SERVICE LEVEL AGREEMENT-BASED MIGRATION FOR DATABASE-AS-A-SERVICE IN THE CLOUD

WONG JIK SOON

UNIVERSITI SAINS MALAYSIA

2015

SERVICE LEVEL AGREEMENT-BASED MIGRATION FOR DATABASE-AS-A-SERVICE IN THE CLOUD

By

WONG JIK SOON

Thesis submitted in fulfillment of the requirements for Degree of Master of Science

October 2015

ACKNOWLEDGEMENTS

In the lengthy period of pursuing master, I have gained a lot of experience in both academically and personality. Thus, I would like to express my sincere appreciation to my supervisor, Associate Professor Dr. Chan Huah Yong. He has provided me many constructive and valuable suggestions to my research work. Next, I am thankful for my parents for all the support provided. And I would also like to express my gratitude to the School of Computer Sciences for providing the environment and equipments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Program Interface
CEP	Cost-Efficient database Placement
DB	Database
DBaaS	Database-as-a-Service
DBMS	Database Management System
EC2	Elastic Compute Cloud
KCCA	Kernel Canonical Correlation Analysis
IP	Internet Protocol
NAS	Network Attached Storage
OS	Operating System
PM	Physical Machine
QoS	Quality of Service
RAM	Random Access Memory
RDS	Relational Database Service
RDBMS	Relational Database Management System
SLA	Service Level Agreement
SQL	Structured Query Language
VM	Virtual Machine

PEMINDAHAN BEBAN KERJA BERDASARKAN PERJANJIAN TAHAP PERKHIDMATAN BAGI PENGKOMPUTERAN AWAN UNTUK PANGKALAN DATA SEBAGAI PERKHIDMATAN

ABSTRAK

Dengan kemunculan pengkomperan awan, wujudnya servis yang baru dipanggil sebagai pangkalan data sebagai perkhidmatan bagi pengkomputeran awan. Pangakalan data sebagai perkhidmatan memindahkan beban pengguna dalam mengurus pelayan pangkalan data kepada pengendali pengkomputeran awan. Pengendali pengkomputeran awan menjana pendapatan daripada langganan pengguna dalam perkhidmatan pangakalan data dengan memenuhi perjanjian tahap perkhidmatan. Demi memperolehi pendapatan yang maksimum, nombor pelayan yang miminal diperlukan untuk memproses beban kerja yang maksimum supaya kos untuk mengendali pelayan pangkalan data boleh dikurangkan. Demi mengatasi masalah-masalah yang disebutkan, satu sistem pemantauan telah dicadangkan dalam tesis ini. Sistem pemantauan ini mampu mengesan beban kerja pangkalan data yang gagal memuaskan tahap perkhidmatan yang dijanjikan. Optimasi akan diaktifkan dengan memindah beban kerja yang terjejas kepada pelayan pangkalan data yang mampu mengendalinya. Sistem pemantauan ini mampu memilih pelayan pangkalan data yang paling sesuai berdasarkan profil beban kerja dan profil pelayan yang dijanakan oleh sistem tersebut. Hasil eksperimen menunjukkan bahawa sistem pemantauan ini boleh mengesan beban kerja yang tidah memenuhi perjanjian tahap perkhidmatan dan memindah beban kerja yang dijejaskan. Kecekapan sistem dalam penempatan beban kerja dibuktikan boleh mengimbankan penggunaan sumber dan mengurangkan nombor kerja yang telah ditangguhkan masa waktu puncak.

Tambahan pula, sistem tersebut boleh mengabungkan beban-beban kerja ke dalam jumlah pelayan pengkalan data yang lebih kecil tanpa mencabuli perjanjian tahap perkhidmatan.

