Recollection Principles

8/14/07

These principles concern how computations store and recall information, and how data layout in the storage system affects their performance.

All computations take place in storage systems.

A. Storage is essential for computation: all representations (of data and instructions) and their states must be held in a medium as the computation proceeds.

B. A storage system consists of various storage devices of different media and access times, with methods for storing and retrieving data.

C. The arrangement of data in storage significantly affects the resolution time of an algorithm. For example, searching for an item in a set takes O(n) time if the n items are unordered, and O(log n) time if they are ordered.

D. A flat storage system is a model in which every item of storage has the same access time. In such a medium, it is reasonable to associate resolution time of an algorithm with the number of steps it takes to complete a computation.

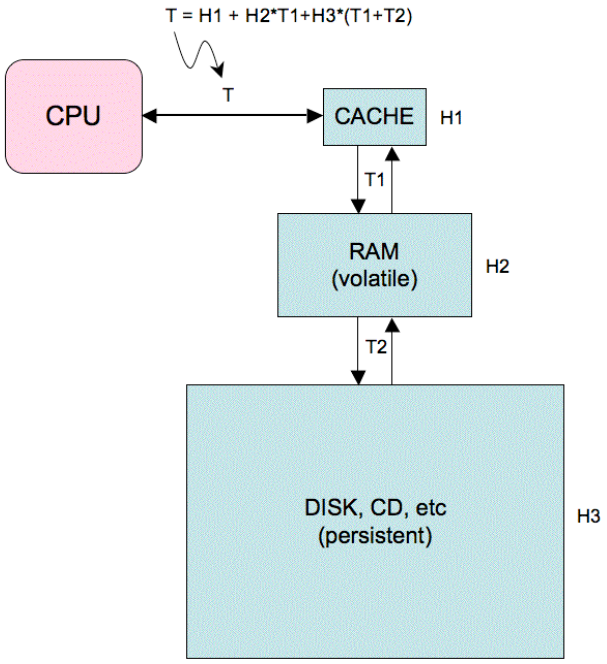
E. Storage systems are not flat.

F. The arrangement of data across devices of a non-flat storage system significantly affects the resolution time of an algorithm. For example, multiplying two nxn matrices takes a simple loop with computation time $O(n^3)$. But if the flat part (RAM) can hold only 3 rows, an additional n^2 load operations will be needed to bring data into the RAM (each of the n^2 result matrix entries requires at least one row or column to be loaded). Because the load operations are much slower than individual instructions (typically by 10^6), the loading cost can easily overwhelm the computation cost. Moreover, planning these overlays and adding the necessary load instructions to the program can double or triple the programming time.

Storage systems comprise hierarchies with volatile (fast) storage at the top and persistent (slower) storage at the bottom.

A. Storage devices are volatile or persistent. Volatile means that the stored data can be maintained only if energy is constantly supplied to the medium (e.g., RAM). Persistent means that the stored data are inscribed in the medium and will remain there without additional energy until erased (e.g., hard disk). Volatile media are fast; persistent media are much slower but also much cheaper. Most computing systems use volatile storage (e.g., RAM) for holding data accessed directly by CPU (or any processing elements), and persistent storage (e.g., disk) for holding data over indefinite periods independent of any CPU.

B. Data are organized into blocks, which are units of storage and transfer. Blocks all of the same size are called "pages". Blocks of different sizes are called "segments". The physical places in which pages are stored are called "slots" or "frames" in RAM, and "records" on disk. We will use the generic term "object" for a block of storage.



C. A storage hierarchy has the fastest (and most expensive) devices at the top and progressively slower (and less expensive) devices farther down. A master copy of data resides in the persistent levels. An UP operation pulls a copy of an object to the top of the hierarchy. A DOWN operation pushes a copy of a modified object down the hierarchy, updating the master when it reaches that level. A typical hierarchy has levels for CACHE, RAM, and DISK.

