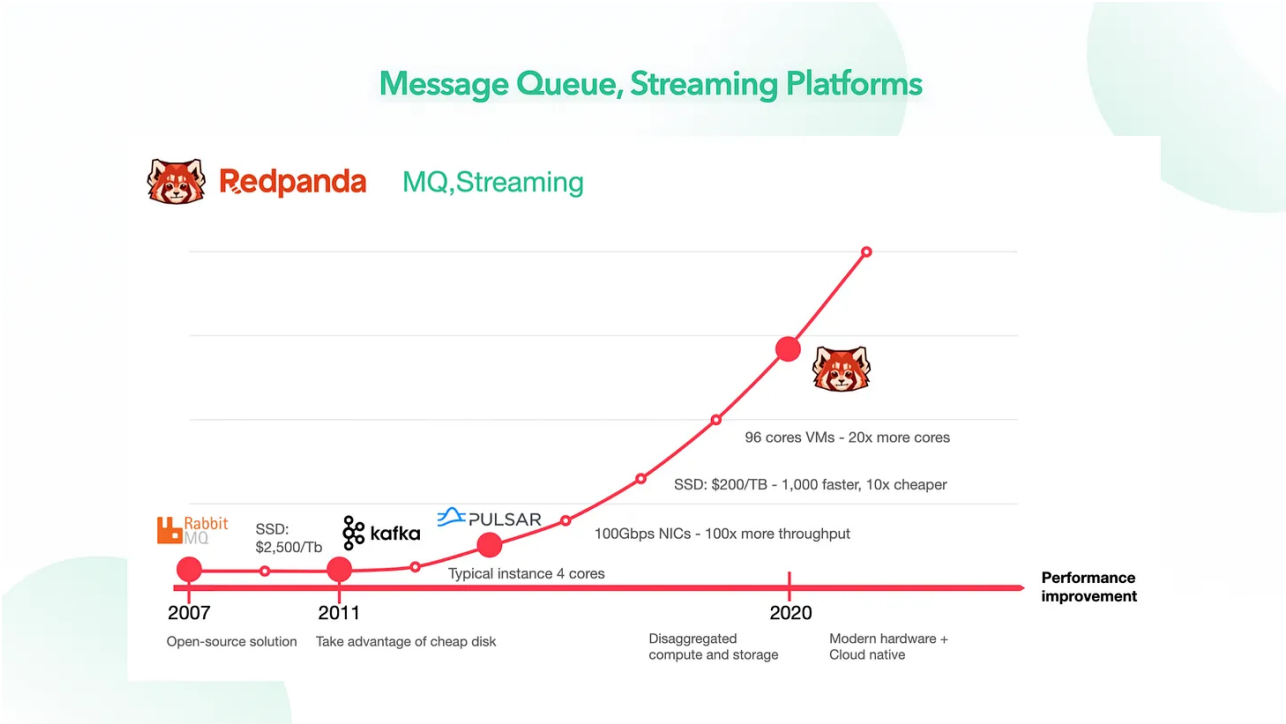
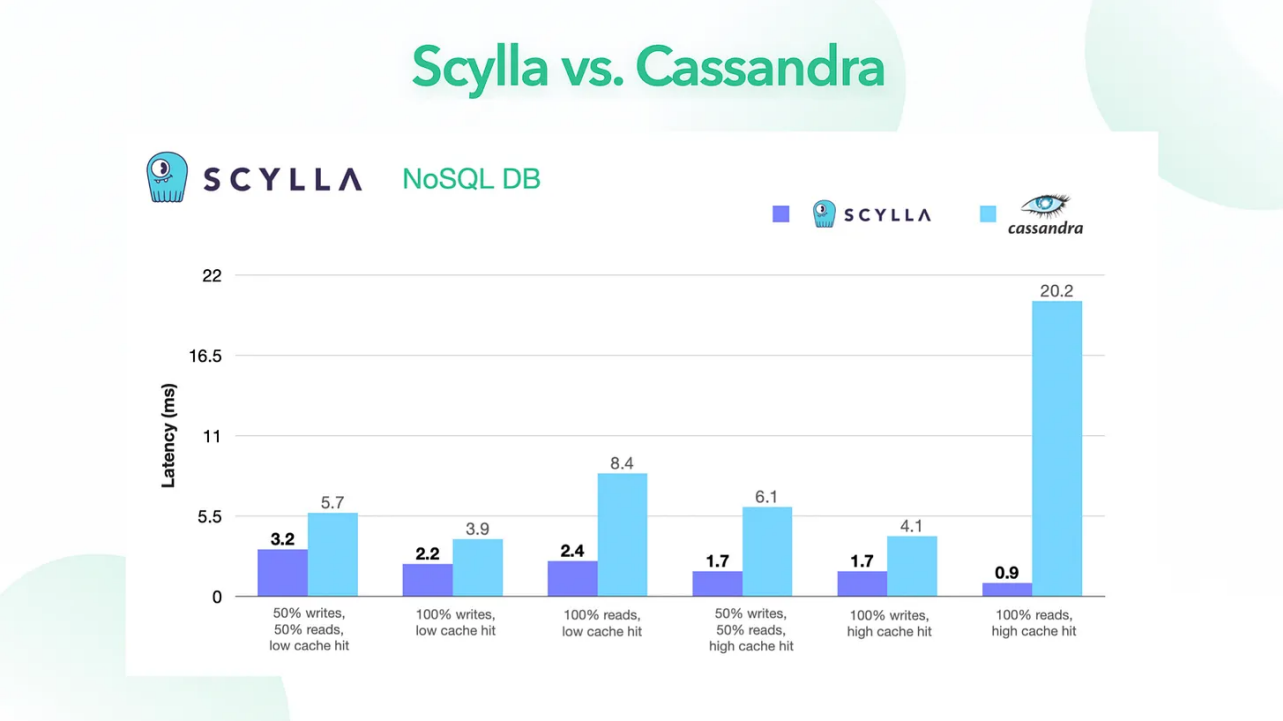
Message Queue and NoSQL with 10x Performance Boost

