2013 IEEE Ninth World Congress on Services

Is Your Cloud-Hosted Database Truly Elastic?

Sherif Sakr   Anna Liu

NICTA and University of New South Wales, Australia
{sherif.sakr, anna.liu}@nicta.com.au

***Abstract*— Elasticity has been recognized as one of the most appealing features for users of cloud services. It represents the ability to dynamically and rapidly scale up or down the allocated computing resources on demand. In practice, it is difficult to understand the elasticity requirements of a given application and workload, and to assess if the elasticity provided by a cloud service will meet these requirements. In this experience paper, we take the position that a deep understanding of the capabilities of cloud-hosted database services is a crucial requirement for cloud users in order to bring forward the vision of deploying data-intensive applications on cloud platforms. We argue that it is important that cloud users become able to paint a comprehensive picture of the relationship between the capabilities of the different type of cloud database services, the application characteristics and workloads, and the geographical distribution of the application clients and the underlying database replicas. We discuss the current elasticity capabilities of the different categories of cloud database services and identify some of the main challenges for deploying a truly elastic database tier on cloud environments. Finally, we propose a benchmarking mechanism that can evaluate the elasticity capabilities of cloud database services in different application scenarios and workloads.**

I. Introduction

Cloud computing is a model for building scalable and highly available services on top of an elastic pool of configurable computing resources such as virtual machines, disk storage and network bandwidth. Therefore, it promises a number of attractive features for their users such as better cost effectiveness, lower time to market, and the perception of (virtually) unlimited resources and infinite scalability. In practice, the advantages of the cloud computing paradigm have opened up new avenues for deploying novel applications which were not economically feasible in a traditional enterprise infrastructure setting. Therefore, the cloud services has become an increasingly popular platform for hosting software applications in a variety of domains such as e-retail, finance, news and social networking. Thus, we are witnessing a proliferation in the number of applications with a tremendous increase in the scale of the data generated as well as being consumed by such applications. Cloud-hosted database systems powering these applications form a critical component in the software stack of these applications [10].

Elasticity is one of the most appealing features and a powerful selling point for users of cloud services. Most of cloud services providers claim that their services are *elastic* in a way that the user can dynamically and rapidly scale up or down the allocated computing resources on demand and being charged according to the *pay-as-you-go* billing strategy [13]. In principle, it is common in modern web-based application to be periodically demanding (e.g. on specific day, month or time of the year) or bursty workloads that may grow very rapidly but fads are fickle and demand can then shrink back to previous normal levels [3]. In such scenarios, traditional on-premise computing infrastructures usually tend to provision their computing infrastructure for the peak situations. There, it risks wasting the over-provisioned resources after the peak time has passed which will remain idle during non-peak periods. Hence, the main advantage of elastic cloud services is that they can optimize the process of dynamic allocation of computing resources by allowing the *pay-as-you-go* billing strategy where the costs are rising or falling according the resource consumptions of the application workloads.

While most of cloud database services are claiming elasticity as a virtue that they possess, it is not clear if they can perfectly serve their customers by enabling their application to gracefully scale up and down the allocated resources according to the characteristics of their workloads without affecting the performance and the quality of services of the running applications, and pay for exactly the needed computing resources to achieve these requirements. As yet, the literature does not contain any comprehensive assessments and measurements of the elasticity capabilities of the different categories of cloud database services. This is the gap that our paper attempts to fill by attracting the attention of the research community to the related challenges of deploying truly elastic database tiers of software applications in cloud environments. We discuss how elasticity capabilities should be assessed with respect to the different categories of cloud database services and with clear consideration for the perspective of the users of the cloud services and their requirements. We envisage our proposal as a first step towards achieving a comprehensive picture in this regard.

II. Cloud Database Services

We classify the approaches for hosting the database tier of software applications in cloud platforms into two main categories, namely, the database as a service (DaaS) platforms and virtualized database servers. In the following, we present an overview of each of these approaches.

A. Database-as-a-Service (DaaS)

Multi-tenancy, a technique which is pioneered by *salesforce.com*[1], is an optimization mechanism for hosted services in which multiple customers are consolidated onto the same

[1]http://www.salesforce.com/

operational system and thus the economy of scale principles help to effectively drive down the cost of computing infrastructure. Therefore, multi-tenancy is an attractive mechanism for both of cloud providers who are able to serve more customers with a smaller set of machines, and also to customers of cloud services who do not need to pay the price of renting the full capacity of a server. Database-as-a-Service (DaaS) represents a multi-tenant cloud database platforms which alleviate the need for their users to purchase expensive hardware and software for hosting their database servers, deal with software upgrades and hire professionals for administrative and maintenance tasks. In practice, DaaS platforms can be classified into two main categories: *NoSQL storage services* and *Relational Database as a Service*.

