Performance considerations

V.Saranya
AP/CSE
Sri Vidya College of Engineering and Technology,
Virudhunagar

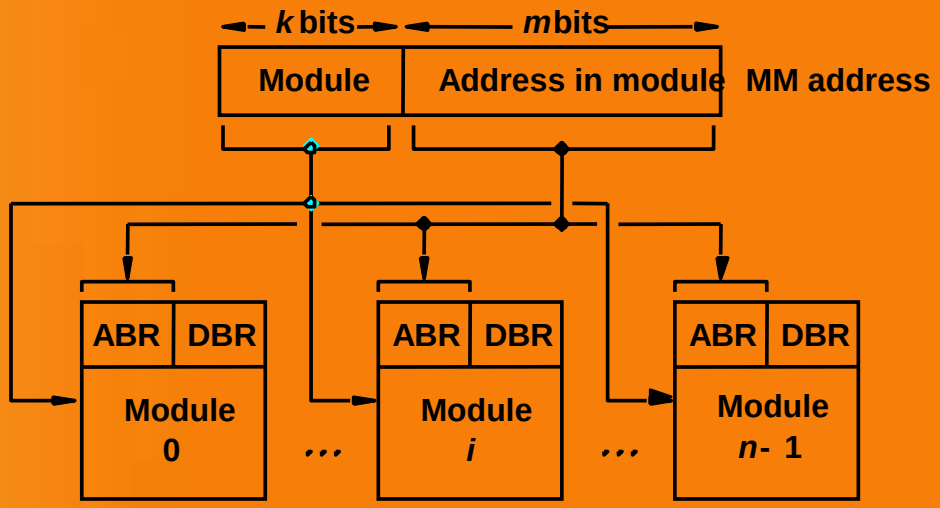
Performance considerations

- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
  - Price/performance ratio is a common measure of success.
- <u>Performance of a processor depends on:</u>
  - How fast machine instructions can be brought into the processor for execution.
  - How fast the instructions can be executed.

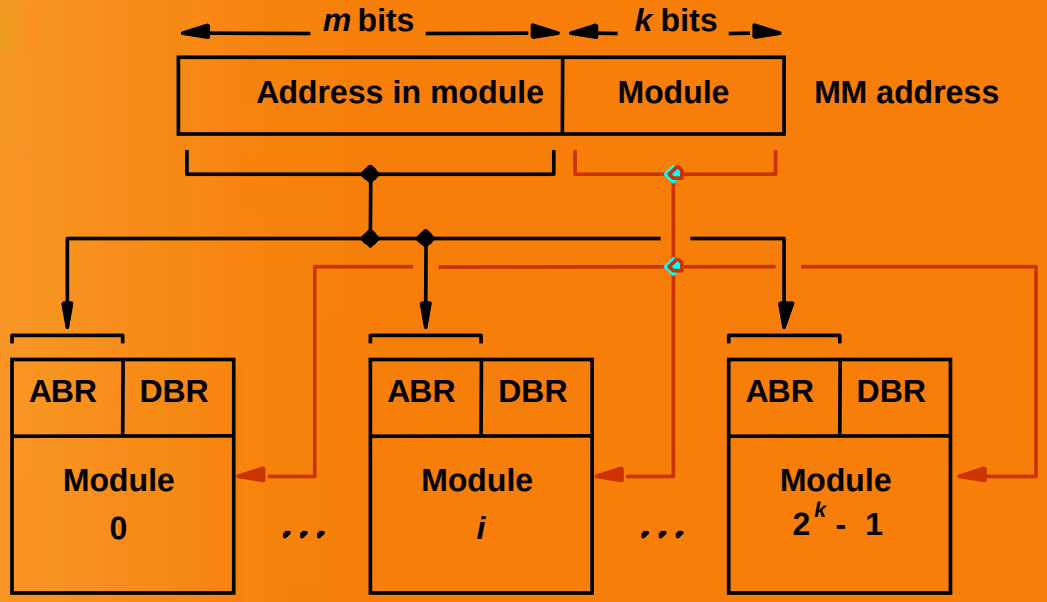
Interleaving

- **Divides the memory system into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).**
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Methods of address layouts



