OpenLDAP Developer’s Day, Aug 2004

Adaptive Cache Tuning in OpenLDAP

Jong-Hyuk Choi (jongchoi@us.ibm.com)
IBM Thomas J. Watson Research Center
Enterprise Linux Group

Contents

- **Application-Level Memory Management**
- **Entry Cache vs. DB Mpool**
- **Adaptive Cache Tuning**
  - Memory Pressure Resilient Entry Cache
  - DB Mpool Resizing Mechanism
- **Summary**

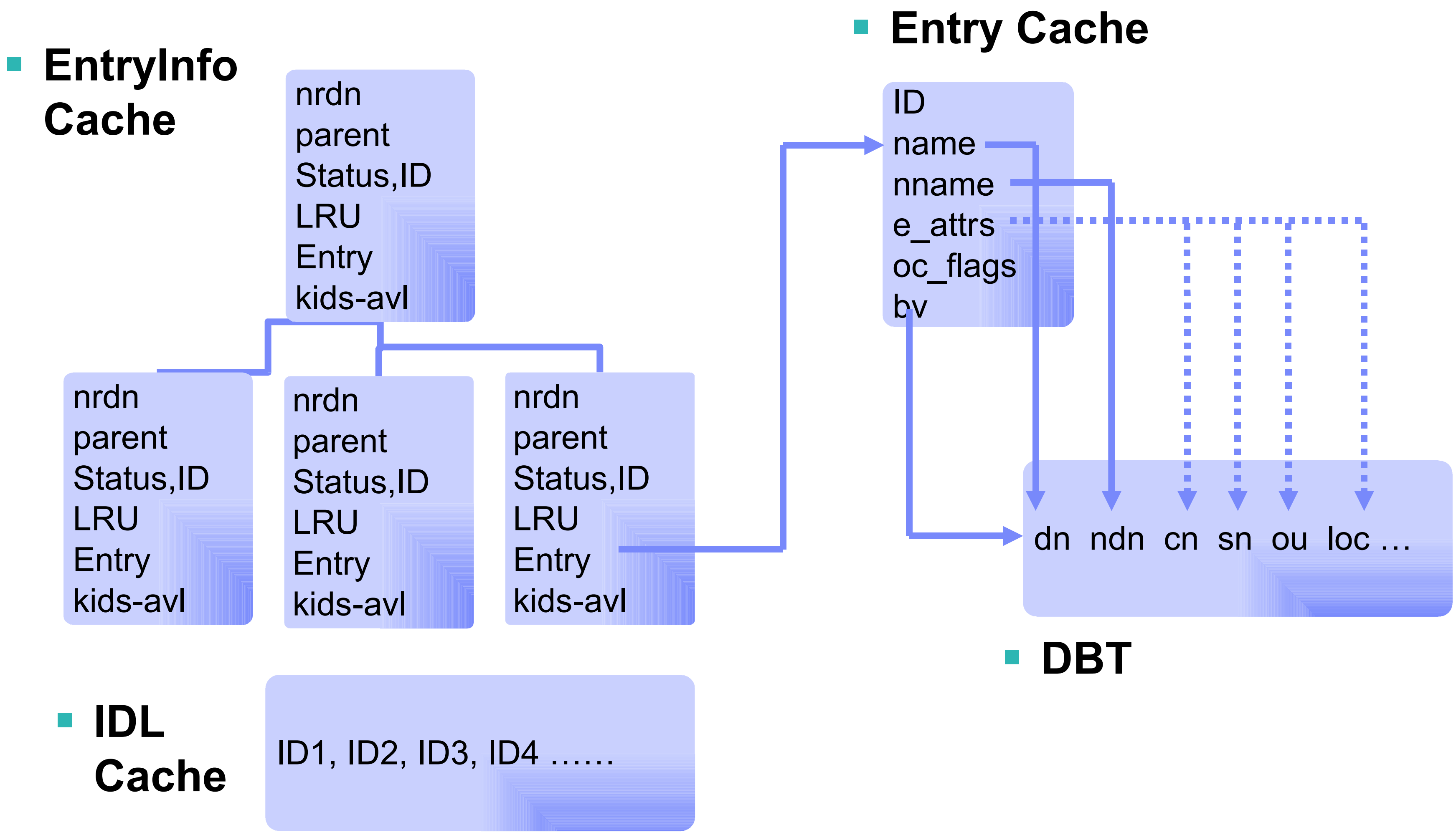
Memory Management

- **Virtual Memory Abstraction**
  - Provides an abstract view of memory
  - Illusion of large address space regardless of physical memory size
  - Does not abstract performance though !
- **64-bit Platforms**
  - Increasing demand of application memory
  - Physical memory size does not scale accordingly
  - Increasing Vitual / Physical ratio
- **Server Consolidation**
  - Over-commit of system memory resource
  - Another level of virtual memory abstraction between OS and VM
  - IBM zSeries zVM, IBM pSeries DLPAR / pHypervisor, VMWare …