1) NoSQL storage services: This category of cloud data storage services relies on a new wave of storage platforms named as key-value stores or NoSQL (Not Only SQL) systems. These systems support flexible data models with the ability to dynamically define new attributes or data schema. They also rely on a simple call level interface or protocol (in contrast to a SQL binding) which does not support join operations. Commercial offerings of this approach include *Amazon SimpleDB*, *Amazon DynamoDB* and *Microsoft Azure Table Storage*. In addition, there is a large number of open source projects that have been introduced which follow the same principles of NoSQL systems [10] such as *HBase* and *Cassandra*. In practice, developing applications on top of NoSQL datastores requires a higher effort when compared to traditional databases because they hinder important factors such as data independence, reliable transactions, and other cornerstone characteristics often required by applications that are fundamental to the database industry [14].

2) Relational Database as a Service: In this category of cloud database services, a third party service provider hosts a relational database as a service. Cloud offerings of this approach include *Amazon RDS*, *Microsoft SQL Azure*, *Google Cloud SQL* and *Heroku Postgres*. For example, Amazon RDS provides access to the capabilities of MySQL or Oracle database while Microsoft SQL Azure has been built on Microsoft SQL Server technologies. The migration of the database tier of any software application to a relational database service is supposed to require minimal effort if the underlying RDBMS of the existing software application is compatible with the offered service. However, some limitations or restrictions might be introduced by the service provider for different reasons[2]. Moreover, the users of these services do not have sufficient flexibility in controlling the allocated resources of their applications. In addition, the whole resource management and allocation process is controlled at the provider side and requires accurate planning, by the user, for the allocated computing resources for the database tier which limits the ability to fully leverage the elasticity and scalability features that should be provided by the cloud platforms.

[2]http://msdn.microsoft.com/en-us/library/windowsazure/ee336245.aspx

B. Virtualized Database Server

Virtualization is a key technology of the cloud computing paradigm. It allows resources to be allocated to different applications on demand and hide the complexity of resource sharing from cloud users by providing a powerful abstraction for application and resource provisioning. In particular, resource virtualization technologies add a flexible and programmable layer of software between applications and the resources used by these applications. The *virtualized database server* approach takes an existing application that has been designed to be used in a conventional data center, and then port it to virtual machines. Such migration process usually requires minimal changes in the deployed application. In this approach, database servers, like any other software components, are migrated to run in virtual machines. In principle, one of the major advantages of the *virtualized database server* approach is that the application can have full control in dynamically allocating and configuring the physical resources of the database tier (database servers) as needed. Hence, software applications can fully utilize the elasticity feature of the cloud environment to achieve their defined and customized scalability or cost reduction goals. However, achieving these goals requires the existence of an admission control component which is responsible for monitoring the system state and taking the corresponding actions (e.g. allocating more/less computing resources) according to the defined application requirements and strategies [4], [9], [15].

III. Elastic Database Services

The two notions of elasticity and scalability are usually mixed up. Therefore, we differentiate between them as follow [8]. Scalability refers to the ability of a system to meet a larger workload by adding a proportional amount of computing resources. However, scalability is a *time-free* notion that does not capture how long it takes for the system to achieve the desired performance level. In contrast, *time* is a central aspect in elasticity, which depends on the speed of response to the changed workload. Therefore, a truly elastic database system must be able to dynamically and rapidly allocate and deallocate computing resources to a running system to handle changing workloads in a graceful manner without significantly interrupting the system performance.

A. Database Elasticity Metrics

Ideally, a cloud database service should be infinitely scalable and instantaneously elastic. A perfectly elastic database service should scale up or out its resources indefinitely with increasing workload, and this should happen instantly as the workload increases with no degradation on the application performance and transaction response times [9]. However, in reality, cloud database services are not be perfectly elastic. In principle, the elasticity of cloud database services can be evaluated according to the following metrics [8]:

- *Speed-up*: refers to the time interval which is needed for the cloud database service to stabilize its performance after the transit phase of dynamically changing its allocated computing resources (e.g., adding or removing a node into a cluster). In principle, the database tier can be considered in stable state when the variations on the transaction response times are equivalent to the variations of state which is known to be stable. Therefore, in order to accurately measure the speed-up metric, it is necessary to determine the performance characteristic of the system stable state of the cloud database service under consideration.

- *Scaleup*: refers to the throughput that can be gained by adding more computing resources as a reaction to the increasing workload volume. It represents the deviation from linear scalability which can be measured and compared with the linear function of throughput depending on the capacity of the allocated computing resources (e.g., number of servers).

B. Database Elasticity Challenges

While most of cloud database services are claiming elasticity as a virtue that they possess, it is not clear, in reality, if they can perfectly serve the requirements of their users. In this section, we discuss some of the issues that need to be addressed before the vision of truly elastic cloud database service can be fully realized.

