Re-architecture of Database Software Stack with Planner module for query optimization in a Cloud Environment

Kamalanathan Kandasamy
Amrita Center for Cyber Security
Amrita Vishwa Vidyapeetham
Kollam, Kerala – 690525, India
kamalanathan@am.amrita.edu

Krishnashree Achuthan
Amrita Center for Cyber Security
Amrita Vishwa Vidyapeetham
Kollam, Kerala – 690525, India
krishna@amrita.edu

ABSTRACT

Nowadays public clouds offer a scalability that is often beyond what a user would be able to afford otherwise. Cloud bursting allows businesses leverage the cloud without losing the comfort and control of in-house data centre operations. Looking at cost, security and resource utilization angles we need a dynamic infrastructure to decide about the hybrid mix ie, private and public clouds. In this paper, we propose a general paradigm where software stacks need to be re-architected to dynamically be able to run either in public or private clouds. The queries get executed in private or public clouds based on cost, security and resource utilization models chosen by the clients.

Categories and Subject Descriptors

C.2.4 [Distributed Databases]

General Terms

Software stack, Query optimization, Query Processing

Keywords

Cloud bursting, Trust model, Cloud security

1. INTRODUCTION

Today lots of companies are looking at how to use the public clouds since the public clouds provide the advantages of elasticity & cost benefits. For small and medium businesses, cost is very important. Cloud bursting is a deployment model which allows the applications run in a private cloud or data center and burst into a public cloud when the demand for computing capacity spikes.

Clients plan to run the steady state business processing on existing systems at their private clouds, and then use the public cloud for periodic or overflow processing. Lots of the companies are hesitant to use the public clouds due to the security problems. For example rogue administrators and rogue clients can modify/tamper the data of the clients.

SecurIT'12, August 17-19, 2012, Kollam, Kerala, India

Cloud sourcing refers to sourcing complete solutions to run the business from the public cloud. The solution provider that offers Cloud sourcing products or services is called "Cloud Provider". The Cloud Provider typically provides solutions that knit together cloud applications, cloud platforms and cloud infrastructure. Cloud sourcing is popular with few companies if they don't have resources privately.

There will be a class of customers who will run their applications in private cloud and then move to public cloud as the need arises for example when their data increases due to on demand basis. Below we explain why we consider security, cost and resource utilization as the three main inputs to the planner module.

Security:

Client bothers about the security of his data in the public cloud. For security reasons, the client might place encrypted data in the public cloud since he doesn't trust the public provider. So security is one of the primary concerns for the client.

Cost :

Network cost, CPU/Computation cost and storage cost are the cost details to be considered by the client when he keeps the data in the public cloud. Mainly it's the network cost that he bothers about.

Resource Utilization :

This is important since the utilization of resources like the storage disks also decides where to run the query for a given situation.

Looking at cost, security and resource utilization angles we need a dynamic infrastructure to decide about the hybrid mix. From the related work we see that this problem has not been solved yet. Some of the unique challenges would

be how to define the security model, cost model and resource utilization models and how to dynamically decide where the query has to be executed which involves searching all the query nodes and coming up with the best cost function node and actually re-architect the database stacks for the execution of the queries.

We are proposing a general paradigm where software stacks need to be re-architected to dynamically be able to run either in public or private clouds, and that where they will execute is based on cost, security and resource utilization models. In this work we are taking one specific example (database) stack to show how this paradigm will work. There can be other example stacks like MapReduce, or online games that can run dynamically decide to run either in public or private clouds or at both.

In the cloud environment, currently the clients run their query at the public provider side. A key insight that we want to explore is our hypothesis that software program stacks need to be re-architected to provide better value for the clients while taking security, cost, and resource utilization input into account. The goal of this work is to build a planning tool and to re-architect the existing database management system software stacks so that the planning tool will dynamically advise where different modules of a particular software stack should execute.

The three main aspects of this work would be :

1. Designing a planner/query optimizer which will take Security, Cost and Resource utilization input requirements to determine where to execute the different modules (i.e. at the public cloud or at the private cloud) for a given scenario.
2. Re-architecting software stacks of the database management systems into different modules so that the query is run as planned by the above planner / query optimizer.
3. Quantitative analysis of different approaches with respect to performance, cost, security and resource utilization.

