Tales of the Tail
Hardware, OS, and Application-level Sources of Tail Latency

Jialin Li, Naveen Kr. Sharma,
Dan R. K. Ports and Steven D. Gribble

UNIVERSITY of WASHINGTON

February 2, 2015
What is Tail Latency?

In Facebook’s Memcached deployment, median latency is 100 µs, but 95th percentile latency ≥ 1 ms. In this talk, we will explore why some requests take longer than expected and what causes them to get delayed.
What is Tail Latency?

In Facebook’s Memcached deployment, median latency is 100 µs, but 95th percentile latency ≥ 1 ms. In this talk, we will explore why some requests take longer than expected and what causes them to get delayed.
What is Tail Latency?

In Facebook's Memcached deployment, median latency is 100 µs, but 95th percentile latency is ≥ 1 ms.

In this talk, we will explore why some requests take longer than expected, and what causes them to get delayed.
What is Tail Latency?

- In Facebook’s Memcached deployment,
  - Median latency is $100\mu s$, but $95^{th}$ percentile latency $\geq 1ms$. 
What is Tail Latency?

In Facebook’s Memcached deployment,
- Median latency is 100µs, but 95\textsuperscript{th} percentile latency ≥ 1ms.

In this talk, we will explore
- Why some requests take longer than expected?
- What causes them to get delayed?
Why is the Tail important?

- Low latency is crucial for interactive services.
  - 500ms delay can cause 20% drop in user traffic. [Google Study]
  - Latency is directly tied to traffic, hence revenue.
Low latency is crucial for interactive services.
- 500ms delay can cause 20% drop in user traffic. [Google Study]
- Latency is directly tied to traffic, hence revenue.

What makes it challenging is today’s datacenter workloads.

Interactive services are highly parallel.

Single client request spawns thousands of sub-tasks.
- Overall latency depends on slowest sub-task latency.
- Bad Tail $\Rightarrow$ Probability of any one sub-task getting delayed is high.
A real-life example

Data dependency DAG for a small request

Nishtala et. al. Scaling memcache at Facebook, NSDI 2013.
A real-life example

All requests have to finish within the SLA latency.

Data dependency DAG for a small request

Nishtala et. al. Scaling memcache at Facebook, NSDI 2013.
What can we do?

- People in industry have worked hard on solutions.
- Hedged Requests [Jeff Dean et. al.]
  - Effective sometimes, but adds application specific complexity.
- Intelligently avoid slow machines
  - Keep track of server status; route requests around slow nodes.
What can we do?

- People in industry have worked hard on solutions.
- Hedged Requests [Jeff Dean et. al.]
  - Effective sometimes, but adds application specific complexity.
- Intelligently avoid *slow* machines
  - Keep track of server status; route requests around slow nodes.

- Attempts to build predictable response out of less predictable parts.
- We still don’t know *what* is causing requests to get delayed.
Our Approach

1. Pick some real life applications: **RPC Server, Memcached, Nginx**.
2. Generate the ideal latency distribution.
3. Measure the actual distribution on a standard Linux server.
4. Identify a factor causing deviation from ideal distribution.
5. Explain and mitigate it.
6. Iterate over this till we reach the ideal distribution.
Rest of the Talk

1. Introduction
2. Predicted Latency from Queuing Models
3. Measurements: Sources of Tail Latencies
4. Summary
What is the ideal latency for a network server?
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.
What is the ideal latency for a network server?

- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.

Given the arrival distribution and request processing time,

- We can predict the time spent by a request in the server.
Predicted Latency from Queuing Models

Tail latency characteristics

What is the ideal latency distribution?

Assume a server with a single worker with 50 µs fixed processing time.

Distribution 1

P[X >= x]

CCDF

Latency in micro-seconds

Distribution 2

Uniform Request Arrival

Poisson at 70% Utilization

Poisson at 90% Utilization

Poisson at 70% - 4 workers

CCDF

P[X >= x]

Latency in micro-seconds
Predicted Latency from Queuing Models

Tail latency characteristics

What is the ideal latency distribution?

Assume a server with single worker with 50 µs fixed processing time.

Distribution 1

Uniform Request Arrival

Poisson at 70% Utilization

Poisson at 90% Utilization

Poisson at 70% - 4 workers
What is the ideal latency distribution?

Assume a server with a single worker with 50 $\mu$s fixed processing time.

Distribution 1

Uniform Request Arrival

Poisson at 70% Utilization

Poisson at 90% Utilization

Poisson at 70% - 4 workers

99th percentile $\Rightarrow$ 60 $\mu$s
What is the ideal latency distribution?

Assume a server with a single worker with a fixed processing time of 50 µs.
Predicted Latency from Queuing Models

Tail latency characteristics

What is the ideal latency distribution?

Assume a server with single worker with $50\,\mu s$ fixed processing time.

