A Survey of Cloud Databases

Zhangbin Cheng, Rui Hu, Zhe Lim

Abstract. The amount of information that currently resides only in the cloud is small, but that is about to change. A recent study by IT industry group IDC estimates that by 2015, nearly 20 percentage of all information will be stored or processed in a cloud. Despite the growing movement toward cloud computing, some IT professionals remain standoffish toward the idea of porting a company's data onto a public cloud computing platform such as Amazon and Rackspace. In this report, we present a survey on the benefits and promises as well as limitations and risks of cloud databases. Meanwhile, a number of commercial products of cloud database are contrasted and analysed from diverse perspectives. Based upon our analysis, we conclude that although cloud databases technology theoretically demonstrates numerous fascinating advantages to both individuals and organisations, it has not reached a mature point that is suitable for all types of businesses. Whether deciding to move a small or significant amount of data to a cloud database really depends on the usage goals and context.

1 Introduction

With the rapid development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. Based upon these emerging advanced technologies, the current computing trend is to host user applications, services, and data in a network core, which is metaphorically referred to as the *cloud*. In the meantime, they also have enabled the realisation of a new computing model called *cloud computing,* in which resources (e.g., CPU and data storage) are provided as general utilities that can be leased and released by users through the Internet in an on-demand basis. With the mass adoption of cloud computing, we are seeing an explosion of Platform-as-a-Service (PaaS) offerings built on top of existing cloud computing infrastructure. One such offering is the availability of managed database service in the form of *cloud database*or *Database-as-a-Service (DBaaS)*.

In this report, we present a survey ofcloud databases, highlighting its key promises and challenges, current offerings, as well as limitations and risks. Our aim is to provide a critical analysis of cloud database and its commercial products from different perspectives, thereby identifying whether this fascinating technology would really propelthe movement from traditional databases towards cloud databases.

In a literal meaning, DBaaS is referring to the delivery of database software and related physical data storage as a service. We believe that a DBaaS must meet the following three requirements. Firstly, the service must be available to the customer on-demand, without any requirement for the installation and configuration of any hardware or software up-front. Secondly, the service must have a pay-as-you-go pricing model, without requiring long-term contract or up-front payment. Thirdly, the provider is responsible for managing the service, without any requirement for the customer to maintain, upgrade and administer their databases. Therefore, a more accuratedefinition of DBaaS could be expressed as: A managed service, offered on a pay-per-usage basis, that takes away the hassles of running your own database server and provides on-demand access to a database for the storage of application data.

Amazon is the first mover in the cloud computing space. They are also the first to provide a DBaaS with a product called Amazon RDS. Over the past few years, Google and Microsoft have played catch up and responded with their own offerings (Google Cloud SQL and Microsoft Azure SQL Database). While they are all cloud database products, they differ in many aspects such as architecture used, performance and pricing model. In section 6, we discuss each of these products in more detail.

In spite of considerable opportunities that cloud database has made available to the IT industry, in its current form, it also brings many challenges, limitations as well as risks that need to be carefully considered and addressed. The biggest drawback is the lost of physical control over data storage and the outsourcing of data security to the DBaaS provider. Data stored off-premise in a public cloud might be accessible by an unauthorised rogue server administrator or be subjected hack attempts. In addition, based upon cloud provider's geographic location, latency problems can also occur due to increased distance between user and application. Another main risk to consider is cloud provider's outages or even bankruptcy. Hence, when looking to implement a database in the cloud, IT professionals should make a trade-off between its actual benefits and potential risks, and a more detailed analysis of these things is given in section 7.

The remainder of this paper is organised as follows. In section 2, a number of related work is described. In section 3, we give a brief premier on different database architectures. In section 4, we provide a list of desirable features in a cloud database service and highlight the challenges of providing these features in Section 5. In section 6, we survey the main players in the DBaaS spacein terms of their performances, supported programming languages, database types and so on. In section 7, we report on the limitations and risks of current DBaaS offerings. Finally, we speculate on the future directions of the DBaaS industry in section 8 and end with our conclusions in section 9.

2 Literature Review

With the emergence of cloud computing, several studies have been carried out to explore the benefits and promises that cloud database can make. Based on a comprehensively theoretical analysis on cloud database, Hogan (2008) points out that although potential benefits of cloud database are overwhelming, attaining these benefits requires that each aspect of the cloud platform support the key design principles of the cloud model, such as dynamic scalability, but unfortunately, the majority of today's database servers are incapable of satisfying this requirement [4].

It is worth noting that the commercial products used to evaluate in this work are out-dated, but in our work, up-to-date cloud database products are discussed. The results of a related study on the economics of cloud computing have been reported in [34], which deeply analyses the cost advantages of cloud database over conventional in-house application deployment and identifies a set of key factors affecting the costs of a deployment choice when considering a move to the cloud.

In [33], cloud features such as scalability, elasticity and pay-per-use pricing are experimented by trying to design a database system to support these features, from which, it is concluded that achieving all cloud features is more easily said than done. Database systems on a cloud computing infrastructure still face many new challenges with large scale operations, such as lightweight elasticity and autonomic control to minimise the operating cost [33].

To help us better demonstrate the cloud database characteristics and its related concepts, we reviewed work done by [7], which examines distributed database architectures that are widespread used in current database systems. Also, it conducted comprehensive experiments among a variety of cloud database products, which inspired us of carrying out investigation and comparisons concentrating on relational cloud database services.

On the technical side, [17, 23] investigated the use of virtual machines to support multi-tenancy. Techniques for live migration of virtual machines between different physical machines were investigated in [18, 19]. Different schemes to support automatic data partitioning schemes were identified in [20, 24]. [27] presented ways to support query processing on encrypted data.