This work helps the clients by allowing them to have better control - what to run and where based on the above mentioned three inputs. Clients will be able to do better match with security/resources etc. and will save money and time. This is done in a way which is cost & security conscious. Currently we don't see the whole notion of this way of dynamically running the queries in either public or private or partly in both the clouds.

The rest of the paper is organized as follows. Section 2 discusses the related work in this area. Section 3 presents the software architecture. Section 4 describes the implementation. Section 5 presents the Experiments and finally section 6 concludes with the future work.

2. RELATED WORK

Since this work is about the query processing in cloud environment based on the security, cost and resource utilization as the inputs, the related work has been categorized into the following three buckets :

1) Distributed Query processing :

This gives the idea of the work done in query processing in the distributed environment since we are dealing with the query processing in the cloud environment which is also a distributed system. Lot of work had been done on the hybrid shipping, which can execute queries at clients, servers, or any combination of the two. Hybrid-shipping is shown to be the best of the query shipping and data shipping policies in [6]. An initial investigation into the use of a 2-step query optimization strategy has been described as a way of addressing the query optimization issues. Partitioning of client application functionality between client and server is suggested in [8]. A novel algorithm is discussed in [7] for hybrid shipping based on the available literature.

The details of the dynamic query execution engine is presented in [10] within the data query infrastructure that dynamically adapts to network and node conditions. The query processing is capable of fully benefiting from all the distributed resources to minimize the query response time and maximize system throughput.[11] proposes an adaptive software architecture, which can effortlessly switch between MapReduce and parallel DBMS in order to efficiently process queries regardless of their response times. Switching between the two architectures is performed in a transparent manner based on an intuitive cost model, which computes the expected execution time in presence of failures. This has the similarity that our architecture allows the query processing to be switching between client side, server side and partly on both sides.

The three different execution policies are identified and evaluated in client-server database systems in [9]. However this work focuses on rather complex queries and disk-bound computation. [12] presents an overview of the basic techniques used to support SQL DML (Data Manipulation Language) in Microsoft SQL Server. The focus is on the integration of update operations into the query processor, the query execution primitives required to support updates, and the update-specific considerations to analyze and execute update plans. Full integration of update processing in the query processor provides a robust and flexible

framework and leverages existing query processing techniques. The issue of how to intelligently manage the resources in a shared cloud database system is addressed in [13] and a cost-aware resource management system is presented as well. [14] talks about a data management platform in the cloud where the clients use just a simple, standard, and uniform language API to access data management functions as a service. Application only needs a logical specification of the data access layer and the data access requests are handled in a declarative way.

Several design issues related to querying encrypted relational databases are addressed in [15] & a new method is proposed based on schema decomposition that partitions sensitive and non-sensitive attributes of a relation into two separate relations. This method improves the system performance dramatically by parallelizing disk IO latency wit CPU-intensive operations (i.e., encryption/decryption). The query optimization problem for the encrypted database systems is modeled and solved in [16] .

2) Trust models in the cloud :

Since security is based on trust between the client and the public cloud we would want to refer the work done related to the trust model. There are several existing trust models for the cloud environment. [3] proposes a trust model of cloud security in terms of social security. Specifically, the social insecurity is classified as the multiple stakeholder problem, the open space security problem, and the mission critical data handling problem. By adding security guarantee to conventional service oriented clouds, a security aware cloud is obtained which is used in deployment of a cloud in mission critical business scenarios. [4] introduces a cloud model into the trust domain, while cloud could maintain the original characteristics and behaviors of the agents. This paper talks about the cloud model that combines two kinds of uncertainties together, which are fuzziness and randomness. To ensure the correctness of users' data in the cloud, [17] proposes an effective and flexible distributed scheme with two salient features, opposing to its predecessors. By utilizing the homomorphic token with distributed verification of erasure-coded data, this scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s).[18] argues that fundamental risks arise from sharing physical infrastructure between mutually distrustful users, even when their actions are isolated through machine virtualization as within a third-party cloud compute service. Number of approaches for mitigating this risk is discussed.

