RPC Latency: Measurements and Optimizations

Henry Qin
Advisor: John Ousterhout

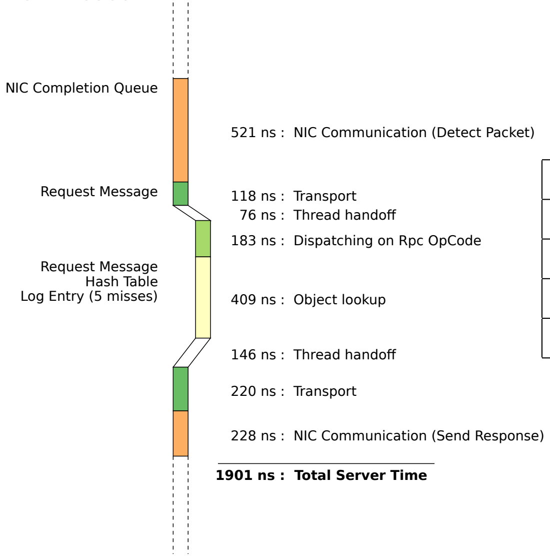
January 30, 2015

Outline

- What have we learned?
    - Detailed latency breakdown for read and write RPC.
- Optimizations and Results
    - Random reads are now 4.75 μs.
    - Random writes are now 13.4 μs.
- Future Work (Time Permitting)

Deep Dive into a Read RPC

Cache Misses



NIC	749 ns (39%)
Threading	470 ns (25%)
Cache Misses (est.)	300 ns (16%)
Other	382 ns (20%)
Total Server	1901 ns (20%)

Observations

NIC Drivers and hardware introduce considerable latency

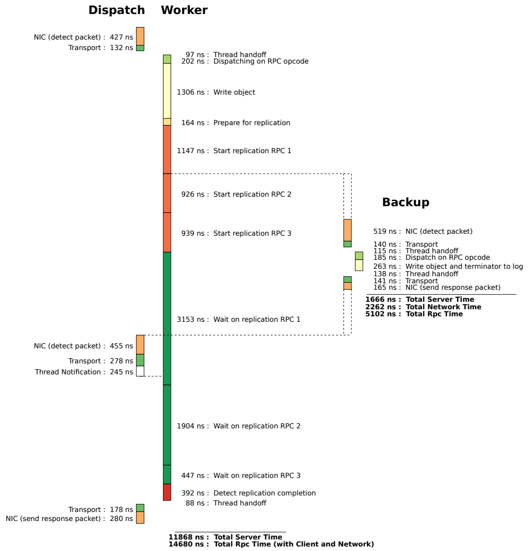
Reads are close to optimal

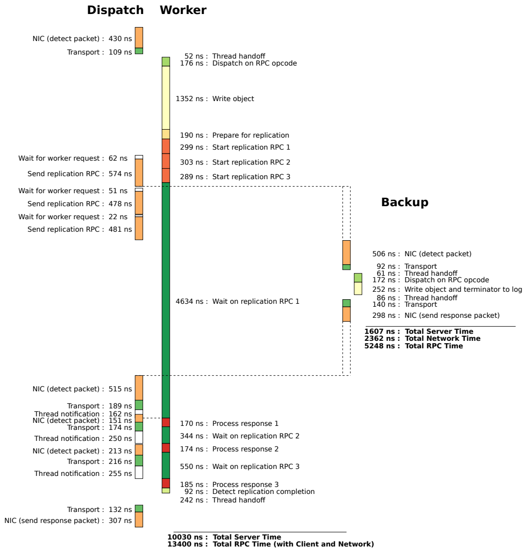
Thread handoffs and L2 misses are costly.

Read Optimization

Prefetch on the incoming packet, log entry saves **190 ns**, from **4.94 μs to 4.75 μs** for random reads.

Old vs New Write RPC


Dispatch
Worker
NIC (detect packet) : 427 ns
Transport : 132 ns
97 ns : Thread handoff
202 ns : Dispatching on RPC opcode
1306 ns : Write object
164 ns : Prepare for replication
1147 ns : Start replication RPC 1
926 ns : Start replication RPC 2
939 ns : Start replication RPC 3
Backup
519 ns : NIC (detect packet)
140 ns : Transport
115 ns : Thread handoff
185 ns : Dispatch on RPC opcode
263 ns : Write object and terminator to log
138 ns : Thread handoff
141 ns : Transport
165 ns : NIC (send response packet)
1666 ns : Total Server Time
2262 ns : Total Network Time
5102 ns : Total Rpc Time
3153 ns : Wait on replication RPC 1
NIC (detect packet) : 455 ns
Transport : 278 ns
Thread Notification : 245 ns
1904 ns : Wait on replication RPC 2
447 ns : Wait on replication RPC 3
392 ns : Detect replication completion
88 ns : Thread handoff
Transport : 178 ns
NIC (send response packet) : 280 ns
11868 ns : Total Server Time
14680 ns : Total Rpc Time (with Client and Network)


Dispatch
Worker
NIC (detect packet) : 430 ns
Transport : 109 ns
52 ns : Thread handoff
176 ns : Dispatch on RPC opcode
1352 ns : Write object
190 ns : Prepare for replication
299 ns : Start replication RPC 1
Wait for worker request : 62 ns
303 ns : Start replication RPC 2
Send replication RPC : 574 ns
289 ns : Start replication RPC 3
Wait for worker request : 51 ns
Send replication RPC : 478 ns
Wait for worker request : 22 ns
Send replication RPC : 481 ns
Backup
506 ns : NIC (detect packet)
92 ns : Transport
61 ns : Thread handoff
172 ns : Dispatch on RPC opcode
252 ns : Write object and terminator to log
86 ns : Thread handoff
140 ns : Transport
298 ns : NIC (send response packet)
1607 ns : Total Server Time
2362 ns : Total Network Time
5248 ns : Total RPC Time
4634 ns : Wait on replication RPC 1
NIC (detect packet) : 515 ns
Transport : 189 ns
Thread notification : 162 ns
NIC (detect packet) : 151 ns
Transport : 174 ns
Thread notification : 250 ns
NIC (detect packet) : 213 ns
Transport : 216 ns
Thread notification : 255 ns
170 ns : Process response 1
344 ns : Wait on replication RPC 2
174 ns : Process response 2
550 ns : Wait on replication RPC 3
185 ns : Process response 3
92 ns : Detect replication completion
242 ns : Thread handoff
Transport : 132 ns
NIC (send response packet) : 307 ns
10030 ns : Total Server Time
13400 ns : Total RPC Time (with Client and Network)

Observations

Server must start three replication Rpcs and wait for them.

Before Optimization: Replication takes 9072 ns 76% of all write time.

Write Optimization

Pay additional thread handoffs to increase pipelining

Saves **1** μ**s** , to **13.4** μ**s** for random overwrites.

Replication drops from 9072 ns to 7230 ns 72% of all write time.

NB: Difference in above numbers includes additional optimizations.

Conclusion

Fine-grained breakdown of read and write RPC latency in RAMCloud.

Random reads are now 4.75 μs.

Random writes are now 13.4 μs.

Current work: Alternate threading architectures to reduce overhead of threads more.

Questions?

Future: Birth of a new RPC System

A general purpose, independent RPC system that will benefit from the lessons we learned in RAMCloud.

Usable by RAMCloud but also by any other system that would benefit from fast RPC.

Design Questions

- What is the correct networking protocol?
- Which thread should handle the sending of sub-Rpcs?
- Which thread should receive the responses?
- How do we handle packets for partially received requests?