SERVICE LEVEL AGREEMENT-BASED MIGRATION FOR DATABASE-AS-A-SERVICE IN THE CLOUD

ABSTRACT

With the emergence of cloud computing, comes a new kind of cloud service namely Database-as-a-Service. Database-as-a-Service removes users the hassle to own their own database servers. The responsibility of maintaining and monitoring the database workloads thus transferred to the Database-as-a-Service cloud operator. Service operator generates income through the database service they have provided by meeting the service level agreement (SLA). In order to maximize the income generated, a cloud service operator needs to utilize the least amount of machines to host the maximum amount of database workloads. In this regards, this thesis proposes a model which monitors the database workloads performance and server resources utilization. Database workloads migration will be triggered when the model detects any database workload that has failed to meet SLA agreement. The model is able to choose a server which serves the database workload better based on the workload profile and server profile generated by the monitoring module. Experimental results proof that the algorithm is able to detect workloads that violated SLA and migrate the affected workloads. The efficiency of decision making in workload placement is proofed to balance the resource utilization and reduce pending jobs during peak hour. Additionally, the proposed model can consolidate multiple database workloads into fewer machines without having the workloads violate SLA in the new environment.

CHAPTER 1

INTRODUCTION

1.1 Overview

Cloud computing has gained momentum since its introduction. Cloud computing is popular with its on-demand characteristic and pay-as-you-go pricing model. With cloud computing, users are leaving the trouble of maintaining hardware and software to the service providers while putting more focus on the business logic. Without the upfront cost of procuring hardware and software, cloud computing lowers the barrier for business startup. A cloud database is a database-as-a-service provided by cloud service providers, such as Amazon Relational Database Service (RDS) and Microsoft SQL Azure. Database-as-a-Service hosts databases software in the cloud environment and provides database features such as data definition, storage and retrieval, on a subscription basis over the Internet. (Mateljan, Cisic & Ogrizovic, 2000) The database software used is different for respective service providers, Microsoft SQL Azure offers SQL Server with extended capabilities while Amazon RDS supports MySQL, Oracle and SQL Server.

Database-as-a-Service usually has a multitenancy architecture. Server resources are shared among other users: i) A physical server usually hosts multiple logical databases, each with their own SLA. ii) Virtual machine is created for each user; each virtual machine has a database instance preinstalled; user uses the virtual machine just like a normal operating system. In order for service provider to maximize revenue gain while minimize the number of resources used, an optimization strategy is needed to consolidate user’s workload into proper resource.

This work uses database workload profiling strategy to consolidate database workloads into fewer machines without violating Service Level Agreement (SLA). If database workload migration is initiated, this strategy is able to choose the ideal machine to fit the workload requirement where imminent migration is avoided.

1.2 Background Information

There are several types of service models in cloud computing as shown in Figure 1.1: Infrastructure-as-a-Service (Iaas), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

Software-as-a-Service
Platform-as-a-Service
Infrastructure-as-a-Service

Figure 1.1: Cloud Computing Service Models

Infrastructure-as-a-Service (IaaS) is designed to augment or replace the functions of an entire data center (Zhang, Cheng & Boutaba, 2010). This service reduces the maintenance cost and eliminates the upfront cost for enterprise to purchase hardware. However, the process of configuration, integration and management must be performed through internet.

Platform-as-a-Service provides (PaaS) Application Program Interface (API) or development platforms to create and run applications in the cloud (Zhang, Cheng & Boutaba, 2010). Users are able to run existing applications or develop new ones without having to worry about maintaining the operating systems, server hardware, load balancing or computing capacity. The service providers usually provides intuitive user interface for user to monitor their application through web.

Software-as-a-Service (SaaS) eliminates customer concern about application servers, storage, application development and related, common concerns of IT (Zhang, Cheng & Boutaba, 2010). It provides all the functions of a traditional application to many customers through a web browser. Salesforce.com, Google's Gmail and Skype are examples of SaaS.

Database-as-a-Service currently can be divided into two categories: virtual machine image and shared database instance. With the virtual machine approach, users can either upload their own virtual machine image or use prebuilt machine images with preinstalled database, which is optimized in cloud environment. For example, Amazon Elastic Compute Cloud (EC2) provides a wide selection of virtual machine instance types with varying combinations of CPU, memory, storage and networking capacity with Oracle Database 11g Express Edition. With the shared database approach, application owners do not have to install and maintain the database on their own. Instead, the database service provider takes that responsibilities, such as backup and recovery away from the users, and users pay according to the usage. In this environment, a database instance is usually shared among many users. Users only have the visibility and control of their own database. For example, Amazon Relational Database Service (RDS) provides multiple choices relational database for users to choose from. Complex administration process like software patching, backing up databases and enabling point-in-time recovery are managed automatically. Scaling storage and compute resources can be performed by a single API call. Supported databases included MySQL, Oracle Database and Microsoft SQL Server while PostgreSQL will be supported in near future.