3) Cloud Databases :

Cloud Computing Dynamic Route Scheduling is presented in [20] for Optimization of Cloud Database for enhancing the efficiency searching database of cloud computing. [21] discusses Cloud Database-as-a-Service (DaaS) which hosts databases in the cloud environment and provides database features such as data definition, storage and retrieval, on a subscription basis over the Internet. The problem of resource provisioning for database management systems operating on top of an Infrastructure-As-A-Service (IaaS) cloud is discussed in [22] .

3. SOFTWARE ARCHITECTURE

With the emergence of public cloud computing phenomena, currently, customers are a) running their programs in their private clouds and storing the results in the public cloud, or b) running the programs and also storing the results in the public cloud or c) running the programs in the public cloud and moving the results to the private cloud. Thus, customers have many options with respect to where to run their programs and where to persistently store the results.

We have come up with the new architecture of the existing database to consider the cost, security and resource utilization inputs and decide where to execute the query (either client side or public cloud side). This is shown in the figure 1 below.

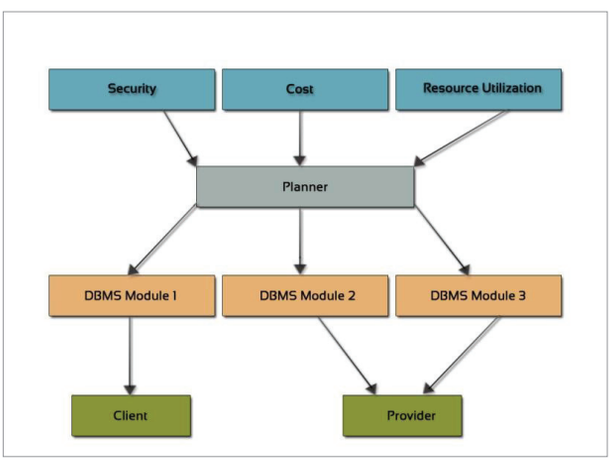

Fig 1. New Cloud Architecture

Data Placement Model :

To execute a query, we need the data in proper place. We need to move the data if it doesn't exist in the right place.

Well ahead of time we decide where to place the data ie, depending upon the security constraints whether to place the data in the public cloud or private cloud or in both the

places. There are three primary factors which can help to determine where and how to run the client program and they are:

Security model, Cost model and Resource Utilization model

Trust relations	Provider Functions	Planner module output
Client trusts the cloud provider	Isolation	Yes :Directly query from the server No :Encrypt data in public cloud, decrypt it in the private cloud for querying
Client trusts the sys admin	Authentication	Yes :Directly query from the server No :Encrypt data in public cloud, decrypt it in the private cloud for querying
Client trusts the other client in the cloud	Auditing	Yes :Directly query from the server No :Encrypt data in public cloud, decrypt it in the private cloud for querying

Table 1 : Trust relations and Planner output

The following section describes each model in detail.

Security Model: Based on the security (threat perception) a client wants, one can run a program at the client side or at the provider side. For example, if the client trusts the provider, then the client can persistently store data in an unencrypted format at the provider. On the other hand, if the client feels that he cannot trust the provider, then the client has to encrypt the data before sending it to the provider.

The planner will see the functionality provided by the cloud provider and decide the output. Here we consider only the security model as the input. Different combinations of trust relationship between client, cloud provider, system administrator and other clients in the same cloud are taken into account.

	client	Sys admin	Cloud provider	Other clients
client	NA	No/Yes	No/Yes	No/Yes
Cloud provider	No/Yes	No/Yes	NA	No/Yes

Table 2 : Client / Cloud provider Trust relations

Having come up with the trust model for the simple cloud environment, we would now like to see the planner output for each of the input. For example when the client doesn't trust the cloud provider, then the planner should say that the public cloud should have the encrypted clients data and that it should be decrypted in the private cloud.

Different trust scenarios and the Planner output are shown in the table 1. Possible trust relations between different parties are given in table 2.

Cost Model: Providers charge clients money for running their programs, or storing their data. Furthermore, clients also have to pay money for transferring data across the networks to/from the providers. Thus, depending upon how much the client is willing to pay, the client might want to do some processing at the server in order to reduce the amount of data that it has to transfer to the client. For example, if the client wants to execute a database query, he might want to execute the select operation at the server in order to reduce the amount of data that needs to be sent back to the client, and then, subsequently perform the join and project operations at the client.

