WIKIPEDIA

Thrashing (computer science)

In computer science, **thrashing** occurs when a computer's virtual memory subsystem is in a constant state of paging, rapidly exchanging data in memory for data on disk, to the exclusion of most application-level processing.[1] This causes the performance of the computer to degrade or collapse. The situation may continue indefinitely until the underlying cause is addressed.

The term is also used for various similar phenomena, particularly movement between other levels of the memory hierarchy, where a process progresses slowly because significant time is being spent acquiring resources.

Contents

Overview

If a process does not have access to a sufficient number of memory pages, a futile, repetitive swapping condition known as "thrashing" often arises, and the page fault rate typically becomes high. This frequently leads to high, runaway CPU utilization that can grind the system to a halt. In modern computers, thrashing may occur in the paging system (if there is not sufficient physical memory or the disk access time is overly long), or in the I/O communications subsystem (especially in conflicts over internal bus access), etc.

Depending on the configuration and algorithms involved, the throughput and latency of a system may degrade by multiple orders of magnitude. Thrashing is a state in which the CPU performs 'productive' work less, and 'swapping' more. The overall memory access time may increase since the higher level memory is only as fast as the next lower level in the memory hierarchy[2]. The CPU is busy in swapping pages so much that it can not respond to users' programs and interrupts as much as required. Thrashing occurs when there are too many pages in memory, and each page refers to another page. The real memory shortens in capacity to have all the pages in it, so it uses 'virtual memory'. When each page in execution demands that page that is not currently in real memory (RAM) it places some pages on virtual memory and adjusts the required page on RAM. If the CPU is too busy in doing this task, thrashing occurs.

Causes

In virtual memory systems, thrashing may be caused by programs or workloads that present insufficient locality of reference: if the working set of a program or a workload cannot be effectively held within physical memory, then constant data swapping, *i.e.,* thrashing, may occur. The term was first used during the tape operating system days to describe the sound the tapes made when data was being rapidly written to and read. An example of this sort of situation occurred on the IBM System/370 series mainframe computer, in which a particular instruction could consist of an execute instruction (which crosses a page boundary) that points to a move instruction (which itself also crosses a page boundary), targeting a move of data from a source that crosses a page boundary, to a target of data that also crosses a page boundary. The total number of pages thus being used by this particular instruction is eight, and all eight pages must be present in memory at the same time. If the operating system allocates fewer than eight pages of actual memory, when it attempts to swap out some part of the instruction or data to bring in the remainder, the instruction will again page fault, and it will thrash on every attempt to restart the failing instruction.

Other uses

Thrashing is best known in the context of memory and storage, but analogous phenomena occur for other resources, including:

Cache thrashing

Where main memory is accessed in a pattern that leads to multiple main memory locations competing for the same cache lines, resulting in excessive cache misses. This is most problematic for caches that have low associativity.

TLB thrashing

Where the translation lookaside buffer (TLB) acting as a cache for the memory management unit (MMU) which translates virtual addresses to physical addresses is too small for the working set of pages. TLB thrashing can occur even if instruction cache or data cache thrashing are not occurring, because these are cached in different sizes. Instructions and data are cached in small blocks (cache lines), not entire pages, but address lookup is done at the page level. Thus even if the code and data working sets fit into cache, if the working sets are fragmented across many pages, the virtual address working set may not fit into TLB, causing TLB thrashing.

Heap thrashing

Frequent garbage collection, due to failure to allocate memory for an object, due to insufficient free memory or insufficient contiguous free memory due to memory fragmentation is referred to as heap thrashing.[3]

Process thrashing

A similar phenomenon occurs for processes: when the process working set cannot be coscheduled – so not all interacting processes are scheduled to run at the same time – they experience "process thrashing" due to being repeatedly scheduled and unscheduled, progressing only slowly.[4]

See also

- Page replacement algorithm
- Congestion collapse
- Resource contention
- Out of memory
- Software aging

References

1. Denning, Peter J. (1968). "Thrashing: Its causes and prevention" (http://www.cs.uwaterloo.ca/~brecht/courses/702/Possible-Readings/vm-and-gc/thrashing-denning-afips-1968.pdf) (PDF). *Proceedings AFIPS, Fall Joint Computer Conference*. **33**: 915–922. Retrieved 2012-02-15.