D. The hit ratio H of a level is the fraction of references from CPU whose data are in that level but not higher or lower. The hit ratios sum to 1. The transfer time T of a level is number of cache reference times to move a requested block UP to that level from the next lower level. The effective access time seen by the CPU is a hit-ratio weighted sum of the times for CACHE, RAM, and DISK.

E. UP (and DOWN) operations can be automated. Typically UP operations are issued on demand -- when CPU attempts to access object that is not in top-level memory. When confidence in prediction is high, UP operations can be issued well before the object is needed.

F. A replacement policy determines which object in CACHE or RAM must be moved down to make way for an up-coming object. Replacement policies try to maximize their hit ratios (H) or equivalently minimize their miss ratios (1-H).

G. Typically the DISK level transfer time T2 is 10^5 or more times T1; DISK transfers are costly and dominate the performance. The DISK hit ratio must be much less than 1/T2 to keep the effective access time under twice the cache access time. This is a tough performance requirement for replacement policies.

The principle of locality dynamically identifies the most useful data, which can be cached at the top of the hierarchy.

A. The optimality principle for replacement policies is to replace the object that will not be needed again for the longest time. By deferring the recall of an object as long as possible, the rate of recall (total number of UPs after initial loading) will be minimum. Real replacement policies cannot use lookahead; they make approximate predictions about the future.

B. Locality is the principle that a computation clusters its references during every phase of execution into small subsets of its objects. called locality sets. Since objects in the current locality set are all much more likely to be referenced before objects outside, a replacement policy that learns locality sets and protects them from replacement will be near optimal.

C. Locality is a fundamental behavioral property of all computations. Its primary cause is limitation of human attention span: humans tend to approach problems by focusing on parts and solving them before moving to other parts. This is called temporal locality.

D. The secondary cause of locality is organization of data. Each data object is linked to a limited number of “neighbor” objects. A

computation is most likely to access a neighbor of a current object in the near future. This is called spatial locality.

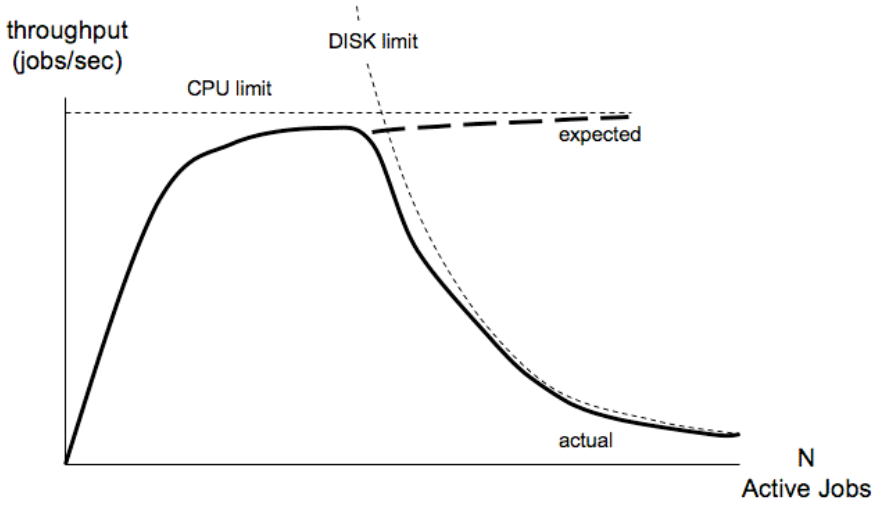
E. A cache is a (hardware or software) memory device designed to hold locality sets of a computation. CACHE is very fast relative to the RAM.

F. Working set is a measure of locality set. It is usually determined from the usage bits of objects during a fixed time window into the recent past. High performance memory management seeks to hold working sets in cache.