Resource Utilization Model: Depending upon how resources are utilized, a client can decide to offload processing to the server. For example, if most of the resources at the client site are heavily utilized, the client can temporarily offload work to the provider, and then subsequently, move the work back to its private cloud once the peak load subsides.

3.1 Query Optimizer :

The query optimizer component of an RDBMS chooses an access plan that specifies the operators that will be used to execute a query. We have the additional three inputs namely cost, security and resource utilization.

We can make two levels of decision:

1) Entire query runs either at the public cloud or on

the private cloud

2) A query is broken into parts and part of it runs at the public cloud and the other part runs in the private cloud.

Based on the above decisions and by considering all the input cases and combinations, there are three possibilities of output as given below.

1) Entire query can be executed at the server/public cloud
2) Entire query processing can be handled at the client side/private cloud
3) Part of the query can be executed at the client and the rest at the server

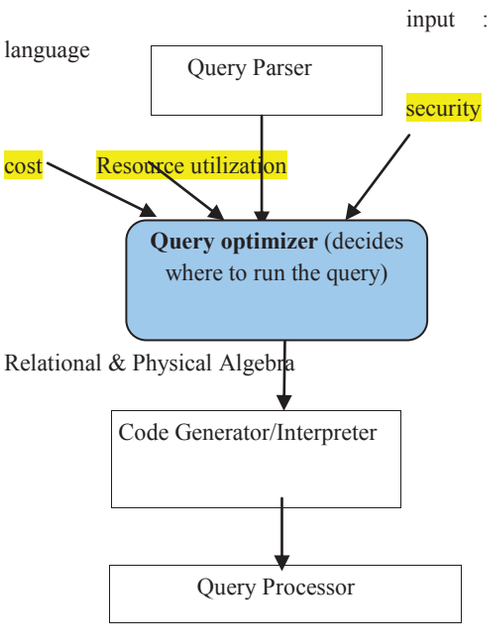

Fig 2 : Query optimizer with NEW inputs (cost, security and resource utilization)

The planner module will find the next possible state from the current state based on the security constraints, cost and resource utilization. We use the greedy approach or brute force method to select the next possible state (ie, the best optimal solution) for a given input case.

The query optimizer can work in the following manners.

- static : planning is well done before the query starts executing
- dynamic : Just before the query starts executing, the planning is done quickly.
- Real dynamic :During the execution of the query, the plan gets changed and these changes get reflected.

As given in fig 2, planner is the main module of the ordering stage. It examines all possible execution plans for each query produced in the previous stage and selects the overall cheapest one to be used to generate the answer of the original query. It employs a search strategy, which examines the space of execution plans in a particular fashion. This space is determined by two other modules of the optimizer, the Algebraic Space and the Method-Structure Space. For the most part, these two modules and the search strategy determine the cost, i.e., running time, of the optimizer itself, which should be as low as possible. The execution plans examined by the Planner are compared based on estimates of their cost so that the cheapest may be chosen. These costs are derived by the last two modules of the optimizer, the Cost Model and the Size-Distribution Estimator.

Search Space Traversal :

We have seen that the query planner takes 3 inputs and it tries to optimize the cost function.Travelling thru' each node we want to choose best node ie, we would like to find the node that gives the best cost.This method is called “search space traversal”.

Here each node corresponds to a query plan. A possible query plan may be :Select 2 tables from public cloud ,join them in the public cloud itself (since they are in non encrypted form and the client trusts the provider) and with this result join another table in the private cloud. This entire operations can be called as a “query plan”. Brute force search is done on all the nodes to see which is the best node. This is almost same as greedy approach. Greedy approach will select the lowest cost node but this is one level deep and chooses the best possible node for a given scenario.

If we have 100 possible nodes then due to the security input, 50 nodes might become irrelevant. Security, being one of the main constraints will thus prune the search space.

The search traversal for a given scenario is explained below. For eg, consider three tables R1,R2 and R3. For the given input constraints (say the input constraints are :R1 U R2 to be executed in the public cloud and R3 at the private cloud and the network cost is to be minimized and if the public cloud is busy) Each node is represented as client or server for each query operation. Depending upon the input conditions, pruning of the tree nodes is done to get the desired result. Using the greedy algorithm the search space will find the most optimal solution for the given input conditions. Figure 3 gives an idea of this principle.

