Towards Understanding the Performance of Distributed Database Management Systems in Volatile Environments

Jörg Domaschka
joerg.domaschka@uni-ulm.de
Ulm University, Ulm, Germany

Daniel Seybold
daniel.seybold@uni-ulm.de
Ulm University, Ulm, Germany

Abstract

Cloud computing provides scalability and elasticity mechanisms on resource level and has become the preferred operational model for many applications. These, in turn, are based on distributed architectures trusting that this leads to scalability and elasticity and hence, good performance.

Many applications rely on one or multiple database management systems (DBMS) as storage backends in order to manage their persistent state. Hence, the selection of a DBMS for a specific use case is crucial for performance and other non-functional properties. Yet, the choice is cumbersome due to the large number of available systems and the many impact factors ranging from the size of virtual resources, the type of the DBMS, and its architecture and scaling factor.

In this paper, we summarise our experiences with performance evaluation for cloud-hosted DBMS in order to find well-suited configurations for specific use cases. We demonstrate that the overall performance of a distributed DBMS depends on three major domains (workload, cloud environment, and DBMS) with various parameters for each dimension.

1 Introduction

Database Management Systems (DBMS) are a major building block of today's applications. They sit at the heart of any Web-business application, of many types of IoT applications, including geographically spread-out installations such as Smart Cities and Industry 4.0 sites; they are the backbone of serverless application.

Currently there are more than 200 distributed DBMS available as commercial and open source products that all claim to be reliable, scalable, and elastic while providing superior performance [7]. Yet, analyses of existing systems show that they differ a lot and suggest that the DBMSs need to be carefully selected based on the use case characteristics [3]. Only then can the DBMS deliver sufficient throughput and latency while at the same time satisfying potential non-functional requirements such as availability.

Cloud computing, virtualisation, and containerization offer appealing approaches to host DBMS in particular as they offer a natural way to quickly provision compute, storage and networking resources, and hence, realise the technical underpinning for dynamic scaling and adaptations. On the downside, relying on such resources introduces volatility in service quality as many critical aspects of the infrastructure are no longer in the hands of the DBMS operator. Further, there is an overwhelming selection of different cloud offerings with different impact on the performance of DBMS's service quality.

Thus, the decision for a cloud-hosted DBMS requires an understanding of the DBMS domain—which DBMS provides which features and what feature has which impact on performance; the cloud domain—which cloud configuration has which impact on performance; and finally the workload domain—what is the read/write/update ratio facing the DBMS and what consistency and reliability demands exist.

In this paper, we summarise the key insights gained over a series of papers [4, 6, 8] investigating the impact of the overall problem over the DBMS and the cloud domain. Due to limitations of space, we leave out the workload domain, as this is not something that can be influenced by an operator. Understanding the workload is hence a prerequisite for the selection process and benchmarking done in that process [5].

2 DBMS Impact Factors

The performance of non-distributed applications is mostly determined by the capabilities of the particular hardware (CPU, memory, I/O) running the application. For distributed applications the communication network plays a role as well as communication protocols causing e.g. queuing, drops, latency, and waiting time. In contrast to stateless applications, stateful applications may need to coordinate the state of their component instances causing traffic and load that are only mediately rooted in external workload, but are a consequence of internal processes such as garbage collection and consistency protocols.

In distributed DBMS (DDBMS), multiple instances of a DBMS provide the client with the impression of a logical DBMS. DDBMS provide two basic mechanisms to ensure scalability, availability, and reliability: sharding and replication. Hence, when designing a cloud-hosted DDBMS, the operator needs to size the individual instances (memory, #cores, type

and amount of storage), and decide on the cluster size (cf. the number of shards), the replication degree (cf. reliability and availability), and read and write consistency (cf. programmability and availability) [3, 8]. These decisions result in the following impact factors for operating DDBMS.

2.1 Sharding and Scale

The use of sharding distributes the data set of the logical DDBMS over the available DBMS instances. Consequently, sharding increases the overall capacity of a DDBMS when more instances are added. Sharding increases the overall throughput in cases where the workload is uniformly distributed over the shards.

