Optimizing Redis for Locality and Capacity

Kevin C., Yoongu K. Lavanya S.

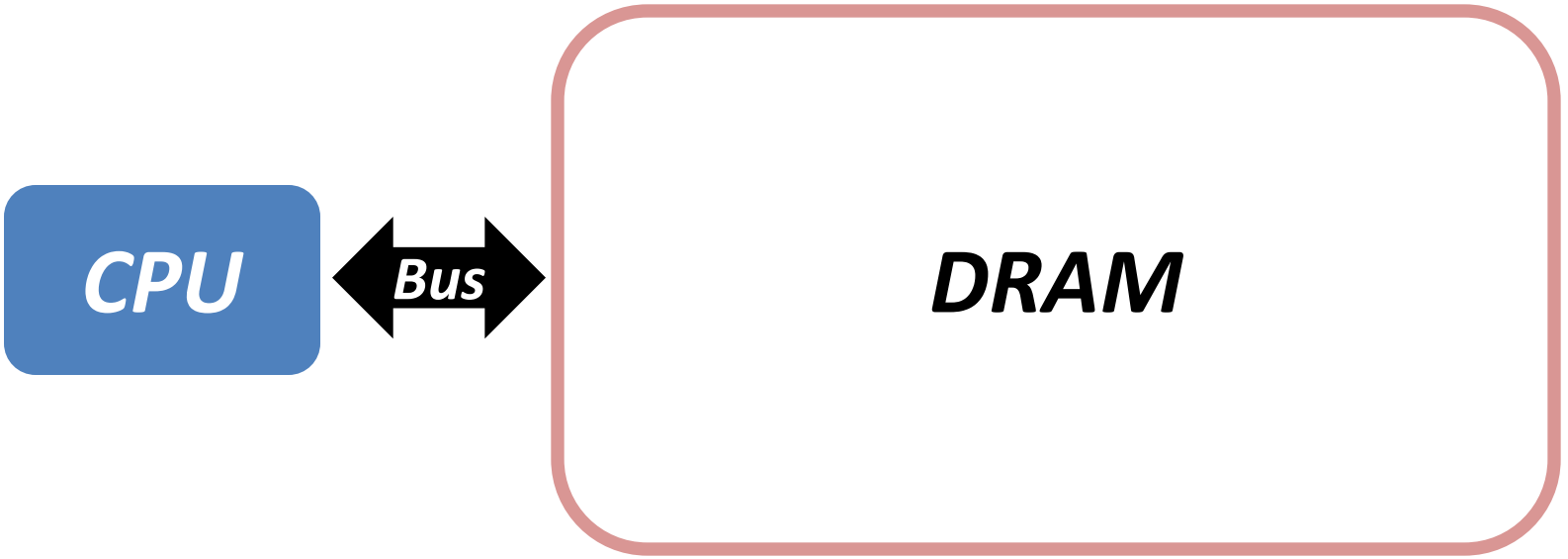
15-799 Project Presentation

12/4/2013

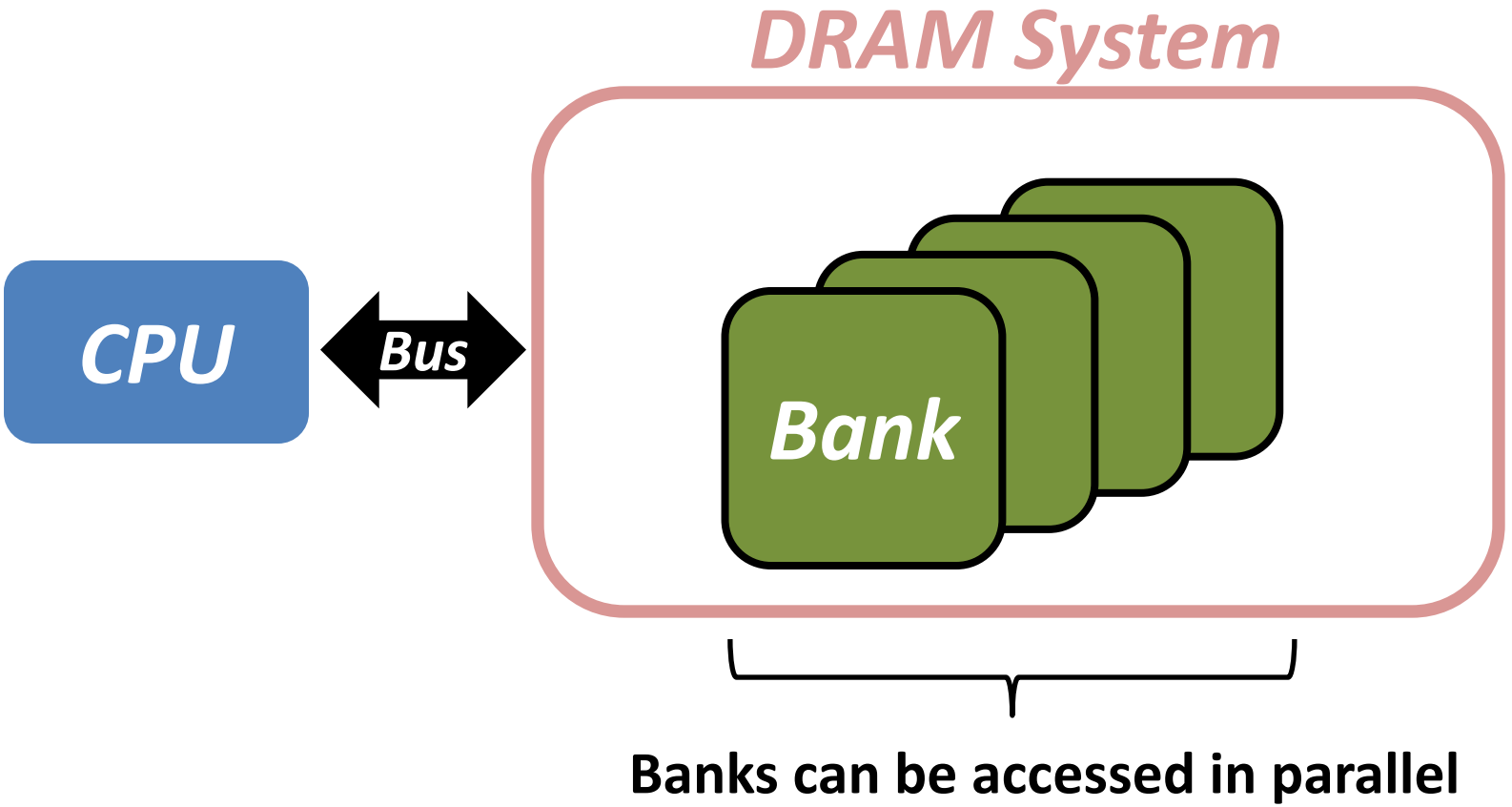
Goals of Our Project

- Leverage DRAM and dataset characteristics to improve performance of **in-memory database**
- **Locality**: Exploit DRAM internal buffers
- **Capacity**: Exploit redundancy in dataset

