Buffer-based End-to-end Request Event Monitoring in the Cloud

Kaihui Gao

Chen Sun, Shuai Wang, Dan Li, Yu Zhou, Hongqiang Harry Liu, Lingjun Zhu, Ming Zhang
**RPC: Fundamental Component in Clouds**

- **Client**
  - Cloud Native
  - APP
  - Web

- **LB**
- **Service**
- **Compute**
- **Storage**

**RPC tools**
- Thrift
- SOFA
- finagle
- redis
- DUBBO
- gRPC

**440M RPC per hour per cluster**
RPC Latency is Crucial for App Performance

5 × 30 = 150ms vs. 5 × 200 = 1s

44k (0.01%) RPC latency anomalies per hour per cluster
RPC Latency Anomaly is **Difficult to Diagnose**

**Possible problem at all layers managed by different teams**
RPC Latency Anomaly is **Difficult to Diagnose**

Any ideas what happened to RPC#2?

Not sure... Got IP and time?

Yeah, check **192.168.1.22 @15:00**?

Emm, I see a burst, RPC#2 *maybe* affected.

**Incomplete Visibility**

**Inconsistent Semantics**

**App**

**Network stack**

- UDP
- TCP
- IP
- Ethernet

**NIC**

**Network**

**NIC**

**Network stack**

- UDP
- TCP
- IP
- Ethernet

**App**

**Telemetry**

- SNMP counters
- Active probing
- Sampling
- Telemetry

**Statistical analysis**

- TCP profiling
- TCP replay
- Kernel statistics
Why Difficult?

- Application
  - Network stack
    - NIC
      - Black Box
    - Switch
      - White Box

- Heterogeneous model
- Distinct semantics

Any Opportunity?

- Application
  - Network stack
    - Programmable NIC
    - Programmable Switch
  - White Box

- Uniform modeling
- Consistent semantics
What's the Right Uniform Model for RLA Locating?

Data processing

Diversified and complex in different layers

Logic

Buffer

Speed mismatch

Downstream logic

Homogeneous in all layers

Indicative of RLA location

Buffer overflow

Slow downstream logic

Out of order

Loss in upstream logic

Buffer overflow

Out of order

...
Motivation for Using Buffer to Locate RLAs

~500 Production RLA tickets
in Alibaba Cloud Storage Service

Root Causes

- Burst
- Code bug
- Disk full
- Hardware failure
- Incast
- Network update
- Out of memory
- Polling hang
- Priority contention

Anomaly Locations

<table>
<thead>
<tr>
<th>NIC buf</th>
<th>NIC hw</th>
<th>SW buf</th>
<th>SW hw</th>
<th>Stack buf</th>
<th>code</th>
</tr>
</thead>
</table>

90% expose buffer anomalies among all RLA tickets
Buffer Chain Abstraction

Application

Network stack

NIC

RPC
Packet

Application

Network stack

NIC

Switches
BufScope: Diagnosing RLAs based on Buffers

- Complete visibility at all layers
- Consistent RPC-level semantics
- Low overhead
Design Challenges

**How to define events for various buffers**

- *Drop* for lossy buffer
- *Pause* for lossless buffer
- *Out-of-order* for TCP buf
- ...

**How to obtain RPC semantics from packets in NIC and switch**

- RPC semantics are encapsulated in packet payload

**How to reduce CPU overhead**

- For software monitoring and semantic operations
Event Definition based on Buffer *Classification*

- Congestion for all buffers
- Priority Awareness
  - NIC
  - Priority contention
  - Switch 1: Pause
  - Switch 2: Congestion
- Enqueue features
- NIC
  - Drop
  - Out-of-order

Network stack: Congestion aware vs. unaware
NIC: lossless vs. lossy
Switches: sensitive vs. insensitive
Event Definition based on Buffer *Classification*

- Congestion for all buffers
- Priority Awareness
  - Priority unaware
  - Priority aware
- Enqueue Features
  - Lossless
  - Lossy
- Order Sensitivity
  - Order-sensitive
  - Order-insensitive

5 types of events:
- Congestion
- Priority contention
- Pause
- Drop
- Out-of-order
Event Definition based on Buffer \textit{Classification}

\textbf{Example#1: multi-priority queues in lossy network}

- **Congestion for all buffers**
  - Priority unaware
  - Priority aware
  - \textit{congestion}, 3 events

