Query Optimization in Cloud Environments: Challenges, Taxonomy, and Techniques

Mushtaq Ahmad

May 19, 2020

Query optimization in cloud environments: challenges, taxonomy, and techniques

Author: Mushtaq Ahmad
Webeng.mushtaq@gmail.com
CMS:406484
Riphah International University , Pakistan

Abstract

Improving query performance remains one of the most interesting and challenging goals for both the academic and industrial communities. Indeed, cloud computing has complicated the traditional process of query optimization since many new challenges must be considered. Great efforts have been made to address this problem in the context of cloud computing. The present article aims to provide a complete view of query optimization in cloud computing. It provides a systematic survey on query processing in cloud environment through three main phases. It first identifies the specific cloud challenges facing query processing techniques. Then, it reviews and classifies the current query optimization techniques based on a proposed taxonomy. Finally, it compares and discusses the surveyed techniques based on the specific challenges related to the cloud environment. This paper provides readers with some recommendations which must be considered in future work.

Introduction

Cloud computing is one of today's fastest growing technology. Its ability to allow users to perform massive-scale and complex computing without owning expensive computing hardware and
dedicated software make it more attractive. Interest in this new technology is primarily due to the economies of scale and for the hardware and software costs incurred by users. Since such users are likely to be much lower when they are paying for a share of a service rather than running everything themselves . Secondly, the costs incurred in well-designed cloud-based data store and management systems will be proportional to concrete usage according to "pay-per-use" paradigm for both software licensing and administrative costs. Hence, cloud solution provides the illusion of infinite pools of highly reliable, flexible, and scalable computing, storage, and network resources . However, even huge storage capacities are guaranteed in such technology, end-users need also to query stored data and get fast query processing. Thus, one of the major challenges in cloud storage components development is to develop effective methods and strategies for query processing and optimization.

Query optimization has been intensively investigated in different contexts, in centralized databases , parallel databases , and large databases . Moreover, other studies have focused on query optimization techniques for a specific type of queries such as top-k queries , recursive queries and continuous queries . In big data context, Lee et al. have given a thorough discussion about mapreduce paradigm-based query processing, its pros and cons, and its optimization strategies.

Traditional query optimization solutions are generally based on several techniques such as indexing, caching, fragmentation, and view materialization, etc. However, implementation of these techniques in cloud environments without adaptation can have a poor performance since they cannot anticipate future availability and release of resources . In the cloud computing context, query optimization is more challenging since a query execution involves many factors and multiple data stores. Furthermore, diverse challenges related to the cloud environment must be considered during query processing. To deal with data storage and querying issues in cloud environments, many studies have suggested adopting big data technologies as a solution to the development of the cloud. Thus, the status of big data in cloud computing is given in. The authors discussed the relationship between big data storage systems, cloud computing, and Hadoop technology. Sakr et al. provided a survey summarizing various strategies and mechanisms for processing and deploying data-intensive applications in the cloud. Attasena et al. focused on the security aspect of the cloud. They surveyed the secret sharing schemes related to the query and data processing with respect to data security, data access and costs in the pay-as-you-go paradigm. Recently, a survey paper has pointed out and evaluated big data indexing techniques for query optimization in cloud computing. The resulting taxonomy of indexing techniques categorization is based on three approaches which are non-artificial-intelligent, artificial-intelligent, and collaborative artificial-intelligent approaches.

It is true that the indexing techniques have been largely used for cloud query processing. However, in recent years, many other query optimization techniques for cloud computing have been carried out. Moreover, even if these different techniques have been studied individually in the past, the literature still lacks a unified view of query optimization techniques in cloud environments. Even though there have been some surveys and reviews regarding query and data processing in the cloud, they fail to cover many techniques and do not provide a holistic view of the research about query processing in the cloud. Moreover, no one of the previously presented surveys directly focused on specific cloud challenges to investigate query optimization techniques. Thus, a comprehensive survey of this field is required.

Mainly, this study contributes to understanding the principal of query optimization techniques in cloud environments. It firstly identifies challenges facing query optimization techniques in cloud environments and then surveys and classifies such techniques with respect to their strategies. A focus on the recent techniques that deal with the specific challenges of cloud environments is made. Noting that, works related to the effect of hardware improvement on query optimization techniques such as memory and access costs are not considered in this paper.

In order to achieve the contributions mentioned above, we carry out our research investigation by querying scholarly online electronic databases using the keywords "Query optimization" and "Cloud computing." The queried scholarly electronic databases include *ScienceDirect, SpringerLink database, IEEE Xplore, ACM Digital Library, and Google Scholar*. The papers were from different journals and conferences. Returned articles were downloaded and carefully examined. The selected papers were carefully read and analyzed to extract the addressed challenges, considered aspects and methodologies used to optimize queries. Complete classification and a comprehensive review of various query optimization techniques are then obtained.

In this paper, we make the following contributions:

- We give some basic backgrounds, related concepts and the main features of cloud platforms, which are necessary for the understanding of the paper.

- We provide a detailed description of the query optimization problem considering the specific cloud challenges.
- We review and classify the most recent techniques of query optimization in the cloud, scrutinizing their main aspects.
- We compare and discuss current methods, taking into account the described challenges.
- Finally, resulting from this study, we analyze the limitations of current query optimization techniques. Then, we give some important opportunities and recommendations for future research directions.

The rest of this paper is organized as follows. In Sect. , we introduce the most important concepts related to cloud computing. In Sect. , we first define the query optimization problem and then highlight the specific challenges related to cloud query processing. In Sect., we review and classify the most recent approaches for query optimization. In Sect. , we provide the overall discussion of the limitation of the literature approaches. In Sect. , we present future directions in the context of query optimization in cloud environments. Finally, we conclude the paper.

Background

For a better understanding of the rest of the paper, we provide in this section a brief background on cloud computing models and querying within the cloud environments.

Cloud computing

The National Institute of Standards and Technology has defined cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction." Also, Armbrust et al. summarized the cloud computing definition as “both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services.”

