DeTail
Reducing the Tail of Flow Completion Times in Datacenter Networks

David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, Randy Katz
A Typical Facebook Page

Modern pages have many components
Creating a Page

Internet

Datacenter Network

Front End  News Feed  Search  Ads  Chat
What’s Required?

• Servers must perform **100’s of data retrievals***
  – Many of which must be performed **serially**
• While meeting a deadline of **200-300ms****
  – SLA measured at the **99.9th percentile****

• Only have **2-3ms** per data retrieval
  – Including communication and computation

*The Case for RAMClouds [SIGOPS’09]
**Better Never than Late [SIGCOMM’11]
What is the Network’s Role?

- Analyzed distribution of RTT measurements:
  - Median RTT takes $334\text{µs}$, but 6% take over $2\text{ms}$
  - Can be as high as $14\text{ms}$

Network delays alone can consume the data retrieval’s time budget
Why the Tail Matters

- Recall: **100’s** of data retrievals **per page** creation

- The **unlikely** event of a data retrieval taking too long is **likely** to happen on **every page creation**
  - Data retrieval dependencies can **magnify** impact
Impact on Page Creation

- Under the RTT distribution, 150 data retrievals take 200ms (ignoring computation time)

As Facebook already at 130 data retrievals per page, need to address network delays
App-Level Mitigation

• Use **timeouts & retries** for critical data retrievals
  – **Inefficient** because of high network variance
  – Choose from conservative timeouts and **long delays** or tight timeouts and **increased server load**

• **Hide the problem** from the user
  – By caching and serving **stale data**
  – Rendering pages **incrementally**
  – User often notices, becomes **annoyed / frustrated**

**Need to focus on the root cause**
Outline

- Causes of long data retrieval times
- Cutting the tail with DeTail
- Evaluation
Causes of Long Data Retrieval Times

• Data retrievals are **short, highly variable flows**
  – Typically under 20KB in size, with many under 2KB*

• **Short flows** provide insufficient information for transport to agilely respond to packet drops

• **Variable flow sizes** decrease efficacy of network-layer load balancers

*Data Center TCP (DCTCP) [SIGCOMM’10]*
Transport Layer Response

Transport does not have sufficient information to respond agilely
Network Layer Load Balancers

• Expected to support single-path assumption
• Common approach: hash flows to paths
  – Does not consider flow size or sending rate

• Results in uneven load spreading
  – Leads hotspots and increased queuing delays

The single-path assumption restricts the ability to agilely balance load
Recent Proposals

• Reduce packet drops
  – By cross-flow learning [DCTCP] or explicit flow scheduling [D³]
  – Maintain the single-path assumption

• Adaptively move traffic
  – By creating subflows [MPTCP] or periodically remapping flows [Hedera]
  – Not sufficiently agile to support short flows
Outline

• Causes of long data retrieval times

• Cutting the tail with DeTail

• Evaluation
DeTail Stack

• Use **in-network** mechanisms to maximize **agility**
• **Remove restrictions** that hinder performance

• **Well-suited** for datacenters
  – Single administrative domain
  – Reduced backward compatibility requirements
Hop-by-hop Push-back

- Agile link-layer response to prevent packet drops

What about head-of-line blocking?
Adaptive Load Balancing

• Agile network-layer approach for balancing load

Synergistic relationship: local output queues indicate downstream congestion because of push-back
Load Balancing Efficiently

• DC flows have **varying timeliness** requirements*
  – How to efficiently consider packet priority?

• **Compare** queue occupancies for **every decision**
  – How to efficiently compare many of them?

*Data Center TCP (DCTCP) [SIGCOMM’10]
Priority in Load Balancing

How to enqueue packet so it is sent soonest?
Priority in Load Balancing

• Approach: track how many *bytes to be sent* before new packet

• Use *per-priority* counters
  – Update on each packet enqueue/dequeue
  – Compare counters to find least occupied port
Comparing Queue Occupancies

- Many counter comparisons required for every forwarding decision

- Want to efficiently pick the least occupied port
  - Pre-computation is hard as solution is destination, time dependent
Use Per-Counter Thresholding

- Pick a **good** port, instead of the **best** one
Reorder-Resistant Transport

• Handle packet reordering due to load balancing
  – Disable TCP’s fast recovery and fast retransmission

• Respond to congestion (no more packet drops)
  – Monitor output queues and use ECN to throttle flows
DeTail Stack

Layer
- Application
- Transport
- Network
- Link
- Physical

Component
- Reorder-Resistant Transport
- Adaptive Load Balancing
- Hop-by-hop Push-back

Function
- Support lower layers
- Evenly balance load
- Prevent packet drops
Outline

• Causes of long data retrieval times

• Cutting the tail with DeTail

• Evaluation
Simulation and Implementation

- **NS-3 simulation**

- **Click implementation**
  - Drivers and NICs buffer hundreds of packets
  - Must rate-limit Click to underflow buffers
• FatTree: 128-server (NS-3) / 16-server (Click)
• Oversubscription factor of 4x

Reproduced From: A Scalable Commodity Datacenter Network Architecture [SIGCOMM’08]
Setup

• Baseline
  – TCP NewReno
  – Flow hashing based on IP headers
  – Prioritization of data retrievals vs. background

• Metric
  – Reduction in 99.9th percentile completion time
Page Creation Workload

- Retrieval size: 2, 4, 8, 16, 32 KB*
- Background traffic: 1MB flows

DeTail reduces 99.9th percentile page creation time by over 50%

* Covers range of query traffic sizes reported by DCTCP
Is the Whole Stack Necessary?

• Evaluated push-back w/o adaptive load balancing
  – Performs worse than baseline

DeTail’s mechanisms work together, overcoming their individual limitations
What About Link Failures?

• 10s of link failures occur per day*
  – Creates permanent network imbalance

• Example
  – Core-AGG link degrades from 1Gbps to 100Mbps
  – DeTail achieves 91% reduction in the 99.9th percentile

DeTail effectively moves traffic away from failures, appropriately balancing load

*Understanding Network Failures in Data Centers [SIGCOMM’11]
What About Long Background Flows?

- Background Traffic: 1, 16, 64MB flows*
- Light **data retrieval** traffic

---

*DeTail’s adaptive load balancing also helps long flows

*Covers range of update flow sizes reported by DCTCP
Conclusion

• **Long tail harms** page creation
  – The **extreme** case becomes the **common** case
  – Limits number of data retrievals per page

• **The DeTail stack improves** long tail performance
  – Can reduce the **99.9\(^{th}\)** percentile by more than **50%**