2.2 Consistency and Replication

A consistency model defines in which order operations issued from multiple users to different data tuples may interleave. Hence, the consistency model defines which changes to a tuple are visible to a user.

Replication creates multiple copies of a single data tuple. This protects the tuple against data loss in case the DBMS instance hosting the shard containing that tuple fails. Hence, replication increases reliability of both the DDBMS and individual tuples. Depending on consistency requirements, the use of replication may either increase or decrease throughput. In case of weak consistency, operations may be targeted to different replicas of a tuple. For strong consistency more than one replica needs to be read/written.

The use of replicated tuples introduces the need to keep the tuples in sync with each other. According to the CAP theorem, availability and consistency are mutually dependant in any distributed and stateful system [1]. This leaves a continuum of possible trade-offs which has caused an increasing heterogeneity in the DDBMS landscape [7]. In consequence, many DDBMS have introduced specific configuration options to optimize for specific use cases. Often, this makes consistency and availability guarantees incomparable and evaluations even more challenging.

2.3 Resource Sizing

When operating a DDBMS, there is a trade-off between using multiple compute resources (e.g. virtual machines) or fewer larger ones. While the overall performance increases the fewer and the larger the individual resources are, the more vulnerable the system gets for failures. Also, replacing failed instances takes longer when larger portions of the overall state fail and need to be synced. Besides, these general considerations, the number of compute cores influences the parallelism in processing client requests, while the amount of memory influences the ratio of the data set that can reside in memory.

For storage, considerations are even more challenging due to different types of available cloud offerings [2]. In IaaS clouds usually three options exist: (i) use a virtual machine with large storage, (ii) attach volumes to the virtual machine as block devices, (iii) include (mount) a remote file system into the virtual machine. Users of cloud services may further build custom storage hierarchies from these basic services. For instance, they may run their own volume-based distributed file system within their virtual machines and mount this into other virtual machines.

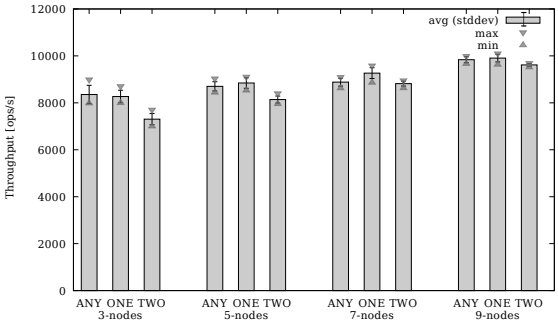

Figure 1: Impact of DBMS Domain: Cassandra [8]

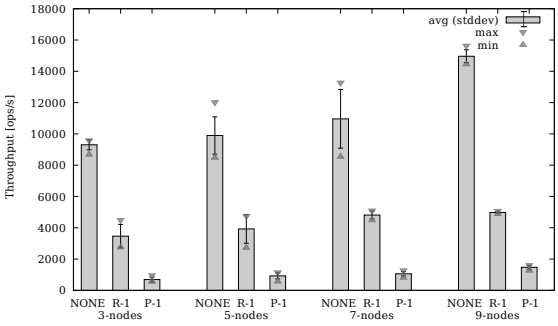

Figure 2: Impact of DBMS Domain: Couchbase [8]

2.4 Examples

Figures 1 and 2 show the average throughput over five experiments (including the standard deviation) of three different write consistency settings for Apache Cassandra (CA) and Couchbase (CB). While their consistency mechanisms can not be mapped 1:1, the applied write consistency increases from the first to the third setting. Moreover, each plot comprises the throughput of different cluster sizes, showing the scalability of the respective DBMS under a fixed workload. The results clearly show that a stronger write consistency has a significant throughput impact for while for CA the impact is neglectable. More details and results are available in [8].

3 Cloud Impact Factors

The performance of an application running on a cloud infrastructure heavily depends on many cloud provider design decisions, but also on operator design decision as discussed in the following [6, 8].