A sample Scenario :

Security model : Client trusts the provider (ie, data kept in normal form without encryption)

Cost model : Client wants to go for lowcost plan.

Resource utilization : None specified.

Output of the query optimization module : Execute the query at the Public cloud

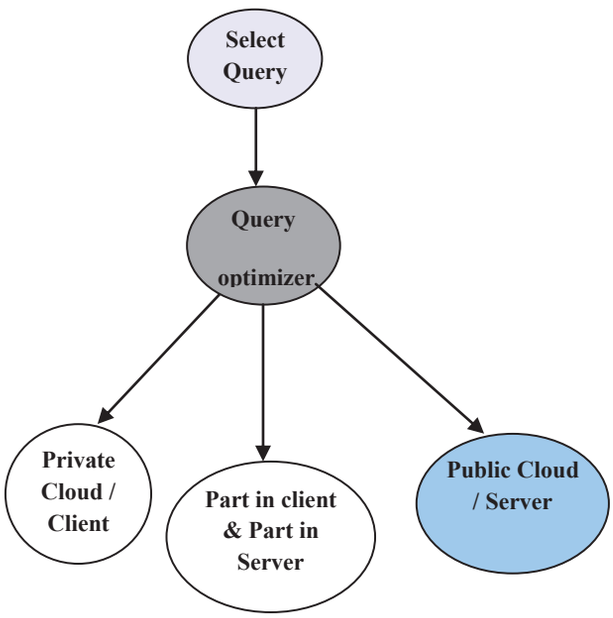

Fig 3 : Query execution plan for Query shipping

3.2 Re-architecture of Database stacks:

Once the planner decides where to execute the given query based on the search traversal as we have seen in the previous section, now we need to re-architect the DB stack so that the query is actually executed either at the client or at the public provider site or at both sides (hybrid shipping).

Existing DB stack contains the log manager, query optimizer etc. Components that need to exist in the public and private cloud need to be decided.

We need to have all of the database components (like transaction processor, query optimizer, query executer) have to reside at both private and public cloud. Moreover, the query executor needs to be able to transfer the query and data between the private and public clouds as necessary.

DB engine needs to execute the entire query in either/or and need to be able to execute in a hybrid manner. Once a part of the query is executed, the DB engine should be able to ship the query results, state of the query plan and the remaining part to be executed. Then the DB engine on the other part can continue to execute rest of the query.

4. PROPOSED TEST CASES

Based on the above design, we are proposing the following test cases to experiment with the different scenarios of query handling.

Test case 1 (With security constraints) :

Depending upon the data placement model, we would want to compare the different query processing options (ie, whether the query processing done at private cloud is better than the query processing done at the public cloud or both).For example some of the tables are kept only in the private cloud because of security reasons. So in this case we select and do the join operations first in the private cloud and with this result join the data brought over from the public cloud .

Number of queries Vs cost of the query and Throughput can be decided. The latency can also be tested

Test case 2 (Without security constraints) :

If there is no security constraints, the data is kept at the public cloud or we can consider the scenario where the data is replicated at both private and public clouds. Given this condition, we would want to again compare the Cost, Throughput and Latency with the number of queries like we did in the case of experiment 1.

Here in this experiment we would want to include the resource utilization constraints (ie, if the data is kept at both sides and the disk space is too full at the private cloud, we would want to figure out which query option would suffer.

Here also we would want to compare the Cost, Throughput and Latency with the number of queries.

Additional Test cases :

We would also compare the throughput for the static and dynamic modes of the query plan/optimizer. By doing so we would know the difference of the output (ie, actual query execution) between the scenarios where the query hasn't started running yet (static) and the queries are already running but due to some resource utilization constraints, the planner needs to dynamically say where to run the current query.

We would like to compare the output between the existing system and the system with the new DB architecture.We would as well compare the output when the client gives the first priority to security model and when the client gives priority to cost model. Based on the search space traversal, the algorithm chooses the best query plan using brute force method. We'll come to know the different nodes that the algorithm chooses for security and cost models.

A histogram that shows the security model, cost model and the resource utilization models for the given set of queries can also be drawn.