Application-Level Memory Management

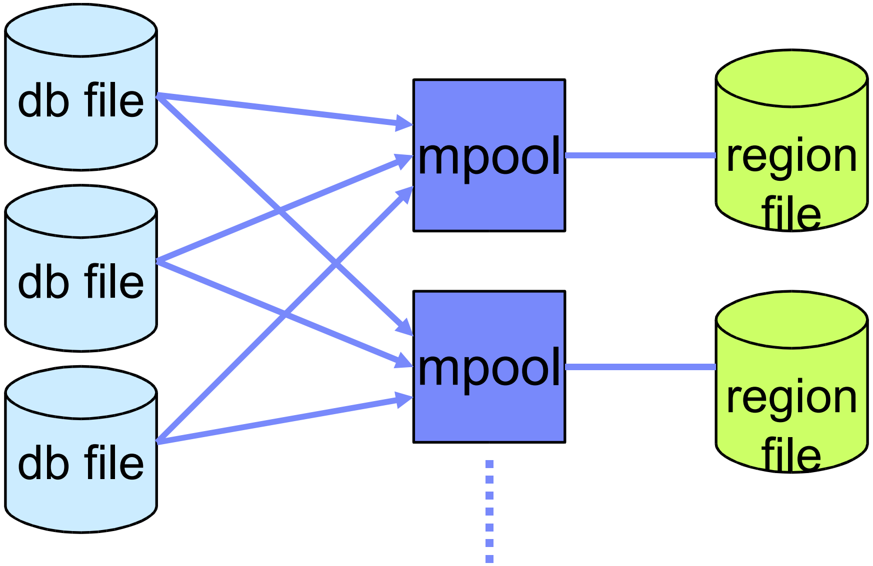
- **Collaborative Memory Management**
  - Collaboration between system layers are essential
    - Applications – Operating System
      - When it is more efficient to construct an object than to rely on lower layer paging mechanism, discard in-memory object
    - Operating systems – Virtual Machine
      - Ballooning driver : relying on OS paging mechanism to collect memory and redistribute to other OS images
      - DLPAR : dynamic resizing of memory resources between LPARs
    - A rule of thumb: it's better for a higher layer to collaborate with the memory management at a lower layer, because the higher the layer is, the more domain-specific information is available

Caches in OpenLDAP



BerkeleyDB Caching

- **Berkeley DB mpool subsystem**
  - General purpose shared memory buffer pool
  - B+Tree, Hash, Recno
  - File mapped / shared memory backed
  - Size is determined upon DB_ENV creation

