Data Migration: Connecting Databases in the Cloud

Farah Habib Chanchary
Department of Computer Science
Najran University
Najran, Kingdom of Saudi Arabia
chanchary@gmail.com

Samiul Islam
Department of Computer Science
Najran University
Najran, Kingdom of Saudi Arabia
samiul80@gmail.com

Abstract—The introduction of new technologies, architectures and systems or organizational restructuring often requires a company to reorganize its databases either by transferring data from old applications to the new ones, or reorganize data in an existing application according to the new organizational structure. Migration of data, services, and processes to the Cloud platform must be achieved based on well-defined models/strategies. Every migration model involves specific objectives, according to the policies of the organization, the control, and security of information. It must be achieved by keeping the optimal balance between data accuracy, speed of migration, time of non-operation and minimal costs. The challenge in all these cases lies in guaranteeing complete and accurate migration of all data, in other words, ensuring that no data is lost, placed in the wrong location or altered in the course of the migration process. The main vision of this work is to integrate data management functionality for large organization and also to provide outsourcing data management to a cloud based service provider for smaller organizations. Users should have access to database functionality without worrying about provisioning hardware and configuring software, while providers should be able to manage several databases without dedicating hardware and administrators to each database. A design has been proposed, which hosts multiple data centers in different geographical region where each data center may hold multiple databases on a pool of commodity servers.

Keywords—Cloud database; DBMS; data migration; Universal DBMS platform; Universal data modeling platform; database partitioning.

I. INTRODUCTION

Cloud computing is an elevated form of grid computing, parallel or distributed computing which evolve to accelerate the idea of sharing resources expeditiously by offering use and access of multiple server-based computational resources via digital networks where user may access the server resources using any kind of computing devices. In a cloud environment, on demand services provided to user that categorized as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Cloud application services or "Software as a Service (SaaS)" deliver software as a service over the Internet, eliminating the need to install and run the application on the customer’s own computers and simplifying maintenance and support. Cloud platform services, also known as Platform as a Service (PaaS), deliver a computing platform and/or solution stack as a service, often consuming cloud infrastructure and sustaining cloud applications. It facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers.

Cloud infrastructure services, also known as Infrastructure as a Service (IaaS), deliver computer infrastructure – typically a platform virtualization environment – as a service. Rather than purchasing servers, software, data-center space or network equipment, clients instead buy those resources as a fully outsourced service [1]. These categories sometimes extend to initiate another sub class that referred to as Database as a Service (DaaS) depends on organizational requirements in cloud environments where IaaS can be extended to provide multiple dedicate database storages, SaaS can widen to support specific database interfaces among users and PaaS can be expanded to allow database management service tools.

This new paradigm for data management in which a third party service provider hosts “database as a service” providing its customers seamless mechanisms to create, store, and access their databases at the host site. Such a model alleviates the need for organizations to purchase expensive hardware and software, deal with software upgrades, and hire professionals for administrative and maintenance tasks which are taken over by the service provider [2].

II. CLOUD DATABASE: REQUIREMENTS AND CHALLENGES

To make database as a service as a renowned cloud service, requirements have been observed from different aspects those can be parted mainly in two domains such as user centric requirements and provider centric requirements. As parts of the first domain, user needs handy application programming interfaces (API) with less governance and maintenance, which can provide high performance in terms of throughput, scalability, latency, availability and reliability whereas the second domain focused on workload handling, effective resource allocation, data security, cost benefit analysis, pricing schemes and the arrangements to meet user service level agreements. Most of the requirements in second domain have an influence over their fixed infrastructures which motivates the providers to design cost effective cloud storage infrastructures and services. Cloud storage service refers to the act of storing data in remote databases those
maintained by third party service providers, instead of storing in local storage devices.

There are hundreds of different cloud storage systems. Some have a very specific focus, such as storing Web e-mail messages or digital pictures. Others are available to store all forms of digital data. The facilities that house cloud storage systems are called data centers. At its most basic level, a cloud storage system needs just one data server connected to the Internet. A client (e.g., a computer user subscribing to a cloud storage service) sends copies of files over the Internet to the data server, which then records the information. When the client wishes to retrieve the information, he or she accesses the data server through a Web-based interface. The server then either sends the files back to the client or allows the client to access and manipulate the files on the server itself. Cloud storage systems generally rely on hundreds of data servers. Because computers occasionally require maintenance or repair, it's important to store the same information on multiple machines. This introduces redundancy. Without redundancy, a cloud storage system couldn't ensure clients that they could access their information at any given time. Most systems store the same data on servers that use different power supplies. That way, clients can access their data even if one power supply fails [4]. Challenges like effective resource utilization, big data handling (handling Exabyte or Zettabyte of data per second), data privacy, faster transition are also key issues in cloud database.