Cloud service models

Basically, NIST has defined three service models and four deployment models. The deployment models are public cloud, community cloud, private cloud, and Hybrid cloud. The service models include IaaS, i.e., Infrastructure as Service, PaaS, i.e., Platform as a Service, and SaaS, i.e. Software as a Service. Thus, IaaS means that computers are proposed as physical or virtual machines to support end-users’ operations. The service provider rents the use of these machines to customers in a pay-as-you-go manner. In another hand, PaaS provides computing platforms to customers such as operating systems, hardware, programming language execution environments, servers, and databases. In SaaS, software applications are installed in the cloud, operated and maintained by service providers. Customers can access the software from cloud clients. Recently, new expanded services model have emerged such as StaaS, i.e., Storage as a Service, and DaaS, i.e., Database as a Service. Indeed, StaaS forms one of the critical components of IaaS. It provides a series of cloud-based data stores that differs in data consistency semantic, data model, data transaction support, and price model. Furthermore, the DaaS model provides database functionalities such as data definition, storage, querying and user interface functionalities. To implement such a service, the majority of cloud service providers use big data hybrid architectures. Such architectures combine structural, non-structural, and semi-structured data storage systems and also open-source and commercial cloud platforms. Thus, cloud computing is closely related to the new provisioning paradigm of computing infrastructure and big data processing technique given the variety, volume, and veracity of data across cloud environments.

Querying in the cloud

In the cloud computing context, querying consists in retrieving of the data records or files that match the specified condition. In the cloud, queries can take different forms such as:

- Ad hoc query means one-time executed query, it is often expected with results at interactive latencies.
- Recurring query, i.e., periodically deployed query to get daily or hourly reports.
- Continuous query, this form of query is executed in real-time and incrementally compute results as it is received.

There is various type of queries considered in the literature. Those that have been widely discussed in the context of the cloud computing are SPJ query, aggregate query, top-k query, range query, multi-keyword query, skyline query, and k-nearest neighbor query.

- *SPJ query* is the most popular query in the database, it includes selection, projection, and join operations.

- *Aggregate query* is defined with a function where the values of multiple rows are grouped together according to some criteria to get a single value with significant content.
- *Top-k query* is to seek and retrieve the k elements with the highest selecting score.
- *Range query* is to seek and retrieve a set of database elements that fall within a range of ndimensional space of attributes.
- *K-nearest neighbor query* KNN query is to seek and retrieve the top k most similar documents in an outsourced database to a given document or a relatively small set of documents.
- *Skyline query* assume that every user has a set of preferences over the attributes of data.

  The sought skyline set is the subset containing the most preferred items of all the preferences of all users .
- *Multi-keyword query* is to seek and retrieve files or documents containing a particular set of keywords.

Problem definition and related challenges

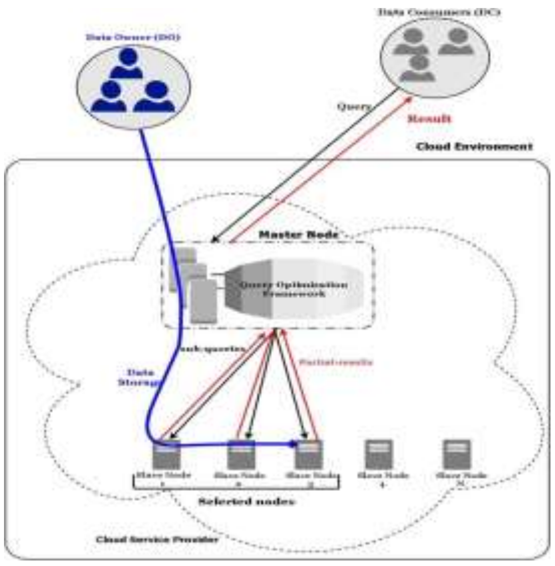
In recent years, there is increasing research interest in query processing over cloud environments. Such research studies have focused on different challenges related to query processing in cloud computing while recognizing diverse hypothesis on the interacted entities and actors of the cloud. Here, we make a brief overview of (1) the different entities considered when addressing query optimization issue, (2) a description of the query optimization problem, and (3) most important challenges when designing a query optimization technique.

Cloud system model

Cloud computing cluster contains hundreds or more of heterogeneous hardware in the form of nodes. Each node has the ability to store data and perform a specific computation.

A representative scheme of cloud querying system and its components are illustrated in Fig. 1. It includes two types of nodes namely, master nodes and slave nodes. Master node, which is responsible to store some meta-data about all data and clusters. Slave node is responsible to store the regular data. Moreover, three different entities can be identified as follows:

Fig. 1



Cloud query processing system and its components

- *Data Owner (DO)* is the landlord of the stored data or files in the cloud. In the case of files, the landlord must identify its files and specify a set of keywords to be used for possible queries.
- *Data Consumer (DC)* is an authorized entity—individual users or enterprises—which has the right to query the cloud data and search for specific information in the cloud.
- *Cloud Service Provider (CSP)* an entity which offers storage and computational resources and services.

Generally, a data owner stores its data in a specific CSP to provide its service to diverse customers who access these services and data over the internet. To ensure a minimum quality of service, it negotiates an SLA, i.e., a service level agreement with the provider on which they specify all monetary costs and service level that must be satisfied.

Query optimization problem

In this sub-section, we give a short description of the query optimization problem in cloud environments.

In cloud environments, query processing can be either centralized or distributed. To perform efficient optimization, it requires statistics from the different data storage nodes. As illustrated in Fig. 1, a DO stores its data on a CSP, and multiple DCs can query this data. A DC submits a query using a high-level language such as SQL, HiveQL or a set of keywords in case of multi-keyword query. Usually, this query is submitted to the master nodes. When the master node receives a query, the optimizer framework—located in the master node—produces a detailed query execution plan (QEP) that can be performed by a number of nodes. To determine an optimal plan, the optimizer framework uses a cost model. This cost model estimates the system resources used for query evaluation operators. Due to the distributed environment, the processing of distributed queries requires data communication between diverse nodes through a network. Thus, according to the generated QEP, the optimizer framework decomposes the query into several sub-queries. Then, for a parallel and concurrently processing, the master node dispatches sub-queries to the available slave nodes based on diverse factor including load balancing strategy and paying capability. Finally, the slave nodes evaluate these sub-queries based on the proposed plan and then return partial results to the master nodes. When all sub-queries on the slave nodes are executed completely, the master node merges the partial results, and the result of the query is returned to the DC. Noting that slave nodes can exchange sub-results between them if necessary.