[28] discussed the limitation and risks of outsourcing data management.

[20] argued that the current DBaaS efforts do not adequately address the three important challenges of efficient multi-tenancy, elastic scalability and database privacy, and describes a system called Relational Cloud that they are building that addresses these challenges.

3 Database Architecture

In order to get an insight into the cloud database ecosystem, let us revisit a few database architectures that are commonly employed today. The classic multi-tier database application architecture is used as a starting point and then variations of this architecture such as replication, partitioning and distributed control are depicted along with their characteristics. Knowing these concepts will help us understand how they are packaged and adopted by commercial cloud database services described in later sections.

3.1 Classic

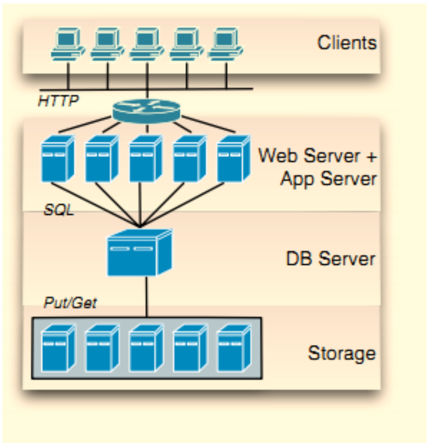

Figure 1: Classic Database Architecture

Figure 1 [7] illustrates the classic database architecture, which is used by most database applications today. Requests from clients are routed to an available web/application server. The application then makes the appropriate SQL queries to the database server. The database server executes the queries and returns the results (if any) to the application. In this architecture, many application servers connect to a single database server and the database server uses one or more disks as the underlying storage. Scaling the classic architecture is easy for the application server and storage layer. These two layers are horizontally scalable which means that to scale the application server layer, one just needs to add additional application servers. Similarly, storage can be scaled by using more or faster disks.

The database server, which sits between the application server layer and storage layer, is the bottleneck in this architecture. The only way to scale the database server is to buy a bigger machine and there is an eventual limit on how big a machine you can buy. Therefore, this architecture has a limitation in its scalability. With scalability being the main feature of a DBaaS, this architecture is not suited for cloud databases.

3.2 Partitioning

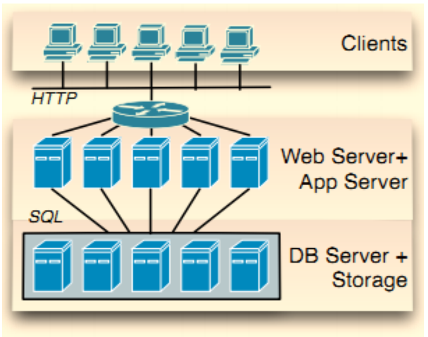

Figure 2: Partitioning

Figure 2 shows one variant of the partitioning architecture. Instead of using a single database server, the database is split into multiple partitions and each partition is managed by a dedicated database server.This architecture removes the single database server bottleneck of the classic architecture and is the architecture of choice for large-scale databases. This makes it a viable architecture for cloud databases. The main drawback is its complexity; adding or removing database servers require data to be repartitioned between machines. The application server must also be modified to be partition aware – to know the correct partition to fetch data from. Traditional join operations might no longer work as data is scattered across the different partitions.

3.3 Replication

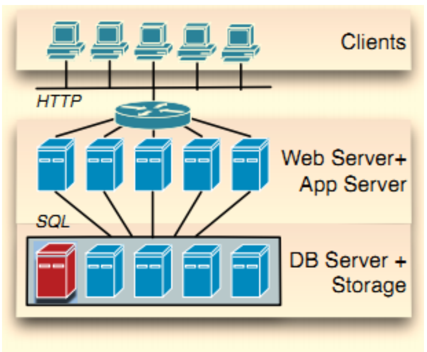

Figure 3: Replication

The replication architecture shown in Figure 3 is also known as master-slave architecture. Instead of splitting data into partitions, each database server has its own copy of the entire database. Read operations can be handled by any of the database servers, which makes it suited for read intensive workloads.All write operations must be directed to the master (show in red). The master will in turn be responsible for propagating all committed updates to its slaves. This architecture is well suited for read intensive applications (most websites are) but falls short when there is a large number of write operations because each write operation has to be executed by all database servers (cannot be load balanced). One positive side effect of this architecture is the replicated copies can be used for backup purposes. In large-scale systems, a combination of partitioning and replication can be used. Data is first split into multiple partitions and each partition itself is setup to have one master and a set of slaves.

3.4 Distributed Control

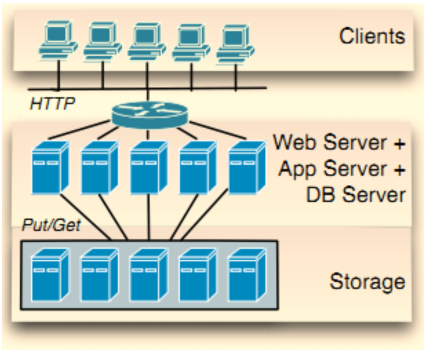

Figure 4: Distributed Control

Figure 4 illustrates a distributed control architecture, which can also be described as a *shared-disk architecture* [8] with loose coupling between the nodes to achieve scalability. Basically, the storage tier is separated from the DB servers and the DB server layer is merged into the web/app server layer. In this architecture, DB servers are allowed to read and write the shared storage concurrently against certain distributed protocols to guarantee different levels of consistency, which on the other hand sacrifices the full consistency and only a consistency level referred as *eventual consistency* [9] was maintained. However, due to its desirable scalability and elasticity at all tiers, this architecture is potentially the best option for deploying database on the cloud. To be noted that, this architecture is mostly adopted by NoSQL databases which is out of scope of this report as we are focusing on relational database.