1.3 Problem Statement

Having multitenancy nature in cloud database environment, service providers needed a strategy to optimize user workloads among servers. This is needed so that some servers are not overloaded with users' workload while some are underutilized. An overloaded server will affect the server performance to serve its users, which may cause violation of service level agreement (SLA). Database-as-a-Service provider gain revenue by meeting the service level agreement and will be penalized financially if failed to do so. Different databases have different kinds of usage pattern; some may have more users on the daytime while others may achieve peak usage at nighttime. Not only the usage pattern is different across databases; the resources consumed also vary; transactional databases may have disk I/O intensive workloads while analytical database may be CPU I/O intensive workloads. Non optimized cloud environment may have a large proportion of analytical databases grouping together in a server while transactional databases in another server. Such environment is not only inefficient in resource optimization, but also causes resource congestion in that particular server, which in the end affects overall system performance. (Soror et al., 2010)

Migration is a technique used to move either virtual machine or database to another server. It is often employed when a server is overloaded or not optimized. Cloud administrator has to determine which workload, in this case, either virtual machine or database, to be migrated and to which server. One or more workloads have to be migrated in order to free enough resources for the system to operate, without further violating the service level agreement for that server's remaining users. Next, the administrator has to determine which server is capable of handling the to-be migrated workload. If the new host cannot handle the workload well enough, either the workload has to migrate again or another cycle of migration has to be initiated. Migration may also incur additional cost to the system. (Curino et al., 2011)

The main research question for this study is "How to distribute database workloads across cloud servers so that service provider will generate maximum revenue while minimize resource underutilization?"

The sub research questions to be answered are as follows:

- How to accurately monitor the resource usage of each database and to estimate the utilization of a combined set of databases?
- How to choose which databases are to be migrated and to be placed on which hardware?
- How to maximize the revenue generated while minimize hardware underutilization without service level agreement violation?

1.4 Objectives

The objectives of this research are as follows:

- To improve database workload distribution across cloud server while ensuring no service level agreement has been violated.
- To develop a monitoring module that is able to accurately capture the performance and workload arrival rate for each database workload.

1.5 Research Motivation

Exponential growth of Internet data has raised the problems of storage and usability of data. Although only a fraction of enterprises have moved into cloud database solution, there has been a sight of trend that more enterprises will consider cloud based storage solution. The advantage for existing enterprise to migrate data to cloud database is to reduce maintenance cost while for new business startup, it eliminates the upfront cost for purchasing new expensive hardware. As more and more existing data have been shifted and consolidated in cloud storage service providers, the responsibility for them to optimize and maintain these data has grown significantly as well. As cloud computing is still an emerging technology, current cloud based optimization strategy is not mature enough, especially for database management software in cloud.

1.6 Scope of Research

The scope of this research covers the following:

- Review the existing cloud database optimization strategies in order to propose an improved strategy that enables better database workload distribution across cloud servers.
- Analyze the resource consumptions for database workloads when combining different types of queries.
- Primary focus is on improving the performance of database performance and load balancing of servers in cloud environment

1.7 Thesis Organization

This thesis consists of five chapters namely Introduction, Literature Review, Research Methodology, Implementation of the Proposed Algorithm and Experiment Result, as well as Conclusion and Future Work.

In this chapter, the overall idea of this thesis that includes the general introductions and background of our research work are presented. In the next chapter, literature review covers the overview of Database-as-a-Service (DBaaS). The several types of multi-tenancy model used by DBaaS providers are described. The different types of Service Level Agreement (SLA) that are used by DBaaS providers are elaborated. Meanwhile, the study of other researchers is described in related work.

In chapter 3, an optimization method which depends on the monitoring data is presented to solve the research problem of this research work. Additionally, the details of the method are elaborated in this chapter as it is the core of this thesis. The workflow of the optimization method is also illustrated with figures. At the end of the chapter, the workflow of migration strategy is also explained in detail.

