Tales of the Tail

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

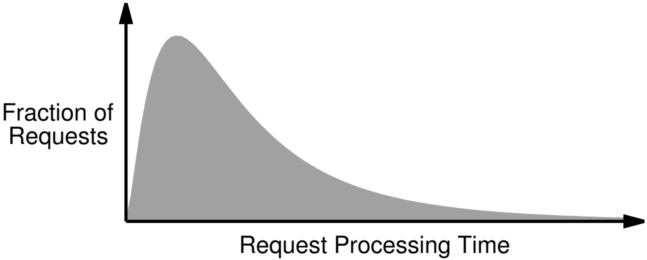
Jialin Li, **Naveen Kr. Sharma**,
Dan R. K. Ports and Steven D. Gribble

UNIVERSITY *of* WASHINGTON

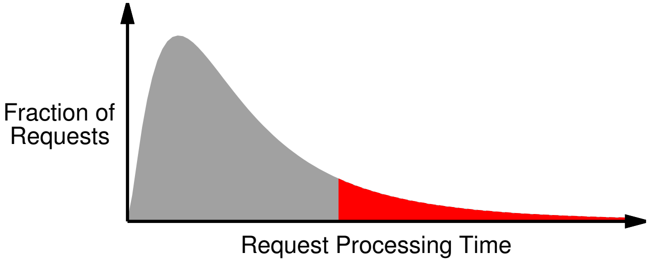
February 2, 2015

What is Tail Latency?

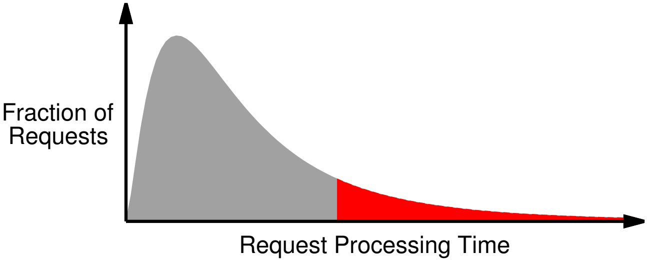
What is Tail Latency?



What is Tail Latency?

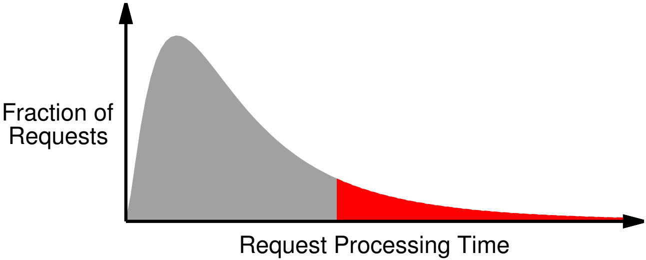


What is Tail Latency?



- 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 $100\mu s$, but 95^{th} percentile latency $\geq 1ms$.

In this talk, we will explore

- Why some requests take longer than expected?
- What causes them to get delayed?

Why is the Tail important?

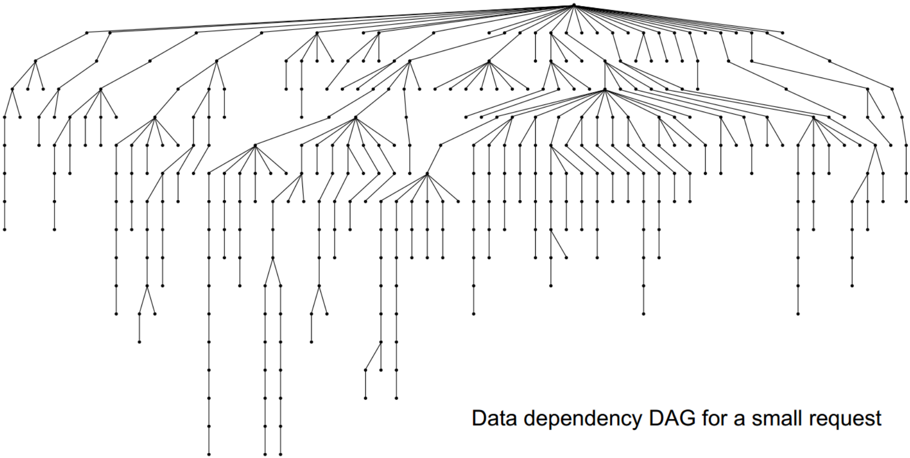
- Low latency is crucial for interactive services.
    - 500ms delay can cause 20% drop in user traffic. [Google Study]
    - Latency is directly tied to traffic, hence revenue.

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.

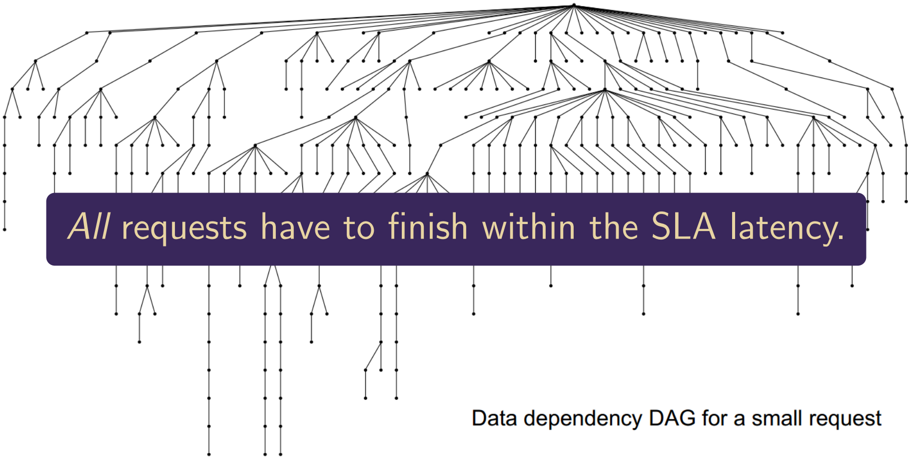
- What makes it challenging is today's datacenter workloads.
- Interactive services are highly parallel.
- Single client request spawns thousands of sub-tasks.
  - Overall latency depends on slowest sub-task latency.
  - Bad Tail $\Rightarrow$ Probability of any one sub-task getting delayed is high.