G. It is sometimes argued that flat memory (no hierarchy) would obviate the need for measuring working sets because all objects are available in the one space with uniform access times. However, locality tells us that some objects will be used more than others. With multiple users, that implies queueing and congestion at the most popular objects, which can significantly increase effective access time to those objects. This congestion can be avoided by caching copies of popular objects at multiple points in the flat memory space. The benefit of caching in this case is to reduce queueing delay, which is a substantial component of access time. Web caching is an illustration.

Thrashing is a severe performance degradation caused when parallel computations overload the storage system.

A. Operating systems use multiprogramming to load multiple processes into main (top-level) memory so that when one stopped for an UP operation, the CPU could switch to another. Early multiprogramming systems unexpectedly exhibited thrashing, a sharp throughput drop when the multiprogramming level passes a critical threshold. The processes generated high demands for UP operations and overloaded the disk units, creating long delays in waiting for UP operations. The picture shows that throughput was expected to approach the CPU saturation limit, but actually plummeted because of increasing page-fault congestion at the DISK. Thrashing was avoided in systems that measured working sets and limited multiprogramming to processes whose working sets could be fully loaded.



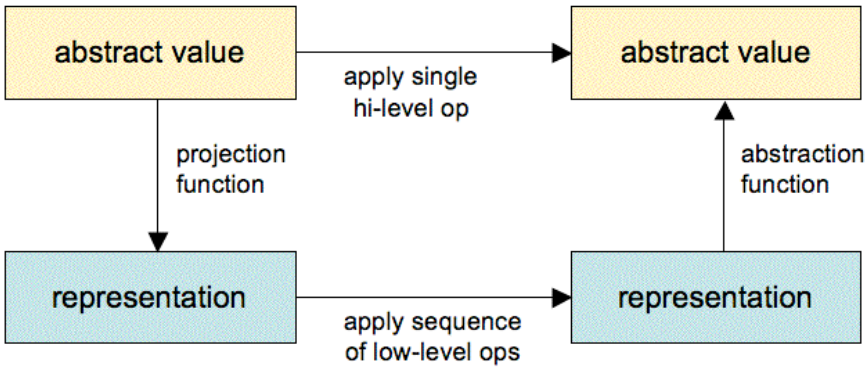
B. Thrashing has been observed in many other contexts where multiple processes contend for a shared resource and the protocol they use to determine who goes next has overhead that increases with the number of contenders. Eventually the overhead of contention resolution reduces the contenders' capacities to do work. Databases with shared locks and early packet radio networks illustrated this form of thrashing.

Access to stored objects is controlled by dynamic bindings between names, handles, addresses, and locations.

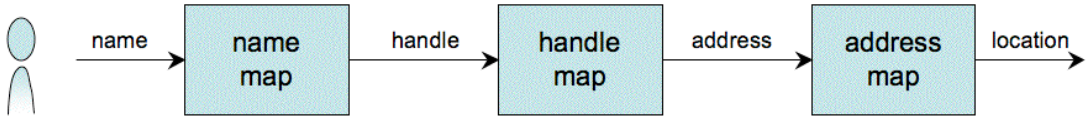
A. A storage object is a data container in some agreed format with a set of allowable operations. For example, a page is a block of consecutive addresses of an agreed fixed length; the allowable operations are read and write for offsets into the page. A file is a sequence of bytes; the allowable operations are open, close, read, and write. A directory is a set of entries that associate symbolic names chosen by users with internal addresses of storage objects; the allowable operations are enter, remove, rename, and search.

B. A virtual (storage) object is a simulation of the object using the available mechanisms of the storage system. A virtual memory, for example, simulates a large main memory using a small RAM, a hard disk, an address mapper, a page fault handler interrupt routine, and a page replacement algorithm. The CPU simply issues read or write requests for addresses in the virtual memory; the system automatically converts each request into an appropriate sequence of address mappings, UP commands, and DOWN commands.

