Analysis of Noisy Neighbor Effect in Scheduling Caused by Persistent Memory I/O

Hyunwoo Ahn
Sungkyunkwan University
Suwon, Republic of Korea
ahw9925@skku.edu

Jongseok Kim
Sungkyunkwan University
Suwon, Republic of Korea
ks7sj@skku.edu

Euiseong Seo
Sungkyunkwan University
Suwon, Republic of Korea
euiseong@skku.edu

Abstract

Linux's Completely Fair Scheduler (CFS) schedules tasks considering their processor loads so that the loads are evenly balanced on the processors. The CFS manages the processors into multiple levels of scheduling domains, such as simultaneous multi-threading (SMT) domain, symmetric multiprocessing (SMP) domain, and non-uniform memory access (NUMA) domain. It performs load balancing at each scheduling domain level from the bottom to the top.

I/O-bound tasks generally have lower processor loads than processor-bound tasks. This is because they spend most of their time waiting for I/O operations to complete. Therefore, the CFS distinguishes processor-bound tasks from I/O-bound tasks by using their processor load.

However, when using persistent memory (PM) as storage, we found out that the CFS cannot successfully classify I/O-bound tasks when the I/O operations are performed to the PM storage. This is due to the direct access (DAX) structure of the PM-aware file systems. Unlike the conventional file I/O path, the DAX architecture bypasses the OS kernel I/O paths to reduce access latency and page cache management overhead [1]. In the DAX architecture, I/O operations are performed through `load` and `store` instructions. Because of this, the I/O-bound threads seem to continuously consume processor cycles from the viewpoint of the OS kernel. As a result, the difference in processor load between I/O-bound and processor-bound threads becomes small, and thus the CFS handles the PM I/O-bound tasks the same as the processor-bound ones.

We analyzed the performance impact from this phenomenon. In the SMT domain, tasks scheduled on sibling cores that share a physical core can achieve good performance if there is no internal resource contention among them. Thus, it is desirable to mix processor-bound tasks and I/O-bound tasks during execution in the SMT domain. In conventional OS, the CFS schedules them evenly mixed, taking into account the load differences between processor-bound tasks and I/O-bound tasks. However, we found that the CFS places processor-bound tasks with other processor-bound tasks and PM I/O-bound tasks with other PM I/O-bound tasks.

We also analyzed performance degradation in two higher-level scheduling domains, the SMP domain and the NUMA domain. Scheduling that disregards PM I/O can lead to tasks of the same type being concentrated on a single NUMA node. For example, if PM I/O tasks are placed on a single node, it can cause contention in the memory bus, interconnect, PM bandwidth, etc. On the other hand, if processor-bound tasks are placed on a single node, it can cause severe last-level cache (LLC) contention in the SMP domain.

To understand the impact of the above phenomena on performance, we conducted experiments using the SPEC CPU benchmark suite and the flexible IO (FIO) tester. Considering various levels of cache sensitivity, we selected six different processor-bound workloads from the SPEC CPU suite. For PM I/O-bound workloads, we used FIO random write. Each experiment consisted of a pair of workloads, one processor-bound workload and FIO random write.

The experiment results showed that co-scheduling different types of workloads in the SMT domain improved overall performance by an average of 18% compared to co-scheduling the same types of workloads. Among the processor-bound workloads, those with low cache sensitivity showed higher performance improvements than those with high cache sensitivity. However, in the case of `povray`, despite its low cache sensitivity, the performance improvement was minimal. This is because `povray` is a very lightweight workload, so even when scheduled with the same types of workloads, its performance degradation is relatively low compared to other workloads.

For the other two higher-level domains, we conducted experiments using eight processor-bound workload tasks and eight random write tasks for each workload pair. To exclude the effect of resource contentions in the SMT domain, each task was scheduled on a different physical core. Also, to alleviate performance degradation caused by remote access in NUMA, we interleaved all PMs across all NUMA nodes. Our experimental results showed that evenly distributing the two types of workload tasks across all NUMA nodes resulted in up to 16.6 times better performance than concentrating the same type of tasks on each node. Furthermore, we found that processor-bound workloads with higher bandwidth consumption show greater performance improvement.

References

[1] Subramanya R Dulloor, Sanjay Kumar, Anil Keshavamurthy, Philip Lantz, Dheeraj Reddy, Rajesh Sankaran, and Jeff Jackson. 2014. System software for persistent memory. In *Proceedings of the Ninth European Conference on Computer Systems*. 1–15.