A real-life example



Nishtala et. al. Scaling memcache at Facebook, NSDI 2013.

A real-life example



All requests have to finish within the SLA latency.

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

What is the ideal latency for a network server?

What is the ideal latency for a network server?

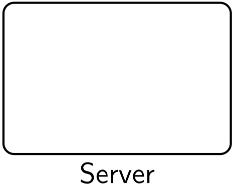
- Ideal baseline for comparing measured performance.

What is the ideal latency for a network server?

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

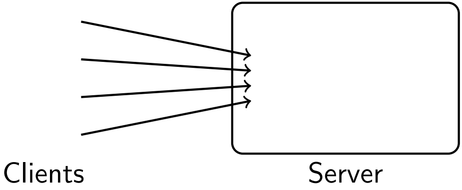
What is the ideal latency for a network server?

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



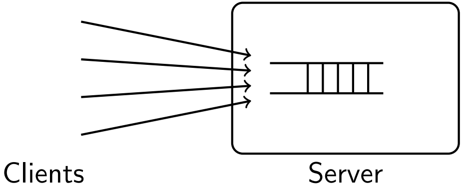
What is the ideal latency for a network server?

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



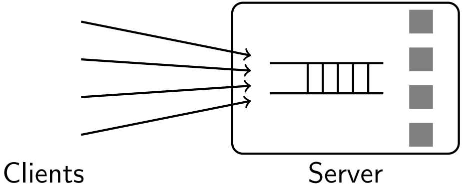
What is the ideal latency for a network server?

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



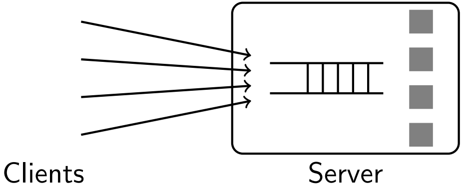
What is the ideal latency for a network server?

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

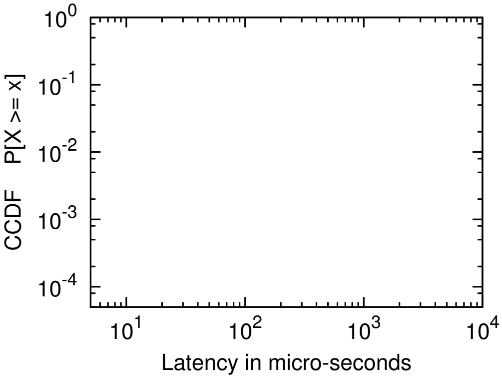


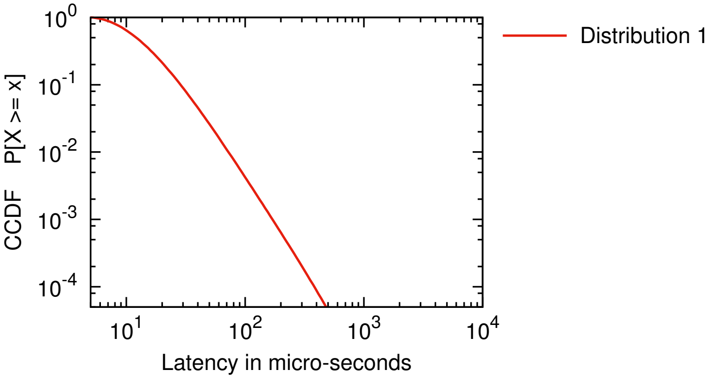
What is the ideal latency for a network server?

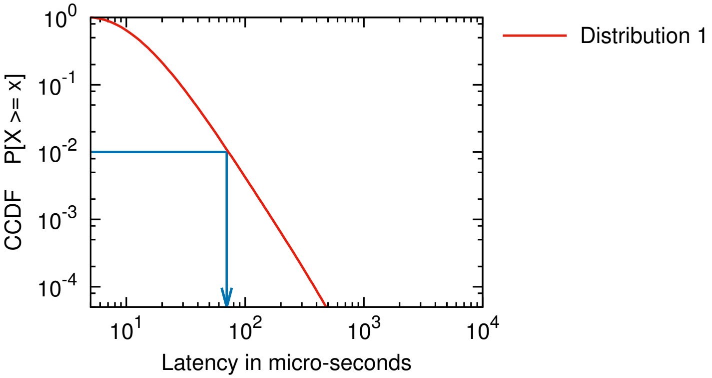
- Ideal baseline for comparing measured performance.
- Assume a simple model, and apply queuing theory.