In chapter 4, the design of the implemented optimization method is presented. The hardware specification for the machines used in the experiment is listed. The software needed in the implementation is also discussed. The pseudo code of the migration strategy is also included in this chapter. Written further in this chapter are the testing scenario and evaluation results for the proposed optimization method. The setup of the experiments and results are discussed in details.

In the last chapter, the research work is concluded and the future direction of how the research can be carried is also discussed.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter presents the literature review of this research. The definition of Database-as-a-Service (DBaaS) is described, followed by the multi-tenancy model used in DBaaS in order to serve multiple database workloads with fewer machines. Service level agreement (SLA) established between DBaaS providers and their customers are explained. Subsequently, the problem of this research is defined. The last part of this literature review summaries the related work done by recent researchers on solving the optimization issues in DBaaS.

2.2 Database-as-a-Service (DBaaS)

Database-as-a-Service is a service that is managed by a cloud operator that supports applications which utilize database management system, without the application team assuming the responsibility for traditional database administration functions (ScaleDB, 2015). DBaaS subscribers do not need to be a database expert nor have to hire a database administrator to maintain the database. Maintaining, upgrading, backing-up and handling server failure will be the DBaaS operator’s responsibilities. From the perspective of subscribers, that is the definition of DBaaS. However, DBaaS subscribers also carry some risks by having their database management system being hosted by another company which may be located in different geographic locations. These risks are additional overhead of remote access to data, an infrastructure to guarantee data privacy and user interface design for such a service (Hacıg ̈um ̈us, Iyer & Mehrotra, 2002). DBaaS subscribers can mitigate the privacy risk of their DBMS application by employing data encryption for their system. For example, CryptDB is a set of techniques designed to provide privacy, it employs different encryption levels for different types of data, based on the types of queries that user run. Queries are evaluated on the encrypted data, and sent back to the client for final decryption (Bijwe & Ramteke, 2015).

Managing large systems poses significant challenges in monitoring and managing system operation. In order to provide a complete DBaaS solution across a large number of customers, the DBaaS operators need a high-degree of automation (Oracle White Paper, 2013). Operation such as database backup can be scheduled in a regular-time based interval while other functions such as elastic scale out can be automated based on certain business rules. Meeting certain quality of services (QoS) according to the service level agreement (SLA), will require DBaaS to automatically shifting database workloads around servers or adding a new server to share the load. As SLA violation has been detected, there should be some kind of mechanisms that respond to the event automatically.

Database scalability in the cloud has been a growing concern since conventional DBMSs and RDBMSs are not easily scalable. Adding new server to RDMBSs does not guarantee improved performance or throughput. Current DBaaS that offers RDBMS as backend database engine support database scalability by encapsulating RDBMS into VM and replicate the VM to scale out during sudden peak of workload. NoSQL database or “Key-Value-Store” system is also being used as an alternative to relational database management system (RDBMS) for DBaaS backend engine (Chandra, Prakash & Lamdharia, 2012). NoSQL database offer better scalability and availability than traditional RDBMS. However, it lacks the database schema which many developers are already used to, in returns NoSQL stores data without explicit and structured mechanisms to link data together (Padhy, Patra & Satapathy, 2011). On a basis level, there are three core categories of NoSQL databases: Key-value Stores, Column-oriented Databases and Document-based Stores.

2.3 DBaaS Muti-Tenancy Model

Multi-tenancy is an optimization method for DBaaS in which multiple database workloads owned by different subscribers are consolidated into a server. Multi-tenancy allows pooling of server resources, which improve utilization by eliminating the need to provision each subscriber for their maximum load (Jacobs & Aulbach, 2007). There are three approaches to multi-tenancy database implementation: shared machine, shared process and shared table.

In the shared machine approach, each database workload gets its own database process and multiple database workloads shared the server's resources. Utilizing virtual machine (VM) is under this approach, where each database workload in placed in its own virtual machine, and the virtual machines are hosted in a server. Using VM per tenant can leverage VM migration technique for elastic load balancing. However, experiment studies have shown that stronger isolation comes at the cost of increased overhead during normal operation resulting in inefficient resource sharing between tenants (Curino et al., 2011). This overhead due to redundant components among VMs, specifically the operating system (OS) which is installed in every VMs. On average, each OS installation in a VM consumes around 2GB of hard disk space. The various server resources' utilization also needs to be taken into consideration in a shared machine approach that uses VM per tenant approach. The shortcomings of this approach can be significantly reduced with shared process model.

