RamCloud Latency: Early Analysis

Henry Qin
Advisor: John Ousterhout

November 15, 2014

This talk is about *latency*, not *throughput*.

Measurement, not optimization.

The results here are preliminary, not complete.

Ultimate Goal: Where are the costs?

Where is that time going?

How much can it be improved?

What are the fundamental limits?

Ultimate Goal: Where are the costs?

Where is that time going?

How much can it be improved?

What are the fundamental limits?

We will not answer these questions today.

Ultimate Goal: Where are the costs?

Where is that time going?
How much can it be improved?
What are the fundamental limits?

We will not answer these questions today.
But today's talk used these questions as a guide.

Contributions

Thread handoff latency is primarily from cache misses.

We have 1.5 μs of overhead and 460 ns of RPC servicing work.

A measurement system which can be selectively disabled at compile time.

The One Number

The most touted number about RamCloud read latency is 5 us.

The One Number

The most touted number about RamCloud read latency is 5 us.

It is the *median* of measurements from reading the same 100-byte object with a 30-byte key over Infiniband, 100000 times.

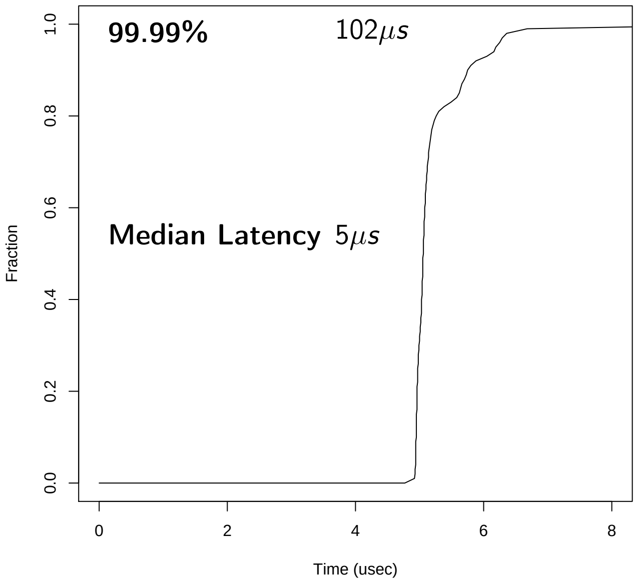
The One Number

The most touted number about RamCloud read latency is 5 us.

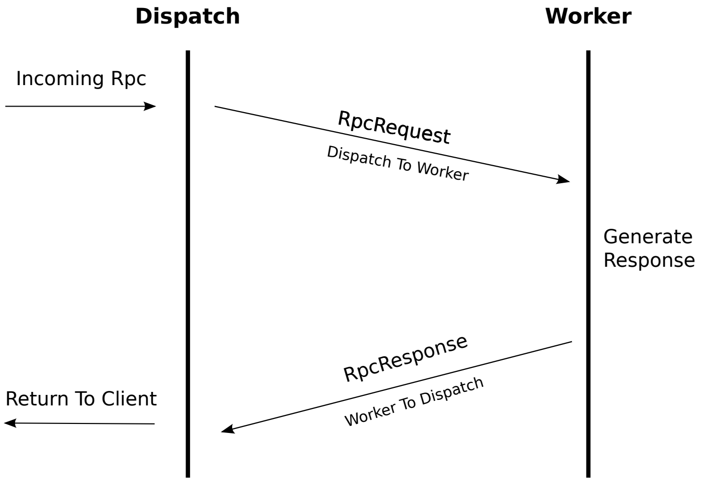
It is the *median* of measurements from reading the same 100-byte object with a 30-byte key over Infiniband, 100000 times.

What do the other 99999 measurements look like?