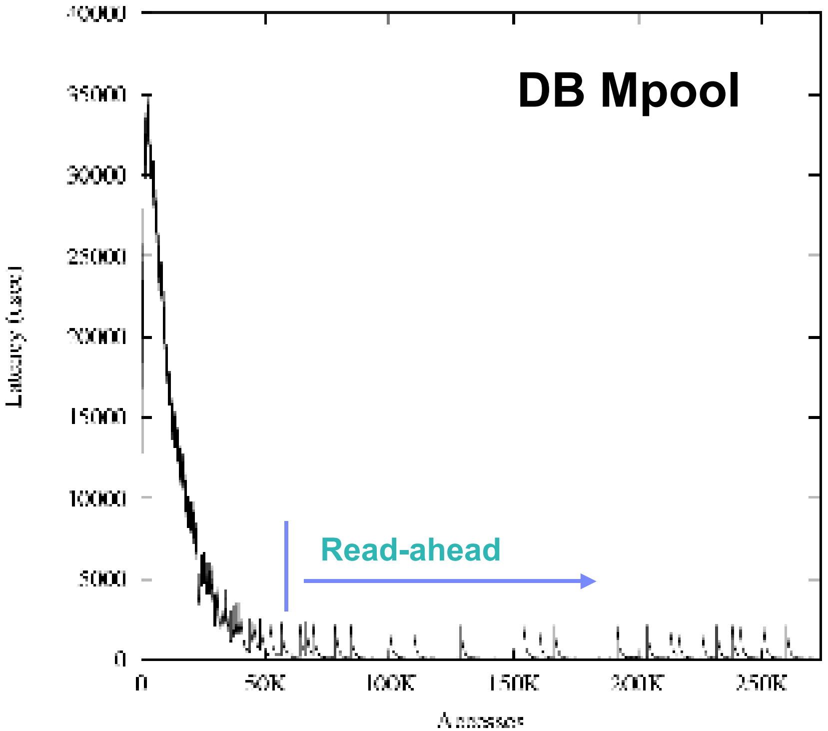


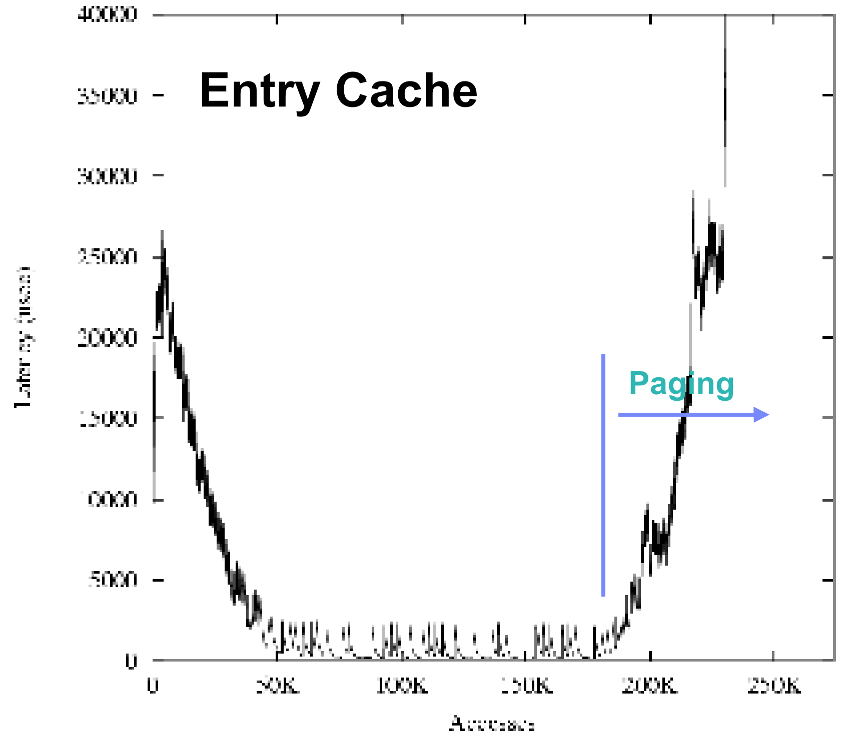
Entry Cache vs. BerkeleyDB Mpool

- **Entry cache**
  - Provides low latency access method for small working set sizes
  - Low hit latency
  - Poor performance under memory pressure – swapping havoc
    - Entry load from DB : write access -> dirty pages -> needs write back
- **DB mpool**
  - Provides caching for large working sets
  - Higher hit latency than the entry cache (10 ~ 1000 times)
    - Access method overhead
    - Data copying from DB mpool to application buffer
  - Good performance under memory pressure
    - Entry load from region: read access -> clean pages -> no write back

Entry Cache vs. BerkeleyDB Mpool: Swapping

- **Sequential access, cold run**
- **Working set > available physical memory size**