Distribution 1

Distribution 2

Poisson at 70% Utilization

Poisson at 90% Utilization

Poisson at 70% - 4 workers

CCDF $P[X \geq x]$
What is the ideal latency distribution?

- Assume a server with single worker with 50 \( \mu s \) fixed processing time.
What is the ideal latency distribution?

- Assume a server with single worker with 50 $\mu$s fixed processing time.
What is the ideal latency distribution?

- Assume a server with single worker with 50 μs fixed processing time.

Inherent tail latency due to request burstiness.
What is the ideal latency distribution?

- Assume a server with single worker with 50 $\mu$s fixed processing time.

Tail latency depends on the average server utilization.
What is the ideal latency distribution?

- Assume a server with single worker with 50 $\mu$s fixed processing time.

Additional workers can reduce tail latency, even at constant utilization.
1 Introduction

2 Predicted Latency from Queuing Models

3 Measurements: Sources of Tail Latencies

4 Summary
Testbed

- Cluster of standard datacenter machines.
  - 2 x Intel L5640 6 core CPU
  - 24 GB of DRAM
  - Mellanox 10Gbps NIC
  - Ubuntu 12.04, Linux Kernel 3.2.0

- All servers connected to a single 10 Gbps ToR switch.
- One server runs Memcached, others run workload generating clients.
- Other application results are in the paper.
Timestamping Methodology

- Append a blank buffer \( \approx 32 \) bytes to each request.
- Overwrite buffer with timestamps as it goes through the server.

Very low overhead and no server side logging.
How far are we from the ideal?
How far are we from the ideal?

Single CPU, single core, Memcached running at 80% utilization.
How far are we from the ideal?

Single CPU, single core, Memcached running at 80% utilization.
How far are we from the ideal?

![Graph showing CCDF and latency in micro-seconds for Ideal Model and Standard Linux. The graph indicates a 30x difference.]

**Single CPU, single core, Memcached running at 80% utilization.**
## Rest of the talk

<table>
<thead>
<tr>
<th>Source of Tail Latency</th>
<th>Potential way to fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Processes</td>
<td></td>
</tr>
<tr>
<td>Multicore Concurrency</td>
<td></td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
<tr>
<td>Source of Tail Latency</td>
<td>Potential way to fix</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td><strong>Background Processes</strong></td>
<td></td>
</tr>
<tr>
<td>Multicore Concurrency</td>
<td></td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
</tbody>
</table>
How can background processes affect tail latency?

- Memcached threads time-share a CPU core with other processes.
- We need to wait for other processes to relinquish CPU.
- Scheduling time-slices are usually couple of milliseconds.
How can background processes affect tail latency?

- Memcached threads time-share a CPU core with other processes.
- We need to wait for other processes to relinquish CPU.
- Scheduling time-slices are usually couple of milliseconds.

How can we mitigate it?

- Raise priority (decrease niceness) ⇒ More CPU time.
- Upgrade scheduling class to real-time ⇒ Pre-emptive power.
- Run on a dedicated core ⇒ No interference what-so-ever.
Impact of Background Processes

Interference from background processes has a large effect on the tail.

Single CPU, single core, Memcached running at 80% utilization.
Impact of Background Processes

Interference from background processes has a large effect on the tail.

Single CPU, single core, Memcached running at 80% utilization.
Impact of Background Processes

Single CPU, single core, Memcached running at 80% utilization.
Impact of Background Processes

![Graph showing CCDF vs. latency for different scheduling models]

- Ideal Model
- Standard Linux
- Maximum Priority
- Realtime Scheduling
- Dedicated Core

Interference from background processes has a large effect on the tail.

Single CPU, single core, Memcached running at 80% utilization.
Impact of Background Processes

Interference from background processes has a large effect on the tail.

Single CPU, single core, Memcached running at 80% utilization.
<table>
<thead>
<tr>
<th>Source of Tail Latency</th>
<th>Potential way to fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Processes</td>
<td>Isolate by running on a dedicated core.</td>
</tr>
<tr>
<td>Multicore Concurrency</td>
<td></td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
<tr>
<td>Source of Tail Latency</td>
<td>Potential way to fix</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Background Processes</td>
<td>Isolate by running on a dedicated core.</td>
</tr>
<tr>
<td>Multicore Concurrency</td>
<td></td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
</tbody>
</table>
Does adding more CPU cores improve tail latency?

![CCDF graph](image)

**CCDF** $P[X \geq x]$

Latency in micro-seconds

1 core Ideal Model

Single CPU, 4 cores, Memcached running at 80% utilization.
Does adding more CPU cores improve tail latency?

Single CPU, 4 cores, Memcached running at 80% utilization.
Does adding more CPU cores improve tail latency?

Single CPU, 4 cores, Memcached running at 80% utilization.
Does adding more CPU cores improve tail latency?