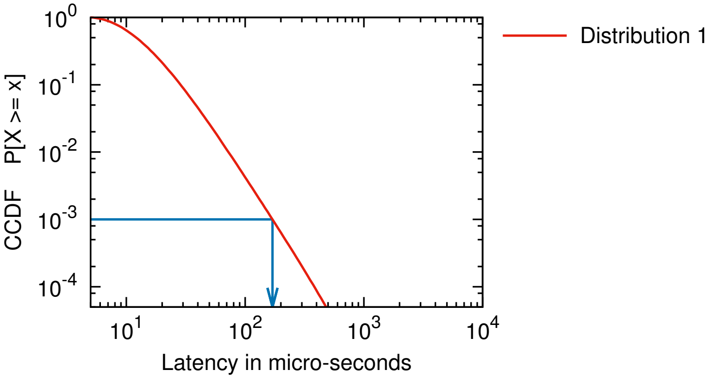
- Given the arrival distribution and request processing time,
- We can predict the time spent by a request in the server.


CCDF P[X >= x]
10^0
10^-1
10^-2
10^-3
10^-4
10^1
10^2
10^3
10^4
Latency in micro-seconds

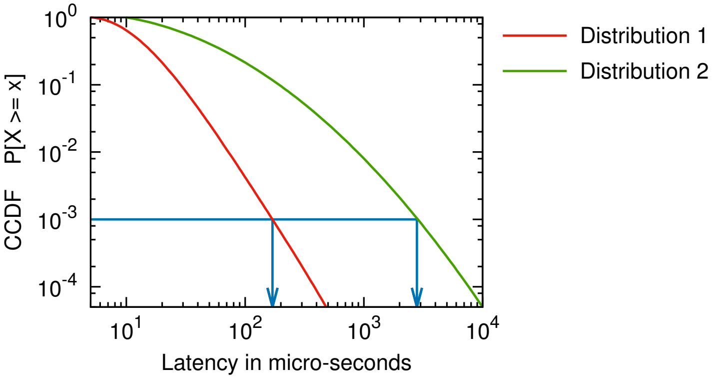

Distribution 1
CCDF P[X >= x]
10^0
10^-1
10^-2
10^-3
10^-4
10^1
10^2
10^3
10^4
Latency in micro-seconds



99th percentile $\Rightarrow$ 60 μs



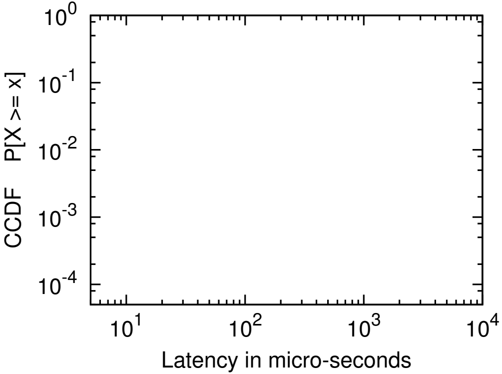
99.9th percentile $\Rightarrow$ 200 μs



Distribution 1
Distribution 2
CCDF P[X >= x]
Latency in micro-seconds
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^1
10^2
10^3
10^4

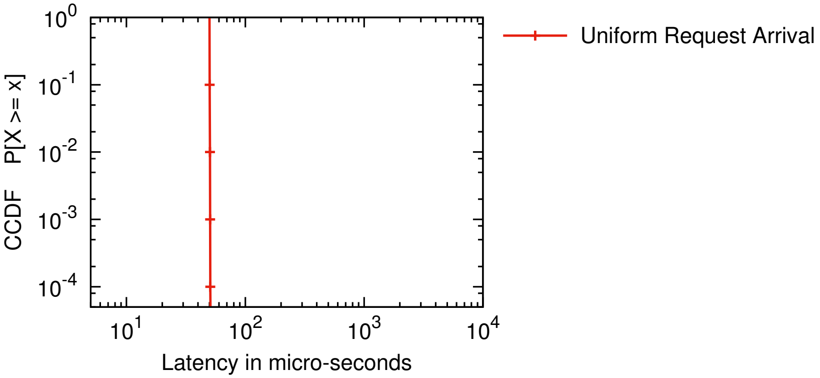
What is the ideal latency distribution?

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



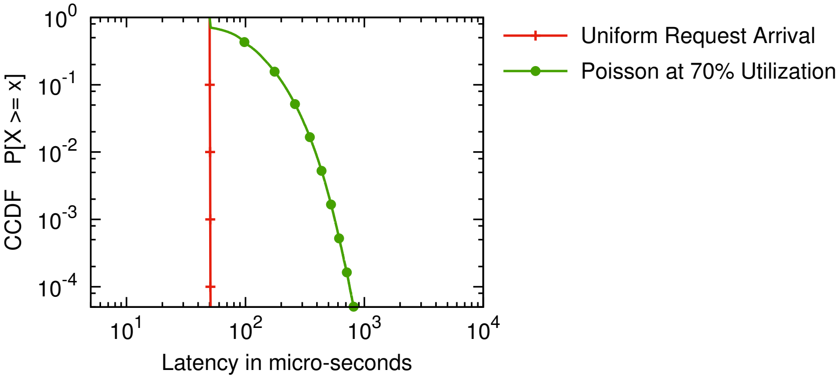
What is the ideal latency distribution?

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



What is the ideal latency distribution?

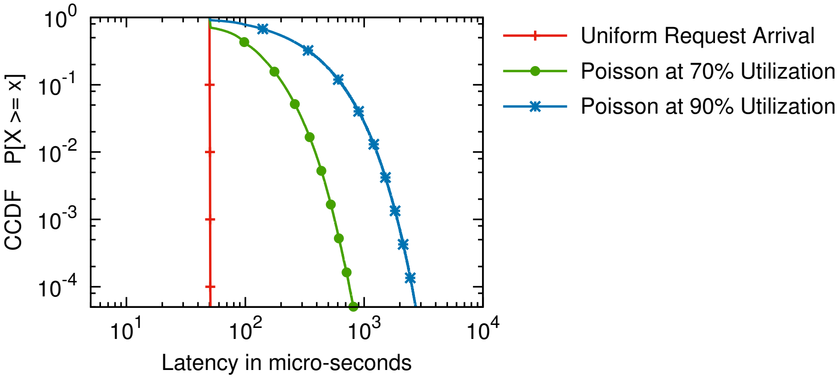
- Assume a server with single worker with 50 μs fixed processing time.



