

Networking Performance Regression Testing

Martin Karsten

David R. Cheriton School of Computer Science
University of Waterloo

October 2026

Motivation

- network stack performance-critical – despite AI
 - complexity → detect performance regression?
 - augment functional testing in selftests & NIPA
 - complement other performance/stress testing

Motivation

- network stack performance-critical – despite AI
 - complexity → detect performance regression?
 - augment functional testing in selftests & NIPA
 - complement other performance/stress testing

- academic papers compare to strawman setups
 - report questionable performance “improvements”

Motivation

- network stack performance-critical – despite AI
 - complexity → detect performance regression?
 - augment functional testing in selftests & NIPA
 - complement other performance/stress testing
- academic papers compare to strawman setups
 - report questionable performance “improvements”
- efficiency matters (cf. data centre backlash)
 - efficiency $\neq$ performance

Objectives

- reference performance/efficiency experiments
 - representative application scenarios
 - including system configuration, etc.
- automatic regression detection
 - multi-host testing scenarios
- benchmark for design proposals
 - for developer and academic communities

Challenges

- many dimensions of testing
 - applications, configurations, workloads
- ⇒ large testing matrix

Challenges

- many dimensions of testing
 - applications, configurations, workloads
- ⇒ large testing matrix
- feasible $\subset$ representative $\subset$ relevant

Challenges

- many dimensions of testing
 - applications, configurations, workloads
- ⇒ large testing matrix
 - feasible $\subset$ representative $\subset$ relevant
- metrics, measurements, tooling
 - integration with selftests & NIPA

Challenges

- many dimensions of testing
 - applications, configurations, workloads
- ⇒ large testing matrix
 - feasible $\subset$ representative $\subset$ relevant
- metrics, measurements, tooling
 - integration with selftests & NIPA
- regression detection: baseline, threshold $\rightarrow$ AI?

Table of Contents

1 Introduction

2 Challenges

3 Proposal

4 Wrap Up

Table of Contents

1 Introduction

2 Challenges

3 Proposal

4 Wrap Up

Canonical Application Types?

- K/V store, RPC-style queries
- large file transfer, streaming
- distributed AI training/inference?

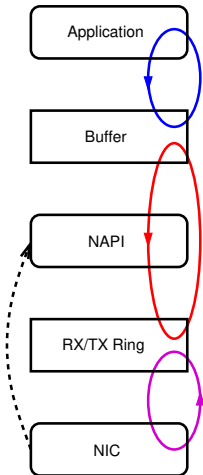
Canonical Application Types?

- K/V store, RPC-style queries
- large file transfer, streaming
- distributed AI training/inference?

- payload access vs offload/bypass (XDP)
- per-packet vs per-byte overhead
- TCP vs UDP vs other – user-space transport?
- epoll vs io_uring – zerocopy?

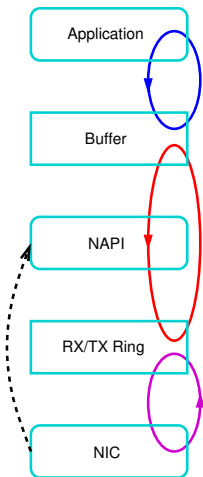
System Configuration – Network Stack

- napi: irq/softirq vs polling
 - threaded?
 - polling: busy vs timer



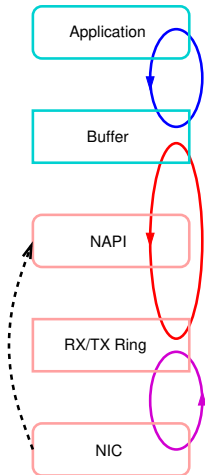
System Configuration – Network Stack

- napi: irq/softirq vs polling
 - threaded?
 - polling: busy vs timer
- spatial alignment
 - application – core – napi



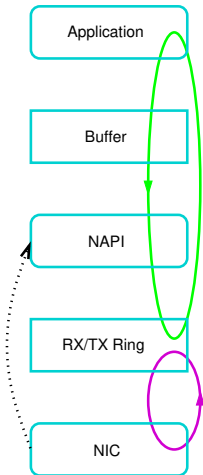
System Configuration – Network Stack

- napi: irq/softirq vs polling
 - threaded?
 - polling: busy vs timer
- spatial alignment
 - application – core – napi
 - application – core / core – napi
 - ▶ pair-wise or pool



System Configuration – Network Stack

- napi: irq/softirq vs polling
 - threaded?
 - polling: busy vs timer
- spatial alignment
 - application – core – napi
 - application – core / core – napi
 - ▶ pair-wise or pool
- temporal alignment
 - application – napi polling
 - with idle sleep: irq-suspend