III. CLOUD DATA MIGRATION

Enterprises need to support hundreds or even thousands of applications to meet growing business demands and this growth has driven up the cost of running and managing the databases under those applications [3]. To meet these challenges, cloud service providers' unleashed different DaaS models those differ from structural points, to the end user services. Among the major players, Salesforce.com launched Database.com that hosted relational database service; Microsoft offers SQL Azure Database as a standalone service; Amazon Web Services has its own Not Only Structured Query Language (NoSQL) cloud database service called SimpleDB and based on MySQL it offers Elastic database service; and Google offering Google AppEngine Data Store based on relational cloud model.

The data migration process can take place in different forms (figure 1) that includes, from organization to cloud databases or cloud databases to the organization, and migration between multiple cloud storages. Several issues have been revisited by the vendors and researchers to make these migration processes worthy according to the organizational needs but in few factors all consumers are rigid to compromise that deal with data accuracy, data privacy and data storing/retrieval time. On focusing these issues, we tried to propose our data migration model for large organizations to manipulate their data efficiently. Recent developments in cloud data migration techniques induce a pattern of relational cloud to provide database as a service in more promising approach. Our proposed work included some of the key features of relational cloud that summarized in section ahead.

IV. LITERATURE REVIEW

With the advent of developing and marketing cloud databases it becomes essential to study the performance (response time) and cost trade-offs of different service providers. Brantner et. al. presented their work [6] which demonstrates the opportunities and limitations of using cloud computing as an infrastructure for general-purpose Web-based database applications. This study also includes alternative consistency protocols in order to build database services on top cloud storage services, alternative client-server and indexing architectures. Many similar studies are also available such as [7] and [8]. Reference [7] describes how changes in the underlying technologies, combined with the economics of cloud computing, reframe the idea of information sharing on cloud platform, whereas J. Varia focuses on highlighting concepts, principles and best practices in creating new cloud applications or migrating existing applications to the cloud [8].

To deal with theft of private information, Popa et. al. presented CryptDB [9] which provides confidentiality against attacks on sensitive online information for applications backed by SQL databases. It works by executing SQL queries over encrypted data using a collection of SQL-aware encryption schemes where encryption keys are chained to user passwords, so that a data item can be decrypted only by using the password of one of the users with access to that data.

In 2010, reference [14] introduces relational cloud which is a transactional database that can be offered as a service to the customer with lower costs. Authors claimed that three important challenges, efficient multi-tenancy, elastic scalability, and database privacy are taken care off in this paper. In 2011 same authors presented a prototype that exploits multiple dedicated storage engines, provides high-availability via transparent replication, supports automatic workload partitioning and live data migration, and provides serializable distributed transactions [5].

Cloud service performance can be disturbed due to congested network traffic. This can be solved by migrating services to different platforms such that the communication cost can be minimized. In 2009, Hao and Yen develops a framework to facilitate service migration and design a cost model and the decision algorithm to determine the tradeoffs on service selection and migration [10]. B. Krishna et. al. presented a scalable data migration service that employs automated schema matching techniques to handle the schema disparities in cloud and enterprise data models, elastic grid infrastructure for on-demand data access, rules driven
based data partitioning algorithm to achieve near-linear performance than existing approaches; ii) the use of a graph-partitions when the load on a database exceeds the capacity of balance load, migrate the partitions as needed without causing periodically determine the best way to: (1) partition each database into one or more pieces, producing multiple partitions when the load on a database exceeds the capacity of a single machine, (2) place the database partitions on the back-end machines to both minimize the number of machines and balance load, migrate the partitions as needed without causing downtime, and replicate the data for availability, and (3) secure the data and process the queries so that they can run on untrusted back-ends over encrypted data [5]. Our proposed prototype comprises encrypted query handling process and data partitioning process and these two algorithmic themes are chartered from relational cloud.