1) Quality-of-Service Management: To unleash the real power of the elasticity feature of clouds, cloud database systems should be able to transparently manage and utilize the elastic pool of resources to deal with fluctuating workloads. For example, to deal with increasing workloads, software applications should be able to simply add more resources and when the workload decreases, they should be able to release some resources back to the cloud provider to reduce the monetary costs. In general, users of cloud services are charged based upon the amount of resources used or reserved but no guarantees are made regarding the performance that the service will provide.

In practice, both of the commercial NoSQL cloud offerings (e.g. Amazon SimpleDB) and commercial relational DaaS offerings (e.g. Amazon RDS, Microsoft SQL Azure) do not provide their users any flexibility to dynamically increase or decrease the allocated computing resources of their applications. While NoSQL offerings claim to provide elastic services for their tenants, they do not provide any guarantee that their provider-side elasticity management will provide scalable performance with increasing workloads [7], [9]. Moreover, commercial DaaS pricing models require their users to predetermine the computing capacity that will be allocated to their database instance as they provide standard packages of computing resources (e.g. Micro, Small, Large and Extra Large DB Instances). In practice, predicting the workload behavior (e.g. arrival pattern, I/O behavior, service time distribution) and consequently accurate planning of the computing resource requirements with consideration of their monetary costs are very challenging tasks. Therefore, the user might still tend to over-provision the allocated computing resources for the database tier of their application in order to ensure satisfactory performance for their workloads. As a result of this, the software application is unable to fully utilize the elastic feature of the cloud environment. The approach of virtualized database server provides software applications with more flexibility and control on being able to dynamically allocate and configure the physical resources of the database tier (database servers).

In the scenario of virtualized database servers where database servers are hosted in virtual machines on public clouds, cloud providers can only guarantee rigid resource availability, i.e., either a machine is up or it is down. However, the performance of the same virtual machine can vary wildly at different times due to different reasons such as over-commitment and interference between virtual machines [12]. Therefore, there might be a discrepancy between what providers provide and what users would might expect. In particular, cloud users typically care only about is the subjective performance of their virtual machines. To bridge this discrepancy, dynamic provisioning techniques have been proposed to base the elasticity behavior rules of the database tier of software application on the service level agreements of the software application [9], [11] instead of the traditional techniques which are are based on the consumption of the computing resources [4]. Ideally, cloud providers will have to forsake the approach of charging users a pre-defined sum in exchange for unknown resources and performance, and switch to a more concrete market-driven model where user can bid for resources according to specified valuations for the performance of those resource, thus affecting their prices [2].

2) Live Migration: Live migration is an important component of the cloud computing paradigm that provides extreme versatility for management of cloud resources by allowing applications and data to be transparently moved across physical machines with a consistent state. For example, live migration techniques can be used to improve compliance with tenant application quality-of-service by migrating the tenant with excessive workload to less loaded server. Thus, it is a main tool for achieving elasticity and dynamic provisioning goals for cloud services. However, live migration is a resource-intensive operation and can come at a price of degraded service performance during migration due the overhead caused by the extra CPU cycles which are consumed on the source and the destination servers in addition to the extra amount of network bandwidth used for the transmission process.

In general, NoSQL database systems follow different mechanisms for achieving the elasticity goal when the allocated computed resources are dynamically changing. For example, when new nodes are added to the cluster, the *Cassandra* systems moves the data stored on the old nodes to the new nodes that have just been bootstrapped. In the case of a perfectly balanced cluster, this would mean moving half of the stored data on each of the old nodes to the new ones. On the other hand, the *HBase* system relies on the notion of region servers where each server is responsible for a list of regions and acts as a cache for the data stored in the distributed file system level. Therefore, when new nodes running both a region server and a datanode are bootstrapped, the new region servers will start to serve a fair share of the available regions but the existing data will not be moved to the new datanode. Therefore, there are no big data transfers during new node bootstrapping. Clearly, reducing the amount of the data that needs to be moved when new nodes are bootstrapped leads to a reduction in the required interval of time for the systems to reach the stable state. Therefore, there is a tradeoff between the migration time, the size of the database and the amount of update transactions in the workload which are executed during the migration process. In a multi-tenancy environment of DaaS platforms, ideally, cloud database services should be able to transparently decide *which* tenant to migrate, *where* (to which server) this tenant should be migrated and *when* the migration should be performed according to the requirement and the performance characteristics of each tenant database in order to fully utilize the elasticity feature of the cloud environment [6]. These challenges remain as open issues for further investigation and the development of sophisticated optimizations and consolidation techniques.