During this process, some traditional issues must be addressed to carry out the optimization operation such as:

- Which is the best cost model to be used?
- How to estimate costs of query execution plans, considering multiple diverse relevant criteria

such as the monetary cost of resources, staleness of data, etc.?

- How to select the best execution plan that could perform the original query?
- How to select available nodes where the query must be processed?

- How to assign sub-queries across different available nodes taking into account load balancing?

These different issues and questions have been well discussed in the literature of distributed databases context. Besides the well-known traditional issues addressed in the conventional and distributed database discussed above, some other issues arise such as the lacking of scalability, reliability, fault tolerance, data partitioning, and replication. Such issues were dealt with big data solutions. However, in addition to all these issues, new considerations and challenges related to cloud environments must be addressed. These new considerations and challenges will be discussed in the next sub-section.

Specific challenges of the cloud

In this sub-section, we identify the main challenges encountered when designing query optimization techniques in cloud environments.

To address the problem of query optimization, many authors adapted traditional query optimization techniques considering cloud computing specificities. Although traditional query optimization solutions present various advantages. However, even with the new big data platforms, the specific features of cloud computing requires rising effectively new challenges. Cloud computing has the features of being a large-scale, distributed and virtual complex information system. In the following, we give a brief description of the most important challenges.

How to deal with DFS and key-value storage systems?

In cloud computing systems, data storage depends largely on the distributed file system (DFS) to store data and adopts a key-value and big-table-like structures. However, such structures have some limitations in data management. For instance, these structures cannot effectively process complex queries and only basic query operations are supported. Moreover, they can only provide key-based insertion. Furthermore, data are horizontally partitioned and redundantly stored in multiple distributed worker nodes. In DFS, there are multiple instances of the same volumes, which can move between the different nodes. When a user query is issued, the system should seek an adequate volume instance while considering load balancing, scalability, reliability, fault tolerance, and data replication. Therefore, query optimization techniques must address problems arising when data volume stored in different nodes is large. Besides, to take the better QEP, the master node must collect statistics from various slave nodes to avoid high cost.

How to preserve elasticity feature of cloud services?

A key asset of cloud computing is its elasticity. Indeed, the cloud elasticity feature means that a service can be scaled up by increasing the set of nodes/resources or can be scaled down by decreasing this set of nodes or resources. Therefore, it enables the system to add and remove resources according to the application's requirements or on user demand in real-time. Therefore, keeping cloud computing elasticity requires observing closely and predicting the required resources for the system in order to decide when to add or to remove resources. Since a node cannot meaningfully participate in the database operations only if it has a substantial proportion of the querying data. Thus, the provisioning of data on available transitory nodes makes elastic a hard problem. This is a crucial task since an under-provisioning or over-provisioning of computing resources will probably adversely affect the cloud's user as well the provider. Furthermore, the speed of elasticity adaption is a crucial quality of a cloud service. Thus, query optimization tasks must be flexible since elasticity requires sporadic and fast shifting of data and data ownership between nodes. Therefore, implementing an adaptable query optimization technique, able to consider the capacity of data provisioning and temporarily available nodes is a real challenge.

How to deal with the pay-as-you-go paradigm?

One of the fundamental features of the cloud is to allow its users to pay the use of short-term computing resources according to their need. Cloud resources are rented in the unit of virtual machine instances, or according to technical criteria such as power, bandwidth, or in flat-rate form. However, using a query optimization technique implies extra cloud resources such as additional storage space, additional structures or more computing power. Thus, such extra cloud resources must be priced. Hence, provide a query optimization technique with minimum extra cloud resources to keep a balance between the customer's ability to pay and supplier's profitability is a great challenge.

How to ensure security and privacy of data?

Data security remains one of the most concerns in cloud computing. Indeed, data security and privacy requirements are quite crucial for many users and often are factors that restrain their adoption of cloud computing solutions. Since cloud computing includes three parties: DCs, DOs, and CSP, and the stored data are not under the direct authority of their DOs. Thus, traditional security solutions are not directly applicable. Indeed, CSPs are a possible threat to the security of data since they can provide a secondary usage of data for advertisement purposes or for governments. Dealing with cloud security issue involves addressing data privacy, availability and integrity issues. In the cloud context, especially for query processing, security concern both storage and computation.

To enforce security, diverse solutions were proposed including, data encryption, data anonymization, data verification, and data separation. However, these solutions make query evaluation more complicated. Thus, a great challenge is how to design and to adapt query optimization techniques to ensure both a high-level of security and flexible query processing. In other words, it consists to find the most trade-off between the high-level of data security and efficient data access.

How to manage the volume and heterogeneity of data?

According to Reinsel et al., the volume of global data will grow from 33 Zettabytes in 2018 to 175 Zettabytes by 2025. Indeed, data and services of the cloud are characterized by their heterogeneities since processed data are in different format including structured, semi-structured, and unstructured data. Thus, there is presently almost no declarative method to define and execute complex queries over many data stores of heterogeneous data models. This is principally due to the lack of a common access model on heterogeneous data stores. Therefore, enhancing cloud data management systems and query processing techniques to be able to follow these growing and heterogeneity trends of the data volume are big challenges.

Solutions to manage such a growing volume of data in cloud environments have been proposed. These solutions consist of data partition to provide ***scalability*** and replication of the partitioned data to get high availability. However, these solutions make query optimization more complex. Since the query optimization techniques have to consider specific criteria such as replication factor, the cost of data movement and transformation which can be very high.