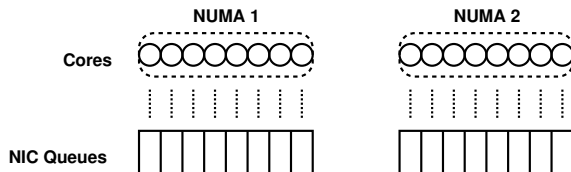


Load and Topology

- workload dynamics: peak vs average load
- concurrent flows $\leftrightarrow$ cache effectiveness

Load and Topology

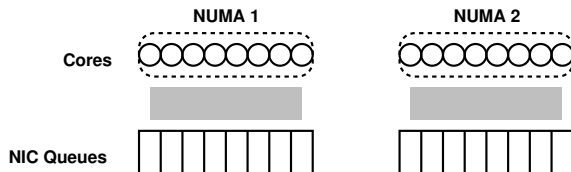
- workload dynamics: peak vs average load
- concurrent flows $\leftrightarrow$ cache effectiveness



- load balancing? nic, stack, application?

Load and Topology

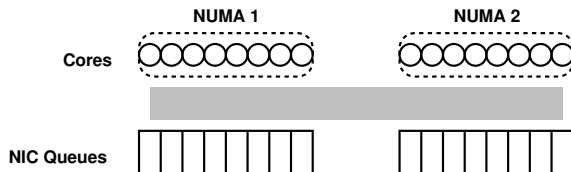
- workload dynamics: peak vs average load
- concurrent flows $\leftrightarrow$ cache effectiveness



- load balancing? nic, stack, application?

Load and Topology

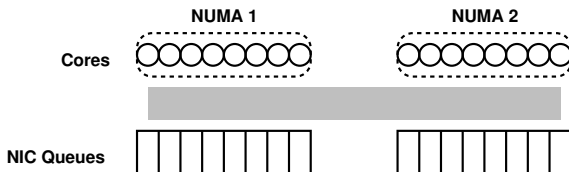
- workload dynamics: peak vs average load
- concurrent flows $\leftrightarrow$ cache effectiveness



- load balancing? nic, stack, application?

Load and Topology

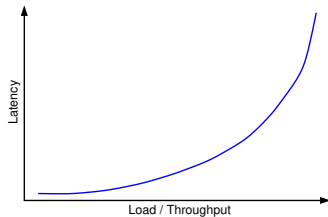
- workload dynamics: peak vs average load
- concurrent flows $\leftrightarrow$ cache effectiveness



- load balancing? nic, stack, application?
- also: isolation vs colocation?

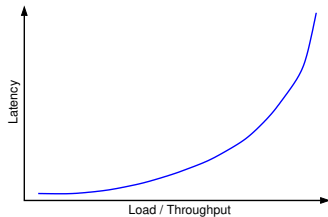
Performance/Efficiency Metrics

- maximum throughput
- latency percentiles at different load levels



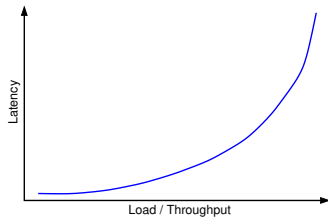
Performance/Efficiency Metrics

- maximum throughput
- latency percentiles at different load levels
- measure loss separately?
 - buffer sizing trades off loss and latency
 - retransmission converts loss into latency



Performance/Efficiency Metrics

- maximum throughput
- latency percentiles at different load levels
- measure loss separately?
 - buffer sizing trades off loss and latency
 - retransmission converts loss into latency
- system efficiency, power consumption



System Configuration - Other

- NIC configuration
 - interrupt coalescing
 - offload processing
 - queue sizes
- power management, frequency scaling
- system timer, clock source, preemption settings

System Configuration - Other

- NIC configuration
 - interrupt coalescing
 - offload processing
 - queue sizes
- power management, frequency scaling
- system timer, clock source, preemption settings
- but also: physical hardware vs virtual machine

Table of Contents

1 Introduction

2 Challenges

3 Proposal

4 Wrap Up

Outcome – v1

- usable and reproducible software stack
 - demonstrated in UW testbed
- set of benchmark experiments
 - applications, workloads, scripting
 - one-way NIPA integration
- written guidance for result interpretation

Outcome – v1

- usable and reproducible software stack
 - demonstrated in UW testbed
- set of benchmark experiments
 - applications, workloads, scripting
 - one-way NIPA integration
- written guidance for result interpretation
- ...v2: automatic regression testing
- ...v3: two-way integration with NIPA?