4 Cloud Database Promises

Firstly, it should be understood that a cloud database is more than simply running an instance of a traditional relational database management system (RDBMS) one a cloud platform like Amazon Web Services (AWS). Such a deployment do not maximise the capabilities of a cloud computing environment. Here, we provide a list of features that we think a DBaaS should have.

4.1 Transparent Scalability

Scalability is a desirable property of a system, which indicates its ability to increase throughput when additional resources, typically hardware, are added. A system, whose performance improves after adding hardware, can be said to be a scalable system.

In general, scaling can be done in two ways: scaling vertically (scale up) and scaling horizontally (scale out).To scale up means to add resources to a single node in a system, typically involving the addition of processors, memory or disk to a single computer. In the classic architecture (section 3.1), buying a bigger machine is an example of scaling up. Scaling out involves placing partitions on different database machines. Each machine is responsible for handling only its subset of data. This is described in detail in the partitioning architecture in section 3.2.

Scaling up is the easier of the two approaches and is typically undertaken before scaling out. In a non-cloud environment, scaling up will involve some lag time for ordering hardware and some downtime for hardware replacement. On the other hand, scaling out is non-trivial. Application code needs to be rewritten to work with partitions, which brings with it numerous restrictions such as not being able to use general `JOIN` clauses on tables that are located on different partitions. There is also the same lag time for provisioning the necessary hardware. Perhaps the most challenging of all is the need to repartition data when the number of database machines changes.

It is important to be able to scale a database to handle more users. However, for most organizations, the scaling of a database is just a form of “muck”; a necessary evil that takes up a lot of time yet do not add any value to the final product.

Ideally, in a cloud database service, the issue of scalability should be handled transparently by the DBaaS provider, which should have their own core competencies in scaling databases. Users of the DBaaS should only need to specify a few parameters such as size of database and estimated workload, and the DBaaS should automatically make the necessary preparations behind the scenes to support the estimated workload. When users wish to increase the capacity of their database instance, it should be as simple as making an API request to the DBaaS or adjusting a dial in a management console.

The scaling that happens in the background should be transparent to the application. From the application’s point of view, it should continue to assume that it is working with a single database server, using the same SQL interface. Existing applications should work without any modifications.

4.2 Automatic Elasticity

One of the major attractions of the cloud computing model is its pay-per-use pricing model. By paying for only what you use, there is no risk of over or under-provisioning for capacity. Elasticity is the key component underpinning the pay-per-use pricing model. Elasticity is the ability to respondto load variations by dynamically adding more resources during high load and removing resources when the load is low. When a database workload can be adequately serviced by a single machine, only a single machine is used. When workload increases and a second machine is needed, it is automatically provisioned and added to the system.

Even though elasticity is often associated with the scalability of the system, a subtle difference exists. Scalability is the ability of a system to accommodate larger loads by either using more powerful hardware or adding additional nodes whereas elasticity is the ability to adjust the amount of resources used on the fly.

The average utilization of servers in a datacentre can be as low as 5-10%. This is because most servers are overprovisioned to handle for peak loads, which might only happen once per day, once per week or even once per year (Cyber Monday for e-commerce websites). In an elastic system, the user can save considerable cost by automatically matching provisioning of resources to actual usage levels.

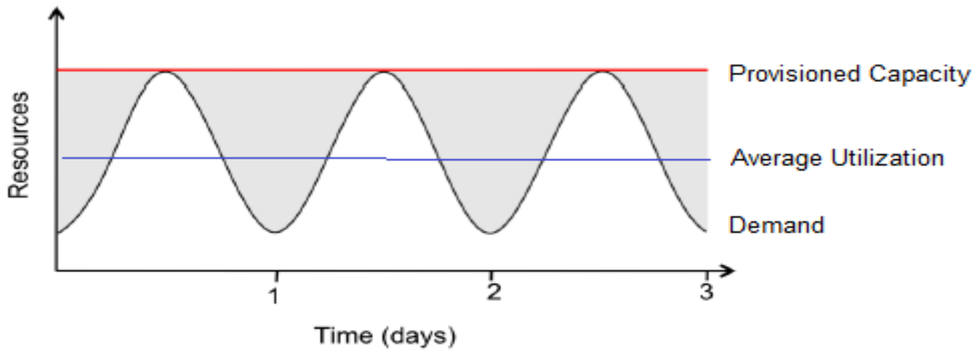

Figure 5 Provisioning for peak load

Elasticity is also useful to handle unexpected load increases either from content that has suddenly gone viral or when a popular website links to a much smaller website. This is typically not handled very well by smaller web applications and has led to the coining of the term *Slashdot effect*; the phenomenon of a website becoming virtually unreachable due to a sudden large influx of traffic. The cost to the web application owner is the wasted opportunity cost for not being able to take advantage of the current publicity. By the nature of it being unexpected, the application owner could not have provisioned resources to handle the extra load. However, with automatic elasticity, this can be avoided.

Unlike scalability, elasticity is not achievable in a non-cloud environment because additional hardware cannot be provisioned on the fly. As per scalability, the application should be able to work unmodified. The application can assume that it will always have the capacity it needs.

4.3 High Availability

Another key promise of cloud computing is increased uptime. Like scalability and elasticity, availability is crucial to successful business. The average downtime costs vary considerably across industries, from approximately US$90,000 per hour in the media industry to about US$6.48 million per hour for large online brokerages.

A database implemented in the cloud should have features that piggyback off of a cloud provider's infrastructure where high availability is concerned. Additionally, that database's degree of availability is v affected by the next cloud computing promise: easy Data distribution and redundancy.