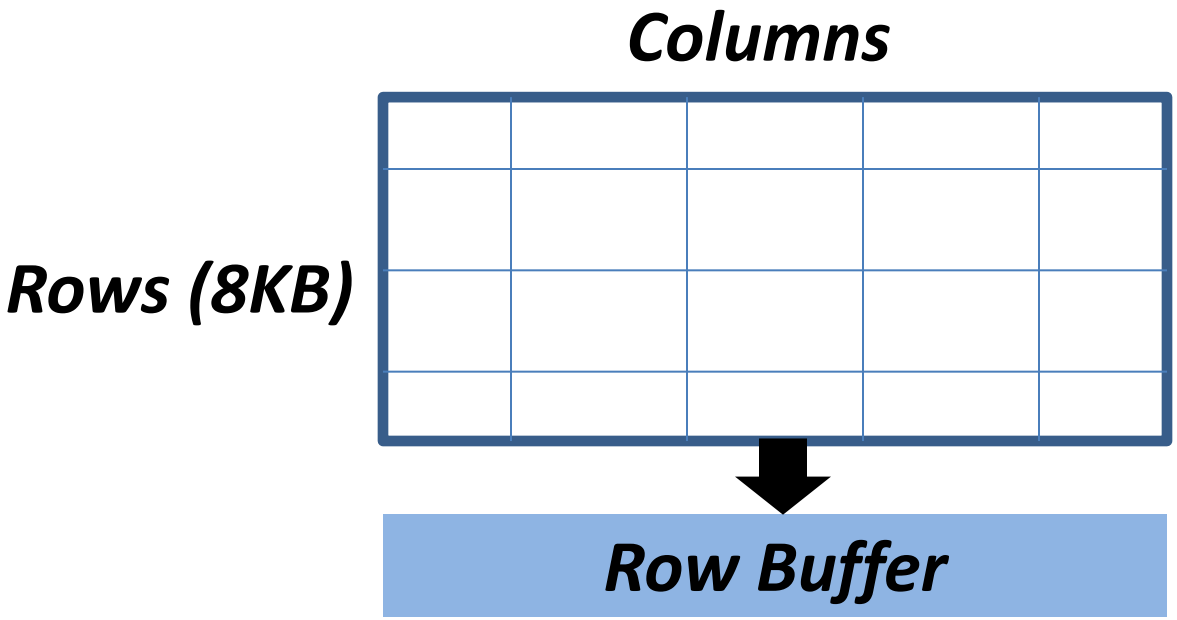
DRAM System Organization



DRAM System Organization



DRAM Bank Organization



- ***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

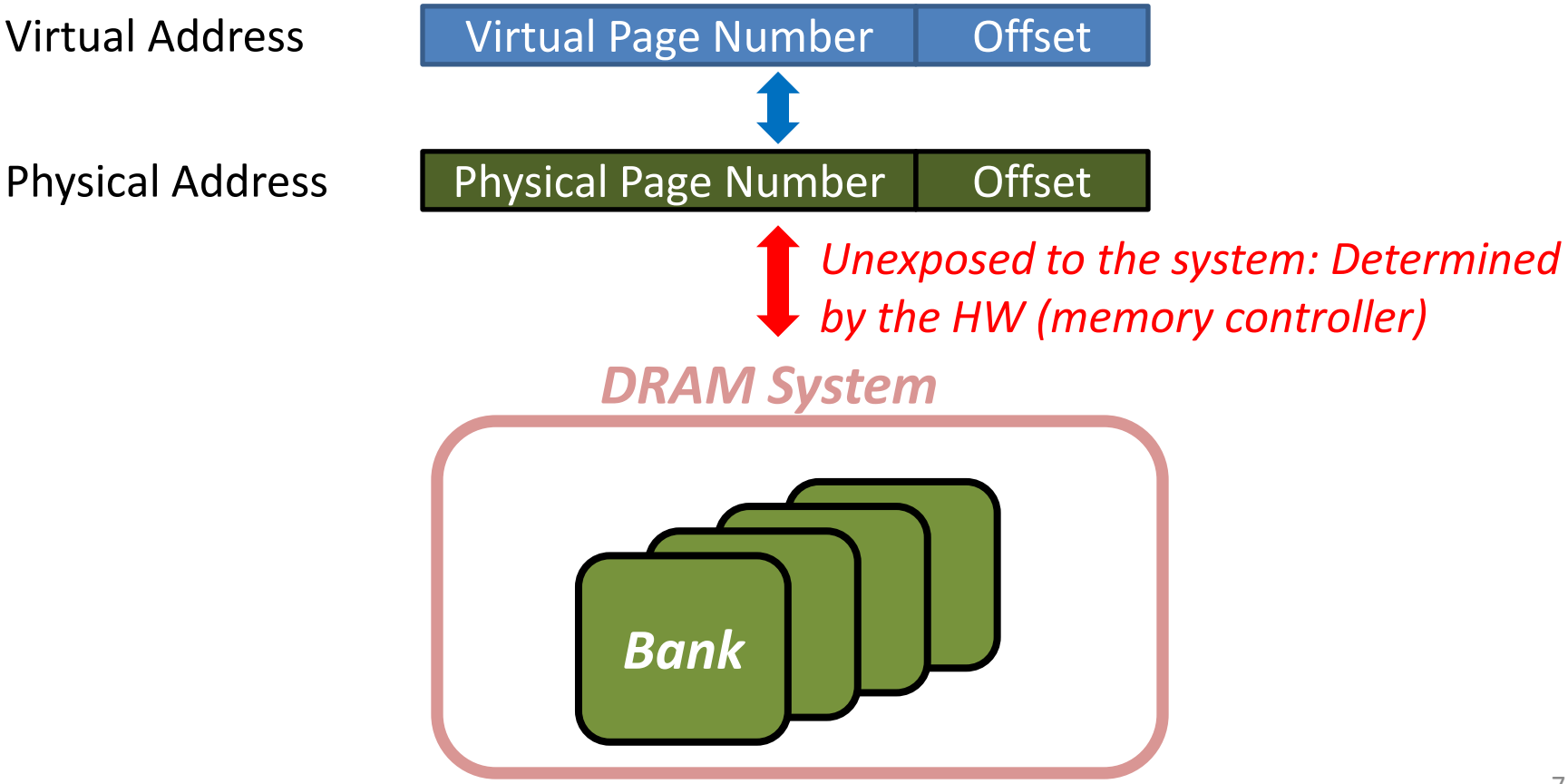
- **Idea**: Map hot data to a few DRAM rows