The *Dolly* system [4] has introduced a live migration technique for virtualized database servers that clones the entire virtual machine of an existing database server which is then started on a new physical server, resulting in a new replica that starts to serve the application requests after a synchronization process. In practice, the cloud infrastructure cannot instantly respond to this request as it has to first find and reserve an available server, clone the virtual machine instance on it, replicate the existing data onto the new virtual machine and include the new replica in a load balancer so that it can be accessed by the application [9]. This is a time consuming process. In order to tackle this challenge, Dolly incorporates a model to estimate the latency to create a new database replica based on the snapshot size of the virtual machine to trigger the replication process well in advance of its necessity to occur according to the anticipated increases in workload. *Slacker* [1] is another migration technique that tries to minimize the performance impact of the migration process by leveraging the migration slack, resources that can be used for migration without excessively impacting performance latency. Since the majority of the resource cost in migration is from reading, writing, or sending a large amount of data, Slacker controls the effect of the migration process by controlling the upper bound rate of data transfer to the point at which performance latency steadily increases indicating that the database is overloaded and cannot maintain both its workloads and migrations.

In general, the virtual machines of the database serves can be located in multiple data centers that are geographically located in different places around the world and are charged on the pay-as-you-go pricing model. Therefore, users of cloud services have the ability to build an affordable geoscalable cloud-hosted database tier that can cope with increasing or bursty workloads. Developing efficient live migration techniques for achieving elastic database tier over multiple data centers is an important challenging issue which is open for further investigation and research.

IV. Benchmark Proposal

The Yahoo! Cloud Servicing Benchmark (YCSB) [5] is the most well known benchmarking framework for NoSQL databases. The benchmark currently supports different NoSQL databases but the schema and workload transactions can be easily adopted for relational scenarios. In addition, the benchmark can be extended to use various kinds of workloads. It is our ongoing work to extend the benchmark with a mix of workloads that vary over time in different ways (e.g., sinusoidal workloads, exponentially bursty workloads, linearly increasing workload, random workload). For example, some workloads will rise and fall repeatedly, others will rise rapidly and then fall back slowly, etc. For each workload, we plan to examine the way the underlying cloud database service will respond to the workload changes, and then we can quantify the defined elasticity metrics of cloud database services in addition to the associated billing cost for each scenario. The benchmark setup will also consider the aspects of the geographical distribution of the application clients and the underlying database replicas so that it can support the scenarios of running elastic database tier over multiple data centers.

References

[1] Barker, S. K., Chi, Y., Moon, H. J., Hacigümüs, H., and Shenoy, P. J. "Cut me some slack": latency-aware live migration for databases. In *EDBT* (2012).

[2] Ben-Yehuda, O. A., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. The Resource-as-a-Service (RaaS) Cloud. In *HotCloud* (2012).

[3] Bodík, P., Fox, A., Franklin, M. J., Jordan, M. I., and Patterson, D. A. Characterizing, modeling, and generating workload spikes for stateful services. In *SoCC* (2010).

[4] Cecchet, E., Singh, R., Sharma, U., and Shenoy, P. J. Dolly: virtualization-driven database provisioning for the cloud. In *VEE* (2011).

[5] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. Benchmarking cloud serving systems with YCSB. In *SoCC* (2010).

[6] Das, S., Nishimura, S., Agrawal, D., and El Abbadi, A. Albatross: Lightweight Elasticity in Shared Storage Databases for the Cloud using Live Data Migration. *PVLDB 4*, 8 (2011).

[7] Floratou, A., Patel, J. M., Lang, W., and Halverson, A. When Free Is Not Really Free: What Does It Cost to Run a Database Workload in the Cloud? In *TPCTC* (2011).

[8] Islam, S., Lee, K., Fekete, A., and Liu, A. How a consumer can measure elasticity for cloud platforms. In *ICPE* (2012).

[9] Sakr, S., and Liu, A. SLA-Based and Consumer-centric Dynamic Provisioning for Cloud Databases. In *IEEE CLOUD* (2012).

[10] Sakr, S., Liu, A., Batista, D. M., and Alomari, M. A Survey of Large Scale Data Management Approaches in Cloud Environments. *IEEE COMST 13*, 3 (2011).

[11] Sakr, S., Zhao, L., Wada, H., and Liu, A. CloudDB AutoAdmin: Towards a Truly Elastic Cloud-Based Data Store. In *ICWS* (2011).

[12] Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *PVLDB 3*, 1 (2010).

[13] Suleiman, B., Sakr, S., Jeffery, D. R., and Liu, A. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *J. Internet Services and Applications 3*, 2 (2012).

[14] Wada, H., Fekete, A., Zhao, L., Lee, K., and Liu, A. Data Consistency Properties and the Trade-offs in Commercial Cloud Storage: the Consumers' Perspective. In *CIDR* (2011).

[15] Zhao, L., Sakr, S., and Liu, A. A Framework for Consumer-Centric SLA Management of Cloud-Hosted Databases. In *IEEE Transactions on Service Computing (TSC)* (2013).