Scalable but Wasteful: Current State of Replication in the Cloud

SEBASTIAN SCHNEIDER, 03783507

1 SUMMARY

State machines are deployed on a wide variety of systems ranging from dedicated hardware deployments in an on-premise data center to fully managed public cloud solutions. When an on-premise cluster has lots of unused hardware, new systems will be deployed in their own isolated environments. Once there is no unused hardware left, the only way to deploy more systems without expanding the datacenter is to use resource sharing. Resource sharing also becomes more relevant as server processors' core counts are increasing while most applications are not able to fully utilize all cores. The public cloud model furhter encourages resource sharing with its cost structure. Given the high usage of replicated state machines in distributed systems, improving the throughput of replication protocols has been a problem with extensive focus during the last decade.

Both of the standard implementations of replication protocols (Mulit-Paxos and Raft) follow a Leader-Follower structure. All of the newly developed protocols take a similar approach to improving performance. They first identify a bottleneck, usually at the leader, after which it is eliminated by shifting work away from the leader towards the followers. Systems like EPaxos do this, by eliminating the leader entirely and replacing in by the possibility of every follower to become an opportunistic leader. This apporoach is called leaderless consensus. It assumes no conflicts occur and only falls back to more traditional Paxos if they do occur. This also decreases latency, because requests do not have to be routed to the centralized leader first. Other protocols keep the central leader to avoid the complexity of dealing with conflicts, and instead offload as much work as possible to the followers. This often involves implementing more roles that fulfill responsibilities the leader used to have, for example letting intermediaries do some communication or processing instead of the leader. The goal of these optimizations is to use resources at the followers, that were previously sitting idle because of the bottleneck, thus achieving more throughput.

The paper runs a test, comparing Multi-Paxos to EPaxos, to confirm this. Both systems are delpoyed on five AWS EC2 m5a.large instances with 2 vCPUs and 8 GiB of RAM. The workload used is a 50% write workload targeting an in-memory key-value store. Up to 90 concurrent clients are saturating the cluster in order to observe its maximum throughput. The workload for EPaxos has a conflict ratio of up to 10%. In this test EPaxos outperforms Multi-Paxos by about 20%, which is consistent with the EPaxos paper.

All original evaluations of newly developed protocols focus on the performance gained by utilizing more resources at the followers in fully dedicated deployments. Modern systems however, are increasingly deployed on the cloud, making these evaluations not suitable for predicting performance in such resource shared environments. On top of that the previously mentioned apporach to increasing performance by shifting work away from the leader has a major disadvantage. It creates complexity requiring expensive algorithms, reducing the resource efficiency of the system.

Knowledge of the resource consuption of the protocol is key to understanding its suitability for usage in resource shared evironments. If we observe the CPU usage of EPaxos in the previous test, we notice that all replicas of EPaxos can fully utilize their CPU. Multi-Paxos on the other hand only fully utilizes its CPU on the leader node, while the followers only use about 25% of their CPU. While indicating worse load balancing, it also means that Multi-Paxos can achieve roughly 80% of the throughput of EPaxos with 40% of its CPU.

Author's address: Sebastian Schneider, 03783507, go69coq@tum.de.

To better understand resource usage compared to performance the paper introduces a new measure: througput-per-unit-of-constraining-resource-utilization. The measure can be applied to any resource such as CPU, network usage or storage IOPS. In our case the CPU is the constraining resource, so we observe it instead of other resources in order to yield the most useful data possible. One could also consider using performance per dollar as a metric. This, however is not best suited for scientific analysis with the goal of performance increase, since cost factors in all of the resources at once, hiding the effect a single resource has on performance.

We now revisit the efficiency of EPaxos and Multi-Paxos with the new metric. It is calculated by measuring the througput of the system and dividing it by the sum of the CPU usages on every node, every time instant. In our case the unit of the measurement is ops/s/CPU%. After measurement this metric shows that Multi-Paxos is about twice as efficient considering the resources consumed, compared to EPaxos being more efficient considering the resources allocated. This makes sence, since EPaxos has to handle conflicts when they arise leading to higher resource requirements, while only yielding a small performance increase. Multi-Paxos on the other hand is comparatively simple.