Inherent tail latency due to request burstiness.

What is the ideal latency distribution?

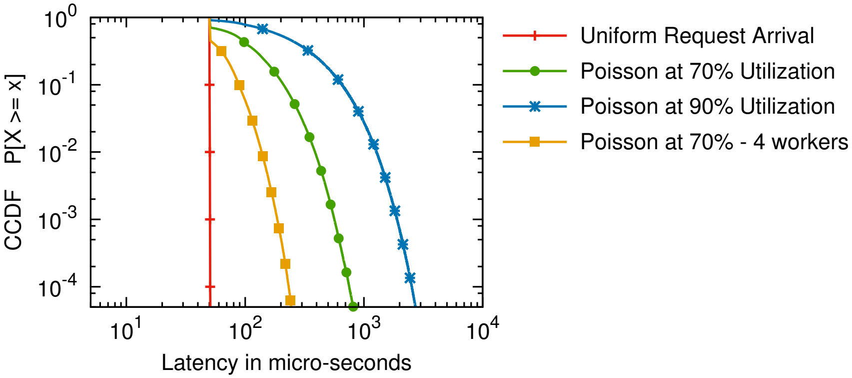
- Assume a server with single worker with 50 μs fixed processing time.



Tail latency depends on the average server utilization.

What is the ideal latency distribution?

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



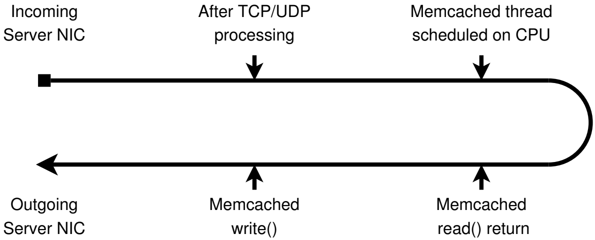
Additional workers can reduce tail latency, even at constant utilization.

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

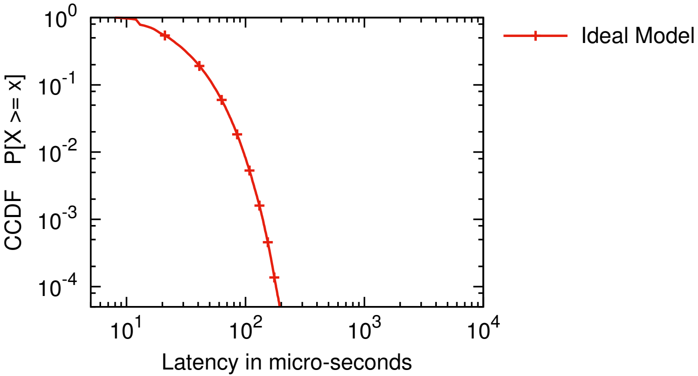
- Append a blank buffer ≈ 32 bytes to each request.
- Overwrite buffer with timestamps as it goes through the server.



- Very low overhead and no server side logging.

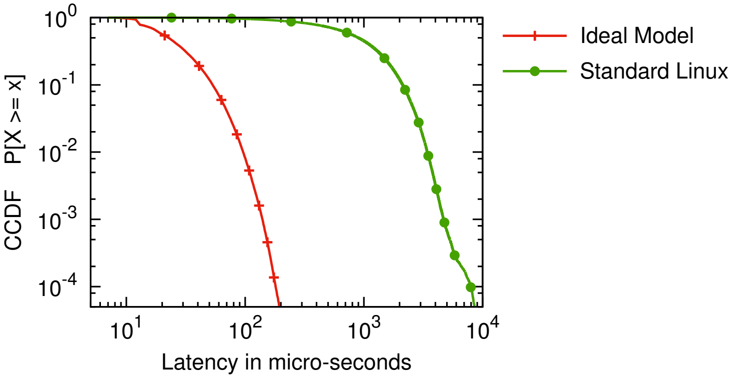
How far are we from the ideal?

How far are we from the ideal?



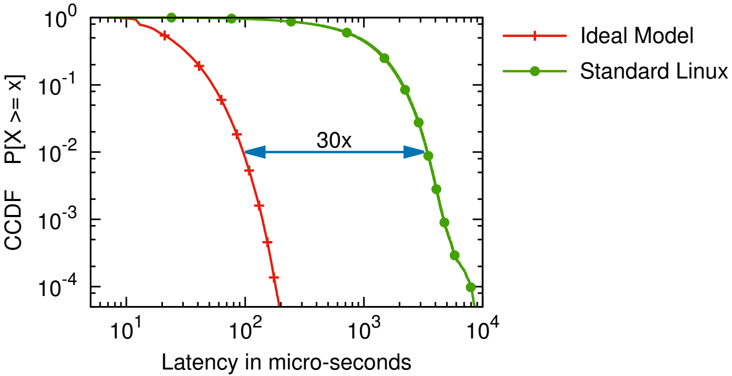
Single CPU, single core, Memcached running at 80% utilization.

How far are we from the ideal?



Single CPU, single core, Memcached running at 80% utilization.

How far are we from the ideal?



Single CPU, single core, Memcached running at 80% utilization.