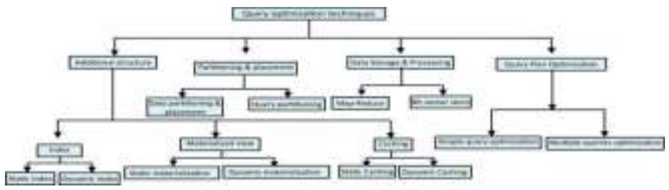
Review of query optimization methods

In this section, we provide a categorization of query optimization techniques in the cloud in a taxonomy that summarizes current research in the field.

This taxonomy is done based on various factors. The first level of the taxonomy captures the basic principle used to optimize queries including design, architecture, and requirements of each technique. The first level provides four main categories. These categories also have different subcategories. Each category was further classified to provide a second level of taxonomy. This second level describes considered information during the optimization process including functionality, used data and meta-data to obtain an execution plan or result. The final hierarchy of the taxonomy was developed by further refining the classifications based on the maintainability on the technique.

Through this taxonomy, we provide researchers with an accurate view about most important contributions on query processing in the cloud environment and their limitations. This taxonomy summarizes similarities and differences between these contributions. Figure 2 shows the resulting taxonomy, which categorizes query optimization techniques in the cloud. The details of these categories are given below.

Fig. 2



Classification of query optimization techniques

Additional structure-based methods

To address the query optimization problem in cloud computing, great efforts have been made to design solutions based on additional data structure such as index, materialized views and caching.

View materialization based methods

In the database field, a view is a derived relation, defined by a query in terms of base relations or other views. It is said materialized if its query result is persistently stored. The views selection consists to select the appropriate set of views to be materialized, considering some constraint such as query workload, storage cost, etc.. Materialized views are useful only if the query optimizer is able to find pertinent views quickly. Based on the maintainability of materialized views set, we distinguish between two kinds of methods; static and dynamic materialization. Materialized views have been used in cloud environments to improve query performance. In this paper, we distinguish between techniques base on static materialization and those based on dynamic materialization.

1. (A)

Static materialization based methods In traditional approaches, to select a set of views to be materialized, authors considered some parameters to estimate the cost model of view materialization including query workload and database size. To adapt such cost models to the cloud environment, Nguyen et al. defined new cost models which consider the pay-asyou-go paradigm of cloud computing. They considered some new parameters related to cloud environments such as cloud pricing model of CPU, storage, and networking. They used these cost models for query optimization process achieving a trade-off between computing power enhancement and view materialization under a budget constraint. The proposed cost models allow achieving a multi-criteria optimization based on materialized view selection. However, the authors did not consider the evolution of their parameters. For instance, an evolution of query workload can be considered due to the evolution of the application requirements. Thus, in this case, they must repeat views selection process from scratch.

2. (B)

Dynamic materialization based methods unlike the previous method, Qu and Dessloch considered view selection problem as the process to find out a position of a materialized view insertion in the view materialization flow in real-time. To select materialized views, they defined a measure called performance gain which depends on some metrics like source update rate and original/incremental operation cost from different platforms. Then, they proposed an incremental on-demand maintenance technique to maintain the materialization point which would be in the flow. The location of a materialization point in transformation flow varies dynamically based on some metrics such as the data source update rates and maintenance cost. This real-time view materialization method took into account the evolution of parameters such as the workload.

Index-based methods

In the database context, different multi-dimensional index structures such as R-tree and B^+-tree have been developed for query optimization. In cloud environments, great efforts have been devoted to adapt these structures in order to address query optimization challenges. Based on the maintainability of an index, we categorized these works into two categories: static index-based methods and dynamic index-based methods.

1. (A)

Static index-based methods In this category of methods, many frameworks have been carried out. Thus, an elastic cloud storage system called EcStore was developed by Vo et al.. EcStore is based on a distributed index called BATON with two new extension functions: optimistic concurrency control and load-adaptive replication. It supports automated data partitioning, load balancing, and replication. It also efficiently handles range queries and transactional access. Users can access data using transactions that combine read and write operations on multiple pieces of data stored on different cluster nodes. Authors argue that ecStore provides two additional features including transactional semantics across multiple keys and load-adaptive replication compared to other closedsource cloud data serving systems such as Dynamo and Pnuts, and open-source systems such as HBase. Another framework based on VF-CAN indexing scheme was developed by Cheng et al.. The proposed VF-CAN scheme includes content-addressable network (CAN)-based routing protocol and an improved index of the vector approximation file (VA-file). The VF-CAN index structure has two levels: a global index level and a local index level. The local index called VAK-file built in each storage node by an improved VA-file. Then, data are quantified and compressed into approximation vectors of VA-file. To group the VA file vectors, they used a *k*-means clustering algorithm considering the degree of proximity of VAK-file. The global index is built on the top of the previously defined local index. For scheme operation, the storage node sends its local cluster center information and its IP address to the overlay network through the CAN interface. Noting that the authors have given an algorithm describing how the local index is built and how to issue a local index in the global index. In order to eliminate bottleneck problems, the global index is distributed in storage nodes. To facilitate data management, each storage node is organized with a structure overlay CDN to manage the global index. According to the authors, VF-CAN reduces index storage space and improves greatly query performance. However, this framework does not sufficiently handle index maintenance, and further effort is required to improve the efficiency of maintenance of the proposed indexing scheme. Guo et al. exploited key-value stores and the map-reduce parallel computing paradigm to implement the KVI-index, which is based on two interval indices: on time dimension and sensor value dimension. Each index has an in-memory search tree and a secondary list materialized in the key-value store. They developed a new approach to perform query processing under modeled data streams by enhancing intersection search algorithm called iSearch that produces consecutive results suitable for map-reduce processing. Noting that the time complexity is O(1) in terms of network I/O, the space cost is O(N) for time index, and the space complexity of vs-tree iSearch algorithm is O(1). Li J et al. proposed a multi-dimensional indexing mechanism PR-Chord. The PR-Chord index includes a PR-Index and a Chord network. It is a hierarchical index structure and based on the improved version of PR quadtree. The multi-dimensional space trained by the range of the multi-dimensional data is split into equal hyper-rectangle spaces. The effectiveness of PR-Chord comes from the load balancing and simplicity of its indexing algorithm. According to the authors, the average transmitting hops of PR-Chord query are O(log N). Furthermore, the time complexity of PR-Index in the worst condition is $O(d \times N^{1 - 1/d})$, in which N is the total node number of index for d-dimension.