4.3 Redundancy and High Availability

A database system is highly available if it remains accessible to its users in the face of hardware failure. High availability is required in cloud database management since it is crucial to successful business. According to [10], the average downtime costs vary considerably across industries, from approximately US$90,000 per hour in the media industry to about US$6.48 million per hour for large online brokerages.

In a cloud environment, high availability is typically achieved by the ability to distribute compute resources and data across different geographies. This kind of multi-geographic data distribution supplies the benefits of high availability if a particular region goes down, and simplified disaster recovery so the data is protected if a particular physical disaster befalls one of the cloud provider's data centres. Definitely, this promise relies on an assumption that the database contains redundant copies of data spread across different data centres, which brings another feature that cloud database should have - redundancy. Redundancy basically means keeping more than one copy of the underlying data, so if the primary copy is destroyed, another copy is available to use. Apart from storing redundant redundancies on different data centres as stated before, it is also necessary to store multiple copies of data on different physical racks within a single data centre. Thus if one rack experiences a machine failure, the other rack can continue to serve data.

On the other hand, in a traditional environment, taking backups and recovering data is often regarded as a manual job of the database administrator. However, finding the correct pieces of data and bringing them back together is a very difficult job, as data is fragmented across the storage platform and constantly moving behind the scenes. Moreover, it is extremely time consuming if the data needs to be completely recovered and this operation always produces a long system and database downtime. The ideal scenario of a manual database recovery is to get data back the customer in a timely manner, but even this would bring enormously expensive downtime costs.

4.4 Reduce administrative hassle

The recurring theme in the last three sections is the automation of common database management tasks for the end user. In addition to what was mention, running a database server also involves substantial administrative hassle for tasks such as setting up, upgrading and maintenance. A DBaaS can help automate all these and leave the users with time to focus on their applications.

5 Challenges

Having gone through the list of desirable features in a DBaaS, we discuss the challenges of providing these features.

5.1 Efficient Multi-tenancy

Multi-tenancy means co-locating several database instances on the same machine. The typical way of providing multi-tenancy is by using virtual machines (VMs). Each individual database instance is packed into a VM and multiple VMs are placed into a single physical machine. The challenge for a cloud database provider is to determine how best to place a set of VMs into the least number of physical machines, while maintaining performance for users.

Each database instance is typically promised a certain amount of CPU, RAM and I/O resources. A simple approach for VM placement is to divide up the resources of a physical machine and allocate them to a fixed number of VMs. This approach is far from efficient because the VMs do not fully utilize all their resources all the time.

In order to achieve efficient multi-tenancy, the cloud database provider would need to (a) detect (or predict) workload for each database instance (b) determine the allocation of VMs for each physical machine and (c) perform live migration of VMs to their target physical machine.

For (a), workload detection has to be automatically. This can possibly be done via black box machine learning [1] or to rely on the cost models produced by the database's built-in query optimizer [2].

In a typical application, the workload level changes throughout the time of day or day of week. The workload pattern of each database instance should be identified to help determine which set of databases has uncorrelated workloads, with the aim of co-locating them in a single machine. For example, a database instance powering an office application that has peak usage during working hours can be co-located with a database instance of an entertainment application that is only used during non-working hours.

When workload changes, a database instance might need to be moved to a different machine. The challenge is to perform live migration of VMs from one machine to another with no perceptible downtime for the end user. This has been shown to be possible in [3, 4, 6] with downtime as low as 60ms.

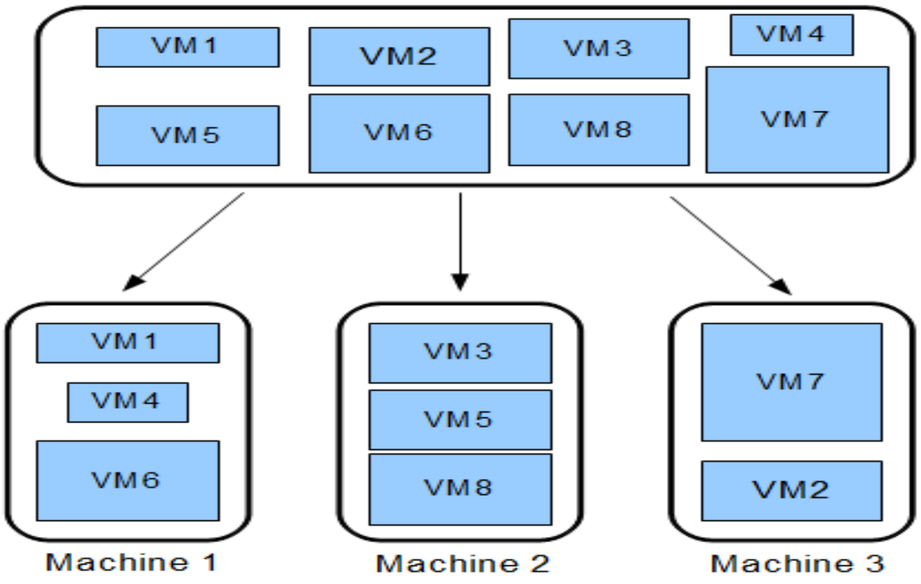

Figure 6 Virtual machine allocation strategy

5.2 Scaling Horizontally

The challenge is for the cloud database provider to support scaling out; to be able to scale a single database instance beyond a single physical machine.