Shared process approach allows independent schemas for tenants while sharing the database process amongst multiple tenants. This approach provides better isolation compared to shared table approach while allowing effective sharing and consolidation of multiple tenants in the same database process (Elmore et al., 2011). However, this approach cannot leverage on VM migration to move individual tenants from a shared database process. Tenants' migration is done by moving database data from one database process to another. Since there is only one database process in this approach, subscribers can share connection pools. Data security has to be handled at the application layer to prevent one subscriber to access another subscriber's data.

Shared table approach is primarily used in the context of Software-as-a-Service (SaaS) such as in Saleforce.com (Weissman & Bobrowski, 2009). All tenant's data is stored in a few big tables to provide row level isolation between tenants. This approach is the best compared to the other two at pooling resources. The ability to scale up is limited only by the number of rows the database can hold. However, data migration is more complex than the other two approaches. It requires executing queries against the operating system as data is intermingled from multiple tenants.

In this study, shared process approach is focused as it is widely adopted by many database management systems, such as MySQL and PostgreSQL. Experiment implementation is easier compared to shared table approach and shared machine approach.

2.4 Service Level Agreement (SLA)

A service level agreement (SLA) is a contract between a service provider and its customers. SLA capture the quality of service (QoS) guaranteed by the service provider and accepted by its customers (Sakr & Liu, 2012). Cloud service provider gains revenue by satisfying the quality of service guaranteed as stated in SLA. If failing to provide its customer with the guaranteed service, cloud service provider will be penalized financially. Not only does cloud service provider gain less revenue, its service reputation will be affected, as future customers prefer reliable service compared to the financial penalty the customer is received if SLA is violated. For example, Google Cloud Platform and Amazon RDS define its SLA by service availability and a percentage of monthly bill is refunded to its subscribers once the availability is reduced under 99.9% (Amazon RDS, 2015; Google Cloud Platform, 2015). In DBaaS, providing QoS according to SLA may require limiting databases to a certain number of connections or peak level of CPU utilization, or some other criteria. When this criterion is exceeded, the DBaaS might automatically add a new database instance to share the load. DBaaS provider needs a highly automated and intelligent system to ensure that all tenants' SLAs have been satisfied and triggers appropriate actions to mitigate the risks if SLA has been violated.

SLA can be defined in terms of various criteria, such as service latency, throughput, consistency, security, and etc. Service latency is the focus of this study. Even with latency alone, there can be multiple specification methods: Mean-value-based SLA (MV-SLA), Tail-distribution-based SLA (TD-SLA), Individual-job-based (IJ-SLA), Hard SLA and Soft SLA (Moon, Chi & Hacigümüs, 2010). Table 2.1 summarizes different categories of SLA and descriptions.

Table 2.1: SLA Categories and Descriptions

SLA	Description
MV-SLA	QOS is measured based on mean response time for each job class. This is least robust type of SLAs from the customers' perspective.
TD-SLA	QOS is measured based in terms of the portion of jobs finished by a given deadline. For example: a user may want 99% of job to be finished within 100ms.
IJ-SLA	QOS is measured using the response time of individual jobs. Any single job with a poor service quality immediately affects the measured quality of service and incurs some SLA penalty cost.
Hard SLA	It has a single hard deadline to meet, violation occurs when the deadline is missed. The violation of hard SLA may not correspond to financial penalty in the client contracts.
Soft SLA	It corresponds to agreed level of service in the contract. Different from Hard SLA, the penalty cost may continue to increase as response time increases.

This work adopted the IJ-SLA as it is believed to be widely accepted from both customer and cloud provider perspective, such as Rackspace Cloud Databases (Rackspace Cloud Databases, 2014) and Microsoft Azure (Microsoft Azure, 2014).

2.5 Research Problem Description

Since workload peaks tend to be transient in nature, a prior static provisioning for the peak result in substantial waste of resources sit unused at non peak duration. Typical server utilization in real data centers has been estimated at only 5-20% (Armburst et al., 2010). This is where cloud computing's elasticity comes in, the ability of the cloud platform to adjust an application's resource allocation on the fly when responding to long term workload growth or sudden load spikes. However, a resource allocation strategy is need to optimize database workload distribution across cloud servers while ensuring that neither SLA has been violated nor database performance has been compromised.