While Li Y et al. used additional information to a complete binary tree called PBtree (privacy Bloom filter tree) index. It is used to organize indexing elements. Noting that query result will be accompanied by validation information, allowing users to check the integrity of results. To manage the security issue, the authors proposed a leveled verifiable range queries scheme based on a private-preserving scheme. They achieve the property of verification by adding some additional information into the query result, and data users also use additional information to verify the query result. Unfortunately, this means an additional storage cost to this method. Moreover, the time complexity of their algorithm is O(|R| log n), where the number of query result is *R* and *n* is the number of index items. Likewise, Mei et al. addressed the security issue when optimizing range query using a new encrypted-index-based scheme called EIT. Indeed, to execute range queries, a data owner creates an index EIT, i.e., Encrypted Interval Tree for its encrypted data. If the data in the cloud is not evenly distributed, the data owner must complete cipher-texts in each sheet the same size. The filling procedure ensures the security of this data. To perform the EIT index, the authors adopt the KNN search pattern on encrypted texts. Noting that this technique makes encryption schemes support the scenario of multiple users. In this scenario, the proposed scheme supports efficient logarithmic complexity search over encrypted multi-dimensional data. However, additional costs of communication between users and cloud provider are required.

1. (B)

Dynamic index-based methods
This kind of indexing scheme is dynamic and support schema adaptation. Some works have been done in this direction. For instance, to deal with query optimization of range queries considering the security aspect, Talha et al. proposed DISC technique which means dynamic index for spatial data on the cloud. Indeed, DISC is an index based on the Hilbert dynamic tree R-tree. Its structure is encrypted by an orderpreserving encryption (OPE) scheme, while the data are encrypted separately with a secure symmetric encryption method. DISC encrypts the nodes without altering the structure of the index, which explains its fast execution. A spatial transformation is applied to the data and the dynamic spatial index is encrypted using OPE. Such OPE encryption balances between efficient query processing and data confidentiality in the cloud. Noting that this spatial index supports dynamic updates on the outsourced data. To speed up multidimensional query processing through data recording and an efficient indexing scheme, Zhang et al.

developed a multi-dimensional index called EMINC\ EEMINC by combining two structures: R-tree and KD-tree. To address EMINC\ EEMINC index maintenance problem, they proposed a cost estimation-based index update strategy to update the index structure. However, the maintenance cost is high.

Finally, and maybe most importantly, in cloud environments, scalability issue is initially addressed by data partition and availability is addressed by replication of the partitioned data. Index-based methods are known as adaptative for data partitioning, i.e., these methods are able to support automated data partitioning and replication. And therefore the scalability issue is handled through these techniques. On the other hand, the static index is rebuilt periodically from scratch rather than updated incrementally, while dynamic index requires an extra cost for its maintenance. Furthermore, the dynamic index approaches outperform static index approaches by maintaining location updates incrementally. However, dynamic index approaches suffer from scalability cost due to their single-server (one data store) design and the extension of the centralized dynamic index approaches do not scale out when the number of servers increases, while tree-based index methods incur dimensionality problem with the increasing in the dimensionality.

Caching-based methods

Cache is a set of data duplicating original values stored in memory, generally, for easier access and reduce response time. Thus, caching is a technique which consists to replicate most requested content on a system closer. Then, these replicated data will be used to process future queries. Some works have also adapted this technique to process queries in the cloud context. Based on maintainability of replicated data, we categorized these works into two categories: static-cachingbased methods and dynamic-caching-based methods.

1. (A)

Static-caching-based methods In this category, replicated data is not maintained after the modification of original data. In this context, Dash et al. developed a cost model for selftuned cloud caching. Their cost model takes into account all possible computing and infrastructure expenditure. Depending on the business line of the cloud computing, the proposed cost model takes into account the necessary and available cloud resources: including disk space, I/O operations, CPU time, and network bandwidth. This model is self-tuned to the three policies that aim to (1) high quality of individual query service (2) increasing overall quality of query services, and (3) cloud profit. According to the first policy, users are satisfied with the received query services given the amount of money they are charged. According to the second, the overall query service is gradually ameliorated so that individual query charges are minimized. According to the third, cloud infrastructure remains profitable at all times. Therefore, the proposed model is adapted to policies that encourage high-quality individual and overall query services but also brace the profit of the cloud provider.

2. (B)

Dynamic-caching-based methods In this category, the cache element is maintained to store the latest data up-to-date. In this context, Ma et al. introduced a dynamic cache replacement strategy based on the frequency of segment access. They define a semantic segment as a decomposed or coalesced query result. When a query is processed, the semantic cache manager analyzes the contents of the cache and creates two sub-queries: a probe query will be handled using the retrieved data of the cache and a remainder query will be handled using the retrieved data of the cloud. The authors proposed an algorithm that allows the semantic cache to provide enough information for a comparison between the input query and the query of the semantic region without processing all the tuples in each semantic region. During this access algorithm, four types of events can occur: (1) The query is completely processed from the cache. (2) Additional operations and treatments must be performed in order to retrieve the result of the query in its entirety by a different segment of the cache. (3) The result of the query can be retrieved from the cache, but another part has to be recovered from the cloud. (4) The query cannot be handled by the cache. They also examined how to increase cache utilization to further reduce query processing time. To ensure cache consistency, they provided an effective lifecycle tag for the cache element to store the latest data up-to-date, and when the data are updated, the cache element will be automatically updated. However, the consistency issue among different cache nodes and fault tolerance must be addressed.

Partitioning-based methods

In this kind of optimization techniques, searchers do not use additional structure. However, they focused on how organizing and partitioning the data or query, so as to find them more quickly. We distinguish between methods based on data partitioning and query partitioning.