ALEX XU

JUN 3, 2022

There is an exciting class of storage software like **ScyllaDB** and **Redpanda** that boasts at least an order of magnitude improvement in performance compared to Apache Cassandra and Apache Kafka, respectively.





They take full advantage of some of the explosive trends in the last decade in computer architecture. What are these trends?

When Apache Cassandra came out around the late 2000s, AWS EC2 instances with a few physical cores and 64GB of RAM were considered high end.

When Apache Kafka came out in the early 2010s, an SSD was about 30 times more expensive per GB than spinning disks.

What happened in the ensuing decade?

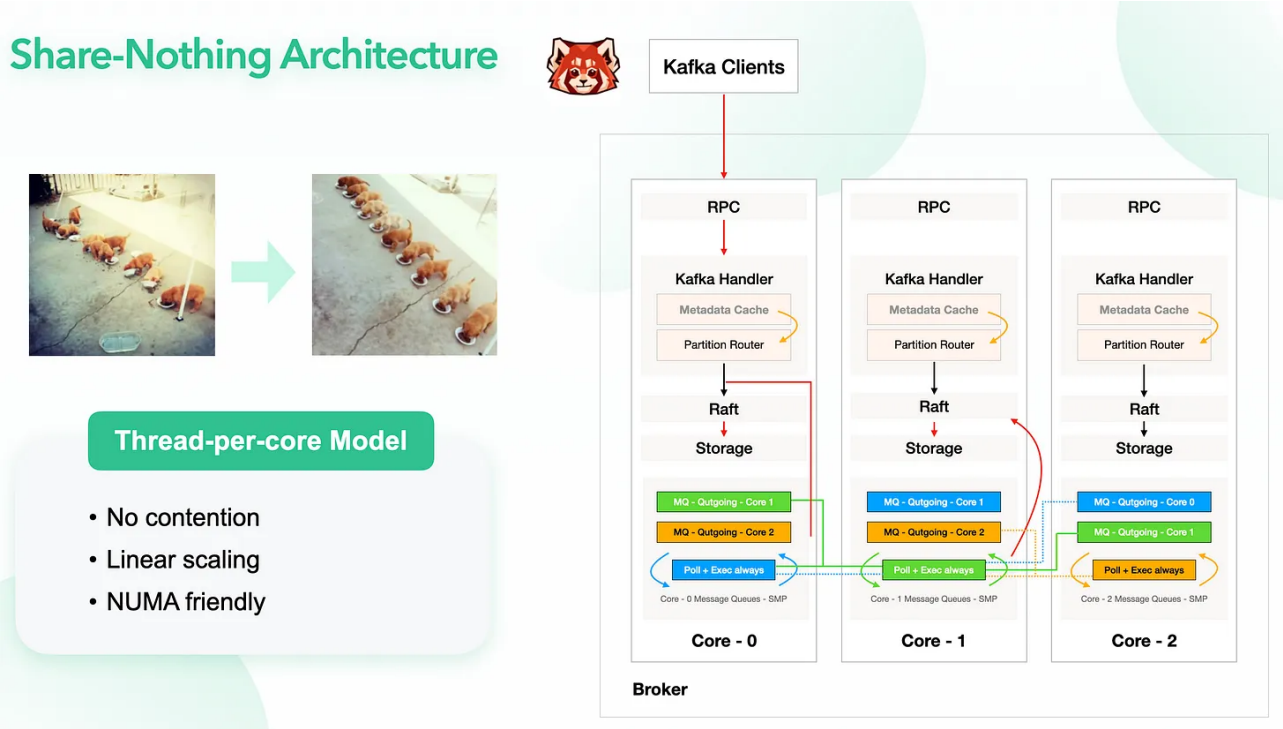
We can now rent an AWS EC2 instance with 36 physical cores and 15 TB of NVMe SSD drives and 512GB of RAM. Network bandwidth at 25Gbps is commonplace, and with some instances supporting 100Gbps. An NVMe SSD drive is about 100 times faster than a spinning disk from a decade ago.

In order to take full advantage of these advances, high performance software requires new designs.

This new class of storage software takes full advantage of these improvements with the following fundamental architectural decisions.

First, they all use shared-nothing architecture. In this architecture, each request is serviced by a single core, and each thread is pinned to a core. Instead of sharding at

the server level, we can think of this as sharding at the CPU core level. There is no memory contention between cores, and the use of locks is practically eliminated.



Also, this architecture recognizes the high cost of traditional threading models. At the high core count of modern servers, context switching is extremely costly, with large thread stacks polluting the caches and slowing everything down.

To complement the shared-nothing architecture, an asynchronous programming model is widely used. In addition to async networking which was already common with the previous generation of storage software, with this class of software, everything is asynchronous. This includes file I/O, and even communication between CPU cores.