When scaling vertically, a database instance is simply given more resources to handle the additional workload. In a VM based multi-tenancy environment, scaling vertically is easy using the virtual machine monitor (VMM), which controls the allocation of physical machine resources to the different VMs [7]. The VMM can be asked to provision more resources to a particular VM without the need to add hardware to the physical machine. For database workloads, I/O is often the main bottleneck and emphasis has been given in scaling up I/O, mainly by the use of fast shared storage devices, for which some are capable of delivering up to 150,000 IOPS [8, 9]. Scaling vertically is relatively easy but this approach has an upper bound of a single physical machine limit.

When scaling vertically is no longer feasible, the database instance needs to be scaled horizontally; where its data and workload are partitioned and processed by multiple VMs. There is a fourfold challenge in providing automatic horizontal scaling: (a) to partition data without expert knowledge of the schema (b) to partition data in such a way that the workload is uniformly distributed across partitions (c) to repartition data when workload changes and finally (d) to transparently route requests to the correct partition and to handle distributed transactions across multiple partitions.

Most of the current partitioning schemes are manual. To achieve (a) and (b), the challenge is for the cloud database provider to identify suitable automatic data partitioning schemes [10, 11]. For example, a workload-aware graph base partitioning strategy that was identified in [10] could be used. When scaling horizontally, it is important for the system to scale linearly to the number of partitions. [10] shows that they were able to achieve a near linear speedup (7.7x speedup with 8 nodes) with their partitioning strategy.

When workload changes, data has to be repartitioned to more (or less) partitions. The challenge is having the ability to serve current requests while repartitioning is ongoing. This is particularly difficult when the underlying data changes during repartitioning. A demand-driven caching strategy [5] can be used to perform live repartitioning of data whereby requests are directed to a new empty partition, and the new partition fetches any data that it does not have from an existing old partition.

Finally, to make the scaling transparent to the application, a transaction coordinator [5] that sits between the application and the database servers needs to be built to route requests across partitions and perform the merging of results from different partitions. This transaction coordinator needs to handle the full range of SQL queries.

5.3 Elasticity

Providing elasticity involves the active monitoring of database workloads and making dynamic adjustments to the resources allocated to a database instance, including performing the necessary data partitioning and live migration necessary when the workload exceeds a single machine limit.

The challenge is to provide a smooth and timely transition among different levels of workload. This is especially challenging for erratic workloads because performing the data migration necessary for elasticity is not without cost, and the frequent scaling out/in will lead to a waste of resources.

5.4 Data Privacy and Security

In the cloud database model, the user's database resides on the cloud database provider's premise. The cloud database provider should ensure data privacy and security for their users but there are a few challenges. The obvious solution to data privacy is to encrypt all the data stored at the cloud database provider. However, the challenge is how to execute SQL queries over the encrypted data. The other challenge is to be able to provide encryption transparently without any modification on the user's side.

The ultimate goal is to achieve fully homomorphic encryption whereby typical SQL operations such as equality and ordering can be done on encrypted data. However, fully homomorphic encryption schemes are still prohibitively expensive [13]. The current solutions being proposed [12, 13] are able to execute SQL queries over encrypted data to varying degrees.

We think the idea of adjustable security in [13] is interesting. It employs different encryption levels depending on the query that is being run. Both [12] and [13] require a query translator to sit between the application and the database in the cloud provider. Current solutions require some modification on the application to pass in the encryption keys.

6 Main Players

This section describes the cloud database services provided by the big three players in the cloud market, namely Amazon, Microsoft and Google, and each of their products will be detailed in following sub sections.

6.1 Amazon RDS

Amazon RDS (Relational Database Service) is a cloud-based RDBMS offered on AWS (Amazon Web Service). According to Forrester Wave [10], Amazon RDS is one of the most widely used cloud database platforms, supporting thousands of customers across many industries. This is primarily because RDS supports several popular RDBMS, including Microsoft SQL Server, MySQL and Oracle Database, offering enterprises more choices when ones require full features and capabilities of a relational database or willing to migrate existing applications and tools utilizing relational database without much efforts. Also, the service is pre-packaged such that users do not need to worry about managing the deployment, software patches, software upgrades, and backups.

Importantly, Amazon RDS (for Oracle and SQL Server engine) employs the Classic architecture (Figure 1), where a single DB server is in charge of all the requests which makes the DB server a bottleneck of this architecture. Therefore, in terms of scalability promised by Amazon, the RDS in fact is only capable of scale-up, allowing users to adjust the instance type (e.g. Micro, Extra large) to meet their needs. Currently, Amazon states the RDS can provision up to 3TB and 30,000 IOPS per DB instance [11]. On the other hand, for MySQL engine deployment, Amazon realizes partially scale-out utilizing MySQL's native replication to propagate changes made to a master DB instance to its associated Read Replicas (slaves) to go beyond the capacity of a single DB instance to serve high-volume application read traffic. Still, as mentioned (Figure 3), the constraint of this replication architecture is the master DB, which will become the bottleneck when overloaded. On top of that, Amazon provides a feature called Multi-AZ deployment, which maintains a geographically separate replica to protect latest updates against unplanned outage and failover quickly.

6.2 Microsoft Azure SQL Database

Windows Azure SQL Database is a relational database service on the Windows Azure Platform. It takes advantages of SQL Server and .Net technologies to benefit users and development community that are familiar with SQL Server engine and .Net framework as a high compatibility is suggested between the products [12]. However, it should be noted that SQL Database exposes only a subset of the full SQL Server functionality and data types [13]. For example, many SQL Server Transact-SQL statements have parameters that allow users to specify file groups or physical file paths are not supported in SQL Database.

