**SSD caching**

SSD caching, also known as flash caching, is the temporary storage of data on NAND flash memory chips in a solid-state drive (SSD) so data requests can be met with improved speed.

In a common scenario, a computer system stores a temporary copy of the most active data in the SSD cache and a permanent copy of the data on a hard disk drive (HDD). A flash cache is often used with slower HDDs to improve data access times.

Caches can be used for data reads or writes. The goal of SSD read caching in an enterprise IT environment is to store previously requested data as it travels through the network so it can be retrieved quickly when needed. Placing previously requested information in temporary storage, or cache, reduces demand on an enterprise's bandwidth and accelerates access to the most active data. SSD caching can also be a cost-effective alternative to storing data on top-tier flash storage. The objective of SSD write caching is to temporarily store data until slower persistent storage media has adequate resources to complete the write operation. The SSD write cache can boost overall system performance.

The form factor options of a flash-based cache include a SAS, Serial ATA or nonvolatile memory express (NVMe) SSD; a PCI Express (PCIe) card; or a dual in-line memory module (DIMM) installed in server memory sockets.

SSD cache software applications, working with SSD cache drive hardware, can help to boost the performance of applications and virtual machines (VMs). They can also extend basic operating system (OS) caching features with Linux and Windows. SSD cache software options are available from storage, OS, VM, application and third-party vendors.

**How does it work?**

Host software or a storage controller determines the data that will be cached. An SSD cache is secondary to DRAM-, nonvolatile DRAM (NVRAM) and RAM-based caches implemented in a computer system. When a data request is made, the system queries the SSD cache after each DRAM-, NVRAM- or RAM-based cache miss. The request goes to the primary storage system if the DRAM-, NVRAM-, RAM- and SSD-based caches do not have a copy of the data.

The effectiveness of an SSD cache depends on the ability of the cache algorithm to predict data access patterns. With efficient cache algorithms, a large percentage of I/O can be served from an SSD cache. Examples of SSD caching algorithms include:

1. Least Frequently Used. Tracks how often data is accessed; the entry with the lowest count is removed first from the cache.
2. Least Recently Used. Retains recently used data near the top of cache; when the cache is full, the less recently accessed data is removed.

**Types of SSD caching**

System manufacturers use different types of SSD caching, such as the following:

Write-through SSD caching. The system writes data to the SSD cache and to the primary storage device at the same time. Data is not available from the SSD cache until the host confirms the write operation is complete at both the cache and the primary storage device. Write-through SSD caching can be cheaper for a manufacturer to implement because the cache does not require data protection. A drawback is the latency associated with the initial write operation.

Write-back SSD caching. The host confirms a data I/O block is written to the SSD cache before the data is written to the primary storage device. Data is available from the SSD cache before the data is written to primary storage. The advantage is low latency for both read and write operations. The main disadvantage is the risk of data loss in the event of an SSD cache failure. Vendors using a write-back cache typically implement protections such as redundant SSDs, mirroring to another host or controller, or battery-backed RAM.

Write-around SSD caching. The system writes data directly to the primary storage device, bypassing the SSD cache. The SSD cache requires a warmup period, as the storage system responds to data requests and populates the cache. The response time for the initial data request from primary storage will be slower than subsequent requests for the same data served from the SSD cache. Write-around caching reduces the chance that infrequently accessed data will flood the cache.

**SSD caching locations**

SSD caching can be implemented with an external storage array, a server, an appliance or a portable computing device, such as a desktop or laptop computer. Storage array vendors often use NAND flash-based caching to augment faster and more expensive DRAM- or NVRAM-based caches. SSD caching is secondary to the higher-performance caching mechanisms and can boost access to less frequently accessed data.

Dedicated flash cache appliances are designed to add caching capabilities to existing storage systems. When inserted between an application and a storage system, flash cache appliances use built-in logic to determine which data should be placed in its SSDs. When a data request is received, the flash cache appliance can fulfill it if the data resides on its SSDs. Physical and software-based virtual appliances can cache data at a local data center or within the cloud.

**SSD caching vs. storage tiering**

By contrast, SSD caching maintains only a copy of the data on high-performance flash drives, while the primary version is stored on less expensive or slower media, such as HDDs or low-cost flash.

SSD caching software or the storage controller determines which data will be cached. With SSD caching, a system does not need to move inactive data when the cache is full; it can simply invalidate it.

Because only a small percentage of data is typically active at any given time, SSD caching can be a cost-effective approach to accelerate application performance, rather than storing all data on flash storage. However, I/O-intensive workloads, such as financial trading applications and high-performance databases, may benefit from data placement on a faster storage tier to avoid the risk of cache misses.

redundant storage – накопитель большой емкости