Patronus: High-Performance and Protective Remote Memory

21st USENIX Conference on File and Storage Technologies

Bin Yan; Youyou Lu; Qing Wang; Minhui Xie; and Jiwu Shu

Paper Notes

By JeongHa Lee

Abstract

RDMA-enabled remote memory (RM) systems are gaining popularity with improved memory utilization and elasticity. However, since it is commonly believed that fine-grained RDMA permission management is impractical, existing RM systems forgo memory protection, an indispensable property in a real-world deployment. In this paper, we propose PATRONUS, an RM system that can simultaneously offer protection and high performance. PATRONUS introduces a fast permission management mechanism by exploiting advanced RDMA hardware features with a set of elaborate software techniques. Moreover, to retain the high performance under exception scenarios (e.g., client failures, illegal access), PATRONUS attaches microsecond-scaled leases to permission and reserves spare RDMA resources for fast recovery. We evaluate PATRONUS over two one-sided data structures and two function-as-a-service (FaaS) applications. The experiment shows that the protection only brings 2.4 % to 27.7 % overhead among all the workloads and our system performs at most $\times$5.2 than the best competitor.

Problem Statement and Research Objectives

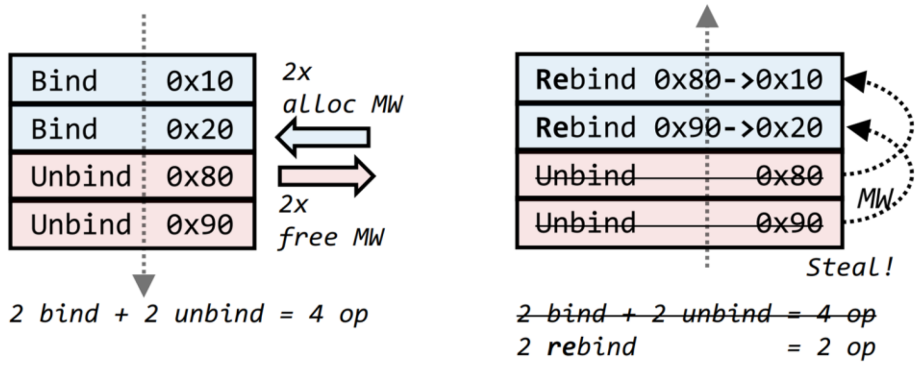
- Remote memory (RM) architecture, which decouples CPU and memory into two independent resource pools (i.e., compute nodes and memory nodes)
- However, there is still an obstacle to cross on the way to practical RM systems: **remote memory protection**.
  - First, buggy or malicious code in clients[1] can **generate illegal one-sided access to the RM**, introducing data corruption or privacy breaches.
  - Second, even if the clients are well-behaved, **concurrent memory reallocations** can turn the in-flight one-sided access illegal.
- It is non-trivial to **simultaneously achieve protection and high performance** in RM systems.
  - First, considering the high throughput of RDMA networks (e.g., ~70Mops/s in 100Gbps ConnectX-5 RDMA NIC), **clients will frequently acquire/revoke permission upon memory allocation/deallocation**. But the common RDMA protection mechanism, **suffers high latency** due to the overhead from OS kernel and RNIC (~1 ms for 256 MB).
  - Second, **on the exception path of RM systems**, i.e., clients fail or access illegal RM addresses, **retaining high performance** with a protection guarantee is **challenging**.

[1] Clients are processes in compute nodes accessing RM.

Proposed Method

Goals for the protective RM system

1. Manage protection fast.
   ➔ Save MW operations without sacrificing protection semantics.
   - For example, when permission requests arrive in a batch, pairs of opposite operations can be **leveraged to reduce the number of MW operations by half** (called **MW handover**).
2. React fast to client failure.
   ➔ Borrow the idea of leases.
   - MW only offers spacewise protection. Lease semantics (i.e., **expire on timeout**) are introduced to MW from the software to enable timewise protection.
   - This allows the system to resume progress by expiring any exclusive permission on timeout, no matter whether the permission is held by a crashed or a slow client.



