

27th Jan 2021

# RDMA vs. RPC for Implementing Distributed Data Structures

<https://arxiv.org/pdf/1910.02158.pdf>

Shubhendra Pal Singhal

# Problem Addressed

If we make a remote call to node A memory's and it has the hardware to support DMA, (RDMA)

vs

We give instructions understandable by that node A to then execute locally and *instantly(as soon as request arrives)* and deliver to the node who made the instructions(RPC)

## **what's better?**

Data Structure used for RDMA is hashtable and queue and RPC has AM and progress thread for attentiveness features on which experiments occur.

# RDMA - Distributed hash-table

1. Explain the CAS, insert
2. Time is  $t(\text{CAS}) + t(\text{Write})$

```
void insert(key, val) {
    slot = hash(key);
    while (!success) {
        rval = CAS(flags + slot,
                    free_flag, taken_flag);
        success = (rval == free_flag);
        if (!success) {
            slot++;
        }
    }
    if (success) {
        RPUT(data + slot, {key, val});
    }
}
```

Fig. 1. Modifying a hash table using one-sided RDMA operations.

3. However, in the case of **hash table collision**, the algorithm will move on to the next available slot, and multiple round trips may be required to perform the insert operation
4. Our **insert operation** to be atomic with respect to concurrent find operations, we will require a second fetch-and-op operation to mark the slot as ready for reading after the remote put operation has finalized. This would increase the best case cost of the remote insert operation to  $t(\text{CAS}) + W + t(\text{FAO})$

# RPC assumptions

1. AM - Active Message

(<http://people.cs.uchicago.edu/~ftchong/290N-W12/isca92.pdf>)

- a. Computation and communication happens concurrent. Like Matrix Multiplication. No buffering for active messages.

Used GASNet as the RPC mechanism <https://gasnet.lbl.gov/dist/docs/gasnet.pdf>

Applications Used : BCL Microbenchmark, GASNet.

# Comparison Standards - Base functions of every..

Communication : averaged over 4 times to equalise network latency.

Follow the operations :

- a. Compare and swap [Usual result]
  - i. The Single CAS experiment
  - ii. Persistent CAS experiment that repeatedly polls until it succeeds in modifying the target value.
- b. Fetch and add [Unusual result]
  - i. Single FAD where it deals with one value per process
  - ii. FAD deals with group of values per process.

RESULT : RDMA :

Single FAD loses to FAD as despite of multiple calls in FAD, target memory locations (and their access) led to loose in performance.

# (Contd.) Performance Analysis

Similar to above operations, for **queue**,

push (write and read n write lock), checksum(to verify whether write is finished or not), AM message queue push

THEORY →

For **hash-table**

1. RDMA- find(RW, R), insert(RW, W).
2. RPC- RTT of Active Messages(AM) - find, insert.

Method	Concurrency Level	Description	Cost
<b>insert</b>			
(a)	Concurrent Read/Write ( $C_{RW}$ )	Fully Atomic Insert	$A_{CAS} + W + A_{FAO}$
(b)	Concurrent Write ( $C_W$ )	Phasal Insertions	$A_{CAS} + W$
<b>find</b>			
(c)	Concurrent Read/Write ( $C_{RW}$ )	Fully Atomic Find	$A_{FAO} + R + A_{FAO}$
(d)	Concurrent Read ( $C_R$ )	Phasal Finds	$R$

TABLE II  
RDMA-BASED HASH TABLE METHOD IMPLEMENTATIONS CONSIDERED IN THIS PAPER.

Method	Concurrency Level	Description	Cost
<b>push</b>			
(a)	Concurrent Read/Write ( $C_{RW}$ )	Fully Atomic	$A_{FAO} + W + A_{CAS-P}$
(b)	Concurrent Write ( $C_R$ )	Only Pushes	$A_{FAO} + W$
(c)	Concurrent Local ( $C_\ell$ )	Local Push	$\ell$
<b>pop</b>			
(d)	Concurrent Read/Write ( $C_{RW}$ )	Fully Atomic	$A_{FAO} + R A_{CAS-P}$
(e)	Concurrent Read ( $C_R$ )	Only Pops	$A_{FAO} + R$
(f)	Concurrent Local ( $C_\ell$ )	Local Pop	$\ell$

TABLE III  
IMPLEMENTATIONS FOR CIRCULAR QUEUE METHODS.