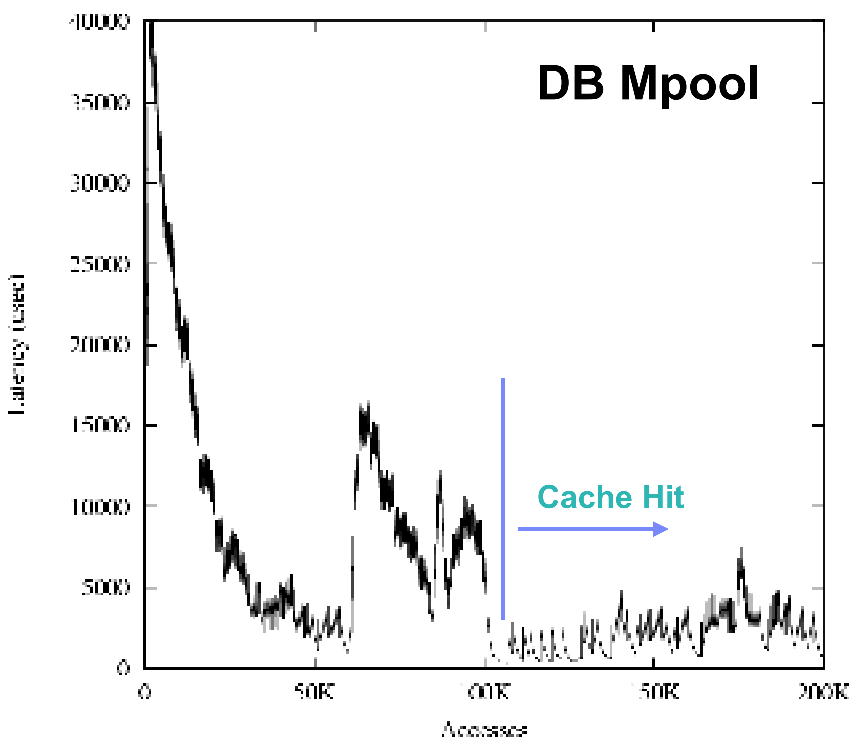


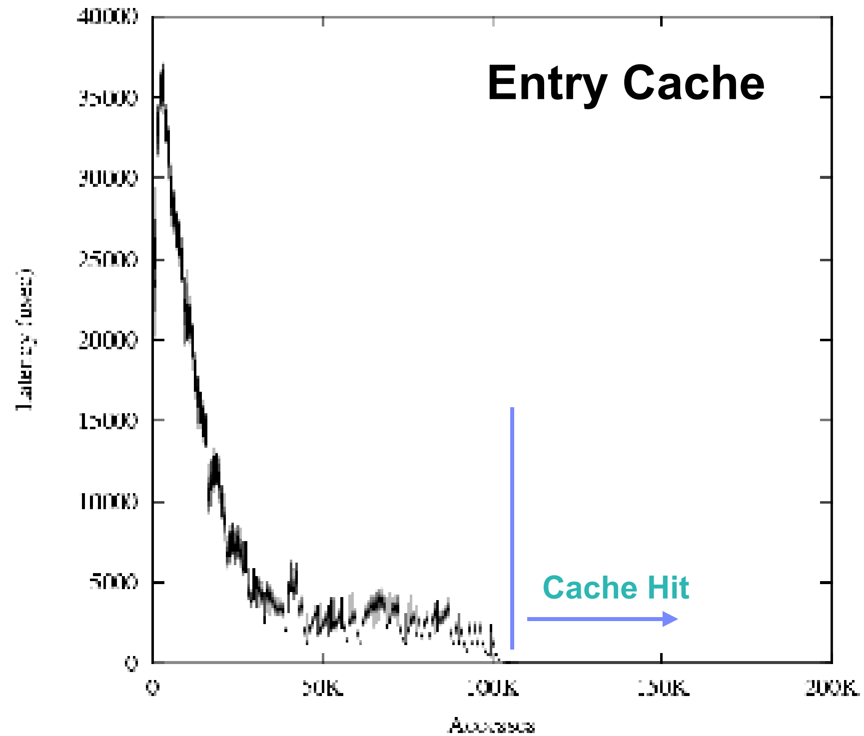


- **Swapping storm can occur even with a balanced initial configuration**
  - Hikes in memory demand due to other applications and/or other OSes

Entry Cache vs. BerkeleyDB Mpool: Latency

- **Non-sequential access, cold run + warm run**
- **Working set < available physical memory size**





