Memory

Main issues:
1) Virtual Memory (Multics)
   - description
   - what does it provide?
2) VA to PA (64-bit PT, Mach, VMS)
3) Abstraction of data (Mach, Multics)
   - memory objects
   - segments
4) I/O buffering (VMS, Mach, Working Set)
5) Locality improvement (CCNUMA, NUMA, Appl. Driven, Working Set)

Provided by virtual memory:
1) Automatic storage allocation
   - everything doesn't have to be in main memory
   - this is what everyone thinks about
   - programmer doesn't have to worry about data placement and overlays
   - programmer doesn't need to know demands or specific details of other segments it uses
2) Modular organization
   - everything is organized in terms of segments
   - allows combination of separately compiled, reusable, and sharable components without prior arrangement and without manual linking into address space ("making a segment known")
3) Protection
   - each segment has its own protection status
4) Object orient programming
   - allows only managers of objects to access and modify them (private data)
   - falls out of 2 & 3
5) Parallel computations on multicomputers
   - all memories joined into single address space, reducing communication due to messages

Virtual to Physical Address Mapping

Address space – set of addresses available for processes use
Ex.) VMS divides address space into 3 components (code, control, system) each having its own page table.
Ex.) Many systems divide address space into user space and system space

Two techniques for mapping:
1) Page table
2) Address Map

Page Tables:

- large, sparse address spaces will get you giant page tables with little locality
- linear bad because gigantic.
- multilevel and forward-mapping bad because they require multiple memory accesses on a page fault to go through levels of page table.
- Hashed bad for dense address spaces because each entry has a fixed (large) overhead.
- Suggested solutions were use of superpages (to reduce overhead) and clustered page tables.
- Clustered page tables good because they amortize the overhead of hashing over a number of pages. Thus, for dense address spaces less room is taken up.

Address Map:

- contiguous portions of memory objects mapped into logical address space
- given VA, memory manager must determine (memory object, offset). This gets sent to the memory object who then handles reading in of the data.
- can have shadow objects (for copies between different address maps) and share maps (one for each shared memory object)
- compact representation of data
- typically small
- make overhead of large sparse address spaces manageable

Abstract Concept of Data
Types:

1) Memory objects (Mach)
2) Segments (Multics)
3) Vs. Memory mapped files

Comments:

- representation of data in secondary storage
- reduces overhead of copying of data
- retains characteristics of data (length, access rights, etc)

Buffering of I/O

1) VMS
   - swapper – swaps out entire resident set of process being swapped out
   - pager – buffers evicted pages, amortizes I/O costs over several pages
2) Mach
   - object cache buffers non currently referenced pages in memory because likely to be used by another process soon.

Importance of Locality

1) Working Set
   - program (over time) consists of phases of locality separated by transition periods lacking locality

- window over time will give us a working set
- requires counter for feedback
- only allow process to run if there is room for its working set. This strategy eliminates thrashing.

2) Application Driven
   - application knows better how to allocate its memory given resource restrictions. Cost model which allows you to appropriately combine processor and memory usage (over time).
3) NUMA, CCNUMA
   - automate paging
   - use migration and replication
   - statistics collection required
   - false sharing is bad