File Systems for Storage Class Memory

David Pease, IBM Almaden Research Center (pease@almaden.ibm.com)
Darrell Long, U. C. Santa Cruz (darrell@cs.ucsc.edu)

IBM Almaden Research Center

School of Engineering
University of California at Santa Cruz

When Storage Class Memories become available, they will drive profound changes in file system architecture and implementation.

We explore ideas on how file systems could change to support such storage.

File Systems Overview

- Modern file systems come in many types:
  - Local, distributed, cluster
  - General purpose vs. special-purpose

- Yet they provide many of same capabilities and services :
  - Consistent programming API
    - e.g., POSIX, Windows NT APIs
  - Name space
    - Way to name, organize, refer to files/data sets
      - Directory-based, tree-structured (Unix, Windows, Mac)
      - Essentially flat (zOS)

File Systems Overview (cont.)

- In-memory Caching
  - Done for two complementary reasons, both related to the disparity in memory vs. disk access time:
    - For input operations, data is cached in the hope it will be read in the future
      - May have been read or written earlier, may be prefetched
    - For output operations, data is cached to isolate application from slow external device write speeds
      - Data is cached, control is returned, flushed to disk later
        - Can be problematic: system failure can lead to lost or inconsistent data
    - Both are problematic when data is shared across systems
      - Complex, often slow or error-prone locking techniques

File Systems Overview (cont.)

- Often tightly tied to Virtual Memory system of the OS
  - Allows features like memory-mapped file I/O
    - But can be very complex
      - Windows file system/VMM interface very difficult
- Finally, I/O device access
  - Could be very simple:
    - Convert file offset to block address, call device driver
  - But typically is not:
    - Must deal with error conditions
    - May involve read-modify-write, network access, multiple device accesses, parity computation/validation, etc.

Storage Class Memory-based File Systems

- Storage class memories available in the next decade may:
  - Be within a factor of 2-10 times speed of DRAM
  - Be large and inexpensive enough to replace disk drives
  - Drive major changes in file system architecture and implementation
- What will change in the file system:
  - Externals (API, name space) will be slow to change
    - Would require changes to applications, user interfaces
  - Internals
    - Levels below the interfaces will change most quickly
    - Will provide the most immediate benefits

Storage Class Memory-based File Systems

- In-memory Caching
  - Sufficiently fast SCM will obviate need for caching (locally)
    - Fast enough to allow direct access to primary copy of data
    - Eliminates unprocessed writes, program flushes to disk
  - Multiple system access to SCM would provide
    - Simpler locking protocol to allow distributed access to data
    - No concerns for cache consistency
- Byte-level Addressing
  - SCM would have no hardware-enforced block boundaries
    - Eliminates need for read-modify-write for small updates
    - Opportunity to rethink data structures for implementing FS
      - B-trees have been the norm for years
      - Structure with different behaviors could be explored
        - Skip lists, Bloom filters, etc., could be considered

Storage Class Memory-based File Systems

- SCM could be treated as main memory
  - In terms of access and addressing
  - Could use virtual memory capability to map ranges of SCM to application's address space
    - Input operations map SCM ranges as read-only memory
    - Applications that use memory-mapped I/O (mmap) could extend idea to output, as well
  - Loading applications would be a memory-mapping operation
    - Logical extension of shared, read-only mapping used in most modern OSes today
    - Memory speed masked by working set in processor cache
  - If memory address mappings were persistent:
    - Useful inter- and intra-file pointers could be implemented
    - Efficient embedded data structures in files possible

Storage Class Memory-based File Systems

- File systems actually become an interface and name space manager for Memory subsystem
  - Block extent lists replaced by memory address range(s)
  - At lowest level, all files processed as memory-mapped I/O
    - Similar to what's done in several modern Unixes
  - If SCM address space sufficiently large, files could always be mapped to contiguous address space
    - Greatly simplifies file management, address space mapping
  - If storage is memory, what are paging/swapping used for?
    - Paging as we know it could be unnecessary
    - Most pages would never move
    - Virtual Memory only used for mapping into address spaces?

Storage Class Memory-based File Systems

- SCM would have very different failure model from today's disk drives
  - Protection and space-efficiency of today's RAID still necessary
    - But implementation will need to be completely rethought
  - Unlikely that large amounts of data (like disk drive) will become suddenly unavailable
    - More likely that bits, or small ranges of bytes will fail together
  - Erasure codes matched to the importance of individual data files could be used
    - Replace indiscriminate use of RAID for all files in a disk set

Storage Class Memory-based File Systems

- Opportunities for new storage paradigms?
  - One such idea: Semantic file system access
    - Ability to find and access data based on the contents or attributes of the data
      - Rather than through file's directory location and name
    - Not new idea, but slow in gaining momentum
      - Because users are content with current paradigm? or
      - Because technology to implement useful semantic access is lagging?
      - Today's model will not go away, but other options needed
    - Ability to read and index data quickly from SCM may make both indexing and searching more practical
    - Some forms of SCM allow huge, fast content-addressable memories, which could be useful in semantic file systems

Thank you!

Questions/Comments?