Azure SQL Database implements a multi-tenanted architecture by hosting several servers created by different users on the same physical machines in a shared environment; comparably, Amazon RDS does it in a different level by provisioning a specialized EC2 instance to the client's AWS account and then the client can create multiple, highly varied, DB instances on that EC2 instance. Nonetheless, Azure SQL Database is said to be more native to cloud platform than Amazon RDS (which basically it is a virtual machine with MySQL etc. installed on it), because it is designed from scratch, specifically for cloud and possesses a better scalability. As stated in [14], SQL Database adopted the Replication architecture (Figure 3) which we know the master DB server is the bottleneck when dealing with update-intensive workloads. According to experiments conducted by [7], however, SQL Database can scale almost linearly as requests increase. The reason is explained in [7] as Azure is using very powerful machines to run the database servers even we can expect a performance drop once requests exceed a threshold. Therefore, Microsoft advises clients themselves to shard/partition (Figure 2) data across multiple servers to scale much better than having it all on a single DB server.

6.3 Google Cloud SQL

Google Cloud SQL is a MySQL database that lives in Google's cloud--Google App Engine [15]. Similar with Amazon and Microsoft's products, Google Cloud SQL is a managed service that maintains, manages and administers clients' databases, allowing clients to focus on their applications and services, achieving faster time-to-market. Basically, it offers the capabilities of a familiar MySQL database with a few additional features and a few unsupported features, for example, MySQL replication and user defined functions are not supported. Importantly, Google Cloud SQL is currently only available for Google App Engine applications that are written in Java or Python, which means that the database cannot be accessed from outside of Google's cloud environment and because they are closely tied, to determine whether go for Google Cloud SQL is not an independent decision—it is part of the decision of whether to write applications for App Engine.

As touted by cloud providers, the elastic pricing scheme is a key factor for us to opt for cloud database services, let us take a close look at each pricing scheme offered by them:

Google offers two billing plans for Cloud SQL: a package plan and a per-use plan. The price is calculated based on the amount of RAM, storage and I/O per day/hour. For example, 40 hours usage of an instance with 1GB data storage, 0.125GB of RAM, and 1Million I/O will cost US$1.34 per month under Google's per use plan. Likewise, Microsoft also has two plans one Pay-As-You-Go and6 or 12-Month Plan. Amazon has some more sophisticated but still similar pricing structures compared with the others.

Table 1 presents a comprehensive comparison of the three providers:

	Amazon RDS	**MS Azure**	**Google Cloud SQL**
Launch Time	2009	2008	2011
Architecture	Classic/Replication	Replication	N/A
Database	MySQL, MS SQL Server, Oracle Database	SQL Database	MySQL
Maximum Size of a Database Instance	3TB	150GB	100GB
Programming Language	Any languages	.Net languages	Java, Python
Database Language	SQL	SQL	SQL
Configurability and Ability to Tune Database	High. RDBMS instantiation in cloud.	Medium. No control over memory allocation.	Low

Transaction Capability	Yes	Yes	Yes
Replication	Yes	Yes	Yes
Datacentre Options	EU, US, South America, Asia Pacific	EU, US Asia Pacific	EU, US
Target Markets	All range of applications	Medium-sized applications	Small to medium-sized

Table 1

7 Limitations and Risks for User Adoption

7.1 Data privacy and security

Corporate databases contain anything from customer data, trade secrets to proprietary information. For this reason, corporations are very protective of their database. Furthermore, some corporations are required to comply with regulations such as Sarbanes-Oxley and Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) with special requirements for auditability.

When using a cloud database provider, application data is placed on a public network and stored on a multi-tenant machine. This makes the data potentially accessible to rouge server administrators and malicious users co-located on the same machine (by breaking out of their VM [17]). There is also the psychological issue of losing control to one's data when it is no longer within the safe confines of corporate firewall.

Some laws may require corporations to keep their data within national boundaries. This is a problem because data stored in a cloud database is often replicated and stored in different (undisclosed) locations in order to provide high availability and backup. A corporation may not want their data to be within the reach of another country's court system. For example, a Swiss bank will not want their list of customers to be one subpoena away from the United States government.

It is better to be safe than sorry. It is essential for all applications that use a cloud database provider to properly encrypt their data before it is sent to the cloud database. A proper encryption would ensure that the cloud database would ever only work with encrypted data [12, 13]. Encrypted data is only decrypted locally upon its return. However, legacy applications may not be modifiable to support encryption.

An application can also be modified to only transmit its non-sensitive data for processing on the cloud. When the processing is done, the sensitive data can be reconnected with the processed non-sensitive data locally. TC3 Health used this approach when they transitioned their HIPAA-compliant application to Amazon Web Services [18].

7.2 Scalability

Despite the marketing hype, none of the cloud database providers we looked at offer scalability beyond a single machine. This takes away the main selling point of using a cloud database and in our view greatly limits user adoption.

The architecture behind a cloud database is generally opaque but we inferred that the single machine limit exists by looking at how database instances are provisioned. When deploying a database instance, the user is asked to select their desired instance size. Using Amazon RDS as an example, the smallest instance has 630MB of memory and a single core CPU and the largest instance has 68GB of memory and an 8-core CPU. However, there is no option to select more than one instance for a single database and this implies the existence of a single server scalability limit. Having a fixed instance size also implies that elasticity is not supported; users pay a fixed per hour cost regardless of usage.

Although a limit exists, it is important to point out that the largest instance might be sufficient to support most applications [quote perfect paper]. With the ability to scale I/O independent of memory and CPU [9], it might take a very high volume usage to hit saturation point.

We argue that without scalability and elasticity as benefits, one is not really using a cloud database but just a database instance hosted in a VM, and therefore one should not pay a per hour premium for using one.

7.3 Availability

One of the main reasons for using a cloud database is high availability. While availability is generally high, service outages do happen from time to time. None of the providers we surveyed provides a Service Level Agreement (SLA) so there is no guarantee of uptime.