- **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?



Task 1: Find the Mapping to DRAM

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

Input: **addr1** & **addr2**

1. Load **addr1** // Fill TLB for addr1
2. Load **addr2** // Fill TLB for addr2

---

3. Flush the cache lines of **addr1** and **addr2**

---

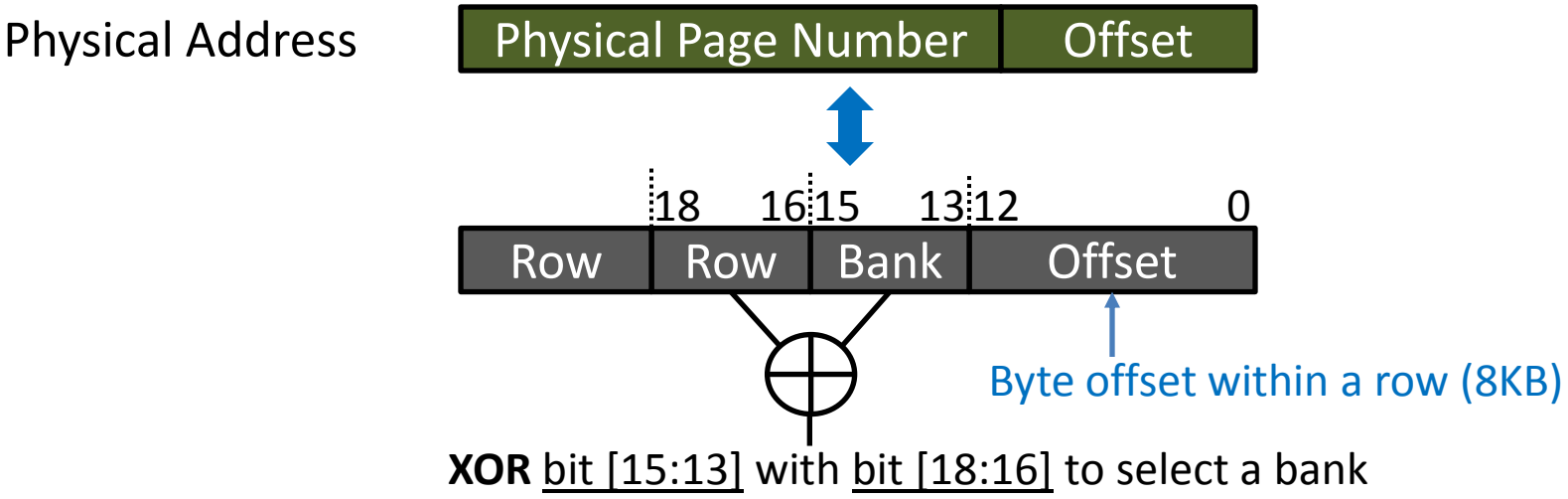
4. Load **addr1**
5. Read CPU cycle counter // Tstart for addr2
6. Load **addr2**
7. Read CPU cycle counter // Tend of addr2