Rest of the talk

Source of Tail Latency	Potential way to fix
Background Processes	
Multicore Concurrency	
Interrupt Processing	

Rest of the talk

Source of Tail Latency	Potential way to fix
Background Processes	
Multicore Concurrency	
Interrupt Processing	

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.

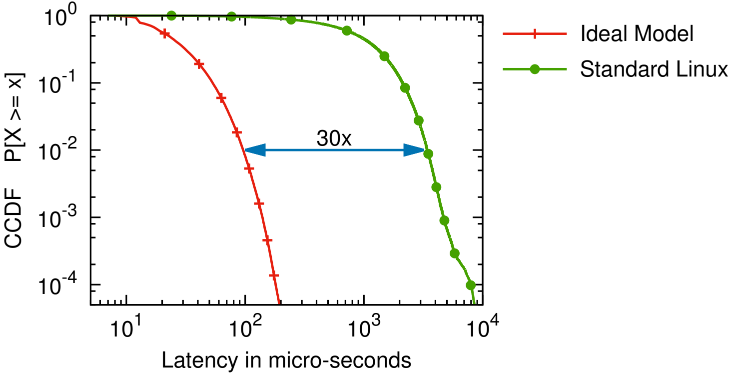
How can background processes affect tail latency?

- 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?

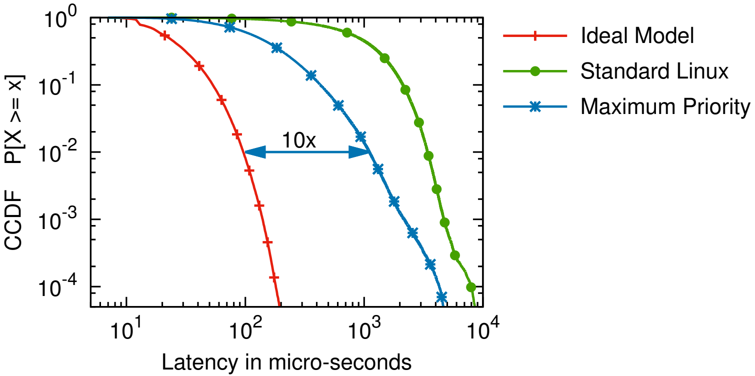
- Raise priority (decrease niceness) $\Rightarrow$ More CPU time.
- Upgrade scheduling class to real-time $\Rightarrow$ Pre-emptive power.
- Run on a dedicated core $\Rightarrow$ No interference what-so-ever.

Impact of Background Processes



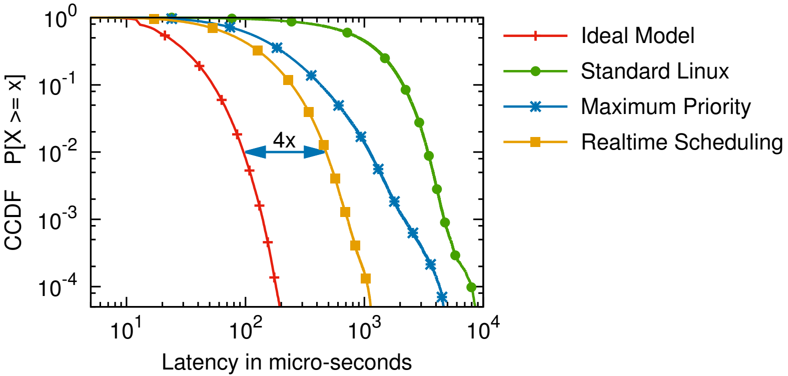
Single CPU, single core, Memcached running at 80% utilization.

Impact of Background Processes



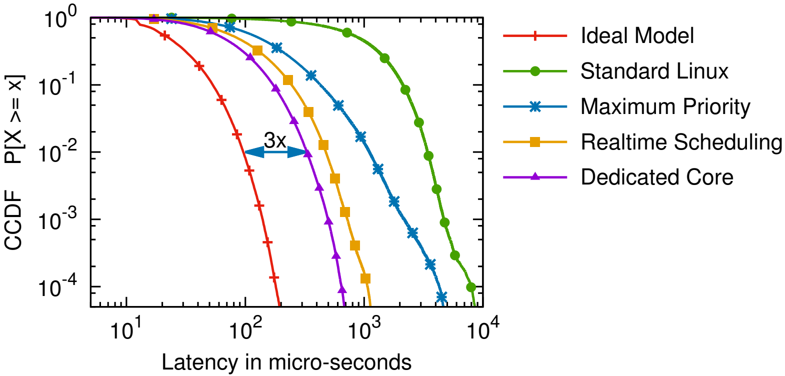
Single CPU, single core, Memcached running at 80% utilization.

Impact of Background Processes



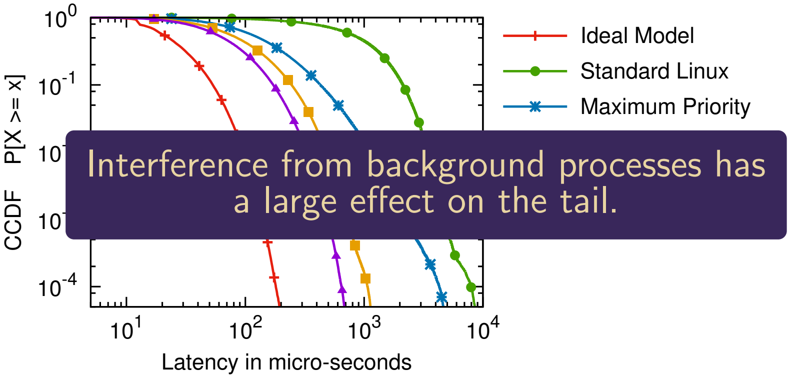
Single CPU, single core, Memcached running at 80% utilization.