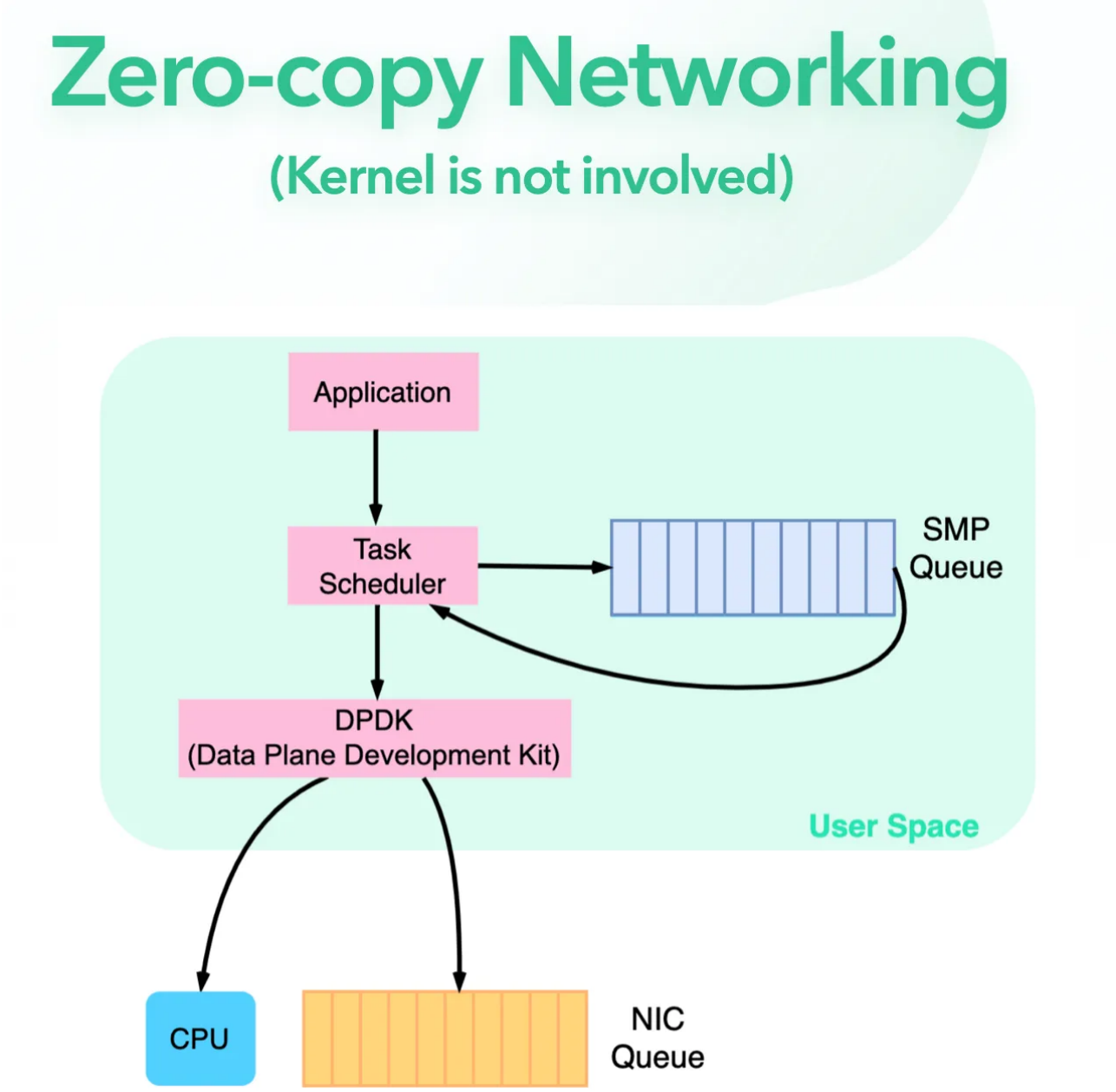
They run their own co-operative scheduler, instead of relying on the general purpose kernel scheduler. ScyllaDB and Redpanda use the same underlying C++ library called Seastar for the implementation of shared-nothing architecture and asynchronous operations.

These two design choices together allow this class of software to fully utilize CPU, memory, and I/O resources of modern servers.

Second, this new class of software keeps the external interface the same as the previous generation of software, but re-implemented everything under the hood in a

low level language. Both ScyllaDB and Redpanda are written in C++. There is no JVM, and there is no production tuning for garbage collection. The tail latency is low and very predictable as the workloads scale.

Third, instead of relying on the kernel to handle file I/O and page cache, this new class of software handles their own I/O and caching. While the kernel is a very capable general purpose operating system, operating at this level of performance requires controlling everything. This includes caching, file I/O, and task scheduling.



What is the **drawback** of this new class of software? Performance does not come for free. The level of complexity of this class of software is higher than the ones from the previous generation. C++ is already difficult to program in. The asynchronous programming model enforced by Seastar makes it even harder to reason about.

Having their own co-operative scheduler means taking full responsibility for managing long running tasks. It is challenging to ensure that every task takes as short as possible to complete. Any latency impact from errant tasks could be felt throughout the entire stack.

References:

[1] Seastar by Cloudius

[2] Redpanda blog

[3] ScyllaDB university

26 Likes

Comments

Write a comment...