- *Consecutive words are placed in a module.*
- *High-order k bits of a memory address determine the module.*
- *Low-order m bits of a memory address determine the word within a module.*
- *When a block of words is transferred from main memory to cache, only one module is busy at a time.*



- *Consecutive words are located in consecutive modules.*
- *Consecutive addresses can be located in consecutive modules.*
- *While transferring a block of data, several memory modules can be kept busy at the same time.*

Hit Rate and Miss Penalty

- Hit rate.
- Miss penalty.
- Hit rate can be improved by increasing block size, while keeping cache size constant
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Caches on the processor chip

- In high performance processors 2 levels of caches are normally used.
- Avg access time in a system with 2 levels of caches is

$$T_{ave} = h1c1+(1-h1)h2c2+(1-h1)(1-h2)M$$

Other Performance Enhancements

Write buffer

- ***Write-through:***
- *Each write operation involves writing to the main memory.*
- *If the processor has to wait for the write operation to be complete, it slows down the processor.*
- *Processor does not depend on the results of the write operation.*
- *Write buffer can be included for temporary storage of write requests.*
- *Processor places each write request into the buffer and continues execution.*
- *If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.*

Write-back

- Block is written back to the main memory when it is replaced.
- If the processor waits for this write to complete, before reading the new block, it is slowed down.
- Fast write buffer can hold the block to be written, and the new block can be read first.

