Clustering the Cloud: A Model for (Self-)Tuning of Cloud Data Management Systems

Siba Mohammad, Eike Schallehn, and Sebastian Breß
Institute for Technical and Business Information Systems, Otto-von-Guericke-University, Magdeburg, Germany
{*smohamma,eike,bress*}*@iti.cs.uni-magdeburg.de*

Keywords: Cloud Data Management, Tradeoff, Optimization, Tuning, Self-tuning, Logical Cluster

Abstract: Popularity and complexity of cloud data management systems are increasing rapidly. Thus providing sophisticated features becomes more important. The focus of this paper is on (self-)tuning where we contribute the following: (1) we illustrate why (self-)tuning for cloud data management is necessary but yet a much more complex task than for traditional data management, and (2) propose an model to solve some of the outlined problems by clustering nodes in zones across data management layers for applications with similar requirements.

1 INTRODUCTION

Cloud-based data management and data processing solutions are significantly different from conventional database management systems (DBMS) by mainly focusing on providing scalability and availability to meet the main requirements of cloud applications while disregarding typical DBMS features such as complex query interfaces, transactional consistency management, and stringent data models.

While tuning of conventional DBMSs is a typical task of administration and its automation as self-tuning has gained attention in industrial and academic research, the (self-)tuning of cloud data management systems (CDMS) is only in its infancy. Because of the typical shared nothing architectures with data partitioning and replication, some performance aspects can be easily addressed for the overall system. Nevertheless, the typical multi-layered architecture of several component systems adds complexity to the tuning tasks. Moreover, if there are several applications with different and possibly changing requirements using the same database cluster, there is little chance to tune for a specific application. Current research efforts on (self-)tuning of CDMSs include approaches that focus on specific aspects such as MapReduce performance (Ahmad et al., 2012), energy conserving (Bostoen et al., 2012), and minmal cluster size (Herodotou et al., 2011). To the best of our knowledge, there is only the work of Florescu and Kosmann (Florescu and Kossmann, 2009) that provides an overall view on the tuning problem of databases in the cloud environment and provides an architecture designed for the new optimization problem, as they call it. The idea of creating zones within a cluster that we present as model for (self-)tuning, is a generalization of the concept of cold, and hot zones presented in greenHDFS (Kaushik and Bhandarkar, 2010).

2 TUNING GOALS AND TRADEOFFS

What makes optimization for a CDMS a complicated task is the fact that it is not a monolithic system, but rather a combination of loosely coupled systems that complement each other. Thus, optimizing a CDMS for a certain goal becomes a task that expanses across several system layers and within each one, where decisions in one layer affect possibilities and decisions in other layers.

Tuning Tradeoffs. Next, we summarize the fundamental tradeoffs for a CDMS:

- *Read performance versus write performance*: In the cloud, challenges related to partitioning and replication add to the problem (Cooper et al., 2010).
- *Latency versus durability*: Many CDMSs choose to write to memory and sync to disk later. This lowers latency but could result in data loss in the case of failure (Cooper et al., 2010).
- *Security versus performance*: Existing models supporting security of cloud data depend mainly on encryption. This affects performance of systems (Yu et al., 2010), because it leads to overhead in reading and writing data and increases the overall latency.
- *Resource utilization versus performance*: Reducing operating cost is important for private and public cloud operators. This usually results in trading performance for higher resource utilization (Chen et al., 2010).
- *CAP tradeoffs*: The CAP theorem (Fox et al., 1997) states that consistency, availability, and partition tol-

