Data Caching Systems for Performance and Cost

David Lomet

Microsoft Research, Redmond, WA

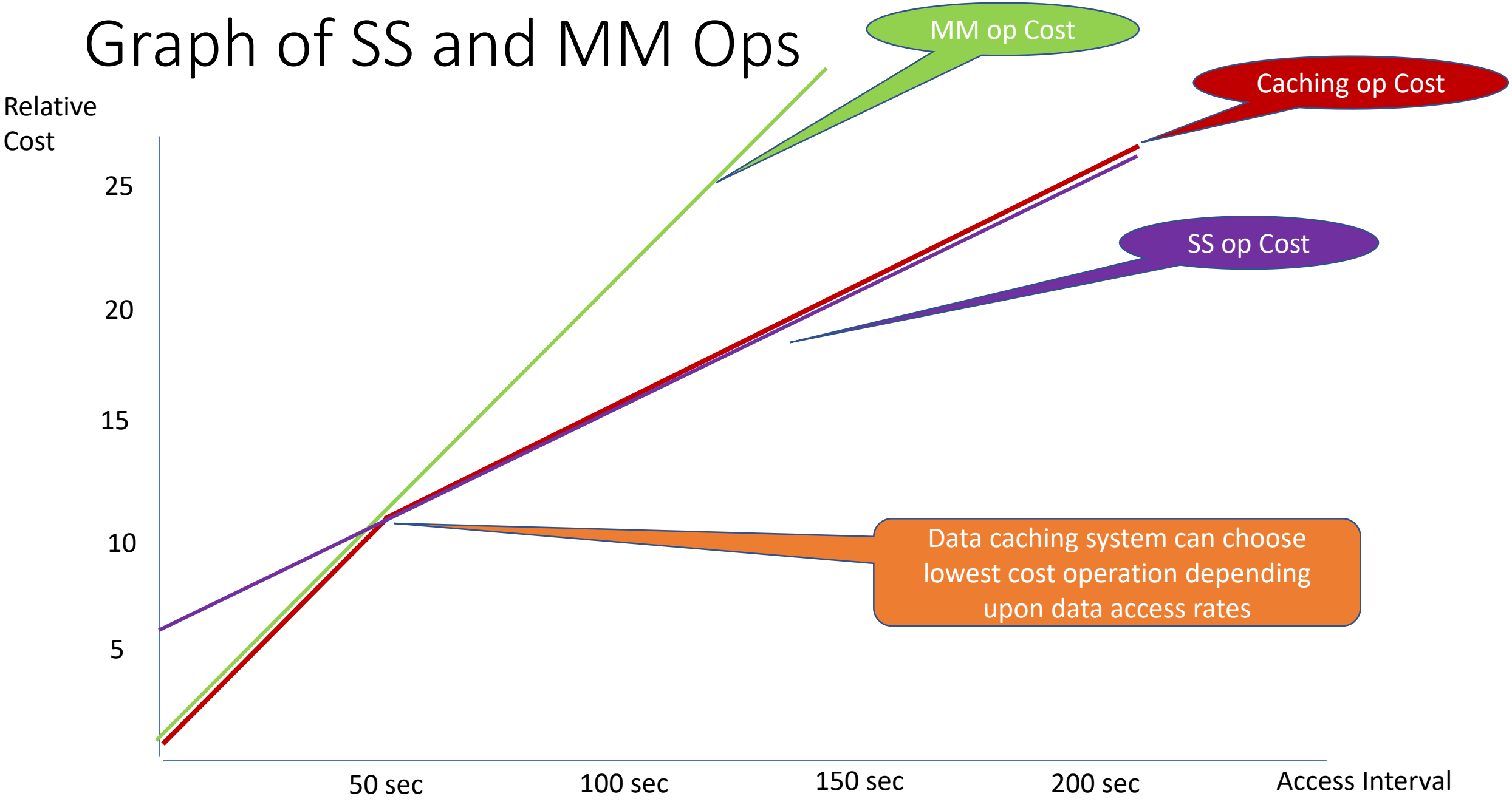
Cost Matter- especially cost/performance

- Data caching systems were designed to use disks for durability
  - But it also was a cost-effective approach
  - Having to store all data in main memory is very expensive
  - Disk storage costs much less
- Strength of data caching systems is
  - Ability to evict data from main memory based on access rates
  - Has a performance impact, raising **execution cost** of operations
  - But it reduces **storage cost**
- **Highlight:** Operations have both **storage** and **execution costs**

Analysing Costs

- **Main memory operation(MM)** finds its data in main memory
  - **Storage Cost:** High cost for main memory between operation accesses
    - Pays also for durability with secondary storage cost as well
  - **Execution Cost:** Low because operations have high performance
- **Secondary storage operation(SS)** finds its data on secondary storage
  - **Storage Cost:** Nothing for main memory between operation accesses
    - Pays only for secondary storage cost
  - **Execution Cost:** High because operations need to include read I/O
- **Breakeven Point:** when operation costs are equal
  - Akin to Gray 5-minute rule, but treats costs away from breakeven
  - Depending on how SSD costs are allocated: access interval ranges from
    - 45 seconds to 100 seconds for average page size of 2.7KB
    - Less than 5 minutes: IOPS cost has dropped faster than DRAM costs
- **Away from breakeven**
  - **Cold data operation cost** depends on relative storage cost
  - **Hot data operation cost** depends on relative execution cost

Graph of SS and MM Ops



Cost Analysis Applied Elsewhere

- Bw-tree (main memory) vs MassTree, --small records experiment
  - **MassTree** 2.6 X faster and lower execution cost than **Bw-tree**
  - **MassTree** takes 2.1 X more memory and higher storage cost than **Bw-tree**
  - Breakeven point (we switch entire DB between MassTree and Bw-tree)
    - Access interval of 1.4*10**(-6) or .73 mil ops/sec for 6.1GB database (Bw-tree)
      - MassTree has lower cost/op at smaller access intervals
      - Bw-tree is lower cost/op at larger access intervals
  - Relating this to caching DB (Bw-tree) breakeven point
    - Access interval breakeven at about 3.1 seconds (for a page of data)
      - MassTree has lower cost for page of data at access interval smaller than 3.1 sec
      - Bw-tree has lower cost for page of data at access interval larger than 3.1 sec
      - Bw-tree has even lower costs at around 45 sec access interval by evicting a page

Analysis Implications

Optimizing Cost to a Performance Requirement

- How does one increase performance at lowest cost
  - Two strategies, depending upon breakeven point
- Hot Data: Increase cache size
  - Bring more hot pages into the larger cache
  - High access rate ensures cost effectiveness
- Cold data: Use more processors
  - Keeping cold data in cache has only a modest impact on performance
    - It decreases only slightly the cache miss ratio
  - Use more processors to access cold data from secondary storage
    - Depends on scalability of system
    - Helped perhaps by data partitioning

Going Forward

- **Data Caching Systems** (like Deuteronomy's Bw-tree) can cover most user performance needs at low cost
  - We should focus on improving data caching systems

NEEDED

- **Main Memory DB Techniques** to improve their main memory operation performance
  - For example– latch free techniques
- **Storage Techniques** to reduce number of I/Os needed
  - Log "oriented" storage to reduce write I/O cost
  - Blind updates and readable recent updates to reduce cache miss rate (uses records already in cache, but not yet integrated with pages)
  - Used by both RocksDB and Deuteronomy's (Bw-tree+LLAMA) key value store
- **I/O operation** execution cost reduction
  - We have been working on that recently