V. RELATIONAL CLOUD

Reference [5] mentioned the key technical features of Relational Cloud as: i) a workload-aware approach to multi-tenancy that identifies the workloads that can be co-located on a database server, achieving higher consolidation and better performance than existing approaches; ii) the use of a graph-based data partitioning algorithm to achieve near-linear elastic scale-out even for complex transactional workloads; and iii) an adjustable security scheme that enables SQL queries to run over encrypted data, including ordering operations, aggregates, and joins. An underlying theme in the design of the components of Relational Cloud is the notion of workload awareness: by monitoring query patterns and data accesses, the system obtains information useful for various optimization and security functions, reducing the configuration effort for users and operators.

Relational Cloud system architecture is shown in Figure 2, which shows that applications communicate with Relational Cloud using a standard connectivity layer such as Java Database Connectivity (JDBC). They communicate with the Relational Cloud front-end using a special driver that ensures their data is kept private (e.g., cannot be read by the database administrator). When the front-end receives SQL statements from clients, it consults the router, which analyzes each SQL statement and uses its metadata to determine the execution nodes and plan. The front-end coordinates multi-node transactions, produces a distributed execution plan, and handles fail-over. It also provides a degree of performance isolation by controlling the rate at which queries from different tenants are dispatched. The front-end monitors the access patterns induced by the workloads and the load on the database servers. Relational Cloud uses this information to periodically determine the best way to: (1) partition each database into one or more pieces, producing multiple partitions when the load on a database exceeds the capacity of a single machine, (2) place the database partitions on the back-end machines to both minimize the number of machines and balance load, migrate the partitions as needed without causing downtime, and replicate the data for availability, and (3) secure the data and process the queries so that they can run on untrusted back-ends over encrypted data [5]. Our proposed prototype comprises encrypted query handling process and data partitioning process and these two algorithmic themes are chartered from relational cloud.

VI. PROPOSED DESIGN

This model best fits for data migration of a small or large organization bounded in a single geographical area but challenges arise for the migration process of data of a large organization that incorporates many offices in different geographical locations having their own data centers to store different data formats. According to the organization policies, data formats and query processing may differ from one location to another and manipulating data between such these locations is a key issue for any large organization that must not be overlooked. To handle such scenarios, we introduce our research prototype that is discussed in the next section.

A. Encryption Enabled Request Handler

Encryption enabled request handler fetch user queries and synchronizes with user's API, launch key generation algorithm to produce cipher queries. The idea behind cipher query request is to provide maximum level of data privacy to the user or to the organizational data. Before handling each cipher query, the request handler acquire the corresponding data location map from local conceptual mapping DB to determine the location of desired data that may store in either on-premises storages or in different location site.

B. Local Conceptual Mapping Database

Local conceptual mapping DB holds the address map of the available onsite storages and their data types using a data
tracker protocol. After every query compilation, data tracker will update information about the data regarding data movement or upgrade and store it in local conceptual DB. Key idea of this algorithm is to provide minimize response time to the user by shortening data retrieval and storing time.

C. Workload Splitter

An organization with thousand of database servers often could have a scenario like 70% of those servers are underutilized, 20% effectively used and 10% of those servers immensely over utilized. As a result, users suffer from poor performance. We initiate workload splitter in our proposed model expecting to handle such scenario. Workload splitter manages the assigned task send by the request handler with ‘n’ number of dedicated processing nodes attached with it. The workload splitter collects information about those consecrated nodes and uses this information to select receivers.

Focused algorithms:

a) Adaptive load distributing algorithm: Our system uses adaptive load-distributing algorithms, which may considered as a special class of dynamic algorithms. Dynamic load distribution algorithm uses system status information (loads at dedicated processing nodes) at a specific time to make load distribution decision. With adaptive load distribution, our system will be able to adapt its activities by dynamically changing its parameters, or even its policies, to suit the changing system state. For example, if some load distributing policy performs better than others under certain conditions, while another policy performs better under other conditions, a simple adaptive algorithm might choose between these policies based on observations of the system state.

b) Load Indexing: A load index predicts the performance of a task if it is executed at some particular node.

c) Transfer Policy: A transfer policy determines whether a node is in a suitable state to participate in a task transfer, either as a sender or a receiver (e.g. a node might be considered a receiver if its load is among the lowest in the system).

d) Selection Policy: Once the transfer policy decides a node as a sender/receiver, a selection policy selects a task for transfer.

e) Information Policy: The information policy decides when information about the states of other nodes in the system is to be collected, from where it is to be collected, and what information is collected [13].