In order to achieve that purpose, DBaaS provider needs to identify which database workload is violating the SLA. Next, DBaaS provider needs to decide that which server should the affected database workload to be migrated. Finally, when DBaaS subscribers create new database workloads, DBaaS provider needs to decide which servers to host the newly created database workloads, so that the placement of new database workloads do not affect the performance of existing database workloads. For DBaaS provider to maintain hundreds to thousands of servers at once, a highly automated system is required. For the system to function properly and accurately, it requires inputs of servers' resource utilization and database workloads performance. Hence, a data monitoring model which accurately records both parameters is required (Mozafari, Curino & Madden, 2013). Server resources utilization to be monitored should include CPU, RAM and Disk I/O, as database workload performance is heavily dependent on those resources. The database workload's turnaround time and arrival rate require monitoring in order to assist the system to manage DBaaS subscribers' workloads.

This research work problem can be refined into: decision making for the placement of new database workload, database workload SLA monitoring, and database workload migration.

2.5.1 Decision Making for the Placement of New Database Workload

The research question to ask is which physical machine should host the new database workload when a new client instantiates it for the first time. Powell et al. show that when a good selection is employed, the database performance will increase (Powell et al., 2000). Among all cloud compute nodes, which physical machine is most suitable to host the new workload, should the new workload be hosted on an idle physical machine or on a not fully occupied physical machine. The amount of server resource which the database workload needed should also be taken into consideration during decision making process. Placing a database workload in a server which could not handle the workload will cause performance degradation across all hosted workloads. This in turn will trigger a series of database workload migration overhead as the database placement does not considering the workload's resource requirements in the first place.

2.5.2 Database Workload SLA Monitoring

In order to minimize the financial penalty of a DBaaS provider for SLA violation, a method to accurately monitoring each database workload performance and SLA is crucially required. The SLA requirement maybe different for every database workloads which are hosted within a server. Some database workloads have stricter SLA objectives and higher financial penalty than the others, which prompts higher attention should an SLA violation occurs. The monitoring method should trigger fast and proper response to amend the situation, such as migrating affected database workload to a proper server. Not only does the performance of database workload require monitoring, server resources such as CPU, RAM and Hard Disk utilization requires monitoring as well. As depleting any of server resources just mentioned, will cause database performance degradation, which will lead to multiple SLA violation for the affected server.

2.5.3 Database Workload Migration

When a server's resource is depleting or the performance of a certain workload is dropping, migrating some database workloads from over occupied physical machine to less occupied physical machine should be a better decision. The decision making process of migrating a database workload from a server to another requires information about the resource requirement of the workload and the resource utilization of the servers. Without this information, database workload maybe wrongly placed to server without the resource capability.

Database workload migration is useful in achieving overall database performance and server utilization; for example: A server is hosting two logical databases with heavy workload for a long period while another server is hosting two logical database which are idle for most of the time. In this situation, the database workloads in the first server are competing for resources which lead to decreased overall performance, while the second server is not fully utilized for most of the time. It is better if one of the heavy duty database workload is swapped with one of the idle database workload, in order to achieve overall improvement.

2.6 Related Work

Cloud computing has been studied with goals such as quality of service awareness, performance and higher resource utilization. Most of the work is done on the optimization of resource allocation. For example, dynamic VM resources allocation among tenants (Aboulnaga et al, 2009; Soror et al., 2010) and database workloads consolidation (Curino et al., 2011; Yu et al., 2012). The purpose of optimizing resources allocation among VM tenants is to improve resource utilization and workload performance. The optimization method focus on shifting the physical resources shared among VM tenants to the tenants which suffer from performance degradation. Database workloads consolidation aims to solve the problem of server underutilization by consolidating workloads from underutilized servers into fewer machines. However, neither the research works of both resource allocation optimization research and database workloads consolidation aim to minimize the SLA violation penalties. Besides the optimization of resource allocation, extensive researches have been done on live migration (Das et al., 2011; Elmore et al., 2011). With live migration technology enabled, DBMS can continue to serve users' requests while database migration is in progress. The objective of live migration research is to minimize DBMS downtime during database migration. Although live migration technology able to minimize service downtime and thus mitigate the risk of SLA violation, the risk of unnecessary database workloads migration still presents as the aspect of database workloads arrival rate and computation power need to be taken into consideration.