Execution Effects

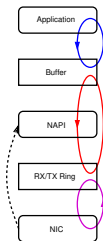
- direct changes
 - different code path
 - different algorithm or data structure
 - best observed at low load via latency?

Execution Effects

- direct changes
 - different code path
 - different algorithm or data structure
 - best observed at low load via latency?
- indirect changes: CPU and other caches

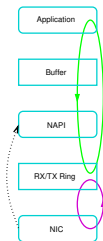
Execution Effects

- direct changes
 - different code path
 - different algorithm or data structure
 - best observed at low load via latency?
- indirect changes: CPU and other caches
 - asynchronous execution: queueing & batching
→ locality vs conflict/coherency misses



Execution Effects

- direct changes
 - different code path
 - different algorithm or data structure
 - best observed at low load via latency?
- indirect changes: CPU and other caches
 - asynchronous execution: queueing & batching
→ locality vs conflict/coherency misses
 - synchronous execution: alignment
→ locality vs capacity misses



Execution Effects

- direct changes
 - different code path
 - different algorithm or data structure
 - best observed at low load via latency?
- indirect changes: CPU and other caches
 - asynchronous execution: queueing & batching
→ locality vs conflict/coherency misses
 - synchronous execution: alignment
→ locality vs capacity misses
 - memory: payload access vs copy/move

Simple Performance Model

application performance: $QPT = \text{queries/time}$

algorithmic cost: $IPQ = \text{instructions/query}$

resource utilization: $CPT = \text{cycles/time}$

resource efficiency: $IPC = \text{instructions/cycle}$

Simple Performance Model

application performance: $QPT = \text{queries/time}$

algorithmic cost: $IPQ = \text{instructions/query}$

resource utilization: $CPT = \text{cycles/time}$

resource efficiency: $IPC = \text{instructions/cycle}$

Sanity Check for Performance Observations:

$$QPT = \frac{CPT \times IPC}{IPQ}$$

Simple Performance Model

application performance: $QPT = \text{queries/time}$

algorithmic cost: $IPQ = \text{instructions/query}$

resource utilization: $CPT = \text{cycles/time}$

resource efficiency: $IPC = \text{instructions/cycle}$

Sanity Check for Performance Observations:

$$QPT = \frac{CPT \times IPC}{IPQ}$$

Breakdown (I,C): kernel, middleware, application?

Benchmark Applications

- microbenchmark: Google neper vs Meta kperf
 - neper: network focus, limited feature coverage
 - kperf: stack focus, limited workloads, no UDP
- “small” application: memcached
 - tunable bottleneck via extstore: network vs storage
- web server/proxy: static vs dynamic content
- other ideas?

Experiments

- receive/reply
 - single flow - stop and go
 - single core/node/server - range of rates
- bulk data transfer
- representative application scenarios
 - “Facebook ETC” workload for memcached
- canonical set of system configurations
 - determine best practice for each scenario

Measurements

- service performance at client
- local performance via tracing?
- kernel statistics
- hardware counters
 - instructions, cycles, memory accesses, cache misses
 - power consumption vs cpu cycles?
- external measurements – power meter?

Regression Testing

- variability vs regression detection
 - perfect case for AI?

Regression Testing

- variability vs regression detection
 - perfect case for AI?
- per-patch testing likely not feasible
 - periodic testing and bisection – also difficult
 - ▶ outlier can send binary search in wrong direction
 - ▶ cumulative effect of multiple commits?
 - length of period vs runtime/overhead?
 - ▶ drill down via smaller periods?

Regression Testing

- variability vs regression detection
 - perfect case for AI?
- per-patch testing likely not feasible
 - periodic testing and bisection – also difficult
 - ▶ outlier can send binary search in wrong direction
 - ▶ cumulative effect of multiple commits?
 - length of period vs runtime/overhead?
 - ▶ drill down via smaller periods?
- any testing better than no testing?

Table of Contents

1 Introduction

2 Challenges

3 Proposal

4 Wrap Up

Status

- NSERC grants for students and equipment
- AMD, Arm, Intel server(s), variety of NICs
- student researchers hired
- community interest and support?

Summary

- automatic performance regression testing
- reference setup for academic research projects

Summary

- automatic performance regression testing
- reference setup for academic research projects
- nuts and bolts of system setup and deployment
- large testing matrix → meaningful subset?
- regression detection → open research question?

Summary

- automatic performance regression testing
- reference setup for academic research projects
- nuts and bolts of system setup and deployment
- large testing matrix → meaningful subset?
- regression detection → open research question?
- if interested, let's chat – or contact me at:
`mkarsten@uwaterloo.ca`