Impact of Background Processes



Single CPU, single core, Memcached running at 80% utilization.

Impact of Background Processes



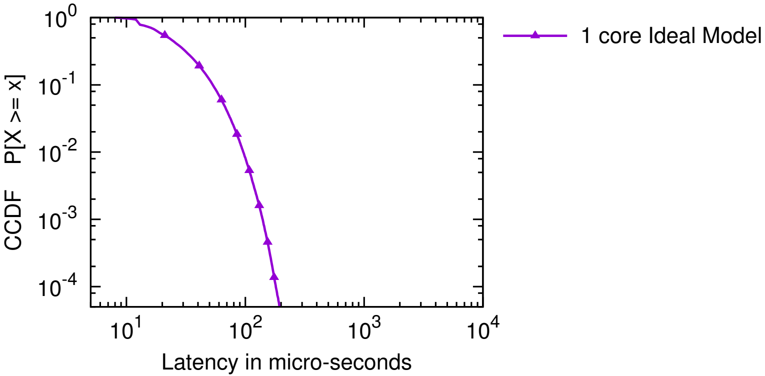
Interference from background processes has a large effect on the tail.

Single CPU, single core, Memcached running at 80% utilization.

Source of Tail Latency	Potential way to fix
Background Processes	Isolate by running on a dedicated core.
Multicore Concurrency	
Interrupt Processing	

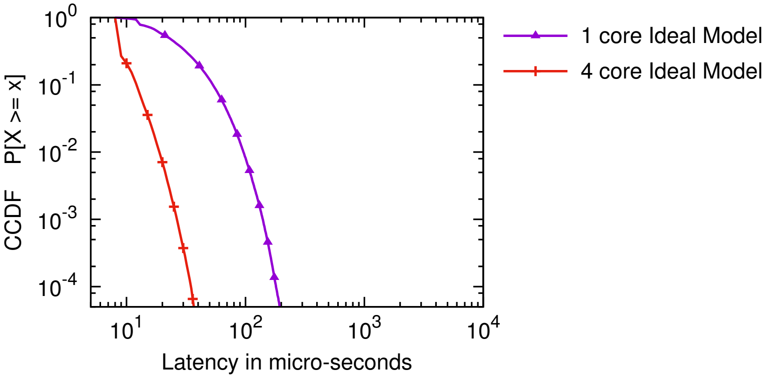
Source of Tail Latency	Potential way to fix
Background Processes	Isolate by running on a dedicated core.
Multicore Concurrency	
Interrupt Processing	

Does adding more CPU cores improve tail latency?



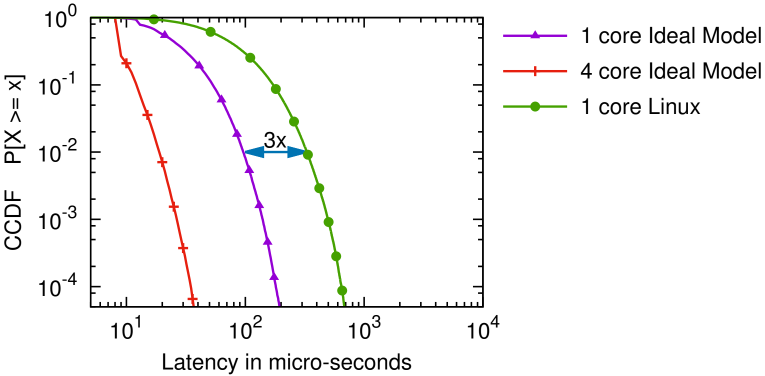
Single CPU, 4 cores, Memcached running at 80% utilization.

Does adding more CPU cores improve tail latency?



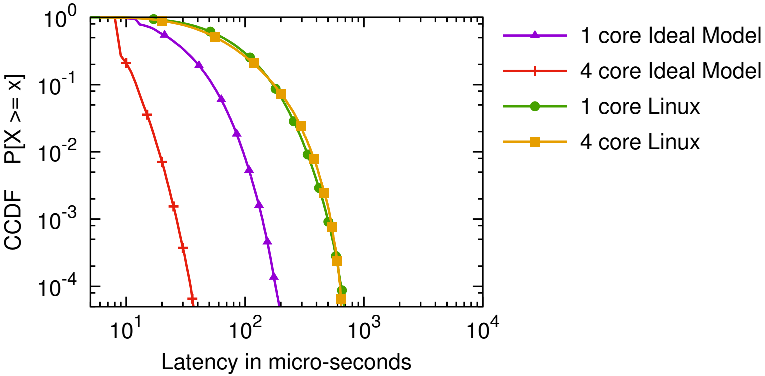
Single CPU, 4 cores, Memcached running at 80% utilization.

Does adding more CPU cores improve tail latency?



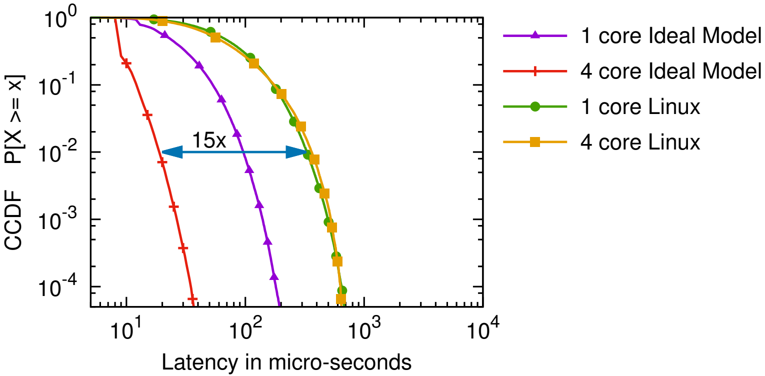
Single CPU, 4 cores, Memcached running at 80% utilization.