In order to generate maximum revenue, DBaaS providers need to satisfy their tenants' SLA requirements to prevent financial penalty. Although there are multiple optimization approaches to satisfy tenants' SLA requirements, they all share a common need for accurate data models of database workloads performance and server resource utilization (Liu et al., 2013; Mozafari, Curino & Madden, 2013; Sakr & Liu, 2012; Xiong et al., 2011). For example, Ganapathi et al. have developed a system that predicts a database query execution time ranging from milliseconds to hours (Ganapathi et al., 2009). With this technique, they are able to identify both the short and long-running queries to inform workload management and capacity planning.

2.6.1 Resource Allocation Optimization

Aboulnaga and his team try to optimize database performance in the cloud by partitioning CPU capacity among virtual machines (Aboulnaga et al., 2009). Aboulnaga et al. use white box model and black box model as its cost model to predict the database performance. White box model is based on the internal knowledge of the database system and black box model is statistical model based on external, empirical observations of the database system's performance. The authors focus on optimizing CPU resource allocation within a physical machine. On the other hand, this thesis not only considers optimization within a physical machine but also multiple physical machines where each containing multiple tenants.

The problem of automatically configuring CPU resource allocation to multiple virtual machines that are all running database systems and sharing a pool of physical resources is considered by Soror and his group (Soror et al., 2010). Soror et al. rely on the cost model of the database query optimizers as its cost model to predict the workload performance under different resource allocations. The authors' work is sensitive to workload resource needs, for example, the system will not allocate more CPU resource to a virtual machine if it does not decrease total execution time. In some cases, the query optimizer cost model is inaccurate, thus affecting the resource allocation decisions. The authors correct the optimizer error through their online refinement approach to fine tune the resource allocation decisions.

Kairos, a database consolidation system is presented by Curino and his team. It uses a technique to measure the hardware requirements of database workloads, as well as models to predict the combined resource utilization of those workloads (Curino et al., 2011). As database management system is usually operated on a dedicated server, the system is configured to use all available server resources. This caused the amount of memory provisioned far exceeds the actual working set at any point in time. They presented a technique called buffer pool gauging to estimate the exact amount of memory the current database working set is utilizing. Experiment has been conducted to compare the DB-in-VM approach versus the shared process approach, and shows that their approach provides between 6x to 12x higher throughput. Their work also shows that past workload behavior is also a good predictor of its future behavior.

Cost-Efficient database Placement (CEP) algorithm is proposed by Yu and his team to help DBaaS providers to achieve effective resource allocation among multiple databases, minimize the disturbance to the system caused by the database migration, and maximize Cloud resource utilization (Yu et al., 2012). Database workloads consolidation can be achieved through Virtual Machine-in-Physical Machine (VM-in-PM) or Database-in-Node (DB-in-Node) approach. In this paper, the authors assume DB-in-Node approach is adopted and focus on deciding when and where to place or migrate the database workloads to which physical machine. Results show that CEP algorithm achieves improvement in reducing the number of nodes needed to support database requests. However, their work does not take into consideration of the resource consumption pattern of each individual database workloads.

2.6.2 Live Migration

Albatross, a live migration technique for shared storage database for the cloud is presented by Das and his team (Das et al., 2011). In shared storage architecture, the persistent data of a database is stored in network attached storage (NAS), thus there is no need for database data migration. Instead, Albatross migrates the database cache to allow the database workload to start "warm" at the new host, minimizing the impact to transaction latency and minimizing the unavailability window as low as 300ms. This can be done by copying the state iteratively while the source continues to serve transactions. Changes made to the source are also copied to the destination node during migration. After the destination node is synchronized with the source node, the source node stops serving requests while the destination node resumes transactions execution.

Zephyr is presented by Elmore and his team. It is a live migration technique in shared nothing architecture for databases in cloud environment (Elmore et al., 2011). Zephyr uses a combination of on-demand pull and asynchronous push to migrate a tenant with minimal service interruption. Compared to traditional stop and copy database migration technique, this work minimize the number of failed SQL operations during migration while also reducing the amount of data transferred. However, it takes more time to complete the migration and extra efforts are needed to ensure data correctness and synchronization during migration.