(a) Before (b) w/ MW Handovers

Figure 4: The *MW handover* technique saves half of MW operations by stealing (reusing) MWs from the unbinding requests to the binding requests. `0xdd` denotes the *addr*.

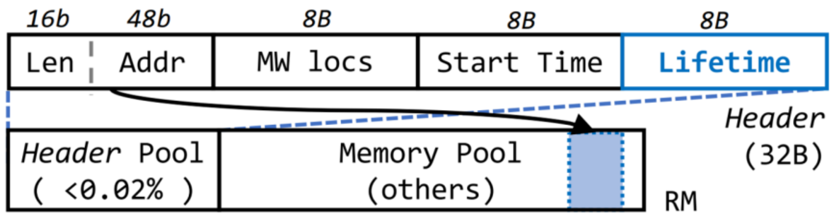


Figure 3: The format of the 32 B *header*. `MW locs` denotes the locations of the MWs. Blue indicates the area exposed to the client, i.e., the *Lifetime* variable in the header and the buffer in the RM.

Proposed Method

Goals for the protective RM system

3. Retain performance under illegal access.
   ➔ Conceal the interruption rather than prevent it.
   - **Spare QP(queue pair)**: Each client is assigned a virtual QP number, which initially maps to a physical QP.
     ➔ **On QP failure, one of the spare QPs is transparently promoted** by altering the virtual-to-physical mapping (❹)

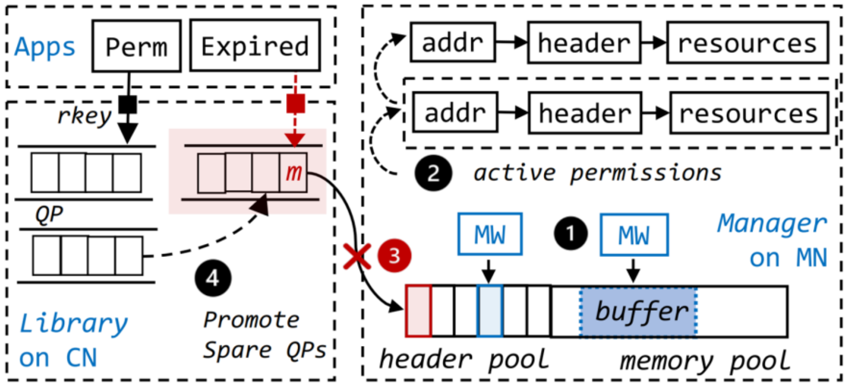


Figure 2: The architecture of PATRONUS. We assume that CNs own a mass of CPU cores (10s - 100s), while MNs use several weak cores to operate the manager.

Category	API	Parameters	Return	Description
Control Path	`allocate`	*size, time, ex/shr*	*Perm*	Allocate memory and acquire permission
	`acquire`	*addr, size, time, r/w, ex/shr*	*Perm*	Acquire permission over ⟨*addr, size*⟩
	`extend`	*Perm, time*	*Success*	Extend the permission lease
	`revoke`	*Perm*	*Success*	Revoke the permission
Data Path	`read/write/ CAS/FAA`	*Perm, addr, size, buffer*	*Success*	Issue remote access (support batching)

Table 2: The PATRONUS APIs. In the parameters, *r/w* specifies read/write permission; *ex/shr* specifies exclusive/shared access mode; *time* specifies the expected lifetime of the permission lease. *Perm* is an opaque object containing the remote address, the `rkey` as the permission token, and the expiration time. *Success* denotes whether the call succeeded.