Data partitioning and placement based methods

In this category, the authors focused on data organization. To optimize query performance and reduce resource consumption, Kumar et al. built up a data placement approach called SWORD of replicated data considering query workload. Their aim is to cluster data required for each query on to as few partitions as possible. To achieve this aim, they developed a partitioning technique based on a weighted hyper-graph. This last is used to model the query workload. Since the resource consumption of a task depends in addition to its characteristics, the overall status of the system such as other simultaneously running tasks. A hyper-graph means that the connected data items, i.e., a set of relation partitions, file chunks, or tuples, appear together in a query workload. Its weights assigned to the edges capture the access frequencies. The authors developed several algorithms for data placement with replication including Dense Sub-graph, Pre-replication, Localbased Motion, and *K*-Way Replication algorithm. They proposed a routing mechanism that minimizes hyper-graphic compression overhead in two steps. Firstly, grouping hyper-graph nodes into groups, secondly, reducing the group of nodes into a single virtual node. While Wang et al. addressed the problem arising out of the low sampling efficiency when using OLA (online aggregation) querying. This last consists to give the user an approximate query result using random samples of the data then refine the result with other samples. The authors have developed an aggregation system called OLACloud coupling two key concepts: repartition and allocation. The base relation is divided into blocks according to its attributes, allowing combining the appropriate tuples together to form a set of new blocks according to these attributes. Then, they proposed a block placement strategy considering block size and variance of storage consumption. However, it would be interesting to improve the online aggregation performance by reducing the redundant statistical computation cost. Oktay et al. proposed a partitioning-based method using division strategy of selection and projection operations. Partitioning of data is achieved using a special column called CPT column. Results, i.e., partitioned data are then stored and placed in a private cloud (trusted part) or in a public cloud (untrusted part) according to data sensitivity. The authors have also developed an approach to execute SQL style queries in a hybrid cloud. They claimed that their approach ensures query processing with a minimum communication between the public and private cloud. Moreover, the partitioning approach improves greatly the overall query execution time, by as much as ten times as compared to the case of working only in a private cloud. However, balance the load between private and public machines is not guaranteed in this approach. Huang et al. proposed an elastic spatial query processing technique in an OpenStack cloud computing environment. They focused on horizontal auto-scaling containers. They implemented a spark-based spatial query processing algorithm (SQPA) and identified the parameters that affect the effectiveness of parallel SQAPA. In their technique, spatial data objects are partitioned and stored in distributed nodes. Then, the query objects are broadcast to the nodes of the partitions reside. They suggested the use of a small number of containers with a reduced number of total executor cores for substantial spatial query processing of big spatial data. However, many parameters must be done manually in this technique which limits its use.

Query-partitioning-based methods

In this category, the authors focused on queries partitioning and management. Guabtni and colleague focused on range queries processing by combined a load balancing technique with a density-based partitioning technique. They introduced a partitioning method which consists to split a range query into smaller ones so that they can be distributed across the available replicas of the database with load-balanced fashion. The authors have developed a density-based query partitioning algorithm enabling to evenly share the workload across a set of database replicas deployed on a cloud infrastructure. Nevertheless, a pre-treatment phase that may take a long time is required to perform the partitioning operation. Da Silva et al. proposed a non-intrusive and adaptive performance monitoring technique which consists to dynamically providing the necessary set of virtual machines able to execute each query. Their technique operates over the relational data model with complete replication of the database. Thus, each deployed virtual machine has a database management system with a complete copy of the database. The authors proposed the architecture of four modules, partition module, a monitoring module, capacity planner module, and orchestration module. The partition module is responsible for virtual partitioning and dividing the query into several sub-queries. The partitioning algorithm distributes a number of partitions to each available virtual machine based on its performance. The capacity planner initially provisions a number of VMs to treat the query while minimizing the computational cost and penalty. The monitoring module instantiated in each virtual machine allocated to process a sub-query, while the orchestration engine is responsible for communication between the previously presented modules. Thus, this technique follows the query response time of the SLA contract and then makes adaptive monitoring, considers that the virtual machines may have different performances. However, it is limited only to select-range queries. Moreover, the authors did not address i/o and storage costs of the proposed technique. In their work, Zhao et al. dealt with SQL query. They proposed to firstly decompose the queries into sub-queries according to the operator and operand request, which can run in parallel. Then, they proposed two scheduling algorithms for query processing procedure providing load balancing and improving query performance. They exploited replication offered by the cloud database system for query processing to provide better alternatives for the scheduler. Moreover, they used a pipeline strategy when the results come back to reduce the response time of the query. However, the query cost does not reduce linearly because of the computation of the large matrix. Besides, the proposed scheduling algorithms select sub-query according to a heuristic method-variance to achieve maximum load balancing.

Storage system improvement based methods

In order to enhance query processing in cloud environments, many improvements to the storage systems have been made in the literature. Since previous investigations and surveys have focused on techniques based on the improvement of the storage system, we analyze and discuss only two categories of methods namely map-reduce based methods and additional storage structure-based methods.

Map-reduce based methods

Map-reduce model has received great interest from academic and industrial. Indeed, in the mapreduce paradigm, programs read input and store output in distributed file systems such as GFS and HDFS. Such distributed file systems represent the storage layer of cloud computing platforms. Map-reduce divides the parallel processing using two functions map and reduce. Several optimizations and improvements of the map-reduce model have been proposed in the literature. For instance, map-reduce-merge is a framework proposed to address the data transfer bottlenecks. It allows merging of already partitioned and sorted data. While HaLoop is an improved implementation of the map-reduce framework by adding Loop control. The HaLoop framework was implemented to support iterative applications. Another framework was developed by Koh et al., who focused on skyline queries optimization. It processes efficiently skyline queries and avoids the bottleneck of centrally finding the global skyline from local skylines using the mapreduce framework. It is based on two algorithms, the first one called MR-DDTP, i.e., map-reduce distributed dominance Test with Projection algorithm and it applies the division of the grid to produce segments assigned to the mappers. Subsets dominated operations are distributed to multiple reducers to find the global skyline from the local skylines objects. Dominator reduction strategies are designed to reduce the amount of data transmission. The transmission of data is from the mappers to the reducers according to the dominance ratio between the resulting segments of the splitting methods. The second algorithm called MR-Sketch avoids high IT costs during local skylines objects finding by computing the skylines objects of the sample points to filter out most non-skyline points in the mappers. However, the reducers responsible for processing these segments need more processing time due to unbalanced loads among the reducers.

