NIL-EXTERNALIZED RAFT PROTOCOL

Rahul Chunduru, Sweksha Sinha, FNU Tabish
University of Wisconsin-Madison
{ssinha, chrahul5, tabish}@cs.wisc.edu

Overview

This project report provides an overview of implementing the Nil-Externalized (Nil-Ext) interface on the RAFT consensus protocol. The report includes benchmarking RAFT performance with Nil-Ext, evaluating correctness, and assessing the impact of deferred execution. It validates results from the Nil-Ext paper [1] and determines which workloads benefit from the Nil-Ext interface. The report also discusses the client-driven approach, introduces a persisted durability log and multi-put consensus, and implements leader recovery using topological sorting of durability logs.

1 Evaluation

1.1 Nil-Ext writes + reads workload (Mysterious graph solved!!)

We assessed the performance of Nil-Ext Raft and Native Raft on a workload comprising a combination of Nil-Ext writes and reads. Throughput and **median latency** for all operations were plotted, considering different percentages of writes. Our observations indicate that, within our setup, writes exhibited latency ranging from $100 - 200$ ms, while reads had latency in the range of $10 - 20$ ms. Consequently, when calculating the median latency across operations with varying write percentages, there is an abrupt shift from the read latency to the write latency. **This elucidates the previously perplexing pattern observed in our graph during the demo.**

To further illustrate this, we separately plotted the median and mean latency for a read-write workload.

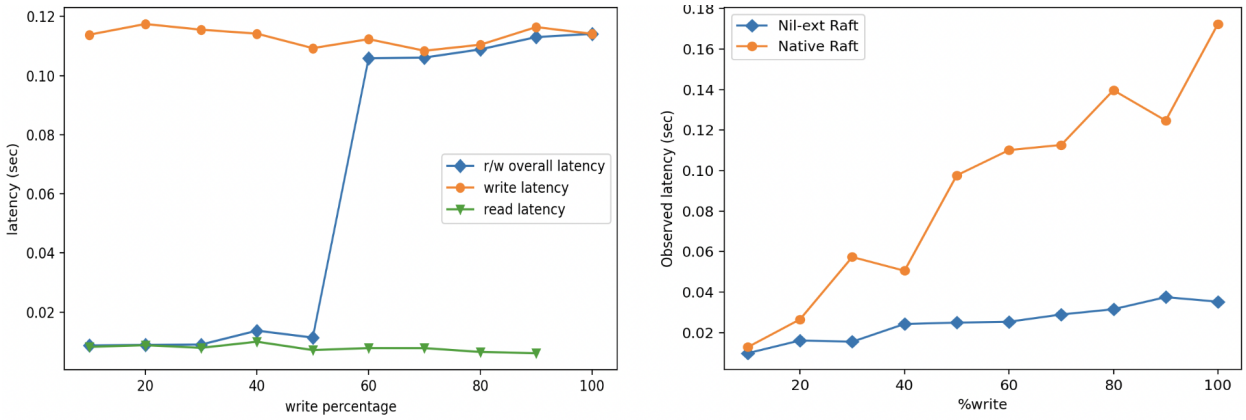

Figure 1: On left, we plot read, write and overall op latency. On right, we plot mean latency with write % mix

Hope this answers the questions raised during our presentation.

1.2 Hardware, Software Setup

Our hardware setup for performance testing consisted of 3 and 5-node clusters hosted on **Google Cloud** with the following configurations (each): e2-medium machines with 4-vCPUs, 16GB memory, 20 GB SSD disk, OS: Ubuntu 20.04 LTS

1.2.1 Performance Overview

Table 1 presents a summary of the performance evaluation conducted for the implementation of the Nil-Externalized (Nil-Ext) interface on the RAFT consensus protocol. For a single client and 3-server setting, the table demonstrates a

significant improvement in performance when using Nil-Externalized (Nil-Ext) updates. Latency shows a remarkable 4x improvement, while throughput experiences a substantial **3x enhancement**. Conversely, there is no noticeable difference observed for Get operations.

	Latency		Throughput (ops)	
	Native	**Nil-Ext**	**Native**	**Nil-Ext**
Put	0.171s	0.04s	6.97	20.91
Get	0.02s	0.03s	66.3	63.4

Table 1: Latency and Throughput Comparison

1.2.2 PUT Latency

The measurement of PUT requests with varying numbers of concurrent requesting clients is depicted in Figures [2]. Nil-Ext PUT latency increases at a significantly slower rate as compared to non-Nil-Ext PUTs with the difference in median and maximum latencies touching 4x.