2.6.3 SLA Management

The problem of virtual resource management for database systems in cloud environments is tackled by Xiong and his team with SmartSLA, a system modeling module that uses machine learning technique to learn the profit gain for each client under different resource allocations (Xiong et al., 2011). Based on the learned model, CPU, memory and number of database replicas will be dynamically adjusted in order to reduce the SLA penalty cost. The study uses linear regression model to present the relationship between resource allocations and SLA penalty cost. However, linear regression model perform poorly as resource allocations do not affect the output in a linear fashion. Hence, they uses regression tree model and boosting approach to further improve the accuracy of the prediction model, however the model still incur around 30% of prediction error.

Sakr and Liu present a framework that adaptive provisioning database replicas of the software applications based on application-defined policies for satisfying SLA performance requirements, avoiding the SLA violation and controlling the monetary cost of the allocated computing resources (Sakr & Liu, 2012). Their approach focuses on the perspective of service providers that utilizes cloud database services. While Sakr and Liu focus on the SLA metric of transaction execution time, other metrics such as CPU utilization, maximum number of database replicas and data freshness can also be implemented and integrated in the same manner.

SLA profit-aware solution for database tenant placement from the perspective of service provider is proposed by Liu and his team (Liu et al., 2013). Their implementation adopts the private table scheme, where multiple tenants share a database while each tenant uses its own set of tables. Two approximation algorithms are provided, one for a special case and one for the general case. A Dynamic Programming (DP) procedure is further provided that can be used to couple with the approximation algorithm for better performance. DP is experimentally shown to outperform the baseline and the approximation algorithms by a large margin. However, the processing time for DP increase with O(n^2m) complexity where number of tenants and servers, whereas the other two algorithms are highly efficient. Although their work is similar to this research work, they focus on placing multiple database tenants at once while this research work places database tenants based on their workload arrival rate and SLA requirements.

A machine learning technique called Kernel Canonical Correlation Analysis (KCCA) is used by Ganapathi and his team to predict metrics such as elapsed time, records used, disk I/Os, etc accurately, for both short and long running queries (Ganapathi et al., 2009). The prediction result can be used as the cost model for optimization decision. Prediction of a single query can be done under a second, which makes it practical for queries that take minutes or hours to run. The authors show that an accurate prediction model can improve the effectiveness of critical tasks including system sizing, capacity planning and workload management. Table 2.2 presents the summary of existing works.

Table 2.2: Summary Table of Existing Works

Source	Study Description	Optimization Method
VM Dynamic Resource Allocation		
Aboulnaga et al., 2009	Performance optimization for database applications deployed in the clouds, which are virtual machines with pre-installed pre-configured database systems.	Use the cost model of the database system's query optimizer to determine the best partitioning of CPU capacity among the VMs which are hosted in a physical machine.
Soror et al., 2010	Consider a common resource consolidation scenario in which several DBMS instances, each running in a separate VM, are sharing a common pool of physical computing resources. Address the problem of performance optimization in such scenario by controlling the VM configurations.	Virtualization design advisor that takes information about different database workloads is implemented. The advisor uses this information to determine how to split the available physical computing resources among the VM. The advisor relies on the cost models of DBMS query optimizers to enable it to predict workload performance under different resource allocations. Techniques that use actual performance measurements to correct the cost model inaccuracies are also presented. A dynamic resource reallocation scheme that use runtime information to react to dynamic changes in the workloads.
Database Replica Provisioning		
Xiong et al., 2011	Intelligently manage database replicas among various clients in a cost-effective manner while satisfying the client SLA.	Present SmartSLA, a cost-aware resource management system. It consists of two main components: the system modeling module and resource allocation module. The system modeling module uses machine learning techniques to learn a model that describes the potential profit margins for each client under different resource allocations. Based on the learned model, the resource allocation decision module dynamically adjusts the number of database replicas in order to achieve optimum profits. Experimental results show that SmartSLA can provide intelligent service differentiation according to factors such as variable workloads, SLA level, resource costs, and deliver improved profit margins.