On the other hand, map-reduce job scheduling has received considerable attention in the cloud context. An efficient job scheduler is crucial to improving resource utilization and system performance considering the importance of the number of jobs.. Kllapi et al. proposed several greedy and probabilistic optimization algorithms. These algorithms explore the space of alternative scheduling of data-flows on the cloud considering time and money constraints. This family of algorithms follows a nested loop approach used in the schedule and stops when the optimality criterion does not improve. Noting that, the proposed solution has considered resource elasticity of the cloud.

Bit vector storage structure-based methods

Bit vector storage methods offer high scalability. Indeed, Yang et al. have been attracted by the issues related to the column storage system including its inability to scale to support the increase storage rates to petabytes and ever-increasing in scale, and also the impedance mismatch that occurs when complex data is normalized in a relational table. The authors realized that these issues do not allow to effectively processing the aggregate query. Hence, they proposed a bit vector storage method to address aggregated query optimization in cloud environments. This method is based on the principle that a bit vector is a sequence of elements of a bit in which each element is referenced by its index. In bit vector storage system, the attribute coding schemes are used to decide how the attribute values are transformed into binary vectors. The appropriate scheme is chosen based on the characteristics of each attribute. The authors claim that the bit vector storage method is better compared to the column storage method in many evaluated parameters such as execution time, processing and i/o costs, and compression time. However, the data compression technique requires further investigation and improvement.

Query plan optimization based methods

Query cost models are used to predict and estimate the cost of the query execution plans in terms of the number of operations (I/O and CPU) needed, considering that physical resources to be employed are known. According to the number of involved queries during one optimization phase, we distinguish between simple query optimization methods and the multi-query optimization methods.

Simple query optimization

In order to provide a time and cost estimation of query plans running on virtual machines from multiple cloud providers, Gounaris et al. have used a bi-objective cost model. The authors extended existing methods—which used only time estimates—to be suitable for multi-cloud environments. In this method, resources used have a monetary cost. However, it suffers from two issues: the cost model must be amortized from the cost of the used resources (rather than the charged price) and it did not take into account all potential objectives, such as the security, reliability, and QoS. Sellami et al. defined a common data model and query algebra to process complex declarative queries. In this technique, queries are processed in multi-data stores named VDS, i.e., virtual data stores. Optimization phase is done by a mediator in two steps. Firstly, select and project operations are pushed down to the local data stores. This allows reducing the size of exchange data. Secondly, an optimal distributed plan is built by a dynamic programming method. The distributed plan seeks to minimize i/o, CPU, data shipping and transformation costs. However, this technique is limited to the ODBAPI query algebra and some query operators. Another approach is developed by Armbrust et al., who proposed an extension of the SQL compiler called PIQL query compiler. PIQL system is based on a declarative language allowing to express relationship cardinality and the result size requirements using a simple query plan optimizer. It provides scale independence by computing an upper bound on the number of key/value store operations that will be performed for any query. It supports three remote operators: IndexScan, Index-FKJoin, and SortedIndexJoin. PIQL optimizer execution engine is implemented as an iterator model allowing the execution of several operators with pipelined processing. However, some useful queries are disallowed by PIQL due to the constraint that the number of operations needs to have a compile-time upper-bound. Ding et al. have developed an efficient query processing optimization solution based on extreme learning machine. This framework called ELM_CMR was built under com-map-reduce framework, which is an improved version of map-reduce. ELM_CMR uses a classifier to build the optimization model. The classifier obtains the query classification results at run-time which will be sent to the query optimizer. The query optimizer uses classification results to select an optimized execution order. They proposed two implementations of ELM_CMR: for one query and for multiple queries. However, ELM_CMR framework requires user intervention.

Multiple-query optimization

To select the most appropriate MQO, i.e., multiple queries optimization able to improve the total execution time in a cloud relational database, Dokeroglu et al. proposed heuristic algorithms to select the best one over many alternative query plans. In their framework, a distributed query engine is used to detect the common sub-expressions. Based on the statistics of the database, the authors developed a cost model based on the total execution time of several queries. To optimize the MQO, they developed four different algorithms: Branch-and-Bound (B&B) algorithm, Genetic Algorithm (GA), Hill-Climbing (HC) algorithm, and Hybrid Genetic Hill-Climbing (Hybrid GHC) algorithm. Based on the total optimization time and solution quality, the authors showed that the GA algorithm has the best performance. A distributed query engine called CloudMdsQL was proposed by Kolev et al. The proposed engine supports querying heterogeneous cloud data stores. CloudMdsQL allows submitting queries using a functional query language to the query engine. In this approach, each engine node is composed of two parts, a master part, and a worker part. Both are able to exchange data or query sub-plans between them. Two levels of optimization are used in this approach. This level of optimization is done using a simple cost model and information stored in the database catalog which is replicated at all master nodes. At the master part level, queries are analyzed and optimized, sub-plans execution by the different workers are also monitored. At the worker' levels, local optimizations of sub-plan are achieved and then executed. Partial results are sent either to another worker or to the master. However, this technique ignores the fault tolerance aspect. Another approach is developed by Silva et al. who extended a Cascade-style optimizer. The proposed optimizer generates many possible query rewrites, then, selects the one with the lowest estimated cost. First, they used sub-expression fingerprints to identify common sub-expressions. The conventional optimization is extended to record the history of physical properties used in the earlier identified shared groups. Then, information about shared groups are propagated and LCA (least common ancestor) groups are identified. Finally, the query enforcing physical properties are re-optimized at the shared groups. This approach has been prototyped in SCOPE Microsoft's system for massive data analysis. Experimental analysis of both simple and large real-world scripts showed that the extended optimizer produces plans with 21–57% lower estimated costs. While Ge et al. focused on the second phase of the MQO technique. This phase consists to generate a global execution plan producing the minimum processing time for all queries when they are executed. They proposed the lineage signature (LS) approach for MQO based on common components specified in the first phase. LS approach is based on AST (abstract syntax tree) of SQL statements. They first used lineage analysis to process the set of recurring queries. Then, using signature extraction, they obtain signature values for each query layer in the original query set. Using the feedback principle of LSShare, the efficiency of the optimization LS approach will be equitably shared over time over the set of recurring queries. However, the LS approach can be combined with other MQO methods to further improve system performance.