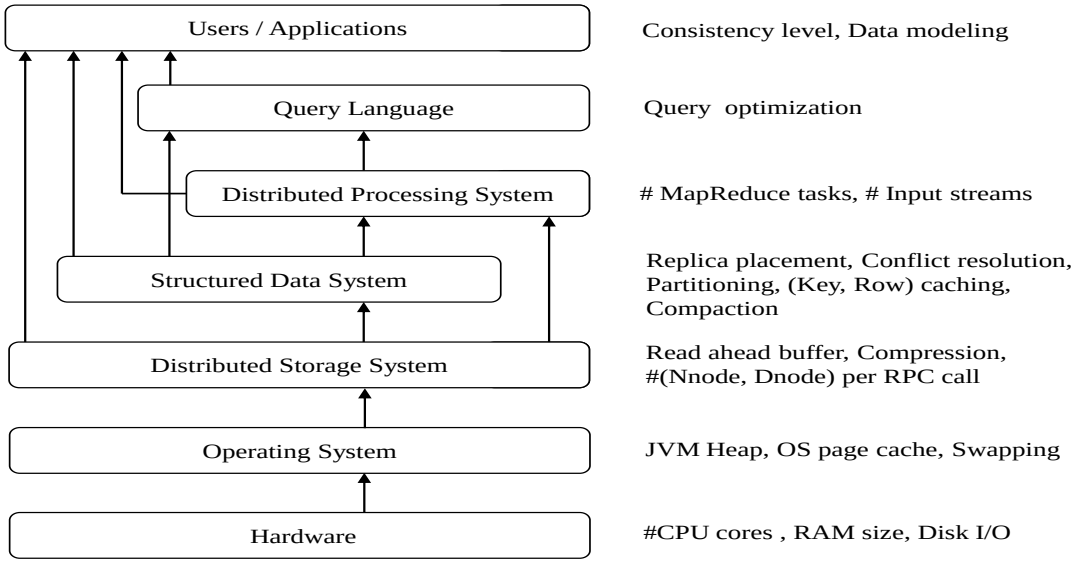

Figure 1: Architecture for Cloud Data Management System (CDMS) extended from (Mohammad et al., 2012)

erance are systematic requirements of designing and deployment of applications in a distributed environment. It also states that a scalable system can fulfill at most two of these three properties. In the context of CDMS, this means in most cases scarificing consistency. More details about this and other implications of CAP on CDMS are in (Mohammad et al., 2012)

Optimization Goals/Opportunities. Next, we summarize the fundamental optimization goals for CDMSs:

Performance: Performance is crucial for CDMSs. It is affected by several aspects such as geographical distribution of data, replication, consistency and durability requirements. Overhead caused by events, such as data compaction and scaling the DB cluster, also impacts the overall performance.

Availability and Fault tolerance: CDMSs work on clusters of nodes where failure is the normal case, not the exception. Most of these systems support fault tolerance and recovery. However, the promised 99.9% availability is not always sufficient. Leading cloud storage providers, such as Amazon, Salesforce.com, and Rackspace, had several outages in the last years causing major websites and businesses to be out of service. Outages result not only in services time outs, but also unrecoverable data losses (Dahbur et al., 2011).

Consistency level: For CDMSs, consistency is not a matter of yes or no question. There are several levels of consistency and one can tune it even on the granularity of a single query or data object. Depending on data type, strict consistency is not always a requirement (Kraska et al., 2009). Since decisions about the level of consistency affect the system performance and availability, it is important to determine the highest achievable level of consistency for specific performance requirements.

Minimum resource consumption: It is very important to minimize resource consumption, but within specified performance thresholds. The first perspective regarding this goal is *monetary costs*. The second perspective is *energy efficiency.* Energy consumption of data centers, whether it is for cooling or operating machines, is high and is estimated to increase by 18% every year (Zhang et al., 2010). Besides, up to 35% of this energy consumption is caused by the storage subsystems. Hence, minimizing energy consumption is getting more important (Bostoen et al., 2012).

3 SELF-TUNING BASED ON CLUSTERING

An example of a cloud data management system is composed of a Cassandra cluster that builds on top of Hadoop Distributed File Systems (HDFS). We take the case that this cluster should be optimized regarding its read performance. Among the factors that should be dealt with are the following (Capriolo, 2011): index, bloom filter, consistency level, replica conflict resolution, caching (row and key caching, Java Virtual Machine (JVM) heap, Operating System (OS) page cache and swapping), compaction, compression, and hardware deployment (Random Access Memory (RAM), Central Processing Unit (CPU), disk and number of nodes). In addition, depending on whether the application works with mainly range queries, it would be better to use an order preserving partitioning technique. In the case that the application will use Cassandra as input for MapReduce, other factors