Other Performance Enhancements

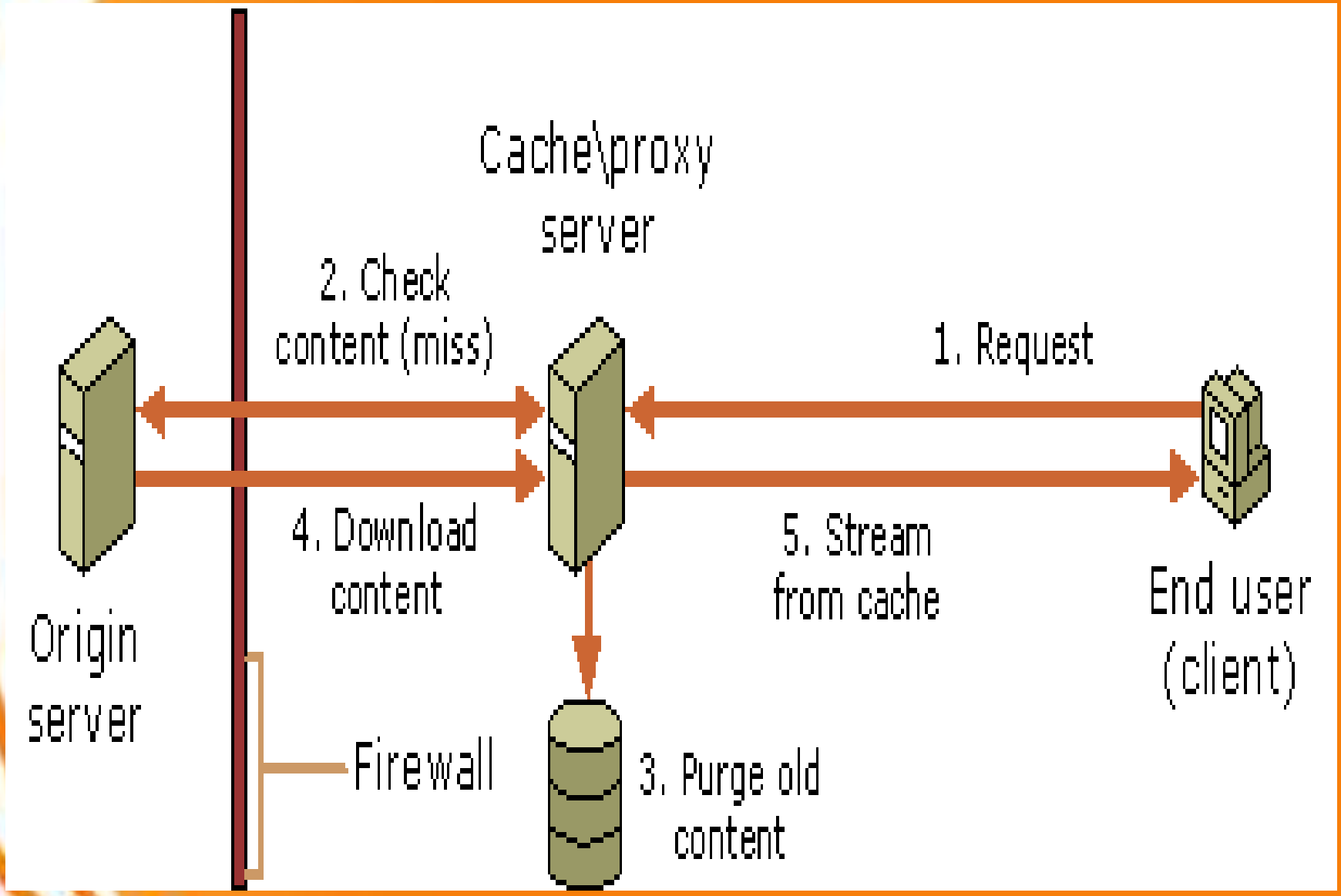
Prefetching

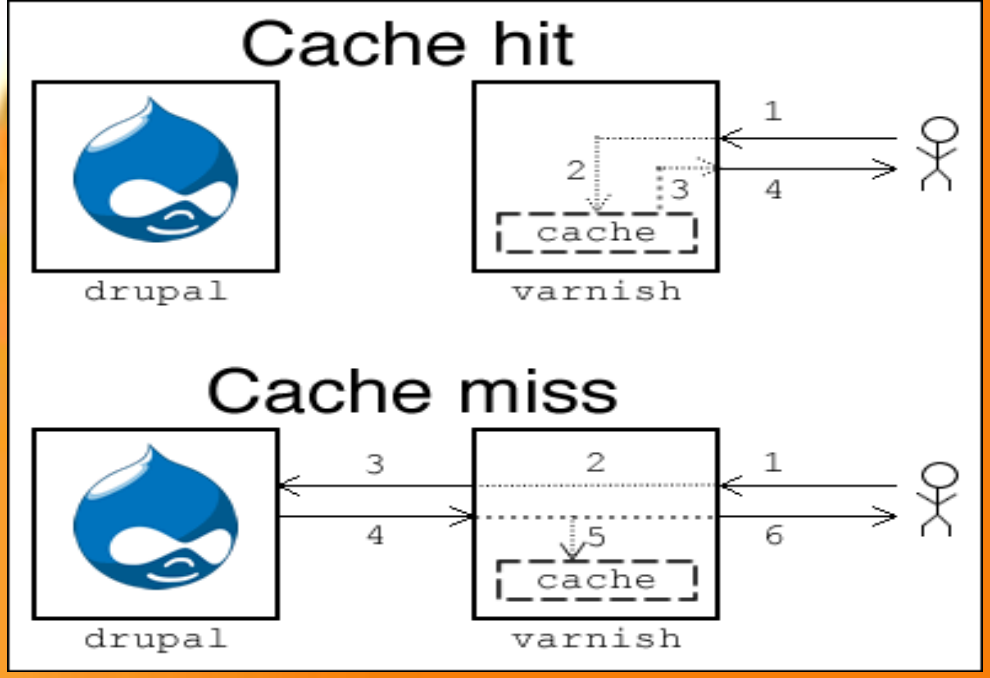
- *New data are brought into the processor when they are first needed.*
- *Processor has to wait before the data transfer is complete.*
- *Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.*
- *Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.*
  - *Inclusion of prefetch instructions increases the length of the programs.*
- *Prefetching can also be accomplished using hardware:*
  - *Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.*

Other Performance Enhancements

Lockup-Free Cache

- *Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.*
- *A cache of this type is said to be "locked" while it services a miss.*
- *Cache structure which supports multiple outstanding misses is called a lockup free cache.*
- *Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.*
- *Special registers may hold the necessary information about these misses.*


Cache\proxy server
2. Check content (miss)
1. Request
4. Download content
5. Stream from cache
Origin server
End user (client)
Firewall
3. Purge old content