Service and Outage	**Duration**	**Date**
Amazon S3	2 hours	February 15, 2008
Amazon S3	6-8 hours	July 20, 2008
Google AppEngine	5 hours	June 17, 2008
Amazon RDS	24 hours	April 20, 2011
Amazon RDS	4 hours	June 14, 2012
Amazon EC2, Amazon EBS	12 hours	October 22, 2012

At the extreme end of the spectrum, service outage could be permanent when the database provider goes out of business. During the writing of this report, a cloud database provider called Xeround announced that they are shutting down their service [15]. Their users were only given a week to migrate data out of Xeround. With this in mind, care should be taken to choose a provider that has strong financial backing.

To achieve high availability, there should be no single source of failure in a system. In some sense, the usage of a single cloud database provider itself is a potential single source of failure. If availability is important, a backup provider should also be used. Here the pay-as-you-go pricing model can be exploited since only storage and minimum computing power are required.

7.4 Latency and Network Limits

If the application is not hosted at the same location as the cloud database provider, depending on the distance between them, network latency might become an issue. Here we are limited by the speed of light, which dictates the minimum latency between two locations. For example, the minimum network round-trip between Melbourne and California (where Amazon RDS is hosted) is 85ms. A web application that makes 5 database queries to generate a webpage will take almost half a second to respond to its users. For most modern web applications, this delay is unacceptable.

Bandwidth limits need to be considered when migrating an existing locally hosted database to a cloud database provider. Assuming a 1TB database and 20Mbits/second of network transfer speed, it would take 4.6 days to perform the initial transfer. Depending on how the transfer is done, the application might have to be taken offline during the initial migration.

Data transfer costs is also an important factor. There is an initial cost for migrating existing data (as per the example above) and the ongoing cost for making queries and receiving data from the cloud database. Amazon charges $0.10-$0.15 per gigabyte transferred. For applications that make a lot of database queries, the ongoing data transfer cost might be substantial and negates any potential cost savings from using a cloud database.

Most cloud computing providers do not charge their users for internal data transfers. Thus, one way to reduce latency and eliminate data transfer costs is to co-locate the application in the same cloud computing infrastructure as the cloud database provider. For Amazon RDS, this means using Amazon EC2 servers for computing

7.5 Interoperability and Lock-in

There is generally no interoperability between cloud computing platforms. Each platform has its own set of APIs for provisioning resources, which means migrating from one platform to another would require some modification to the application. For cloud databases, this is not so much of a problem because the common language for database is SQL and all providers strive to provide compatibility with existing libraries.

There is a risk of being over reliant on a single cloud database provider. If no viable alternative exists, the user is effectively locked in to the current database provider. The user will have no bargaining power if prices were to increase or if service levels were to deteriorate. In the extreme case of a provider going out of business, significant downtime can be expected as the application is modified to use an alternative platform.

Some work has been done to improve on interoperability. The forerunner is a project called OpenStack that seeks to standardize cloud computing platforms. OpenStack has significant momentum and has the support of more than 150 large companies including Rackspace, HP and IBM [16].

8 Future Directions

We think the current incarnation of DBaaS such as Amazon RDS do not live up to the promise of a cloud database service. In order to address the shortcomings of current DBaaS, the RDBMS has to be rethink from the ground up with cloud computing in mind.

Here, we present an interesting project called Relational Cloud [20]. Relational Cloud is a DBaaS prototype that is being developed by a team at MIT that aims to address three important challenges: efficient multi-tenancy, elastic scalability and database privacy. Briefly, efficient multi-tenancy is solved by co-locating uncorrelated workloads on a database server. Elastic scalability is achieved by using a graph-based data partitioning algorithm that they developed and finally database privacy is done using an encryption layer called CryptDB (also developed by the MIT team). Early results are promising; the data partitioning yielded nearly linear speedup (7.7x speedup with 8 nodes) and encryption/decryption can be done with only 15% decrease in speed.

We think Relational Cloud is a ground breaking project and should be a template for future cloud database services.

9 Conclusion

Along with the economic and social development, demands on computing and information technology support are soaring. Importantly, database management systems (DBMS) are an integral and indispensable component in most computing environment facilitating both transactional and analytic functions; meanwhile, the requirement of minimum administrative burden of hardware provisioning, configuration, scaling, performance tuning, backupet cetera, associated with on-premise database have driven the rapid adoption of cloud database services. Scalability, elasticity, availability, pay-per-use pricing and economies of scale from large-scale operations are the major promises of a DBaaS.

Due to these advantages of cloud database and its market potential, many big players of cloud computing have been offering a range of cloud database products with differing feature sets and different pros and cons. It is worth noting that the survey conducted towards the big three cloud providers reveals thatthese commercial products are targeting on different group of users and therefore we recommend a detailed examination of eachDBaaS before making a choice.

In spite of the promised benefits, we conclude that cloud database services are still in its infancy and still have to solve some of the keychallenges such as efficient multi-tenancy, horizontal scaling, automatic elasticity andensuring data privacy and security. Furthermore, some inevitable limitations and risks cannot be ignored, such as compliance with regulations, network latency and operator lock-in.

In summary, though we believe that the growing popularity of cloud databases is going to mark a beginning of new era of databases, a number of unsolved challenges or limitationsand risks will limit its adoption to startups and small businesses.

Reference and Sources

[1] Zhang, Qi, Lu Cheng, and RaoufBoutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of Internet Services and Applications* 1.1 (2010): 7-18.

[2] Hogan, M. (2008). Cloud computing & databases. *ScaleDB Inc*.