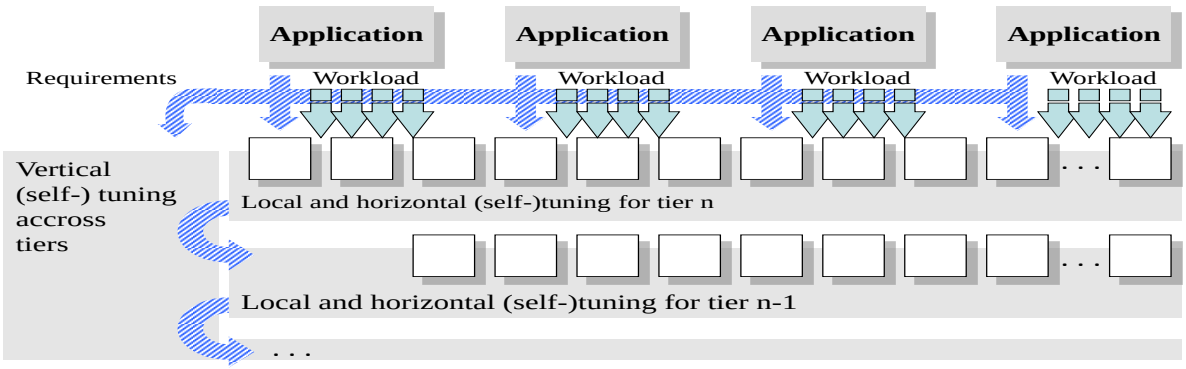

Figure 2: Illustration of complexity of (Self-)tuning multi-layerd CDMSs according to divergent requirements and different and changing workloads

such as the number of input streams, and the number of map and reduce tasks, etc., would have to be considered. Figure 1 illustrates how tuning works across several layers with parameter examples on the right side. As a step toward (self-)tuning CDMSs for divergent requirements across several layers, we propose the concept of logical clustering of shared nothing systems into zones applying the basic principal of divide and conquer. As illustrated in Figure 2, tuning becomes even more complicated when the DB cluster is serving different workload types with different optimization goals: either in the case of one application with shifting workloads or several applications with different workloads. *Horizontal (self-)tuning* includes aspects such as partitioning/load balancing, replication/update strategies, automatic scaling, etc., which are typically better supported because of homogeneous processes of a single component type within one layer. Nevertheless, there are still open research questions, e.g., for heterogeneous resources/hardware across nodes. *Vertical (self-)tuning* represents mapping application requirements expressed as optimization goals, service levels, strategies, etc., to specific tuning measures on each level of the architecture. To support this complex process of (self-)tuning, we suggest the idea of creating logical clusters within a CDMS by clustering nodes based on application requirements as illustrated in Figure 3. A logical cluster (LC) can be identified by applications requirements and assigned specific physical resources. This is a generalization of the concepts introduced by Kaushik and Bhandarkar (Kaushik and Bhandarkar, 2010). Their model depends on an energy-aware data-classification data placement strategy to define two zones, hot and cold, with the main purpose of energy saving.

Clustering based on Application Requirements. We classify applications based on these criteria: data, workload, optimization goals, and thresholds. Then, for each class, an LC is created and assigned specific physical resources. There are common workload patterns that are defined by the write/read mix. We take into consideration other aspects such as the size and freshness of data that will be accessed to define a list of workload types. Typical workload classes are on-line access and batch processing. On-line access is characterized by heavy update, read latest, read ranges (with small ranges it is equal to lookups), and random reads (lookups). Batch processing scenarios typically perform batch writes and scans. Zhang et al. (Zhang et al., 2010) discusses challenges in analyzing and defining data traffic patterns in the cloud computing environment.

Clustering based on Physical Resources. Finding the right cluster size, deciding when to scale, and minimizing the system overhead that results from adding new nodes, should be addressed. All CDM solutions support scalability to satisfy applications growth. However, some systems, such as Cassandra and Yahoo! Pnuts, show degradation in performance and need time to stabilize after adding new nodes (Cooper et al., 2010). For this reason, and for cost and energy conserving, increasing cluster size should not always be the first suggested solution for performance problems. Since the assumption of homogeneous clusters does not stand in real applications, CDM is evolving towards adopting heterogeneity. In addition, systems supporting heterogeneity allow the possibility of improving performance by adding nodes with higher capacity instead of having to upgrade, at once, all nodes within a cluster (DeCandia et al., 2007). However, only with resource-aware scheduling and load balancing performance improves. Various studies (Rasool and Down, 2012; Ahmad et al., 2012) show that the current implementations of data-intensive applications do not take into consideration heterogeneous nodes and show degradation in performance.

Alternatives for Cluster Structures and Clustering Strategies. While the outlined proposal of logical clus-

