation was unexpectedly non-trivial...
  – Spam and/or invalid user queries
  – ASCII vs. UTF-8 vs. ISO/IEC 8859-1
  – URI escape characters, HTML escape characters
  – Running out of memory

• Sanitized dataset
  – 141K key-value pairs: (title, article)
  – 3.6GB (uncompressed)