![Graph showing CCDF of latency in micro-seconds for different CPU configurations.](image)

Single CPU, 4 cores, Memcached running at 80% utilization.
Does adding more CPU cores improve tail latency?

![Graph showing CCDF and Latency in micro-seconds for different configurations: 1 core Ideal Model, 4 core Ideal Model, 1 core Linux, 4 core Linux. The graph indicates a 15x improvement in latency.]
Does adding more CPU cores improve tail latency?

- Yes it does! Provided we maintain a single queue abstraction.
Does adding more CPU cores improve tail latency?

- Yes it does! Provided we maintain a single queue abstraction.

Ideal Model
Does adding more CPU cores improve tail latency?

- Yes it does! Provided we maintain a single queue abstraction.
- Memcached partitions requests statically among threads.

**Ideal Model**

**Memcached Architecture**
Does adding more CPU cores improve tail latency?

- Yes it does! Provided we maintain a single queue abstraction.
- Memcached partitions requests statically among threads.

**Ideal Model**

**Memcached Architecture**

How can we mitigate it?

- Modify Memcached concurrency model to use a single queue.
Impact of Multicore Concurrency Model

For multi-threaded applications, a single queue abstraction can reduce tail latency.

Single CPU, 4 cores, Memcached running at 80% utilization.
Impact of Multicore Concurrency Model

For multi-threaded applications, a single queue abstraction can reduce tail latency.

Single CPU, 4 cores, Memcached running at 80% utilization.
Impact of Multicore Concurrency Model

For multi-threaded applications, a single queue abstraction can reduce tail latency.

Single CPU, 4 cores, Memcached running at 80% utilization.
Impact of Multicore Concurrency Model

For multi-threaded applications, a single queue abstraction can reduce tail latency.

Single CPU, 4 cores, Memcached running at 80% utilization.
<table>
<thead>
<tr>
<th>Source of Tail Latency</th>
<th>Potential way to fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Processes</td>
<td>Isolate by running on a dedicated core.</td>
</tr>
<tr>
<td>Concurrency Model</td>
<td>Ensure a single queue abstraction.</td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
<tr>
<td>Source of Tail Latency</td>
<td>Potential way to fix</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Background Processes</td>
<td>Isolate by running on a dedicated core.</td>
</tr>
<tr>
<td>Concurrency Model</td>
<td>Ensure a single queue abstraction.</td>
</tr>
<tr>
<td>Interrupt Processing</td>
<td></td>
</tr>
</tbody>
</table>
How can interrupts affect tail latency?

- By default, Linux `irqbalance` spreads interrupts across all cores.
- OS pre-empts Memcached threads frequently.
- Introduces extra context switching overheads and cache pollution.
How can interrupts affect tail latency?

- By default, Linux irqbalance spreads interrupts across all cores.
- OS pre-empts Memcached threads frequently.
- Introduces extra context switching overheads and cache pollution.

How can we mitigate it?

- Separate cores for interrupt processing and application threads.
- 3 cores run Memcached threads, and 1 core processes interrupts.
Measurements: Sources of Tail Latencies

Impact of Interrupt Processing

Single CPU, 4 cores, Memcached running at 80% utilization.
Impact of Interrupt Processing

Single CPU, 4 cores, Memcached running at 80% utilization.
Measurements: Sources of Tail Latencies

Impact of Interrupt Processing

Separate cores for interrupt and application processing improves tail latency.

Single CPU, 4 cores, Memcached running at 80% utilization.
## Other sources of tail latency

<table>
<thead>
<tr>
<th>Source of Tail Latency</th>
<th>Underlying Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Scheduling Policy</td>
<td>Non-FIFO ordering of requests.</td>
</tr>
<tr>
<td>NUMA Effects</td>
<td>Increased latency across NUMA nodes.</td>
</tr>
<tr>
<td>Hyper-threading</td>
<td>Contending hyper-threads can increase latency.</td>
</tr>
<tr>
<td>Power Saving Features</td>
<td>Extra time required to wake CPU from idle state.</td>
</tr>
</tbody>
</table>
Summary and Future Works

- We explored hardware, OS and application-level sources of tail latency.
- Pin-point sources using fine-grained timestamping, and an ideal model.
- We obtain substantial improvements, close to ideal distributions.
- 99.9th percentile latency of Memcached from 5 ms to 32 μs.
Summary and Future Works

- We explored hardware, OS and application-level sources of tail latency.
- Pin-point sources using finegrained timestamiing, and an ideal model.
- We obtain substantial improvements, close to ideal distributions.
- 99.9th percentile latency of Memcached from 5 ms to 32 μs.

- Sources of tail latency in multi-process environment.
- How does virtualization effect tail latency?
- Overhead of virtualization, interference from other VMs.
- New effects when moving to a distributed setting, network effects.