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

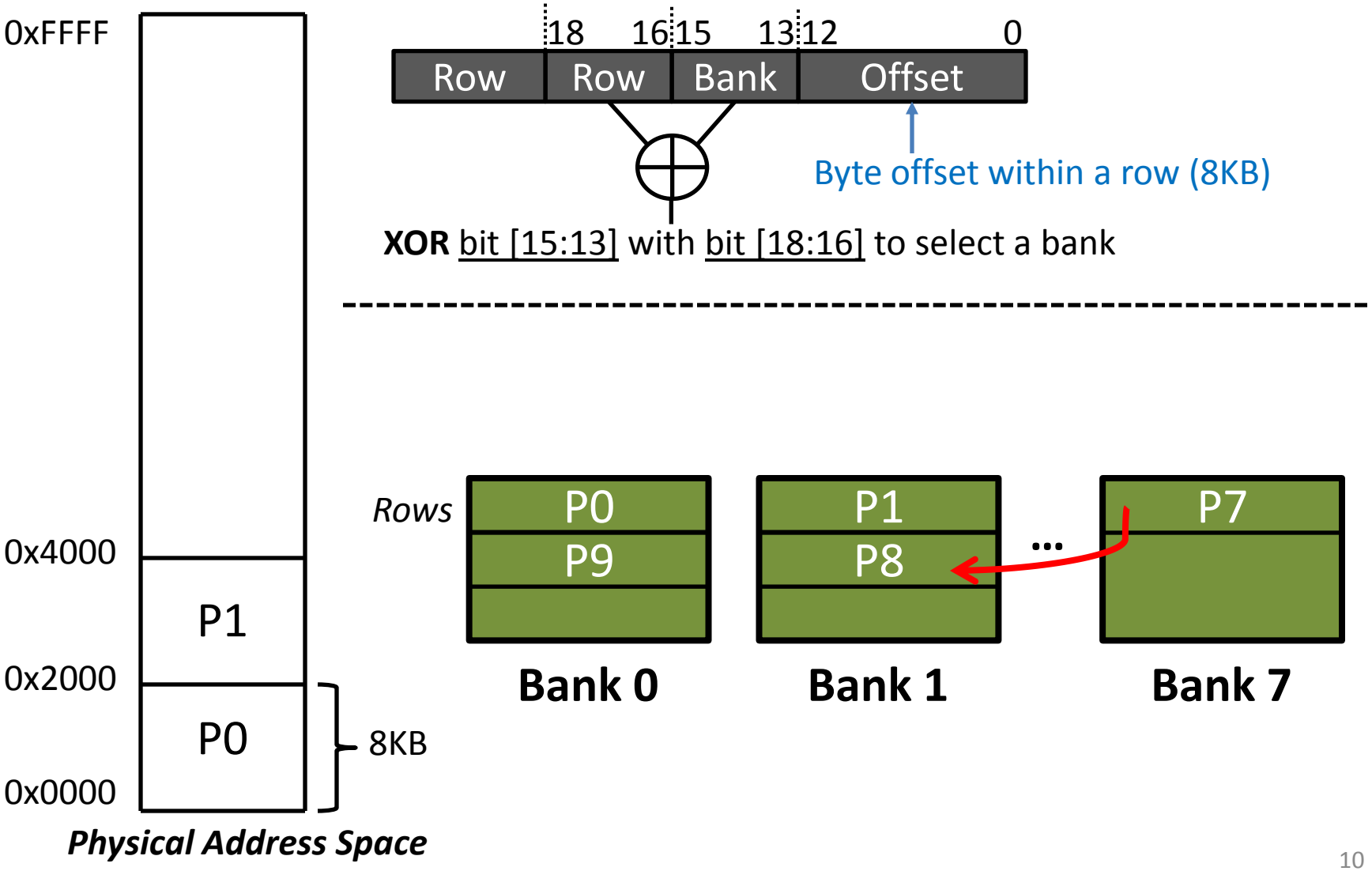
Courtesy: Backbone kernel module is obtained from Hyoseung Kim under Prof. Rajkumar