Evaluation and Results

Overall Performance

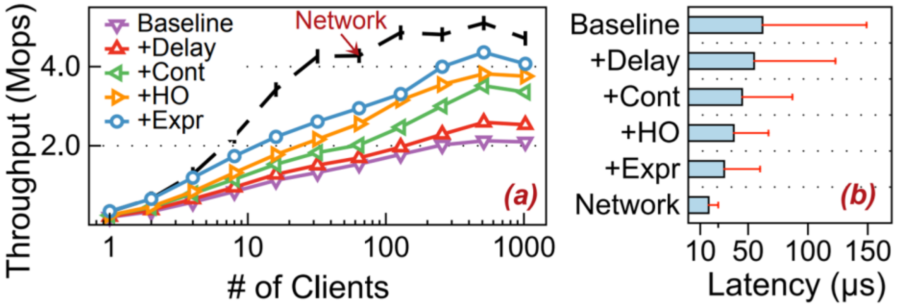


Figure 6: Throughput *(a)* and latency *(b)* of permission acquisition under different optimizations.

Goal 2. Expire on timeout

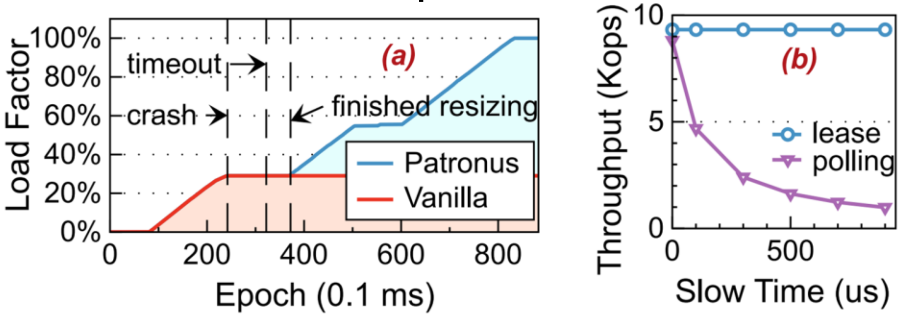


Figure 8: *(a)* The load factor of RACE hashing under client crash for vanilla implementation and PATRONUS. *(b)* The comparison of polling and leases with clients of different fail-slow degrees.

Goal 1. MW handover

Name	**(Abbr)**	**# of MW**	**# of RPC**
Baseline		2 + 2	2
Delay Unbind	(+ Delay)	2 + **1**	2
Use Contiguity	(+ Cont)	**1** + 1	2
MW Handover	(+ HO)	1 + **0** †	2
Lease Expire	(+ Expr)	1 + 0 †	**1**

Table 4: A summary of techniques for reducing the permission management overhead (§5.4). The **# of MW** column reports *binding* + *unbinding* operations. † means at probability.

Goal 3. Conceal the interruption (spare QP)

Name	**w/o**	**w/**
Failure Reported	769 µs	
Promote QP	-	78 µs
Notify QP Failure	8 µs	-
Recover QP	1004 µs	-
Summary	1012 µs	78 µs (8 %)

Table 5: Latency breakdown of handling QP faults with (w/) or without (w/o) the spare QPs technique.

Notes

- RDMA is a high bandwidth (e.g., 200 Gbps) and low latency (~2 μs) networking technology widely adopted in today's data centers.
  - The one-sided verbs: allows direct access to the remote memory while bypassing remote CPUs.
  - The two-sided verbs offer a message-passing interface.
  - basic mechanisms for regulating RDMA verbs
    - queue pair(QP): is the communication endpoint. It offers channel-wise restrictions on the access type (i.e., readable or writable)
    - memory region(MR): represents a memory area registered to the RNIC for remote access
- RM(remote memory) is getting prevalent in the decade because it addresses the problem of memory usage imbalance in traditional data centers.
  - With RM, the CPU and memory are assembled into two separate components.
    - The compute nodes (CN) gather a mass of CPU cores (10s - 100s)
    - The memory nodes (MN) typically have weak and limited computing power.