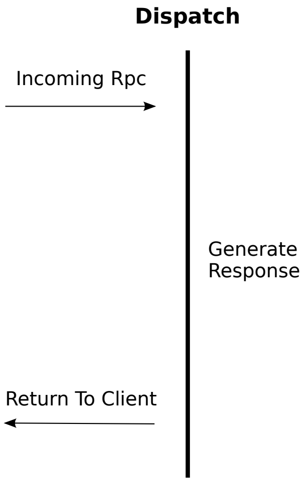
End To End Read of 100 Byte Object

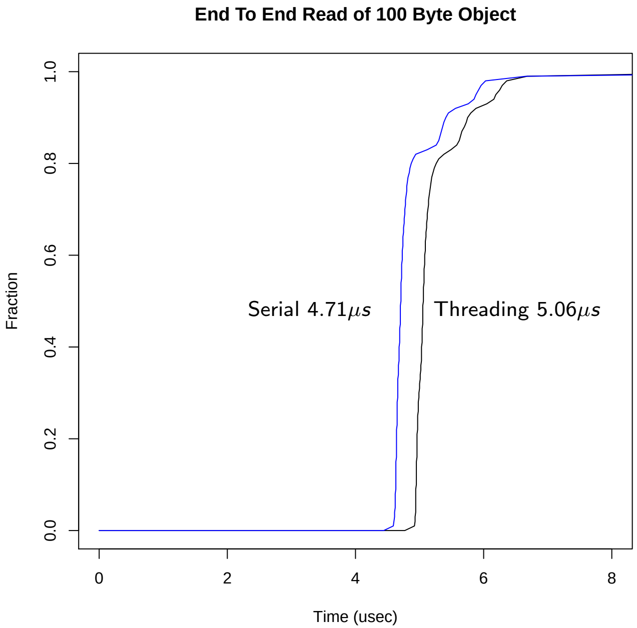


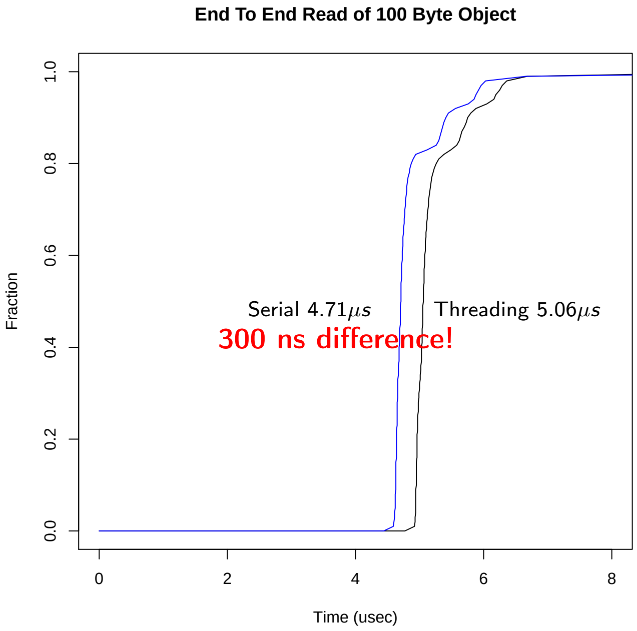
Current Threading Model



Experimental Design




End To End Read of 100 Byte Object
Fraction
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
Time (usec)
Serial 4.71μs
Threading 5.06μs


End To End Read of 100 Byte Object
Fraction
0.0
0.2
0.4
0.6
0.8
1.0
Time (usec)
0
2
4
6
8
Serial 4.71μs
Threading 5.06μs
300 ns difference!

(L2) Cache Misses

(L2) Cache Misses

Serial Execution: 3 misses

Multithreaded: 25 misses

Cost of L2 Cache Miss: 13 ns

$(25 - 3) \times 13 = 286$ ns

Takeaway: Do not run a single instance of RamCloud that spans multiple sockets, or you will get L3 cache misses instead of L2!

(L3) Cache Misses

(L3) Cache Misses

Serial Execution: 3 misses

Multithreaded: 25 misses

Cost of L3 Cache Miss: 85 ns

$(25 - 3) \times 85 = 1870$ ns

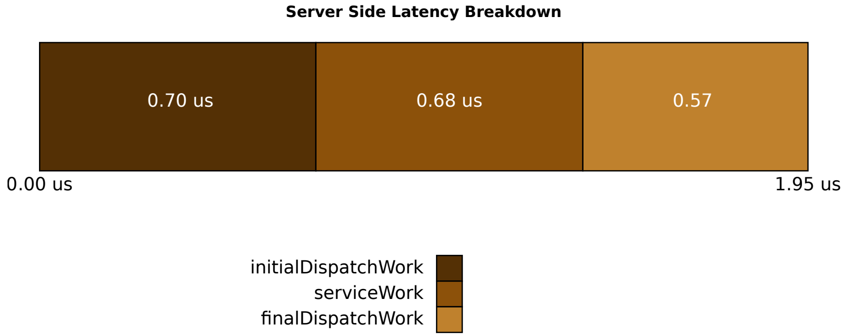
In practice, we observe a 1μs increase in latency when we ran this experiment. We suspect hardware prefetching is improving actual performance.