C. The principle of virtualization is depicted in the accompanying figure. At the simulated (abstract) level, one sees only the abstract object and the high-level operations. The system maps the current value of the abstract object down to a lower-level representation, performs a series of low level operations to simulate the requested high level operation, and maps the result back up to an abstract value.



D. The virtual memory was one the earliest examples of virtualization. The main abstraction was the address space, a simulation of the RAM. The standard operations (read, x) and (write, x) were issued with each instruction to any address x in the address space. The address space is mapped down (projection) to a configuration of pages in RAM and on disk; the exact configuration is recorded in a page mapping table. The system translates the virtual address x to a physical address in RAM or disk using the mapping table. Then it performs low level operations such as presenting the physical address to RAM, creating a page fault, selecting a page to replace, moving the replaced page DOWN, and moving the missing page UP. This sequence of low level operations leaves the system in a new configuration, which, when mapped up (abstraction), looks to CPU like the proper new state of the address space.



E. Modern storage systems, from those on individual computers to the entire Internet, use several levels of abstraction (virtualization) to realize all the goals of the system:

names

handles

addresses

locations

Where

- Names are symbolic strings chosen by users to name their objects. Users have extreme difficulty remembering machine-readable addresses (strings of bits). This level lets them give their own name for objects; the storage system maps the names to locations and accesses the objects for the user.
- Handles are system-generated identifiers that are globally unique and are never reused. Handles distinguish all objects and all versions of the same object, allowing anyone at any time to access the object regardless of any local names assigned it.
- Addresses are bit-strings identifying individual bytes of an address space. The compiler, which creates the address space, initially generates them and thereafter the CPU issues them as it executes instructions.
- Locations are bit strings used by the hardware to identify specific physical locations.

F. We represent the levels of virtualization as a series of dynamic maps: name → handle → address → location. The operations at each level are fixed, but the mappings from the abstract state at one level to the representation at the next lower level change over time. Those mappings, called bindings, enable the same abstract state to be represented by many physical configurations invisible to the user.

G. Mapping tables, which represent dynamic bindings, associate an item at one level with the corresponding item at the next lower level. In a virtual memory, for example, the page table records current associations between pages and memory frames. In a file system, the directory structure records the associations between paths and handles. In the Internet, the Domain Name Service records the association between host-names and IP addresses.

H. The process of searching a map takes time, and since computations are constantly accessing storage objects, accumulated mapping time can easily exceed computation time. To avoid this, most mapping systems incorporate a small high-speed cache that holds the most recent items and their mapped results. For example, the mapping cache in a virtual memory holds recently mapped pairs (address, location); if the CPU generates an address in the cache, it can quickly recover the location without consulting the page table. The principle of locality ensures a high hit rate in the mapping cache, which means that most items can be mapped in miniscule time and the total mapping time can easily be limited to 1% of computation time or less.

I. In a dynamic map, an object named at a level may be unknown at that level. In this case, the mapping tables yield a "not found" indicator when searched. Such a mapping fault creates an interrupt that invokes a process in the operating system to find the missing object from the master copy in the lower levels of memory; thereafter, accesses to that object will map without fault. In a virtual memory, for example, every page table entry contains a presence bit set to 1 when the corresponding page is in RAM; if the page is missing, the bit is 0, and the page fault causes the operating system to fetch the missing page from disk and update the page table.

J. Dynamic bindings provide location independence: neither a user nor a CPU needs to know the physical location of an object to access it. Objects can be relocated to new locations by updating only the final map (address → location).

K. Dynamic binding offers numerous other advantages including spontaneous sharing, logical partitioning, artificial contiguity, and dynamic relocation. It also increases programming productivity since programmers do not have to incorporate solutions to these issues into their programs.

L. In many cases, the bindings from names to addresses are static. A compiler changes all the symbolic variable names of a program to addresses. Symbolic names outside the module being compiled are tallied unresolved in an external symbol table; a separate linker (or make-file) program takes a set of separately compiled modules, resolves their external references by substituting addresses in other modules containing the referenced objects, and produces a complete address space ready to execute. In this case, the only binding that can be managed dynamically by the system is address → location.