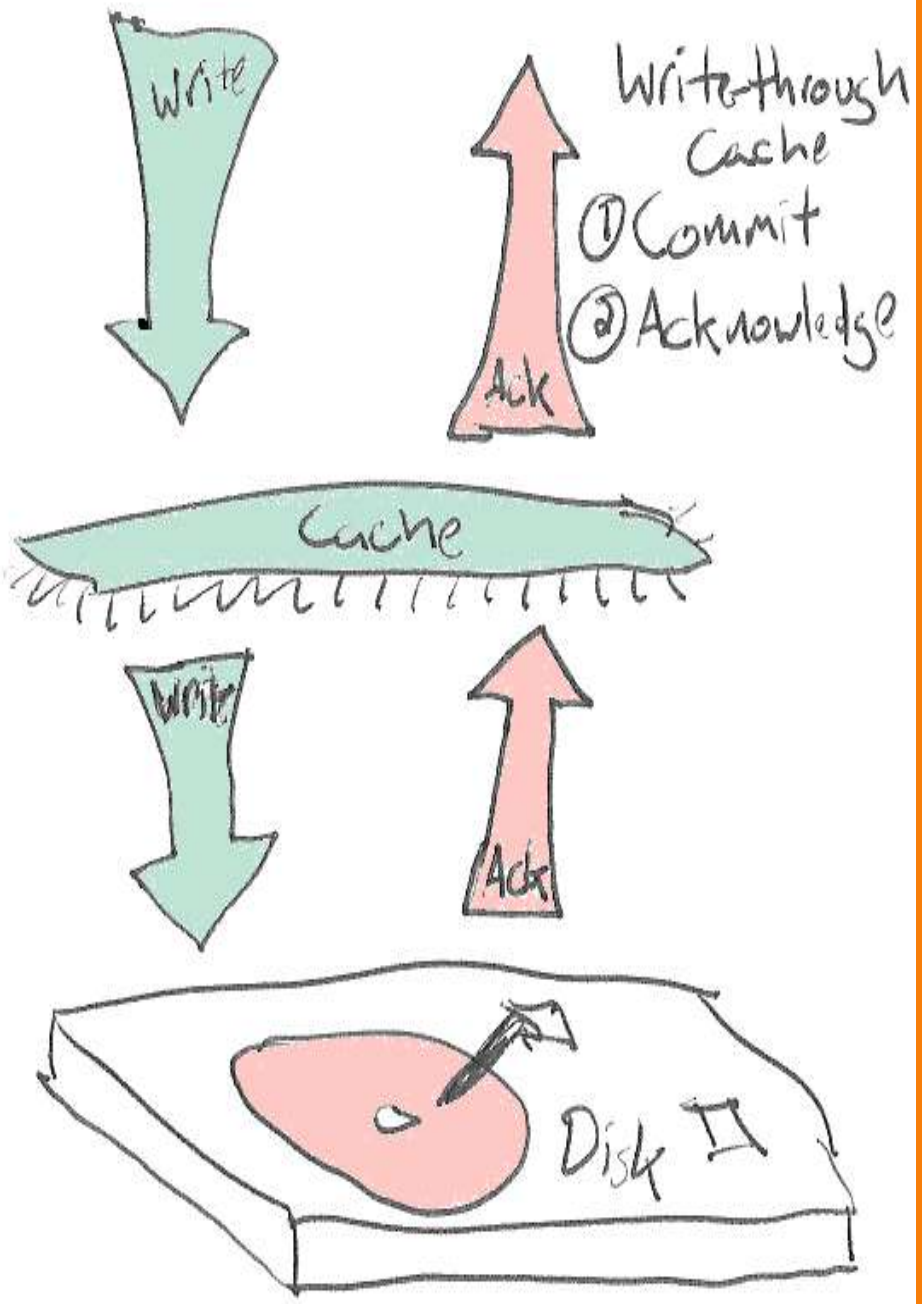
[Varnish](#) is a HTTP accelerator (or reverse proxy) capable of serving 100,000 requests a second. Somewhat faster than Drupal, even with page-caching on!

Cache-hit

- User requests a URL
- Varnish checks it's cache
- Varnish retrieves the data from the cache
- Varnish delivers the data to the user.