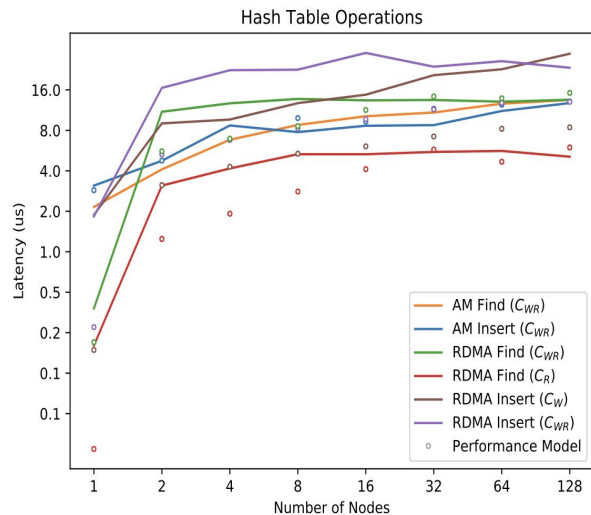


Fig. 5. Latencies for RDMA- and RPC-based hash table operations.

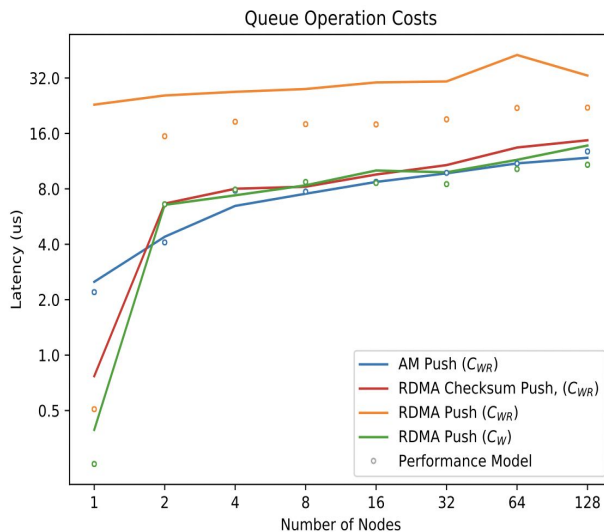


Fig. 4. Latencies for RDMA- and RPC-based queue push operations.

Check who wins?  
RDMA Find(R)...

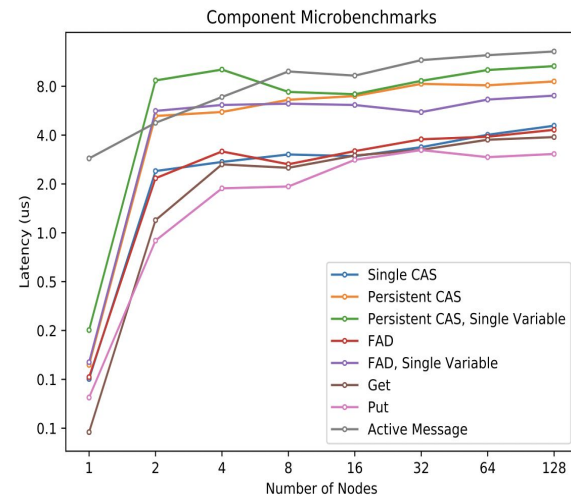


Fig. 3. The component latencies for RDMA operations and AMs on Cori.

Attentiveness Benchmarks

## More to go...

Try out GASNet-1 documentation to see what's new and GASNet-Ex?

Study more benchmarks and find out what has been missed on which might have any interesting side. *RISK!!*

Purpose : Distributed Data Structure, so change the purpose and see(explore)?

Include more system comparisons with detailed difference between each and then compare!! (May be some RPC take more time than efficient RDMA, or best RDMA, best RPC category award!!!) - Table Formulation.



# RPC over RDMA protocol

Just read basics, which is a result of faster RDMA get and put in buffers/memory locations as we could see.

So operations if can send in bulk and then converted into RDMA encapsulation such that its directly on memory location but compoundedly(as sending many as one RPC operation and then do RDMA ops )!

# More Study on RPCs

1. Performance Analysis of RPC systems within themselves with formal verification. (SunRPC, gRPC)
2. RDMA and RPC for distributed data structures.
3. <https://arxiv.org/pdf/2107.01381.pdf> Talks about all comparisons of RDMA, RPC over 2010-20.
4. Performance Perspective - “Designing a Micro-Benchmark Suite to Evaluate gRPC for TensorFlow: Early Experiences”
5. Remote Procedure Call Approach for Extreme-scale Services(<https://arxiv.org/pdf/1510.02135.pdf>)

*MORE TO GO : Better RPC line, High Performance aspect for comparison, bulk benchmark!*