Thread handoff costs do not tell the whole story

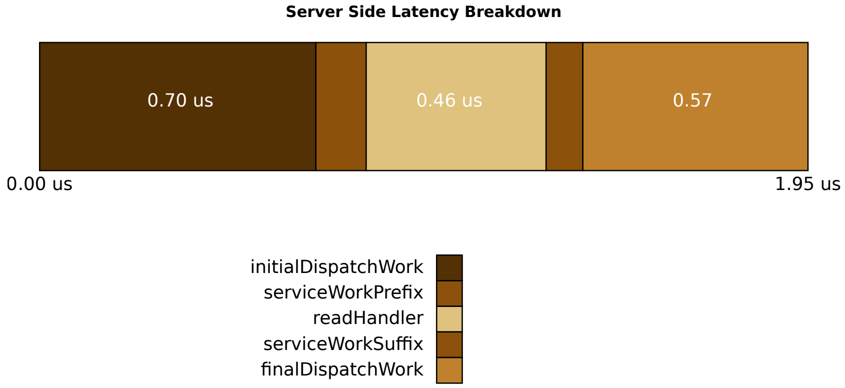
Dispatch To Worker: 81.5 ns

Worker To Dispatch: 93.4 ns

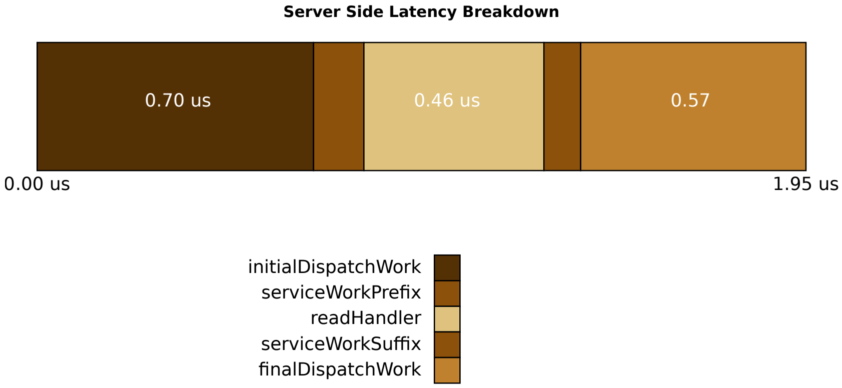
Incidental Tidbits: Overhead of an RPC



Overhead of an RPC: A Closer Look



Overhead of an RPC: A Closer Look



`readHandler` takes only 460 ns out of 1.95 μs!

We have 1.49 μs of potential overhead overhead on the server side.

Measurement System

Goals

Measure cache misses or cycles, or anything that can be used in a start-stop fashion, such as branch mispredictions.

Collect *all* measurements over a given time range, rather than just keeping a running average.

Allow different groups of measurements to be enabled and disabled together.

Allow measurement code to be permanetly left in production code base, with no performance impact when not compiled in.

Measurement System

Components

An framework for creating groups of measurements, and allowing them to be enabled and disabled at compile-time.

Linux kernel modules to select hardware performance counters and enable them to be read from userspace.

Lessons Learned and Funny Stories

Intel's performance counters are somewhat arcane to use, but they do not require black magic.

Ramcloud blows up if all global destructors are called.

The only reliable way to handle signals in a multithreaded environment is to embrace multithreading and start a new thread to do the cleanup.

Conclusion

Thread handoff latency is primarily from cache misses.

We have 1.5 μs of overhead and 460 ns of RPC servicing work.

A measurement system which can be selectively disabled at compile time.

Future Work

Optimization and tuning.

- Can we reduce the 460 ns to read an object?
- Can we work around the threading issues while retaining responsiveness?

Broadening the scope.

- Randomized reads, write RPC's.
- Latency vs throughput: Is there some fundamental tradeoff?

Thank You!

To my awesome advisor John Ousterhout for his careful guidance and eternal patience

To Ryan, Diego, Ankita, Collin, Jonathan and the rest of my lab...for all the obvious reasons

To all of you, for staying awake!