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 (Multi-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: throughput-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 throughput 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, throughput 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

There are many protocols that have been developed since this one was published, that focus on cloud consensus. Two of them are QuePaxa[1] and RACS[2]. In this section their ideas will be presented and since they both cite this paper, their applications of this papers' proposals will be analysed.

QuePaxa aims not to increase throughput by large amounts, but to increase resilience. QuePaxa should perform comparatively to Multi-Paxos and Raft under normal network conditions. Under bad network conditions like a DoS-attack or misconfigurations, QuePaxa aims to retain much more throughput than Multi-Paxos or Raft. It achieves this by reworking the leader change logic from relying on conservative timeouts to ensure liveness, to relying on short hedging delays and by implementing a randomized asynchronous core.

In its performance tests QuePaxa neither shows nor mentions CPU usage. In fact it does not mention resource efficiency in general at all. The test setups are not mentioned in high detail. However, since the test shown in figure 7 has Epaxos outperforming Multi-Paxos by over 20% it is safe to assume that instances were deployed outside of a resource shared environment. In other words, they were deployed just the way this paper criticises.

On the other hand QuePaxa's focus is not on raw throughput, but rather on throughput versus latency. Latency being a key performance metric not addressed in this paper. QuePaxa does not take the advice from this paper to also test in resource shared environments and to measure cpu usage. It does, however, also not only focus on increasing throughput leaving efficiency behind as it measures performance relative to latency.

RACS is an even more modern consensus protocol with partially the same authors as QuePaxa. Its aim is to address the key issue of protocols like QuePaxa being hard to integrate into the existing cloud software stack, without delivering less performance. RACS is even more clear with the focus on latency versus throughput. Since both protocols have very similar performance to Multi-Paxos and Both still have a Leader, one can assume that their efficiency must be close to that of Multi-Paxos. Because of the added complexity of the protocols however, it is probably not as high.

Both papers cite this paper. They do so when evaluating the latency of EPaxos. While they do not acknowledge its advice on testing they do take the advice to design consensus protocols specifically for cloud usage. The main takeaway from this paper for these authors seems to be the latency characteristics of EPaxos.

3 CRITIQUE

3.1 Strong Points

The Metric that the paper puts forward is very well designed. Its use is not predefined neither in the unit of performance nor in the unit of resource usage. This does of course makes it resource specific and results harder to compare but as a tradeoff also more precise, since the specific environmental characteristics get factored in. It is, however, usable in every case where efficiency in the usage of a resource needs to be compared. The accuracy of the comparison is shown off by the tests the paper does. In the first test where Multi-Paxos gets outperformed by Epaxos, its efficiency metric shows that is has about double the efficiency. In the second test, Multi-Paxos achieves about double the performance of EPaxos, indicating that the metric is able to predict performance accurately.

The Paper also does an excellent job at highlighting the problem in current research. It uses large descriptions of the concepts mixed with short interludes on the details of specific protocols and their implementations. The ability

of EPaxos to use all resources allocated while decreasing efficiency and the disadvantage this brings in the cloud is explained very well. Current methods are criticised very directly and called out for ignoring resource efficiency in their work. They do this to such an extent that it reaches the border to insult, staying on the side of criticism by just a margin. The core idea of the paper is impossible to miss.

The paper uses illustration very well to explain its topics. The two figures of Multi-Paxos' bad load balancing and of task packing in a cloud scenario help understand the concepts well. The graphs never show changing values over time, instead only mostly constant values are represented. Only mentioning these numbers in text, however, would greatly impact the understandability of the paper, which is why the inclusion of these graphs and the selection of which to include, is very important and done well.

3.2 Weak Points

The paper only shows the problem with current research but does not meaningfully suggest ways to do better. The only section where they do propose topics for future research is at the end of the conclusion. Half of that section, however, is very generic advice like "Use the increasing core count of modern server processors". They themselves do not attempt to design a protocol more efficient than Multi-Paxos. This limits the scope of the paper to complaining about peers and introducing a metric with which they can do so. Even in the tests they do run, they compare two protocols on an undescriptive key-value store, missing even the chance to compare Multi-Paxos and Raft or to use real world usen cases and datasets.

On top of a limited scope, the paper repeats itself many times. The abstract introduces the topic, the long introduction says the same again, just longer, the main part features everything with detail, and the conclusion then says it all again. This can be confusing on a first read. Some repetition in the form of an abstract or a conclusion is to be expected in academic writing. The degree to which it is present in this paper however, is too much. The repetition leads to the actual content of the paper being much shorter than the paper itself.

The paper boasts two main ideas: The shift of research towards increasing efficiency and the metric to measure efficiency. The part the metric plays in achieving the other goal is not illustrated understandably. The metric is supposed to help identifying bottlenecks in replication protocols. How it does that is not stated at all since the authors themselves use it to compare relative efficiency of protocols and predict their performance in resource shared environments, not to identify bottlenecks within a protocol. Even if it is possible to use the metric for this purpose, it is not visible in plain sight, since the point of the metric is to aggregate its throughput and resource utilization values on the entire system.

4 SUGGESTIONS FOR IMPROVEMENT

In order to get a baseline for how efficient current protocols are, research could be done on a large number of protocols, measuring their relative efficiency. Since the measure is resource-specific absolute values on efficiency cannot be given. Thus one group needs to gather absolute data on the same system or setup, which makes them comparable relative to each other. Multi-Paxos would be a sensible baseline as it is widely used and already quite efficient. Raft could also be considred but it is already quite similar to Multi-Paxos in its efficiency. One could then measure the efficiency of a large number of comparable consensus protocols in both local and wide area networks. The resulting understanding of the relative efficiency of the protocols could help identify practises that lead to high efficiency. Using the same test setup with a newly developed protocol would then make you able to easily compare its efficiency to that of other protocols. In other words, now that there is a standard measure for efficiency in consensus protocols, one can also benefit from a standardised test to quickly identify efficiency and in extent performance on the cloud.

There is also the possiblity to extend the use of the measure to latency. One could normalize the throughput of QuePaxa's and RACS' throughput to latency graphs, to get a reading of how fast the system responds at a specific efficiency. Since efficiency might differ at different throughput levels and efficiency dictates performance in a resource shared environment, low throughput per instance on a shared resource could lead to lower latency in the entire system. There might also be even more elegant ways to incorporate resource efficiency into latency improvement, which should researched.