Page Cache Replication Talk

Jérôme Coquisart Julien Sopena Redha Gouicem

This talk abstract presents our paper “PaCaR: Improved Buffered I/O Locality on NUMA Systems with Page Cache Replication”, which will be presented and published to Eurosys 2026.

Modern applications are increasingly I/O-intensive, driven by the growing demand for data processing in fields such as machine learning, databases, and large-scale web services. These applications rely heavily on efficient file I/O operations to achieve high performance. To mitigate the inherent latency of disk accesses, operating systems usually implement I/O caching mechanisms that buffer file data in memory, significantly improving access times and throughput, e.g., the Linux page cache. This caching layer is particularly crucial for workloads that involve frequent read and write operations, as it reduces the need for costly disk operations and helps maintain high application performance.

However, with the democratization of NUMA architectures in modern multi-socket servers introduces new challenges for the page cache. In NUMA systems, memory access times vary depending on whether the memory is local to the CPU or resides on a remote node, i.e., another socket. Remote memory accesses can incur doubled latencies and halved bandwidth. The Linux kernel community has actively worked on improving the support for NUMA architectures over the last decade. In particular, automatic memory migration and thread placement have been introduced to improve data locality. However, these mechanisms only target anonymous memory and do not apply to the page cache. Consequently, I/O-intensive applications that access the page cache often suffer from degraded performance due to remote memory accesses, particularly in highly parallel workloads where threads on different nodes share the same working set.

To address this gap, we propose PaCaR, a novel page cache replication mechanism that enhances data locality in NUMA systems. PaCaR transparently replicates cached pages across NUMA nodes, ensuring that threads can access data locally without incurring the penalties of remote memory accesses. Our design is guided by three key goals: improving performance by minimizing remote accesses, addressing the limitations of existing NUMA balancing techniques, and ensuring transparency for applications by requiring no modifications to their code. Additionally, PaCaR integrates seamlessly with the kernel's memory management subsystem, adapting to memory pressure and maintaining system stability.

To do so, we implemented a dynamic replication mechanism, where remotely accessed pages are duplicated on the local node, making further accesses from the same node local. The original page is called main, and the duplicated page is called twin. This helps dealing with consistency. When writing to a page, we always write to the main page, and invalidate the twin. Further accesses to the twin page will trigger an update of this page. That ensures the system remains consistent. We are also dealing with memory eviction and memory pressure, by integrating PaCaR to the Linux’s LRU, ensuring cold pages get evicted when memory is needed.

PaCaR is implemented and tested on the latest Linux LTS version. We evaluate PaCaR on microbenchmarks, and real world applications. On simulation workloads like filebench, we observe up to 1.4x bandwidth improvement. Finally, on RocksDB, a production database, we observe up to 8.5% improvement in bandwidth.

By bridging the gap between NUMA-aware memory management and the page cache, PaCaR unlocks the full potential of modern multi-socket servers for data-intensive applications. Our approach demonstrates that it is possible to transparently and efficiently exploit idle memory resources to improve locality, without sacrificing consistency or requiring intrusive changes to the operating system or applications.