[3] Ramanathan, S., Goel, S., &Alagumalai, S. (2011, December). Comparison of Cloud database: Amazon's SimpleDB and Google's Bigtable. In *Recent Trends in Information Systems (ReTIS), 2011 International Conference on* (pp. 165-168). IEEE.

[4] Agrawal, D., El Abbadi, A., Das, S., & Elmore, A. J. (2011, January).Database scalability, elasticity, and autonomy in the cloud.In *Database Systems for Advanced Applications* (pp. 2-15).Springer Berlin Heidelberg.

[5] Dory, T., Mejías, B., Van Roy, P., & Tran, N. L. (2011). Comparative elasticity and scalability measurements of cloud databases.In *Proceedings of the 2nd ACM Symposium on Cloud Computing, SoCC* (Vol. 11).

[6] (2012). Evaluating Apache Cassandra as a Cloud Database. *DataStaxCorporation*.

[7] Kossmann, D., Kraska, T., &Loesing, S. (2010, June).An evaluation of alternative architectures for transaction processing in the cloud.In *Proceedings of the 2010 international conference on Management of data* (pp. 579-590). ACM.

[8] M. Stonebraker. The Case for Shared Nothing. *IEEE Database Eng.Bull.*, 9(1):4–9, 1986.

[9] W. Vogels. Eventually Consistent.*Commun.* ACM, 52(1):40–44, 2009.

[10] The Forrester Wave: Enterprise Cloud Databases, Q4 2012 by noel yuhanna, november 8 2012

[11] Amazon Relational Database Service.*Amazon RDS*. Retrieved May 31, 2013, from http://aws.amazon.com/en/rds/#

[12] Pizzette, L., & Cabot, T. (2012). Database as a Service: A Marketplace Assessment.

[13] Data Types (Microsoft Azure SQL Database).*Windows Azure.* Retrieved May 31, 2013, from http://msdn.microsoft.com/en-us/library/windowsazure/ee336233.aspx

[14] S. Sengupta. SQL Data Services: A Lap Around. In *Microsoft Professional Developers Conference* (PDC), 2008

[15] About Google Cloud SQL. *Google Developers.* Retrieved May 31, 2013, from https://developers.google.com/cloud-sql/docs/introduction

[16] Xiong, P., Chi, Y., Zhu, S., Moon, H. J., Pu, C., & Hacigumus, H. (2011, April). Intelligent management of virtualized resources for database systems in cloud environment. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on* (pp. 87-98). IEEE.

[17] Soror, A. A., Minhas, U. F., Aboulnaga, A., Salem, K., Kokosielis, P., & Kamath, S. (2008, June). Automatic virtual machine configuration for database workloads. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 953-966). ACM.

[18] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., ... & Warfield, A. (2005, May). Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2* (pp. 273-286). USENIX Association.

[19] Voorsluys, W., Broberg, J., Venugopal, S., & Buyya, R. (2009). Cost of virtual machine live migration in clouds: A performance evaluation. In *Cloud Computing*(pp. 254-265). Springer Berlin Heidelberg.

[20] Curino, C., Jones, E. P., Popa, R. A., Malviya, N., Wu, E., Madden, S., ... & Zeldovich, N. (2011). Relational cloud: A database-as-a-service for the cloud.

[21] Das, S., Nishimura, S., Agrawal, D., & El Abbadi, A. (2010). *Live Database Migration for Elasticity in a Multitenant Database for Cloud Platforms*. Technical Report 2010-09, CS, UCSB.

[22] Aboulnaga, A., Salem, K., Soror, A. A., Minhas, U. F., Kokosielis, P., & Kamath, S. (2009). Deploying database appliances in the cloud. *IEEE Data Eng. Bull*, *32*(1), 13-20.

[23] Hogan, M. (2009). Database Virtualization and the Cloud. *Scale DB Inc., December*.

[24] Curino, C., Jones, E., Zhang, Y., & Madden, S. (2010). Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, *3*(1-2), 48-57.

[25] Rao, J., Zhang, C., Megiddo, N., & Lohman, G. (2002, June). Automating physical database design in a parallel database. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 558-569). ACM.

[26] Hacigümüş, H., Iyer, B., Li, C., & Mehrotra, S. (2002, June). Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (pp. 216-227). ACM.

[27] Popa, R. A., Redfield, C., Zeldovich, N., & Balakrishnan, H. (2011, October). CryptDB: protecting confidentiality with encrypted query processing. In*Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (pp. 85-100). ACM.

[28] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... & Stoica, I. (2009). Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, *28*.

[29] Xeround pulls the plug on cloud database service. *GIGAOM.* Retrieved May 31, 2013, from http://gigaom.com/2013/05/01/xeround-pulls-the-plug-on-free-cloud-database-option/

[30] Companies Supporting The OpenStack Foundation. *OpenStack*. Retrieved May 31, 2013, from http://www.openstack.org/foundation/companies/

[31] Webhost hack wipes out data for 100,000 sites. *The Register.* Retrieved May 31, 2013, from http://www.theregister.co.uk/2009/06/08/webhost_attack/

[32] AWS Case Study: TC3 Health. *Amazon AWS*. Retrieved May 31, 2013, from http://aws.amazon.com/solutions/case-studies/tc3-health/

[33] Agrawal, D., El Abbadi, A., Das, S., & Elmore, A. J. (2011, January). Database scalability, elasticity, and autonomy in the cloud. In *Database Systems for Advanced Applications* (pp. 2-15). Springer Berlin Heidelberg.

[34] Tak, B. C., Urgaonkar, B., & Sivasubramaniam, A. (2011, June). To move or not to move: The economics of cloud computing. In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing* (pp. 5-5). USENIX Association.

[35] Confronting System Downtime. *EVOLVEN*. Retrieved May 31, 2013, from http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html