4. CONCLUSION

We have proposed the method of running the queries dynamically either at private cloud or at public cloud or at both the places depending upon the security, cost and resource constraints and the priority of these constraints are to be given by the client. This is quite helpful in case of cloud bursting.

To accomplish this, we have come up with the new planner/query optimizer module and suggested re-architecting the database stacks to actually execute the queries.

As a future work, we would like to implement the design proposed in this paper using Amazon EC2/S3 as public cloud and amrita server as private cloud. We will modify the mySQL server and run it on Amazon EC2/S3 cloud. For the test run we'll keep the data stacks in both public and private clouds.

5. ACKNOWLWDGEMENT

We would like to express our immense gratitude to our beloved Chancellor Sri.(Dr.) Mata Amritanandamayi Devi for providing an excellent motivation and inspiration for doing this research work.

6. REFERENCES

[1] W. Tang, Z. Chen, “Research of subjective trust management model based on the fuzzy set theory[J]”, Journal of Software, 2003, 14(9),pp. 1401-1408.

[2] W. Tang, J.B. Hu, Z. Chen, “Research on a fuzzy logic based subjective trust management model”, Computer Research and Development, 2005, 42(10), pp. 1654-1659Processing, Vol. 56, No. 6, June 2008.

[3] Hiroyuki Sato , Atsushi Kanai , Shigeaki Tanimoto“A Cloud Trust Model in a Security Aware Cloud ”, 2010 - 10th Annual International Symposium on Applications and the Internet.

[4] Yang Mo, Wang Lina et al, “A Novel Cloud-Based Subjective Trust Model”, 2009 International Conference on Multimedia Information Networking and Security.

[5] Yanpei Chen, Vern Paxson, Randy H. Katz ,”What’s New About Cloud Computing Security?” (http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html)

[6] Michael J. Franklin, et al. “Performance Tradeoffs for Client-Server Query Processing”,SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data.

[7] Amit Sharma, et al. Hybrid query execution Model.

[8] Ivan T.Bowman, Hybrid Shipping Architectures : A Survey.

[9] Donald Kossmann, Michael J. Franklin : A Study of Query Execution Strategies for Client-Server Database Systems Technical report, University of Maryland, 1995.

[10] Pawel Jurczyk et al, Dynamic Query Processing for P2P Data Services in the Cloud EXA '09 Proceedings of the 20th International Conference on Database and Expert Systems Applications .

[11] Adrian Daniel Popescu et al,Adaptive Query Execution for Data Management in the Cloud.

[12] A. Galindo-Legaria et al, Query Processing for SQL Updates.

[13] Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hacigumus: Intelligent management of virtualized resources for database systems in cloud environment. ICDE (2011).

[14] Hakan Hacigumus, Junichi Tatemura, Wang-Pin Hsiung, Hyun Jin Moon, Oliver Po, Arsany Sawires, Yun Chi, Hojjat Jafarpour: CloudDB: One Size Fits All Revived. IEEE SERVICES (2010).

[15] Mustafa Canim,Murat Kantarcioglu,Ali Inan : Query Optimization in Encrypted Relational Databases by Vertical Schema Partitioning, ACM SDM '09 Proceedings of the 6th VLDB Workshop on Secure Data Management.

[16] Hakan Hacıgümüş, Bala Iyer and Sharad Mehrotra : Query Optimization in Encrypted Database Systems.

[17] Cong Wang, Qian Wang, and Kui Ren ,Wenjing Lou: Ensuring Data Storage Security in Cloud Computing.

[18] Thomas Ristenpart et al. : Hey, You, Get Off of My Cloud:Exploring Information Leakage in Third-Party Compute Clouds.

[19] Liu Jia; Huang Ting-Lei : Dynamic Route Scheduling for Optimization of Cloud Database , Intelligent Computing and Integrated Systems (ICISS), 2010.

[20] Islam, Md. Ashfakul; Vrbsky, Susan V : Tree-Based Consistency Approach for Cloud Databases, Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference.

[21] Mateljan, V.; Cisic, D.; Ogrizovic, D : Cloud Database-as-a-Service (DaaS) – MIPRO, 2010 Proceedings of the 33rd International Convention

[22] Rogers, J.; Papaemmanouil, O.; Cetintemel, U.:A generic auto provisioning framework for cloud databases, Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference.