Presenter: Owen Hilyard
Advisor: Aleksey Charapko
Program: Computer Science

METRONOME: A Kernel Bypass Distributed Database

University of New Hampshire

Introduction

In distributed storage systems today, the ingestion and ordering of data is often coupled and use of consensus algorithms is a typical way to provide a total-order (i.e., a single history of changes) which is a limiting factor in how much data the system can process. To achieve higher throughput, they must be separated.

Goals

- Separate consensus and data
- High throughput with consistent (not necessarily low) latency
- Keep the storage servers lean and fast
- Provide a comparison point for Ethernet vs RDMA for databases
- Re-use of existing consensus algorithms

Requirements

- Compatibility with public cloud environments (AWS, GCP, Azure)
- Focus on widely-available NIC features while leveraging DPDK.
- Ease of deployment of the distributed system.
- Separation of concerns between security, business and storage requirements to facilitate optimal performance.

Non-Goals

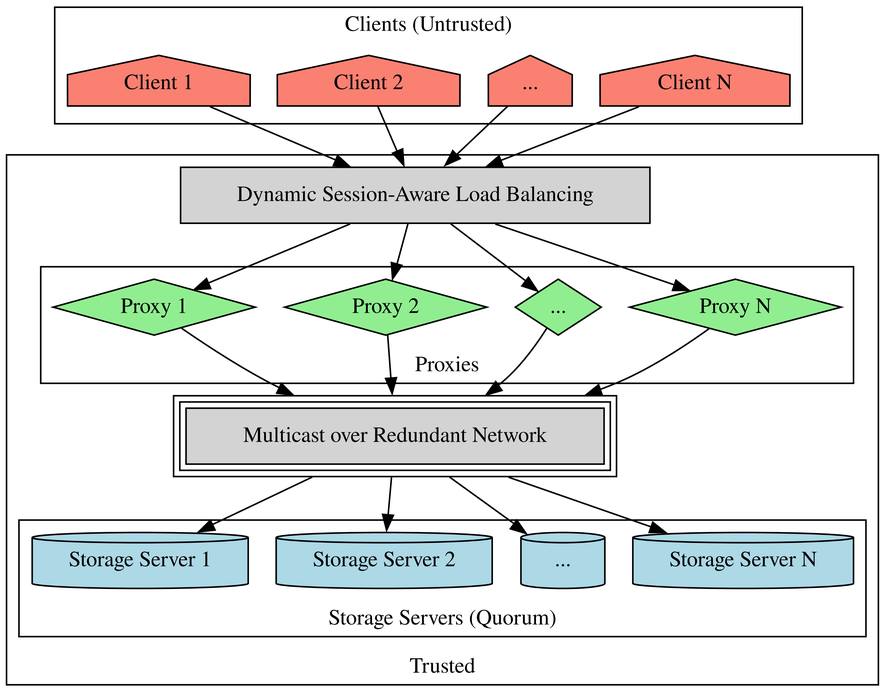
- Low throughput efficiency ($\leq$ 30k requests per second)
- Low latency (nanosecond timescale)
- Non-Linux OS support
- Local developer environment deployments

DPDK

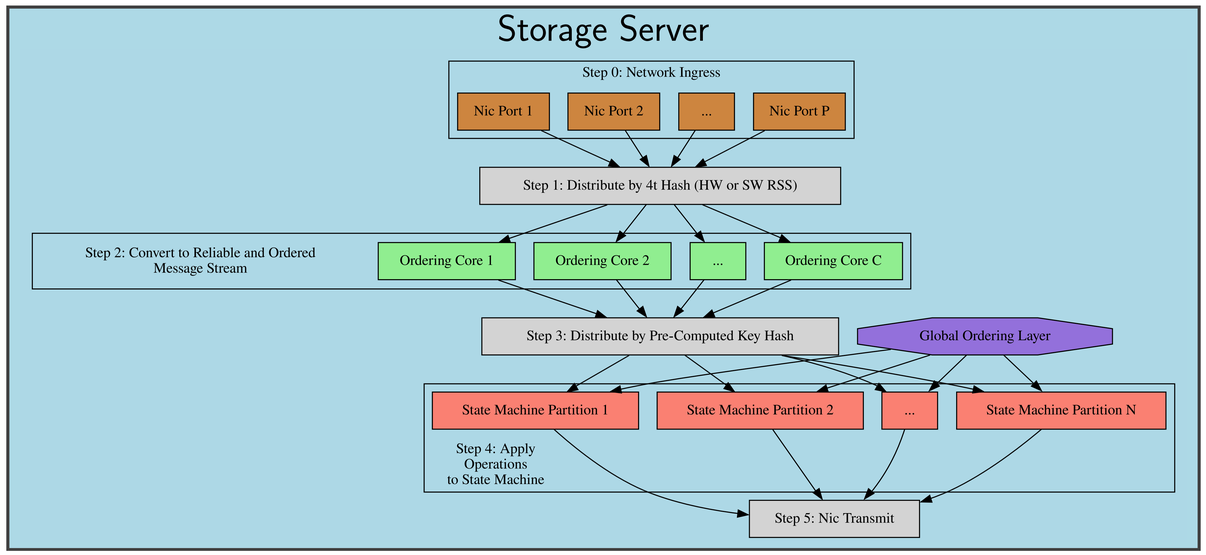
DATA PLANE DEVELOPMENT KIT

- A multi-vendor kernel-bypass framework for high performance packet processing owned by the Linux Foundation
- Takes control of NICs away from the kernel to enable network IO with no syscalls.
- Capable of forwarding IPv4 traffic at 123,220,000 packets per second **per core** (DPDK 22.11 Intel NIC Performance Report)
- Extensive support for hardware offloads, such as the entirety of TLS 1.2.
- **The UNH InterOperability Lab** hosts the CI infrastructure for DPDK as well as providing support for testing. I was the team lead for that project and a DPDK maintainer.

Metronome Cluster Data Ingest



Storage Server



Test Environment (Provided by UNH IOL)

- 2x Gigabyte R270-T60
- Ubuntu 20.04.5 LTS
- 4x Cavium THUNDERX Network Interface @ 10G
- 2 x Cavium ThunderX ARM CPUs per server @ 2.0GHz (2014 Launch)
- 251 GiB Micron 2100 MT/s DDR4 per server

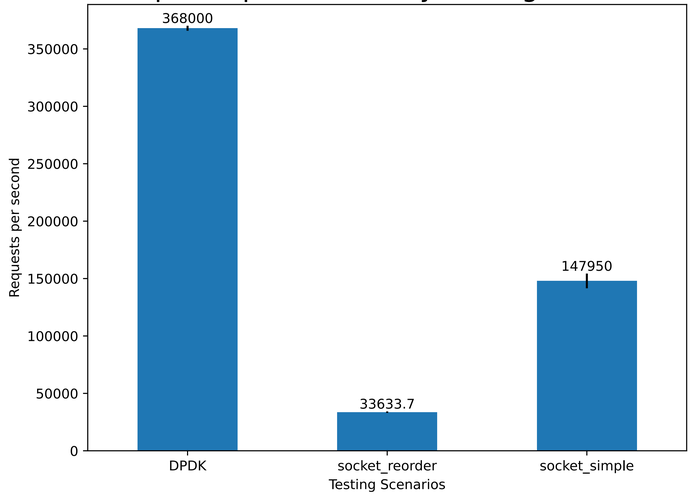
Testing Methodology

- Client uses 1 core as a workload generator.
- Server uses 1 core to accept packets, order per-burst, apply operation to hash table and transmit.
- Both client and server use a 10G interface on the local NUMA node.
- Both client and server are provided 50GiB of contiguous pinned 1GiB pages.
- Both client and server cores are removed from the kernel scheduler.
- Rate was captured by waiting 30 seconds for warmup and capturing the total number of requests handled in the next 60 seconds.
- 10 trials per server implementation.
- The DPDK server implementation has a granularity of 10,000 requests per second per trial as more precise measurements resulted in decreased performance.

Testing Scenarios/Results

- **DPDK**: Uses DPDK for networking, takes advantage of pre-calculated hash optimizations in the DPDK hash table.
- **socket_reorder**: Uses kernel sockets and performs the same work as the DPDK implementation.
- **socket_simple**: Use kernel sockets and treats UDP like a reliable ordered protocol.

Requests per Second by Testing Scenario



Completed Work

- Data ingest pipeline achieves an 11x performance improvement over sockets.
- Custom Ethernet/IPv4/UDP stack
- Implemented custom messaging protocol to minimize serialization and copying of data.
- Guaranteed message ordering
- Infrastructure as code deployments with patched from-source kernel builds for AWS deployments