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?



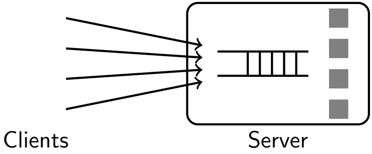
Single CPU, 4 cores, Memcached running at 80% utilization.

Does adding more CPU cores improve tail latency?

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

Does adding more CPU cores improve tail latency?

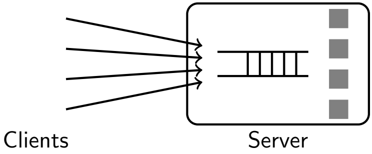
- Yes it does! Provided we maintain a single queue abstraction.



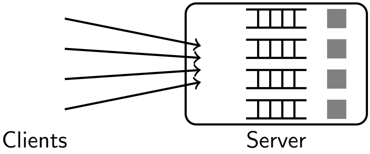
Ideal Model

Does adding more CPU cores improve tail latency?

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



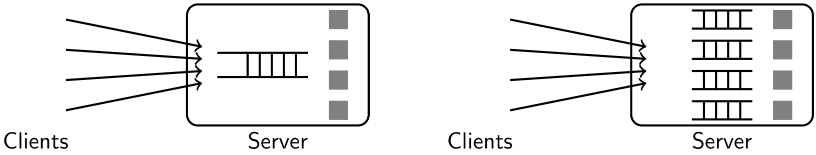
Ideal Model



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.

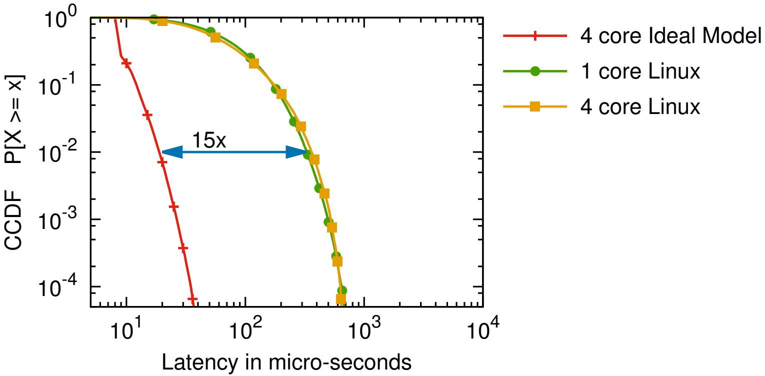


Ideal Model **Memcached Architecture**

How can we mitigate it?

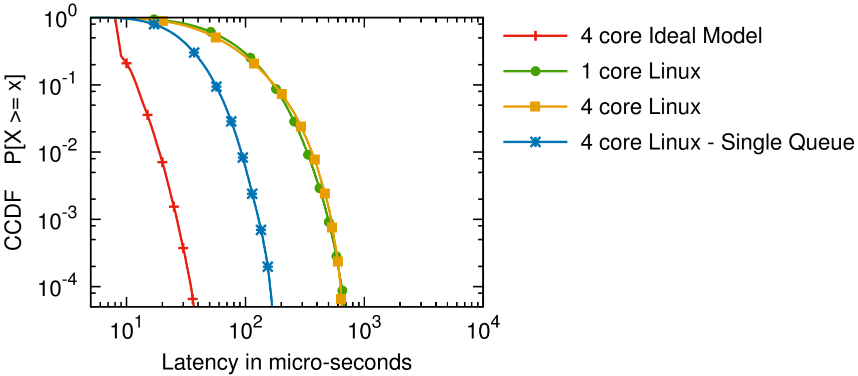
- Modify Memcached concurrency model to use a single queue.

Impact of Multicore Concurrency Model



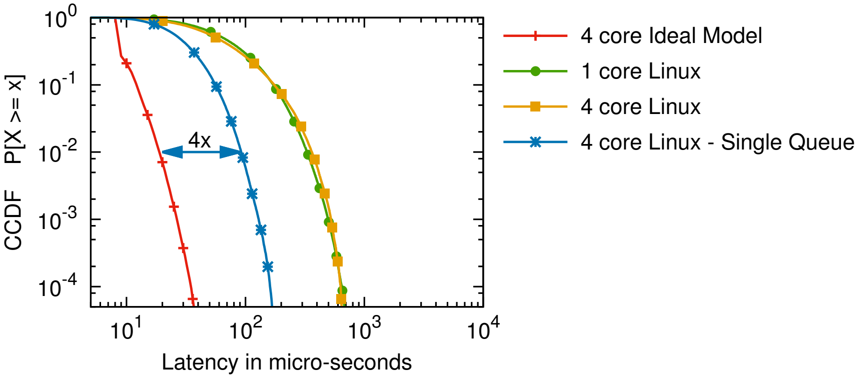
Single CPU, 4 cores, Memcached running at 80% utilization.