Cache-miss

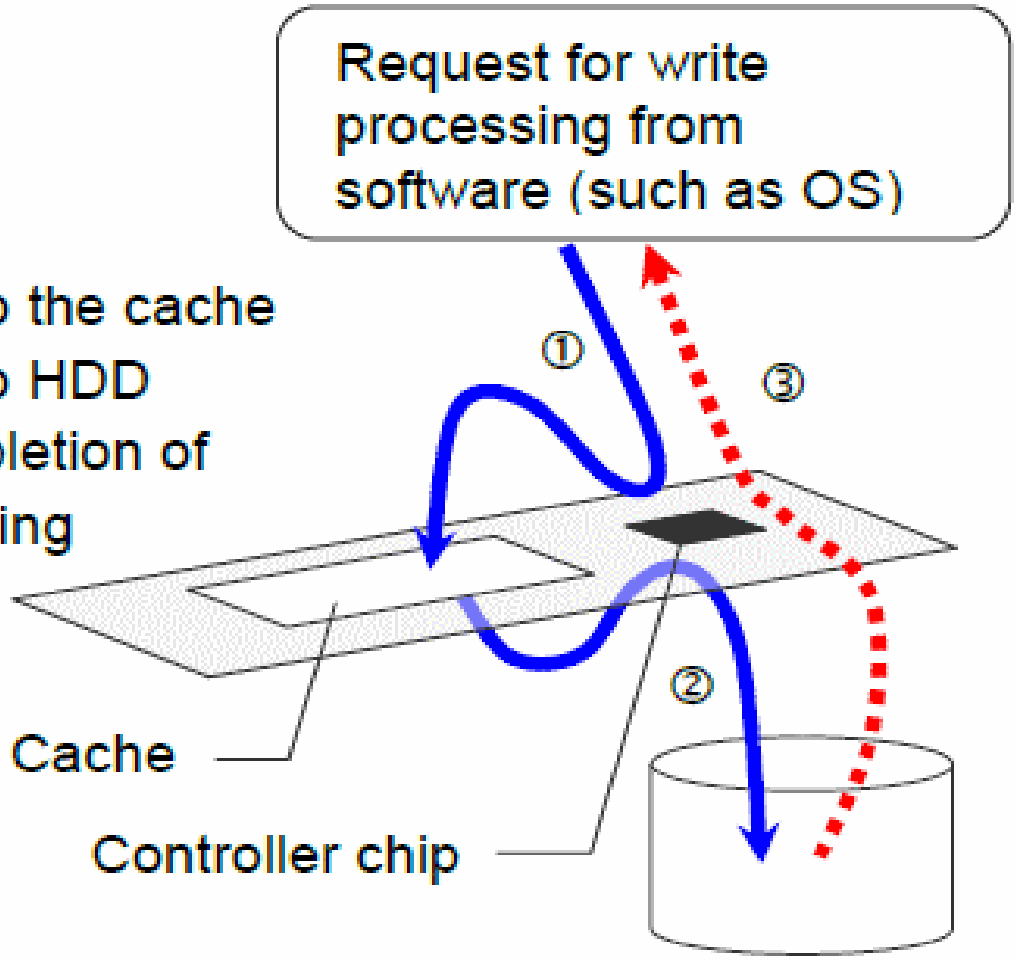
- User requests a URL
- Varnish checks it's cache - but the data isn't cached
- Varnish requests the URL from the backend
- Drupal processes the request and delivers a response to Varnish
- Varnish caches the response
- Varnish forwards the response to the user