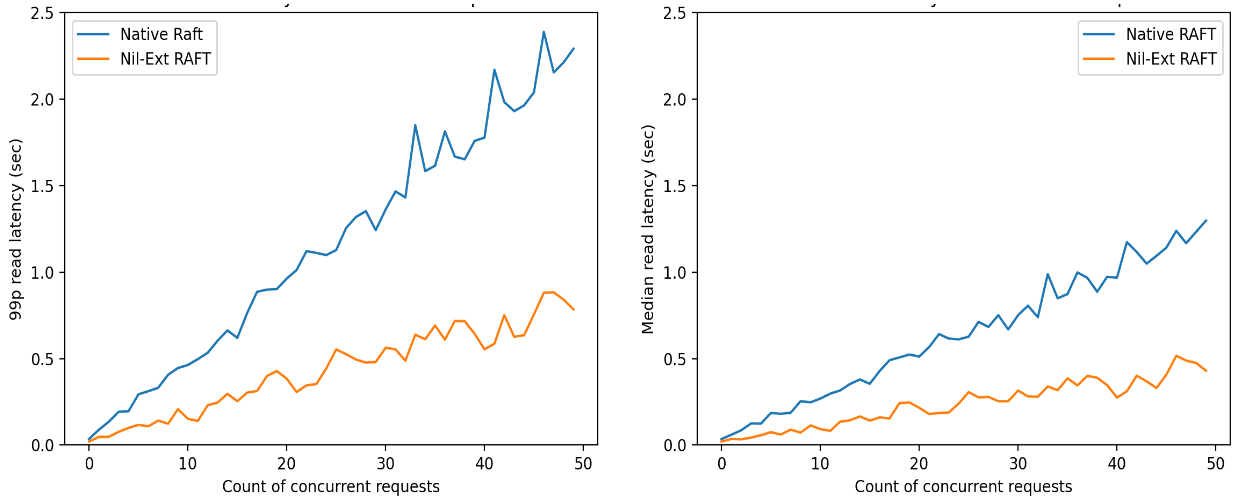

Figure 2: GET and PUT throughout vs Number of concurrent requests

1.2.3 Windowed Get/Put performance

Figure [3] demonstrates the impact of Nil-Ext writes on a workload of 1000 read and 1000 write operations, considering varying percentages of recent-read operations. The leftmost point represents no interleaving of reads and writes (all writes occurring before any reads), resulting in low average latency for the 1000 reads. Conversely, the rightmost end indicates complete interleaving, leading to increased average read latency as each read requires the execution of at least one durability log entry.

1.2.4 Availability

Figure [4] depicts the write availability of native RAFT and Nil-Ext RAFT. With Nil-Ext, writes require a super majority to be successful. For our experiments, we tested write availability on a deployment of 5 servers. In this cluster, the RAFT majority required for writes is 3 servers and the super-majority for Nil-Ext writes is 4 servers. Clearly, Nil-ext availability is reduced due to the super-majority requirement. Further, recovery from failures in Nil-Ext RAFT involves an exchange of durability logs and topological sorting, which has considerable overheads and extends the unavailability window.

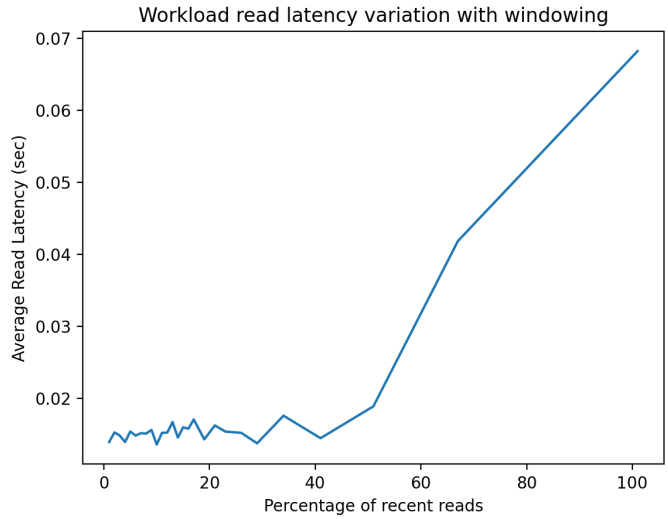

Figure 3: Read latency variation with varying percentage of 'recent-reads'

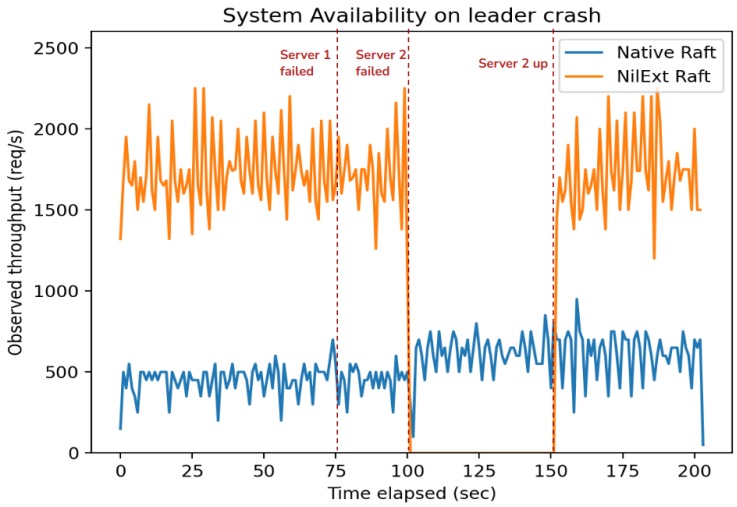

Figure 4: Write availability for native RAFT and Nil-Ext RAFT

1.2.5 Conclusion

The incorporation of the concept of nil-externality presents a valuable enhancement for optimizing write-mostly workloads or workloads that are not characterized by frequent access to the most recent values. However, it is important to acknowledge that this improvement is accompanied by a trade-off in terms of diminished availability. To mitigate this potential limitation, a viable solution lies in reverting to the conventional RAFT protocol in scenarios where a super-majority is absent.

References

1. Aishwarya Ganesan, Ramnatthan Alagappan, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. "Exploiting Nil-Externality for Fast Replicated Storage." *SOSP*, 2021.