- **Priority Awareness**
  - Priority aware

- **Enqueue Features**
  - Lossless
  - \textit{pause}

- **Order Sensitivity**
  - Order - sensitive
  - Lossy
  - \textit{drop}
  - Order - insensitive
  - out-of-order
Event Definition based on Buffer Classification

Example#2: TCP receive buffer

- Congestion for all buffers
- Priority Awareness
  - Priority unaware
  - Priority aware
- Enqueue Features
  - Lossless
  - Lossy
- Order Sensitivity
  - Order-sensitive
  - Order-insensitive

congestion

priority contention

pause

drop

out-of-order

3 events
Obtaining RPC Semantics in NIC and Switch

**Goal:**

*Using RPC ID to correlate events in all layers*

Original RPC stream:
- RPC #1 header
- RPC #1 data
- RPC #2 header
- RPC #2 data
- RPC #3 header
- RPC #3 data

RPCs \(\rightarrow\) packets

Packet X:
- Ethernet Header
- IP Header
- TCP Header
- RPC #1 header
- RPC #1 data
- RPC #2 header
- RPC #2 data (the former half)
- FCS

Packet X+1:
- Ethernet Header
- IP Header
- TCP Header
- RPC #2 data (the latter half)
- RPC #3 header
- RPC #3 data
- FCS

**Difficult to obtain RPC semantics for NIC & Switch**

Missing RPC header

Unpredictable location
Per-packet RPC-level Semantic Injection

Multiple RPCs in packet X

RPC #1 header
RPC #1 data
RPC #2 header
RPC #2 data (the former half)

RPC #2 ID
Remaining size

RPC #2 data (the latter half)
RPC #3 header
RPC #3 data

New packet X+1

Packet Header
RPC #2 ID
Offset

Additional payload (10 Bytes)

Missing header in packet X+1

Packet Header
RPC #2 data (the latter half)
RPC #3 header
RPC #3 data
CPU Overhead

- Event reporting
- Semantic injection
- Semantic recovery
- Large CPU overhead

Per-packet processing
BufScope Implementation with Low Overhead

1. SmartNIC Offloading

2. Zigzag encoding and Batching

Network stack

SmartNIC

Agent

Semantic Injection

Switch

Agent

Event Collector

Semantic Recovery

SmartNIC

Agent
Case Study 1: Polling Hang in Host Thread

Polling hang at host recv thread

5ms congestion in receiver NIC

80μs congestion in one switch

Operator

Tenant

RPC ID
Timestamp

Server team
Network team

UDP TCP IP Ethernet All clear
Case Study 2: Priority Contention inRecv NIC

App#1: Wrong priority configuration in a version update

RPC#2 with low priority

Priority contention from App#1

App#2

Operator

Tenant

RPC #2

Timestamp

Operator
Evaluation Setup

Evaluation Goal
- Coverage & Overhead

Baseline
- Tracing+TCP monitor+
  Network monitor

RPC Framework
- Finagle, Alibaba storage

Traffic Patterns
- DCTCP, VL2, Storage, Web

10 Tofino switches

100Gbps switch links

32 x 25Gbps NFP-4000 SmartNIC

DCTCP, VL2, Storage, Web

Tracing+TCP monitor+
Network monitor

Baseline

Evaluation Goal
- Coverage & Overhead

Baseline
- Tracing+TCP monitor+
  Network monitor

RPC Framework
- Finagle, Alibaba storage

Traffic Patterns
- DCTCP, VL2, Storage, Web

10 Tofino switches

100Gbps switch links

32 x 25Gbps NFP-4000 SmartNIC
Microbenchmarks

Event Coverage

100% event coverage

Performance overhead

<1% throughput drop

Coverage Ratio

BufScope

Dapper

Dapper

NetSeer

BufferScope

Tracing (1.0)

Tracing (0.001)

Dapper

NetSeer

Sampling (0.01)

1.409.81

1.361.98

1.349.36

1.400.29

1.337.40

1.396.08

Application Query Per Second under different monitoring tools
BufScope: Diagnosing RLAs based on Buffers

Boost RLA locating from days to minutes

Buffer Classification

Semantic Injection

Offloading & Compression
Thanks for your interest in BufScope

gkh18@mails.tsinghua.edu.cn