On a dedicated server, resources that are allocated but not used are wasted. This makes protocols like EPaxos desirable. In a cloud-native environment, resources are shared, such that resources left unused by one instance can be used by another, eliminating the need for perfect load balancing on a protocol level. Additionally, resource abundancy is heavily constrained by a budget leading to the need of extracting as much performance from a fixed pool of resources. This makes it possible for Multi-Paxos to achieve higher aggregated throughput despite having worse protocol level load balancing than EPaxos. As an example, the paper considers a five-node cluster. One can use task-packing on five Multi-Paxos instances, so that each node of the cluster hosts one leader and a follower from the other 4 instances. This lets each node of the cluster experience a similar amount of CPU load, negating Multi-Paxos' bad load-balancing characteristics.

The paper demonstrates this by emulating such an environment with an AWS EC2 cluster using five m5a.2xlarge VMs with 8 vCPUs and 32 GiB of RAM. Multi-Paxos is deployed as five instances with the before mentioned task-packing. Similar to the first test, througput of an in-memory key-value store and CPU utilization will be measured and the cluster is saturated using up to 100 clients per VM. Each Multi-Paxos instance can achieve roughly 30 kops/s of throughput, while all CPUs on all nodes are fully utilized. If the same experiment is repeated with EPaxos, each instance produces only 15 kops/s of throughput, while all CPUs on all nodes are fully utilized. The aggregated throughput of all Multi-Paxos instances is almost twice the aggregated throughput of all EPaxos instances, confirming that a more efficient protocol has a significant advantage in a shared environment with proper task-packing.

In summary: Modern consensus protocols try to achieve more throughput by eliminating a bottleneck, creating complexity and reducing efficiency in the process. This works because in dedicated deployments "unoptimized" protocols leave resources idle, which can be used if no bottleneck exists, thus increasing throughput despite being less efficient. In resource shared environments like the cloud however, these "optimized" protocols are at a disadvantage since they can no longer exploit unused resources to negate their lowered efficiency. Tests show that more efficient protocols can drastically outperform less efficient ones in such an environment with proper task packing. In order to better compare and understand protocols based on their resource efficiency the paper introduces a new measure called throughput-per-unit-of-contraining-resource-utilization. It normalizes performance over resource utilization indicating the efficiency of a protocol.

Research in replication protocols should shift to developing protocols specifically for cloud usage. One could try to use the increasing core counts of modern servers to increase performance. Another lead is to explore older ideas that

were previosly scrapped due to different requirements or missing possibilities that have now shifted with time. The main focus of development should lie on increasing resource efficiency.

2 RELATED WORK

The related work section must incorporate at least one related paper that has not been addressed in the document. This section should compare the primary idea of the paper with those of the related works. A good starting point is to explore papers that cite the given paper. The length of this section should be about half a page.

3 CRITIQUE

Critique the paper by identifying **three strong points** and **three weaknesses**. The critique section should be at least one page long.

3.1 Strong Points

- Metric design
  - stretched on 2: Focus on shifting research away from outdated technologies towards already relevant and growing topics - good highlighting of current problems - very direct criticism of current methods - good highlighting of promising directions

3.2 Weak Points

- very limited scope: only showcase of wrong approach - stays theoretical even in tests
  - repetitive
  - very few suggestions for direction of future work other than "more multicore" or "emerging technologies"

4 SUGGESTIONS FOR IMPROVEMENT

Propose ways to **enhance the paper's core idea** or outline **future extensions** that are not covered in the paper. It is also acceptable to discuss **potential use cases** for the proposed idea or system that the paper does not mention. Alternatively, consider suggesting **implementation approaches** to validate the proposed scheme and either prove or disprove its effectiveness (if you believe it may not be effective). This section should be at least half a page long.