Impact of Multicore Concurrency Model



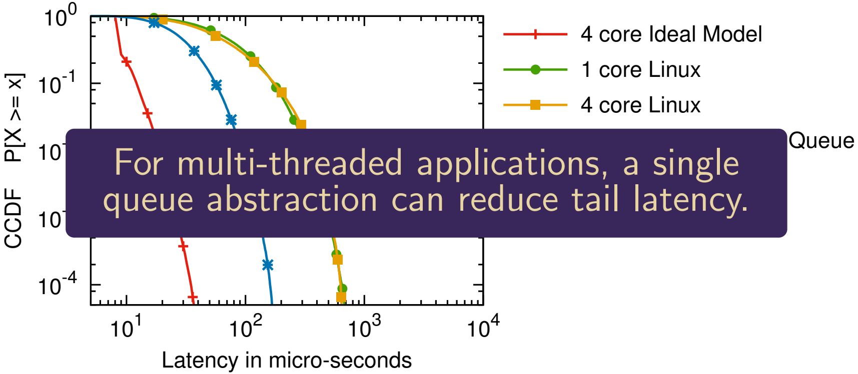
Single CPU, 4 cores, Memcached running at 80% utilization.

Impact of Multicore Concurrency Model



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.

Source of Tail Latency	Potential way to fix
Background Processes	Isolate by running on a dedicated core.
Concurrency Model	Ensure a single queue abstraction.
Interrupt Processing	

Source of Tail Latency	**Potential way to fix**
Background Processes	Isolate by running on a dedicated core.
Concurrency Model	Ensure a single queue abstraction.
Interrupt Processing	

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.

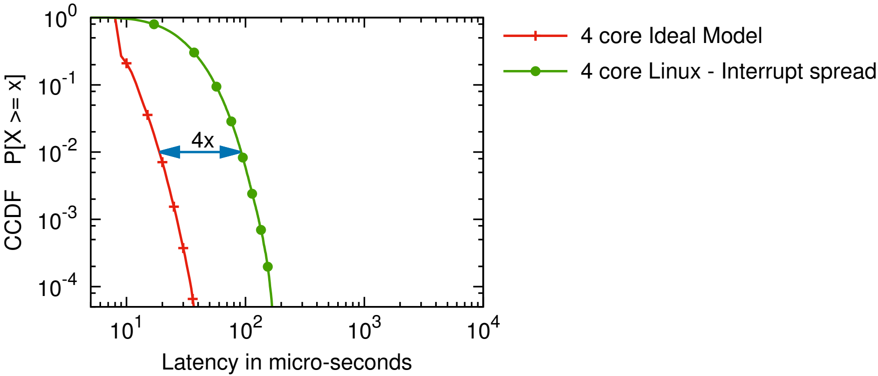
How can interrupts affect tail latency?

- 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?

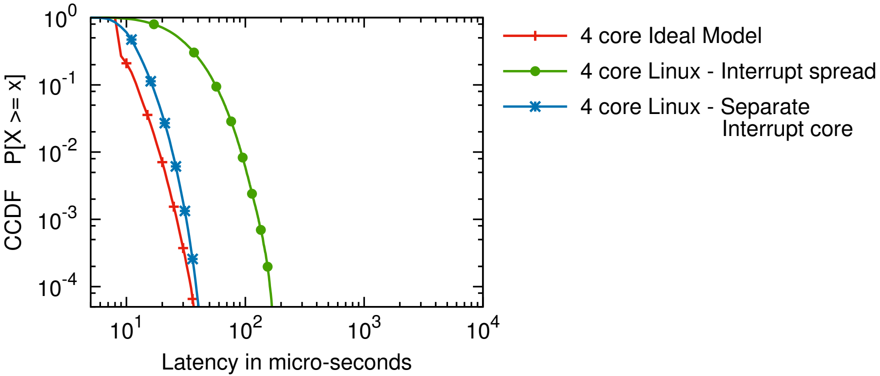
- Separate cores for interrupt processing and application threads.
- 3 cores run Memcached threads, and 1 core processes interrupts.

Impact of Interrupt Processing



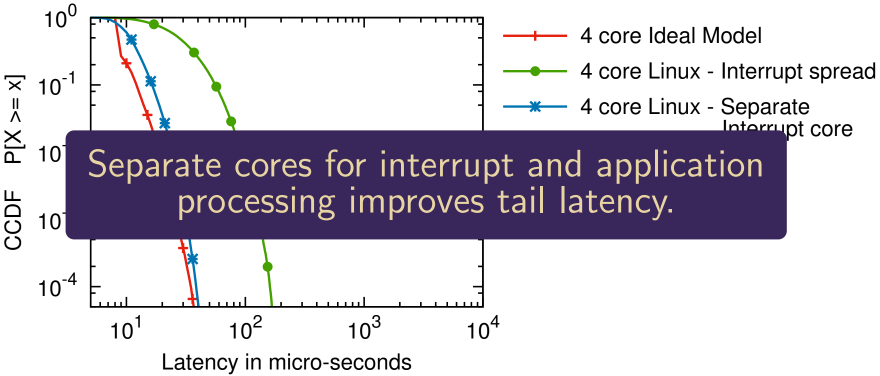
Single CPU, 4 cores, Memcached running at 80% utilization.

Impact of Interrupt Processing



Single CPU, 4 cores, Memcached running at 80% utilization.

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

Source of Tail Latency	Underlying Cause
Thread Scheduling Policy	Non-FIFO ordering of requests.
NUMA Effects	Increased latency across NUMA nodes.
Hyper-threading	Contending hyper-threads can increase latency.
Power Saving Features	Extra time required to wake CPU from idle state.

Summary and Future Works

- We explored hardware, OS and application-level sources of tail latency.
- Pin-point sources using finegrained timestaming, 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 timestaming, 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.