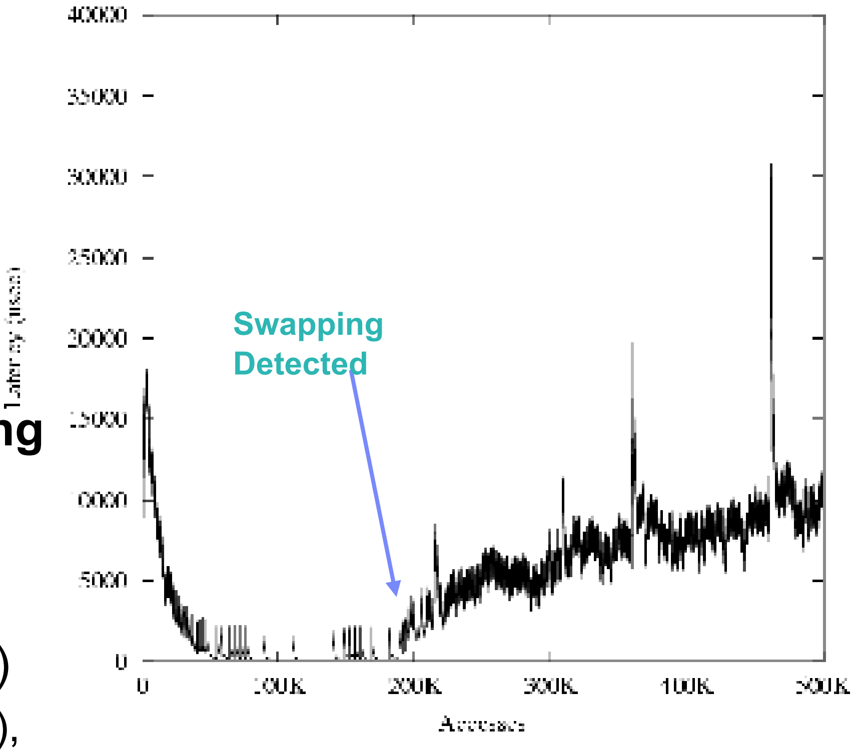
- **Access method overhead / data copying in DB Mpool**
  - Latency increase - Degrades system perf (throughput, cache pollution)

Entry Cache vs. Berkeley DB Mpool

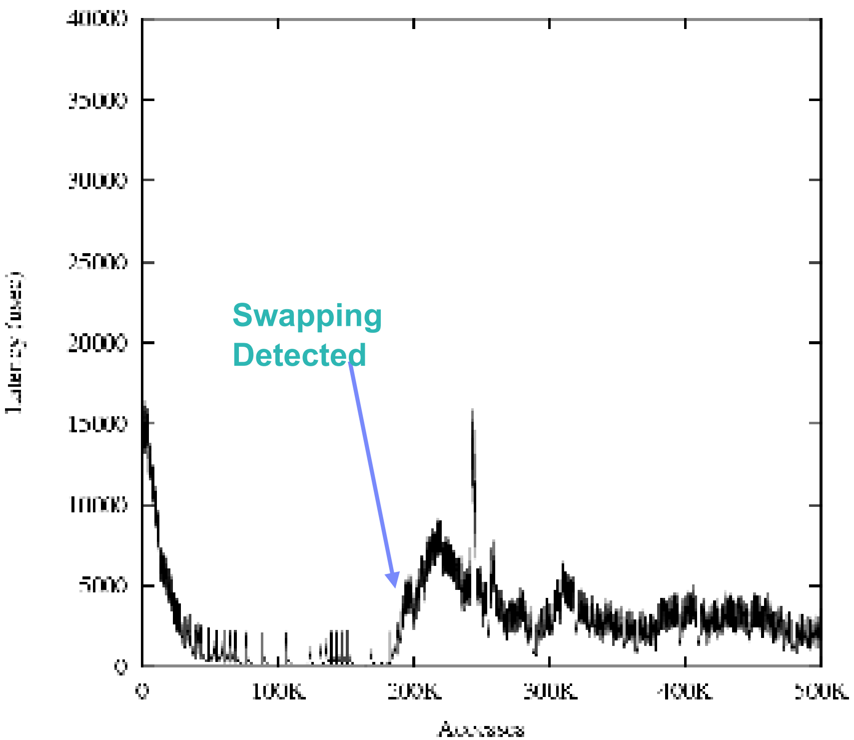
- **How can we utilize both the advantages ?**
  - Entry cache redesign to make it resilient to memory pressure
  - DB cache resizing mechanism

Entry Cache Redesign: Detecting Memory Pressure

- **Limit entry cache size upon detecting memory pressure**
  - OS memory info does not tell the whole story in virtualized environment
  - Monitoring average access latency
  - Sudden incline of average latency curve
- **Huge decrease in swapping storm**
- **Cannot recover completely from swapping storm, because**
  1. OpenLDAP caches are malloc'd
  2. Different OpenLDAP cache objects are collocated and interfere (EntryInfo / Entry)
     - EntryInfo cache has poor locality (AVL tree), hence it makes OS paging algorithm ineffective