Write through cache

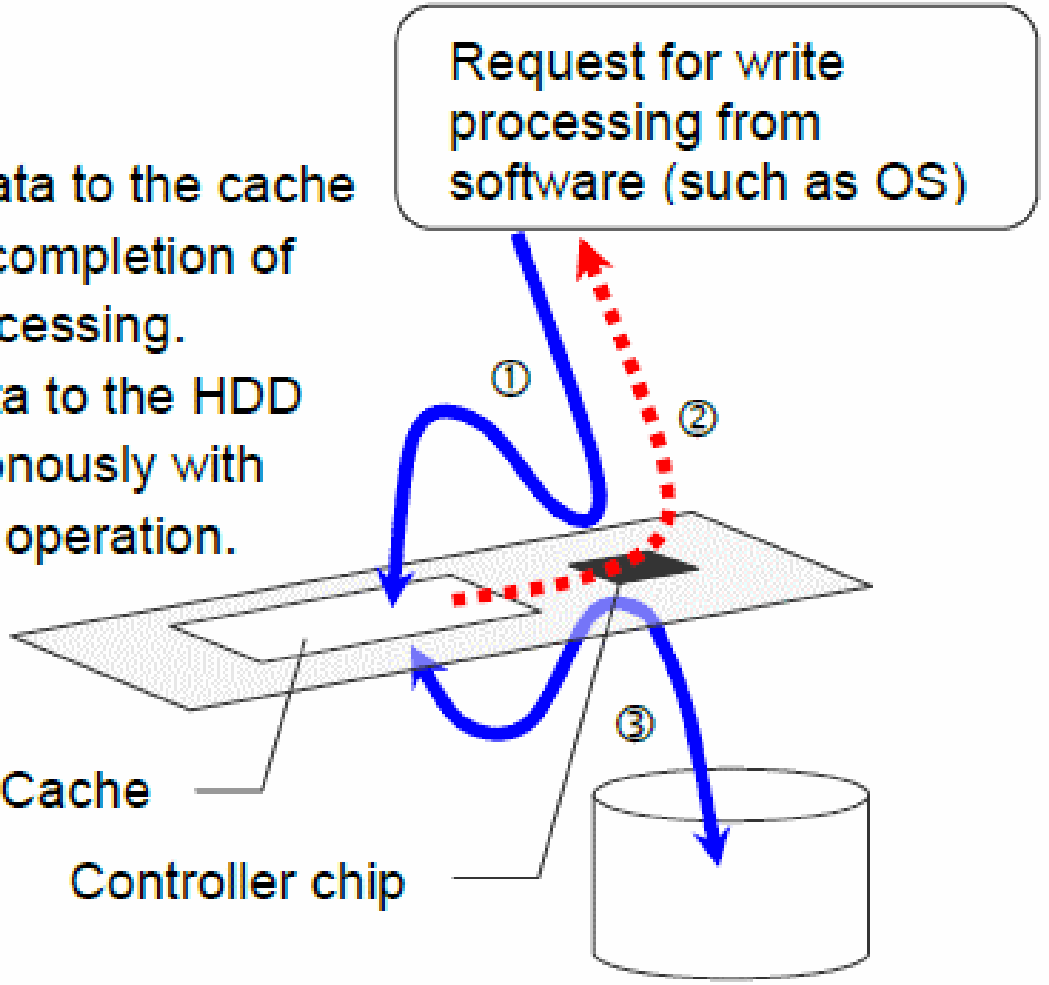
Write through

① Writes data to the cache
② Writes data to HDD
③ Notifies completion of write processing