M. When viewed as a large storage system, the Internet also conforms to this pattern. Names are URLs (uniform resource locators), consisting of a hostname followed by the pathname of that object on the host.

There are no handles in the standard Internet protocols. Addresses are 32-bit IP addresses, represented as four integers (0-255) separated by dots, for example 192.168.1.1. Locations are MAC (media access control) addresses assigned to network connector cards; for example, an Ethernet identifier is 48 bits represented as six two-letter hexadecimal codes (such as 00:0a:95:c4:1f:74). Thus the standard mappings for Internet are: URL (name) → IP (address) → MAC (location).

Hierarchical naming systems allow local authorities to assign names that are globally unique in very large name spaces.

A. Users have to deal with tens of thousands of file names in their own systems, and (potentially) billions or trillions of names in the full Internet. The hierarchical naming principle is a way of constructing a local name that is unique within the entire name space.

B. The hierarchical naming principle organizes all objects in the name space into a tree whose internal nodes are directories. A directory is a list of object names and their handles, with no duplicate names. Pathnames in such a hierarchy are unique. Postal addresses, phone numbers, and organization charts illustrate non-computational, hierarchical naming systems.

C. Thus a directory hierarchy maps pathnames to handles.

D. The pathnames are global symbolic names for objects. Every object has a unique pathname. Therefore pathnames can be used to share objects.

E. The Internet is a large name space with URLs as the names. A URL (uniform resource locator) is a host-name concatenated with a pathname in the host's directory tree. This embeds all host trees as branches from a global Internet tree.

F. Because pathnames can be reused, the same name can designate different objects at different times. Therefore, the Internet URL naming system does not guarantee that a name one acquires points to the same object it did at the time of acquisition. This can be remedied by adding a version number to a directory entry so that, if the owner reuses the pathname for a new object, the old object is still available under a previous version number. In the Internet, which has no handles, it can be remedied by overlaying a system of handles on top of the Internet. A time-unique Digital Object Identifier (DOI) system can be defined for every object along with a mapping service that maps DOI to URL, giving the bindings DOI → URL → IP → location.

Handles enable sharing by providing unique-for-all-time object identifiers that are independent of all address spaces.

A. A handle is a bit-string containing a unique identifier, an access code granting read or write privileges to the handle's holder, and an object type indicator (used by operations to check that incoming handles are of the expected type). The unique identifier, usually composed of a time stamp and a machine identifier, is likely to be several times longer than the normal address length of the machine (e.g., 128-bit handles on a 32-bit machine).

B. Original virtual memory systems relied on the bindings names → addresses → locations. They had no handles. The first binding, names to addresses, was done by compiler and linker and was not dynamic. The second binding, addresses to locations, was implemented dynamically by the virtual memory system. Sharing was problematic because other users could not address anything in the object owner's address space; and putting the shared object in public address created unacceptable security risks.

C. The handle solved this problem by allowing each user to map local names to the global handles. Handles are essential for sharing.

D. If the system prevents users from modifying handles, possession of a handle becomes proof of permission to access an object. This idea was at the core of the capability machines produced in the 1970s and is at the core of modern object-oriented systems.

Data can be retrieved by name or by content.

A. In addition to name-based addressing, memory systems also provide content retrieval for objects.

B. With content retrieval, the user specifies keywords or attributes and the memory returns a set of matching objects. Databases and Internet search are like this.

C. Data can be organized to facilitate fast retrieval. An index, for example, maps keywords to lists of objects containing them. A fast search of the index yields the matching object list quickly. Internet search is like this.

D. Internet search is exceptionally demanding because search engines must infer broader meanings from the few keywords a user provides and because the search database requires huge computing facilities to respond rapidly to queries.