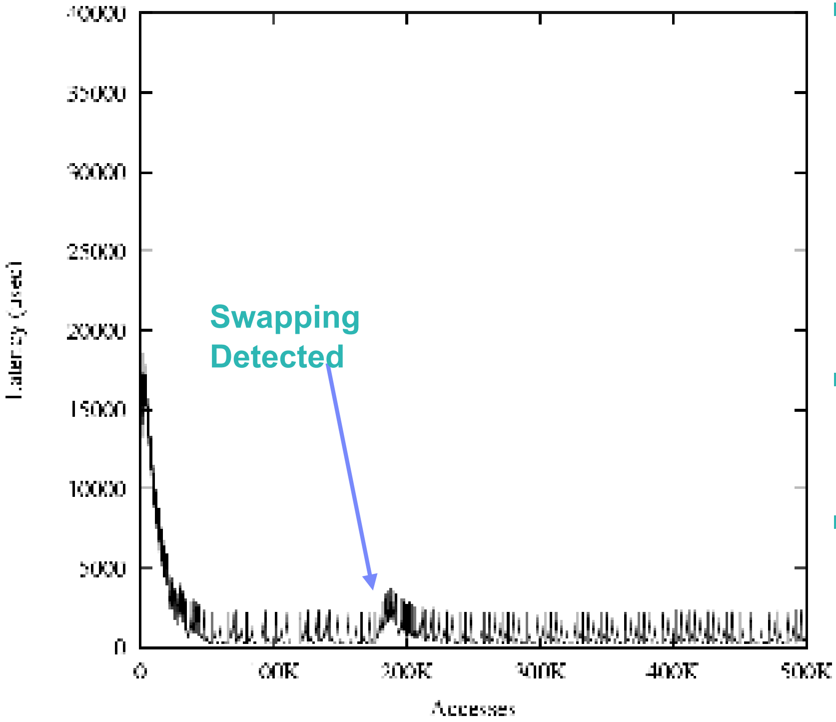


Entry Cache Redesign: Dedicated Object Heaps



- **Dedicated object heap**
  - Breaks interference bw objects
- **Mmap-based entry cache**
  - Allocation / replacement unit: cluster of pages
  - Mapping from /dev/zero
- **Entry cache**
  - Use mmap-based entry cache
  - Entry struct size depends on schema
- **DBT struct**
  - Use mmap-based DBT (DBT_USERMEM)
  - Size depends on stored data (small variance)
- **EntryInfo**
  - Small size, always in addr space, use malloc
- **Much enhanced swapping behavior**
- **Fragmentation problem**
  - Provides slabs for Entry and DBT struct
  - Simple buddy allocator
  - Find cluster size to minimize fragmentation according to the average size of DBT struct
  - Invalidate highly underutilized clusters

Entry Cache Redesign: Avoid Swapping



- **Dedicated object heap + memory use hint to OS**
  - **`madvise(MADV_DONTNEED)`**
  - Zaps the pages in the mapping wo writeback
  - Mapping is still active and COW zero pages will be provided when accessed again
- **When memory pressure is detected**
  - Call `madvise` to release memory wo writeback
- **How to detect an app object is gone ?**
  - Testing non-zero byte in object (Entry, DBT)
  - Compare epoch numbers in EntryInfo and the page cluster
- **Entry cache resizing becomes very efficient**

Resizing BerkeleyDB Mpool

- **BerkeleyDB Mpool can be resized when it's dedicated to a single slapd**
  - Completing outstanding DB operations
  - Removing the DB environment by DB_ENV->remove()
  - Recreating the DB environment with a new cache size
- **The environment resizing overhead turned out to be very small with an appropriate checkpoint setting**
  - Consider resizing when system is under low load condition
- **During DB environment restart**
  - Queues incoming requests temporarily
  - Requests can be serviced out of OpenLDAP caches
  - Return BUSY
- **DB Mpool resizing policy**
  - Increase upon large update latency, Decrease upon small update latency
  - When DB Mpool is resized, resize the Entry cache in the opposite direction

Summary and Further Works

- **Adaptive cache tuning**
  – Taking advantage of both the entry cache and DB mpool
- **Memory pressure resilient entry cache**
  – Use of mmap based memory allocator and memory access hint
  – Entry cache resizing becomes very efficient
- **Resizing DB mpool**
  – DB mpool can be resized by monitoring the latency of updates
- **Further works**
  – A patch for community review
  – Monitoring of cache hit ratio