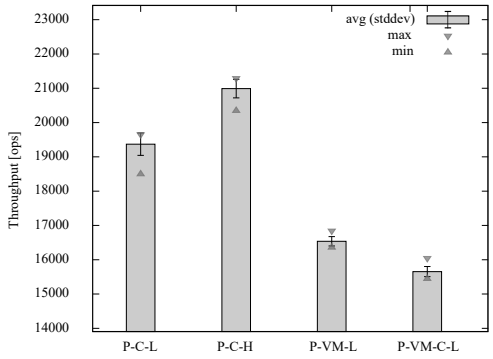

Figure 3: Impact of Cloud Resource Types [6]

3.1 Infrastructure

The maximum overall performance of a cloud-hosted application depends on the physical hardware capabilities of the cloud infrastructure including the type of physical CPUs and storage as well as networking. Further, it depends on configuration choices the cloud provider has made such as the network topology; the physical set-up of storage: SSDs vs. HDDs, hierarchical RAID systems, distributed file systems, hyperconverged infrastructure; etc.

Even though these aspects are mostly differentiating factors between cloud providers, customers can influence these decisions to some extent: for instance, they can select specific virtual machine types, regions, and availability zones that are known to be backed by a certain type of physical hardware. Yet, even if they are known to the clients, their impact on performance is hard to estimate and hence mostly unclear.

3.2 Resource Types

The resource type captures the abstraction a cloud provider offers to its customers. It defines whether the resource provided is a bare metal server, a virtual machine, a (Docker) container, or at an even higher layer. This decision has a significant impact on performance. The type of resource is often an immediate consequence of choosing a particular cloud provider.

3.3 Examples

Figure 3 shows the performance of heterogenous cloud resources based on the avg. write throughput (with stddev.) over ten experiments of a single MongoDB (MDB) instance deployed on the cloud resource types: *(P-C-L)* physical server (P) with a MDB container (C) using the local Overlay2 container filesystem (L); *P-C-H* using the host filesystem (H) of the physical server; *P-VM-L* running MDB in a VM using the VM filesystem; *P-VM-C-L* running MDB in a container on top of a VM using the local Overlay2 container filesystem. The results point out the impact of using Overlay2 with containers or VMs to operate DBMS. More results and conclusions are available in [6].

4 Conclusion

This paper summarises the design space when running distributed database management systems (DDBMS) in (volatile) cloud environments. Using examples, we illustrated that the overall performance (measured by throughput) of DDBMS are dependent on the services offered by cloud providers and the storage types used.Even minor changes in the set-up of experiments may have larger impact on performance. Besides these external factors, there are huge differences between the DDBMS themselves and the choice of a DDBMS needs to be be well considered and measured against the workload and application requirements.

Ongoing and future work extends the results shown here with the consideration of non-functional requirements. Here, we focus on the evaluation of both scalability/elasticity and availability guarantees of different DDBMS under advanced workloads. Finally, we are currently investigating the impact of overbooking and noisy neighbours on the performance of DDBMS.

References

[1] E. Brewer. “CAP twelve years later: How the ”rules” have changed”. In: *Computer* 2 (2012), pp. 23–29.

[2] S. Kächele et al. “Beyond IaaS and PaaS: An Extended Cloud Taxonomy for Computation, Storage and Networking”. In: *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Dec. 2013, pp. 75–82.

[3] J. Domaschka, C. B. Hauser, and B. Erb. “Reliability and Availability Properties of Distributed Database Systems”. In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. Sept. 2014, pp. 226–233.

[4] D. Seybold et al. “Is elasticity of scalable databases a myth?” In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 2827–2836.

[5] D. Seybold and J. Domaschka. “Is Distributed Database Evaluation Cloud-Ready?” In: *European Conference on Advances in Databases and Information Systems*. Springer. 2017, pp. 100–108.

[6] D. Seybold et al. “The Impact of the Storage Tier: A Baseline Performance Analysis of Containerized DBMS”. In: *European Conference on Parallel Processing*. Springer. 2018, pp. 93–105.

[7] S. Mazumdar et al. “A survey on data storage and placement methodologies for Cloud-Big Data ecosystem”. In: *Journal of Big Data* 6.1 (2019), p. 15.

[8] D. Seybold et al. “Mowgli: Finding your way in the DBMS jungle”. In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*. ACM. 2019, pp. 321–332.