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:



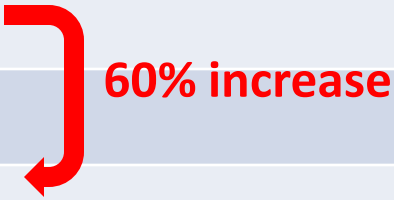
Task 1: Find the Mapping to DRAM



Task 1: Find the Mapping to DRAM

- Measurement:

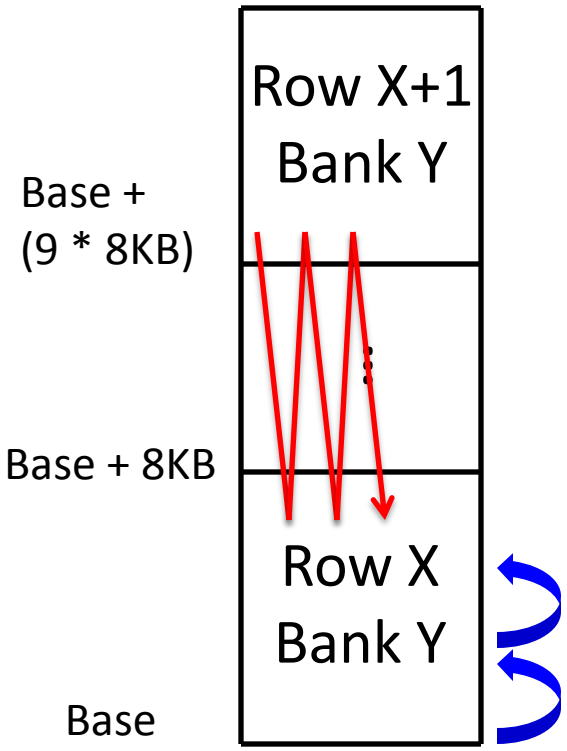
Request Type	Approximate Latency (CPU cycles)
Cache hit	30
Row hit in the same bank	170
Row hit in a different bank	220
Row miss	270