Figure 3: Proposed Cloud Database Architecture
D. Data Partitioning Manager

We tried to develop a method of database partitioning that reflects an image of relational cloud database partitioning to scale a single database to multiple nodes, useful when the load exceeds the capacity of a single machine, and to enable more granular placement and load balance on the back-end machines compared to placing entire databases. Relational Cloud uses a workload-aware partitioning strategy. The front-end has a component that periodically analyzes query execution traces to identify sets of tuples that are accessed together within individual transactions. The algorithm represents the execution trace as a graph. Each node represents a tuple (or collection of tuples) and an edge is drawn between any two nodes whose tuples are touched within a single transaction [5]. This algorithm partitions the entire database or its selected attributes in distinct parts and spread over multiple data storage nodes to increase performance for sites that have regular transactions involving certain views of data.

E. Centralized Conceptual Database

Centralized conceptual database holds the information about all available local conceptual databases connects to it from different location sites.

F. Universal Platform

Our proposed prototype brings up an idea to support different query formats generated in different location sites. Universal platform refers to the node capable of handling huge bundle of data with its high-speed computational process. Key offerings of this platform include the followings:

a) Universal DBMS Support Platform: Large organizations with several offices in different geographical regions may use different database management systems based on few factors like company policies, types of data, service availabilities, technical knowledge, etc. MySQL, MSSQL(Microsoft Structured Query Language), NoSQL(Not only Structured Query Language) are widely used DBMS systems and they differ from each other in many attributes like scaling, replication, distribution, handling concurrency, managing redundancy, consistency maintenance and many others. As an example, MSSQL uses Object Oriented Database Management System (OODBMS) where main data elements are objects that can be manipulated by the commands of a programming language. On the other hand, NoSQL uses a graph database where main data elements are nodes and edges. MySQL uses Relational Database Management System (RDBMS) and the relational database uses a formal system of predicates to address data. Our proposed prototype brings up an idea to support different query formats generated in different location sites served by universal data modeling platform.

b) Universal Data Modeling Platform: Data models and manipulation methods differ along with the database management systems. DBMSs are designed to use one of five database structures, named as network model, relational model, hierarchical model, multidimensional model, and object model, to provide simplistic access to information stored in databases. The aim of our proposed universal data modeling platform is to allow data handling regardless the database models.

c) Universal Operating Platform: Different data servers are designed and developed on different operating system (OS) platforms. Data manipulation among different OS platform is a key issue. We aim to develop protocols that will support most of the operating environments and act as a universal operating platform for different data servers.

It also acts as a transition gateway for the movement of data between multiple sites. Universal platform fetch the map from centralized conceptual DB for the desired data location site to minimize data search overloads.

VII. Working Principles

Request handler fetch encrypted queries from user API and connects to its local conceptual mapping DB to find out the data location of the corresponding queries. If data location is found on site, request handler pass the job to workload splitter who schedules the job through multiple dedicated processing nodes by storing data in on-premises storages tempted by data partition manager or retrieving data from the onsite storages. If data location is not found in local premises, queries are then diverted to the universal platform server that determines the data site location with the help of centralized conceptual DB. The workflow diagram is shown in figure 4.
VIII. CRITICAL ISSUES

In our proposed model we are expecting to achieve high level of data security, accuracy and privacy through secure query generation and handling algorithms. We also gestate to reduce time complexity and effective data manipulation by the data partitioning algorithm along with the idea of conceptual databases. We also predict that our workload splitter will provide utmost resource utilization. Though we have not pointed on few scenarios like serving a series of concurrent queries related to a single database from different DBMS platforms or when multiple DBA’s from different location site try to manipulate a single data attribute/instance.

IX. CONCLUSION

In the last decade, various systems are evolved to meet different business challenges based on cloud environment. We expect that our data manipulation technique with the proposed design will handle most of the traditional database barriers expeditiously. As demonstrated, we parted the whole work in different phases and this system is still under development. Few issues are yet needed to be revised efficiently those are mentioned in section VIII. We hope that shortly we shall be able to bring in a number of acceptable test results from different parts of this system.

REFERENCES


[2] Hakan Hacigümüs, Sharad Mehrotra, Bala Iyer, "Providing Database as a Service," icde, pp.0029, 18th International Conference on Data Engineering (ICDE’02), 2002