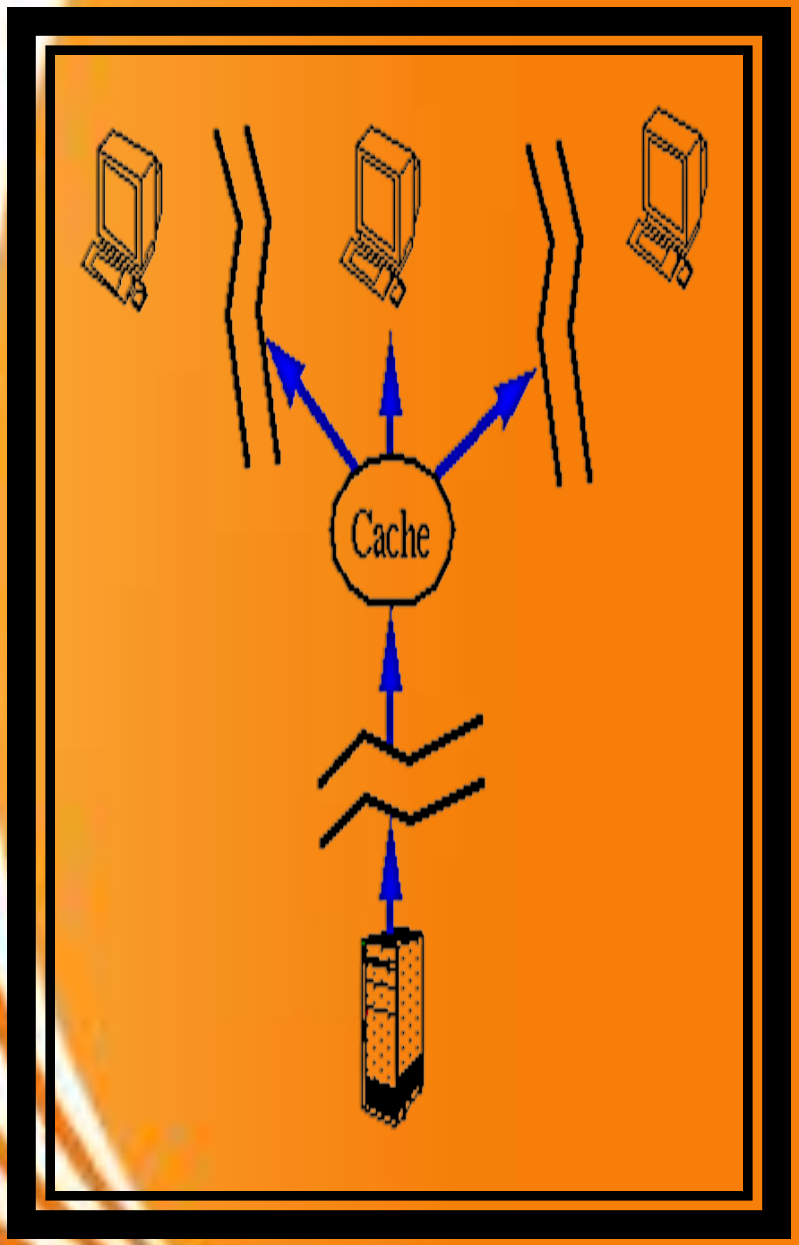


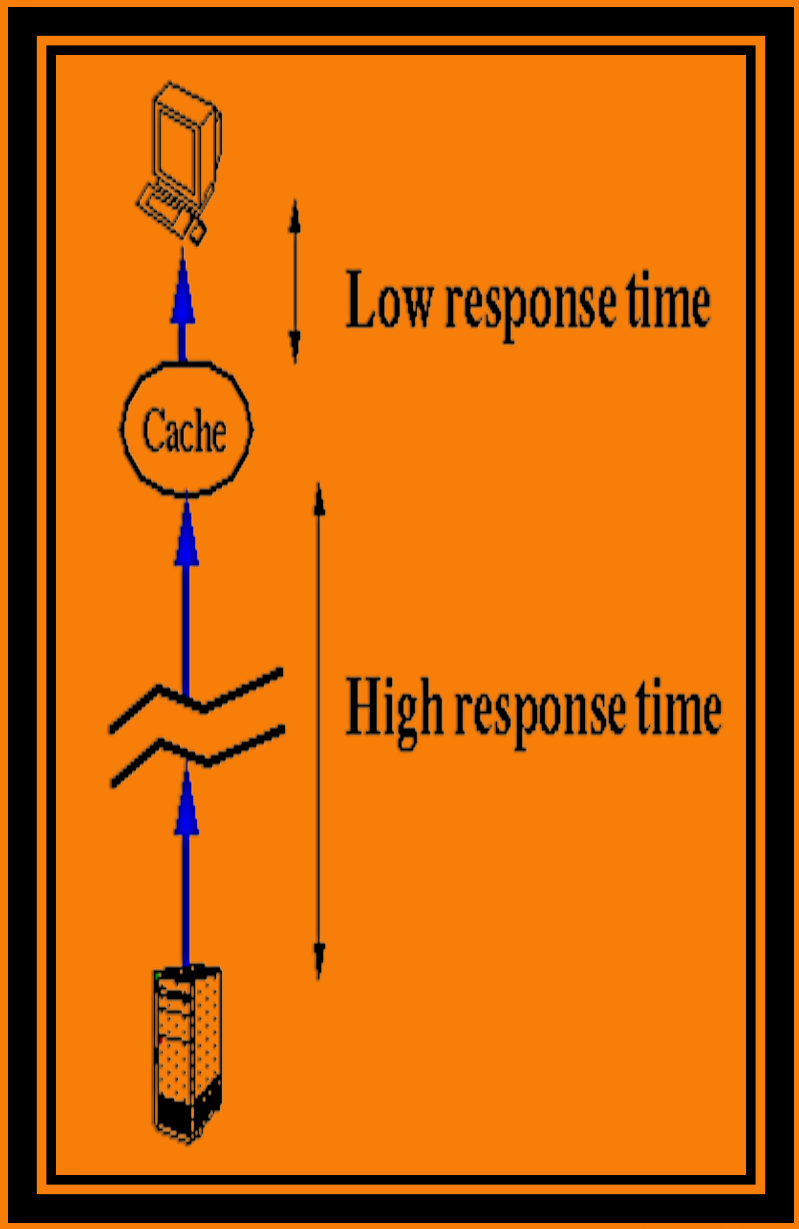
Write Back

① Writes data to the cache
② Notifies completion of write processing.
③ Write data to the HDD asynchronously with software operation.

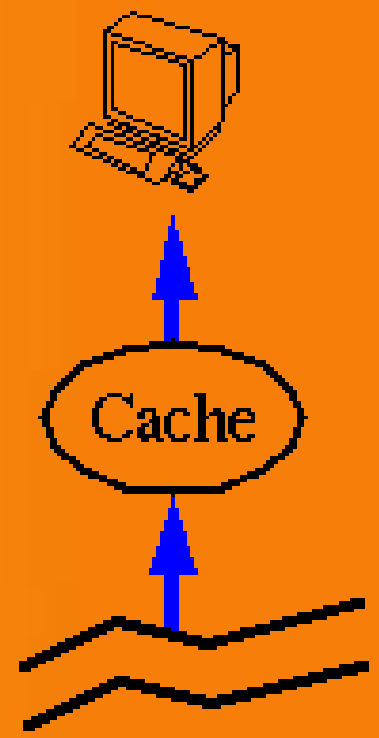


cache





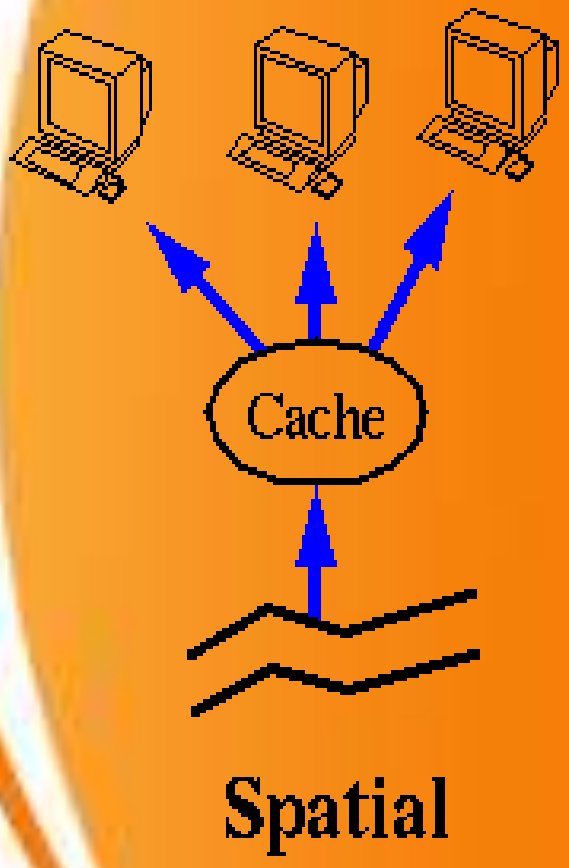
Temporal Locality



Temporal
(built into browsers)

Temporal locality is a property of a single client over time; the same client requests the same response repeatedly over a period of time. This kind of locality is best served by per-client caches, which are already incorporated into most commercial clients (browsers, file system clients, etc.).

Spatial Locality



Spatial locality is a property of a group of clients over time; one client requests what another has requested earlier. This kind of locality is best served by shared caches, known as proxy caches in web systems. They require that the cache be placed central and near to the set of clients, and far from the server.

Cache hit and cache miss