Things Change

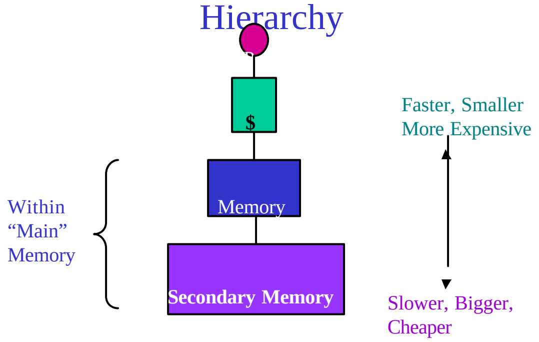
- Myth that placement is irrelevant
- View that OS is concerned only with the main-secondary levels of memory hierarchy
- New architectures / new views of the memory "hierarchy"
- Scale - larger address spaces
- Workload assumptions
  - New things to do with memory management

You are here

Non-Traditional Memory Hierarchies

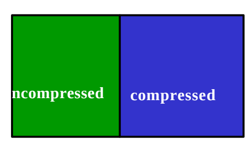
- Rambus Memory - a power-aware hierarchy
- Compression Cache - Douglis
- Somebody-else's memory (remote memory)
  - NUMA (non-uniform memory access)
  - DSM (distributed shared memory)
  - GMS (global memory systems)

Broad Definition of Memory Hierarchy



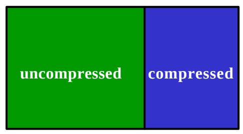
Compression Cache (Douglis)

- Compressed pages in memory form an intermediate level in the storage hierarchy between uncompressed pages and backing store
- Dynamically vary amount for each purpose.

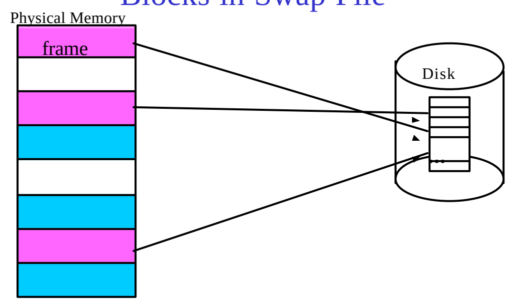


Compression Cache (Douglis)

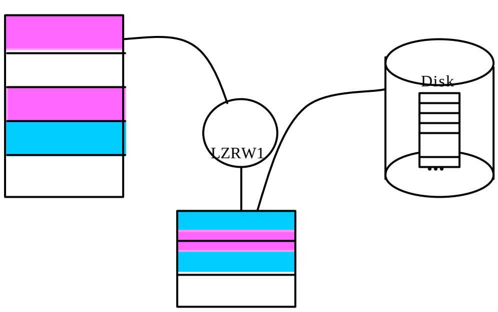
- Compressed pages in memory form an intermediate level in the storage hierarchy between uncompressed pages and backing store
- Dynamically vary amount for each purpose. Based on Sprite.



One-to-one Mapping of Pages to Blocks in Swap File



Compression Cache



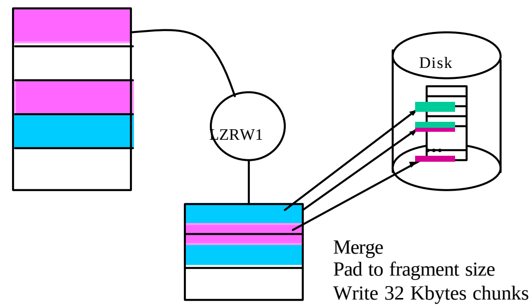
Implementation Issues

- How to balance need for physical memory between uncompressed pages (VM), compression cache (CC), and file system buffer cache (FB)?
- Variable size pages lose the simple mapping to file blocks in swap file.
- Bookkeeping to find compressed pages and maintain structure of CC.

Structure of CC

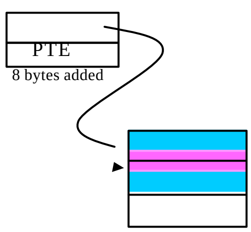
- Variable size circular buffer.
  “Oldest” physical page in CC
- Pages are compressed onto tail of CC
- CC has it’s own “replacement” policy to reclaim CC page frames - oldest clean page
- Grow or shrink CC: compare the *age* of the oldest (LRU) page in each category of memory use with bias of CC pages over VM over FB.
  - Strength of bias determines growth rate. Application dependent

Interface to Backing Store



Bookkeeping & Overhead

- Hash table for compression algorithm
- Page table changes
- Header (24 bytes) in each physical page frame used for CC
- Header (36 bytes) for each virtual page compressed into CC



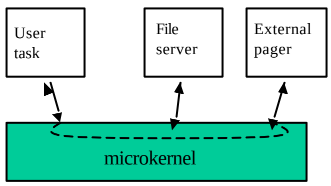
Performance

App	Time std	Time CC	Speed up	Comp ratio	% Un
compare	**16.14**	**6.04**	**2.68**	**31**	**.1**
Sort Random	**26.17**	**28.51**	**.9**	**37**	**98**
Gold cold	[illegible]	**56.4**	**.8**	**60**	**10**

Bad compression

Write I/O intensive

Mach Microkernel



Fault Handler*

- Kernel does lookup of v.a. in task's address map –> object/offset
- Kernel tries to find if it's resident in object/offset hash table –> page, if successful; otherwise request from pager.
- Kernel informs pmap of v.a. –> p.a. mapping to install

*ignoring copy_on_write issues

Kernel - Pager Interactions

- Kernel to Pager
  - pager_init,
  - pager_data_request
  - pager_data_write
    write back
  - pager_data_unlock,
  - pager_create
    accept new responsibility
- Pager to Kernel
  - pager_data_provided,
  - pager_data_lock
    requests cache access
  - pager_flush_request
    invalidate cache
  - pager_clean_request
    force cache writeback
  - pager_cache allow caching
  - pager_data_unavailable

Potential problems

- What if user-level pager doesn't return data?
  - timeout
- What if user-level pager doesn't free memory?
  - Timeout and page it out to a default pager
- What if user-level pager takes a page fault itself?
  - Reserve a memory pool for pager allocations.