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

Task2: 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

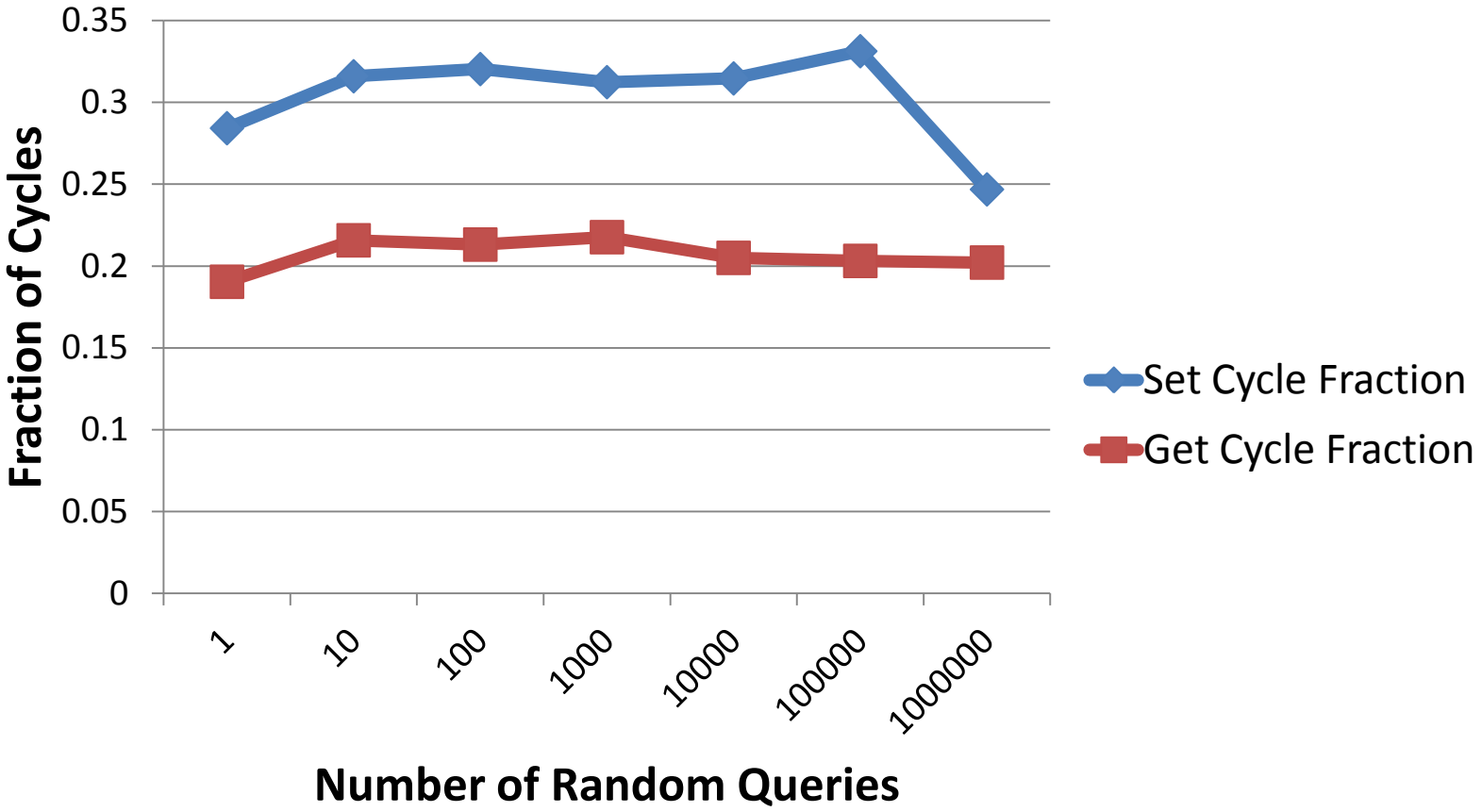
- **Pages mapped to rows:**
  - Data accesses to the same row incur low latency
  - Colocate frequently accessed data in same row

- **Next cache line prefetched:**
  - Accessing next cache line incurs low latency
  - Map data accessed together to adjacent cache lines