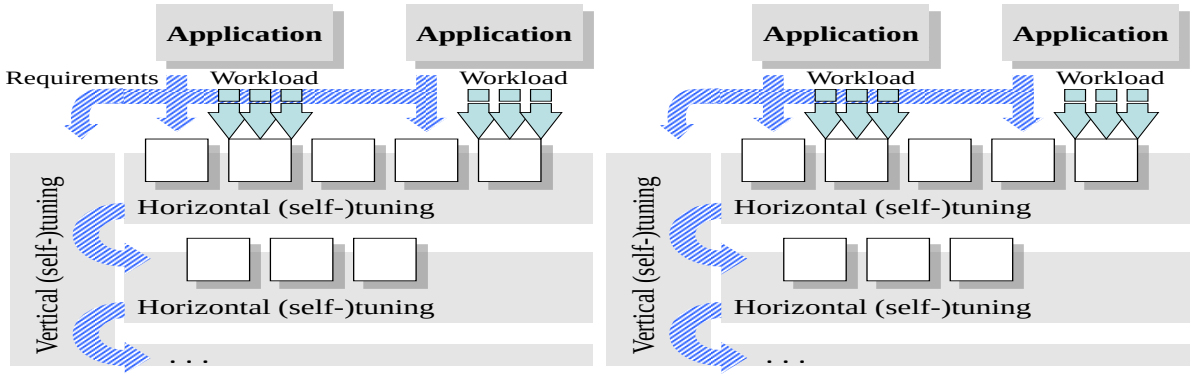

Figure 3: Clustering applications with similar requirements and workloads(divide and conquer) to ease tuning

ters is already supported in a static manner, with our research we want to focus on dynamic clustering to support workload-based optimization. This dynamic clustering requires support for splitting and merging or re-computing clusters. Furthermore, an LC may contain other LCs, so hierarchical structures are desirable for the clustering process as well as the mapping to clustering criteria. Furthermore, the possibility to build clusters only on several layers or independent clusters across layers should be considered.

4 CONCLUSION

In this paper, we outlined the tuning tradeoff decisions and optimization goals for cloud data management systems. The complexity of (self-)tuning for these systems results from their highly distributed multi-layerd architecture. (Self-)Tuning gets even more complicated when one cloud database cluster is serving one application with shifting workloads or several applications with multiple workloads. With the aim of supporting (self-)tuning in such case, we suggested a general model for creating logical clusters within a cloud DB system. To create logical clusters, we depend on clustering of applications based on data, workload, optimization goals and thresholds. Finally, we briefly discussed different problems and alternatives for this model.

REFERENCES

Ahmad, F., Chakradhar, S. T., Raghunathan, A., and Vijaykumar, T. N. (2012). Tarazu: Optimizing MapReduce On Heterogeneous Clusters. *SIGARCH*, 40(1):61–74.

Bostoen, T., Mullender, S., and Berbers, Y. (2012). Analysis of disk power management for data-center storage systems. In *e-Energy*, pages 2:1–2:10. ACM.

Capriolo, E. (2011). *Cassandra High Performance Cookbook*. Packt Publishing.

Chen, Y., Ganapathi, A. S., Griffith, R., and Katz, R. H. (2010). Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. Technical report, EECS Department, University of California, Berkeley.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking Cloud Serving Systems with YCSB. In *SoCC*, pages 143–154. ACM.

Dahbur, K., Mohammad, B., and Tarakji, A. B. (2011). A survey of risks, threats and vulnerabilities in cloud computing. In *Proceedings of ISWSA*, pages 12:1–12:6. ACM.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's Highly Available Key-value Store. *Symposium on Operating Systems Principles*, 41(6):205–220.

Florescu, D. and Kossmann, D. (2009). Rethinking cost and performance of database systems. *SIGMOD Record*, 38(1):43–48.

Fox, A., Gribble, S. D., Chawathe, Y., Brewer, E. A., and Gauthier, P. (1997). Cluster-based scalable network services. In *Proceedings of the sixteenth ACM symposium on Operating systems principles*, SOSP '97, pages 78–91. ACM.

Herodotou, H., Dong, F., and Babu, S. (2011). No one (cluster) size fits all: Automatic cluster sizing for data-intensive analytics. In *Proceedings of SOSP*, pages 18:1–18:14. ACM.

Kaushik, R. T. and Bhandarkar, M. (2010). GreenHDFS: Towards An Energy-Conserving, Storage-Efficient, Hybrid Hadoop Compute Cluster. In *HotPower*, pages 1–9. USENIX Association.

Kraska, T., Hentschel, M., Alonso, G., and Kossmann, D. (2009). Consistency Rationing in the Cloud: Pay only When It Matters. In *VLDB*, pages 253–264. VLDB Endowment.

Mohammad, S., Breß, S., and Schallehn, E. (2012). Cloud data management: a short overview and comparison of current approaches. In *GvD*, pages 41–46. CEUR-WS.

Rasool, A. and Down, D. G. (2012). An Adaptive Scheduling Algorithm for Dynamic Heterogeneous Hadoop Systems.

Yu, S., Wang, C., Ren, K., and Lou, W. (2010). Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *INFOCOM*, pages 534–542. IEEE.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: State of the Art and Research Challenges. *Internet Services and Applications*, 1(1):7–18.