October 06, 2017

PROCESSOR CACHE SNAPSHOT AND RESTORE

David Weekly

PROCESSOR CACHE SNAPSHOT AND RESTORE

When a process is running on a computer processor and yields to another process, an operating system may perform a "flush" of the cache corresponding to that process as part of a context switch. For example, cache space in a processor may be limited so when an application yields the cache corresponding to the application may be emptied or marked as dirty, e.g., marked as invalid so can be overwritten, so that the cache space in the processor can be used for another process. When the process is resumed, the process may be given a flushed cache and may need to rebuild the cache from the results of requests to slower memory, e.g., random access memory (RAM) that is slower than a L1 or L2 cache on a processor, as the process requests data during execution.

Rebuilding a cache from data requests during execution of the process may yield poor performance for the process because any request for a value in a memory address that has not already been requested after the process is resumed may not be in the cache. Accordingly, when a process is resumed with a flushed cache, performance may be slow as there may be multiple separate requests for data from slower memory. Poor performance from rebuilding a cache may be acceptable for systems in which a small number of processes are running and context switches are infrequent. However, in function-oriented computing, e.g., serverless computing, a process may need to be run on-demand and only execute for a very short period of time, making the poor performance while a cache is being rebuilt for a resumed process a meaningful percentage of the time the process may be running.

A system may improve performance in processes that are resumed by providing a cache that stores information that can immediately be used by the process when the process resumes. For example, the system may store a snapshot of at least a portion of a cache in a L1 cache to

RAM for a process before the process is paused and before the process is resumed, restore at least the portion of the cache to the L1 cache from the snapshot stored in RAM.

In some implementations, the system may use processes with corresponding executable files that each define processor-specific cache preload data. The processor-specific cache preload data may indicate what data should be loaded into a cache corresponding to the process before the process is about to be run. For example, the processor may determine that a process for which processor-specific cache preload data is defined is about to be resumed and, in response, reserve space for a cache corresponding to the process in a L1 and a L2 cache, load the processor-specific cache data into the reserved space in the L1 and L2 cache, and then resume the process once the data is pulled.

Additionally or alternatively, in some implementations, a process may be able to request cross-context switch cache preservation, e.g. by setting a bit in the executable's header. When a process that has requested cross-context switch cache preservation is about to yield to another process, the kernel may instruct the processor to dump values of a cache for the process to an area of memory selected and reserved by the kernel for that process, e.g., a portion of RAM. When the kernel determines the process is about to be run again, the kernel may instruct the processor to restore the cache from the area of memory.

Additionally or alternatively, in some implementations, a process may indicate to a kernel when the process has initialized and a cache corresponding to the process stores values for performing the process, e.g., the cache for the process is fully primed. Once the kernel receives the indication that process has initialized and the cache corresponding to the process stores values for performing the process, the kernel may request that the processor store values of the cache corresponding to the process in memory, e.g., store a cache snapshot for the process in

RAM. The process may later yield and when the process is switched to a run state due to a new inbound request, the kernel can request the processor load the stored values for the process's cache when primed from memory.

Additionally or alternatively, the kernel may notice which processes are being run in a repeated, intermittent short loop mode heuristically, e.g., by noting a short period of time between process invocation and voluntary yielding, e.g., by calling select() or poll() or equivalents. For processes detected to be being run in repeated, intermittent short loop, upon receiving a yield request that will yield such a process, the kernel may take a snapshot of the cache for the process and store that in a desired memory address and retrieve the snapshot when the process is to be resumed.

Additionally or alternatively, a processor may experiment with running a process with or without storing a snapshot of a cache for the process based on the above approaches. Storing and retrieving a snapshot for a cache of a process may sometimes perform worse than resuming from an empty or dirty cache as there may be a penalty as the processor may need to write or read more data than might otherwise be needed. For example, if the resumed process doesn't end up using any values in a retrieved cache snapshot, then the storage and retrieval of the cache snapshot reduced performance without providing any benefit.

Additionally or alternatively, the process may experiment with running each of the processes with each of the above approaches, alone or in various combinations, and then use the approach, or combination of approaches, that results in the best performance for each of the processes on an individual basis. For example, the processor may decide to store and retrieve an entire cache for a first process, and only store an entire cache for a second process once the cache is primed.

FIG. 1 below shows an example where process A yields to a new process B, the processor may store a cache snapshot that includes values stored in a L1 cache of a processor for process A to RAM.

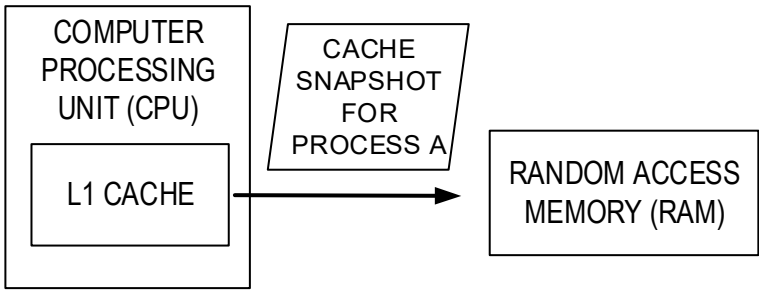

FIG. 1

FIG.2 below shows an example where when process B yields to process A that previously yielded, the processor may store a cache snapshot that includes values stored in a L1 cache of a processor for process B to RAM and restore a cache snapshot for process A from RAM to the L1 cache of the processor.

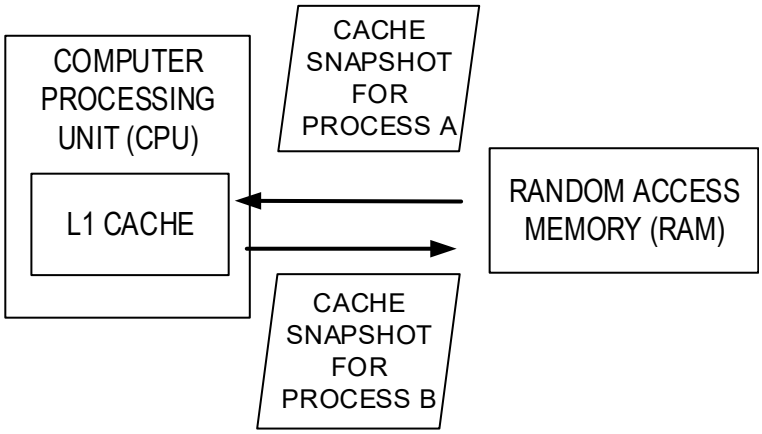

FIG. 2

ABSTRACT

A system is described that improves performance in processes that are resumed by providing a cache that stores information that can immediately be used by the process when the process resumes. A processor may store a snapshot of a cache for a process to memory when the process yields to another process. When performance of the process is later resumed on the processor, the processor may retrieve the cache for the process from the memory and then resume the process.