Data Mapping Benefits in Redis

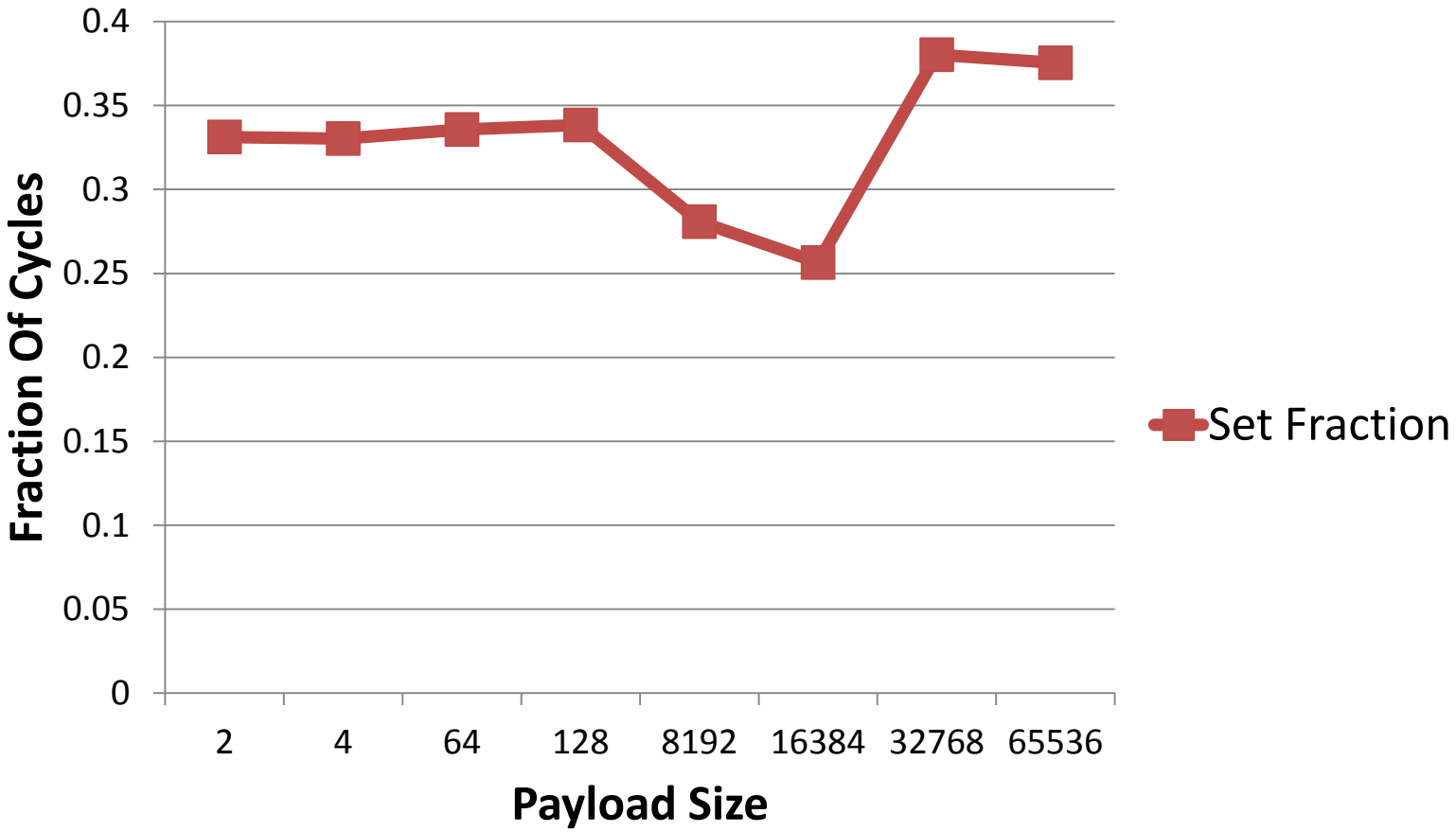
- Is memory access the bottleneck?
- Profiling using Performance API (PAPI)
  - An interface to hardware performance counters
- Profile set and get key functions
  - Determine what fraction of cycles are set and get

Data Mapping Benefits in Redis



Memory is not a significant bottleneck in Redis

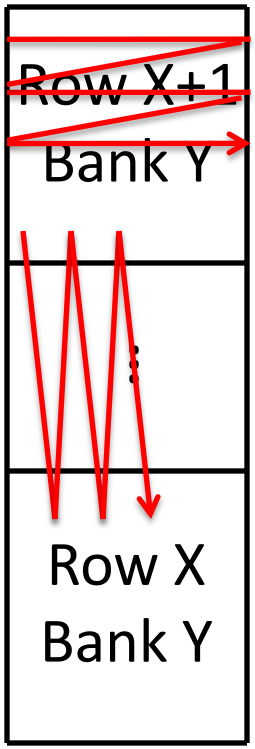
Sensitivity to Payload Size



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
     - http://dumps.wikimedia.org/other/pagecounts-raw/2013/2013-11/pagecounts-20131105-000001.gz

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)