Hybrid methods

In addition to the previously presented approaches, there exist hybrid methods that combine several techniques. For example, to provide on-demand provisioning cloud services while ensuring elasticity, Graefe et al. proposed a query processing approach that combines several prior works including B-trees partitioning technique, an adaptive merging technique for index optimization, views materialization, index, and deferred maintenance. The partitioned B-tree adds an integer indicating the partition to which each index entry belongs. It manages the differential information for multiple helper nodes. Then, adaptive merging optimizes index on-demand manner as a side-effect of query execution. However, the authors did not implement or evaluate their approach to demonstrate performances, scalability, and the elasticity of their solution.

Comparison and discussion

In this section, we propose to conduct an overall summary of the most relevant results about studied and surveyed solutions in Sect. 4. This summary will be done regarding general challenges then against specific challenges related to the cloud environments.

Related challenges

Indeed, many challenges related to the query optimization process including fault tolerance, load balancing, scalability, partitioning, and replication are crucial for cloud environments. Such challenges should be considered in current query optimization techniques.

We have carried out a comparison among the different challenges which are not specific for cloud computing, but important for the proper functioning of query optimization techniques.

Although most of the optimization techniques listed are able to support scalability, partitioning, and replication. However, many of them have not addressed load balancing and most algorithms of them have not designed to run with fault tolerance challenge. Therefore, it is necessary to improve the existing query optimization techniques and propose new techniques able to run with fault tolerance challenge.

Specific challenges

We first examine how the query optimization techniques deal with each challenge separately. Then, we compare them through, which summarizes and reviewed techniques using the identified challenges.

- *Volume and heterogeneity* Cloud computing solutions must be able to carry out, analysis and processing several types of queries over huge volumes of heterogeneous data. However, a dramatic increase in overlap will affect query efficiency when the dimension is high or data volume is large. Indeed, most of the previously presented techniques have addressed how managing issues related to the volume of data during the optimization process and how to assign data across data nodes. This can be explained by the greater effort devoted to developing big data solutions. However, the heterogeneity aspect has not received enough attention. Only a few works have considered heterogeneity aspect. This may be explained by the complexity of dealing with the heterogeneity aspect.
- *Pay-as-you-go* Cloud computing resources are allocated based on a quantum pricing system per period. Indeed, during query optimization steps some techniques use additional data such as caching and materialized views. However, such additional data implies an extra storage cost. As illustrated, some techniques have taken into account the pay-as-yougo paradigm. These solutions have considered pay-as-you-go paradigm in different ways: using a cost model, caching price, data-flows price, views materialization price. However, many other solutions did not take into account this aspect.
- *Elasticity* Elasticity is one of the main features of cloud computing. To properly ensure elasticity, some studies used directly the auto-scaling groups to spawn virtual machines. Some other solutions used horizontally partitioning of tables and tablets to be distributed across multiple nodes. Graefe et al. addressed the sporadic unavailability of the temporary node from elastic service by deferring maintenance of indexes and materialized views. However, if

the adding/removing nodes frequency becomes too high, it would have a considerable negative impact on queries performance, even with the proposed solutions. Therefore, much effort is required to handle the elasticity aspect.

- *Data security* Security and privacy of data are critical aspects that must be ensured by current technologies of data management systems in cloud environments. Only a few techniques of query optimization were proposed which considered the security aspect. When examining such approaches, we found that the encryption method before outsourcing of data is largely the most adopted solution. Thus, these approaches focused on the most effective scheme. The most drawbacks of these approaches are the complex calculation and communication required to achieve encryption and query answering.
- *Types of queries* Various types of queries have been studied in the cloud context. Indeed, great efforts have been made to address range queries. On the other side, some techniques have focused on aggregate queries, which are important for OLAP systems. Some other works addressed other query types including SPJ queries and multi-dimensional queries. However, little effort has been made to investigate skyline queries and top-k queries. As we have seen, optimization techniques typically focused only on one type of query in cloud environments. analyzes and compares these different presented query optimization techniques using the set defined challenges.

Open challenges

Cloud computing has clearly complicated the traditional approaches to query optimization. Although there have been great efforts made in this context, there remains a colossal amount of work in this field to reach more interesting performances in query optimization. Indeed, illustrates that pay-as-you-go paradigm is addressed using materialized views and caching. However, most of the works—which addressing security issue—are based on index techniques. Furthermore, the elasticity challenge is addressed by data partitioning-based methods.

Based on, we can see that there is no work that fulfills all the described challenges. Thus, there are still a number of aspects that require further investigations. Firstly, in the context of query type, current distributed cloud data management systems based on the key-value store cannot support complex query models such as range query, QNN query, and multi-dimensional query. Thus, we highlight the importance of developing optimization techniques dealing with a wide range of query types, especially more complex ones.

In another hand, illustrates that the most query optimization techniques focus on the volume to the detriment of heterogeneity of data, mainly due to the use of big data technologies in cloud platforms. Therefore, future works must address how to consider the heterogeneity aspect of both data and platforms in query optimization techniques. In the context of pay-as-you-go paradigm, we emphasize that future approaches of query processing must keep a balance between cloud resources cost and performance gain. Otherwise, they must get the best formula to price cloud resources as a quality of service delivery. Furthermore, there are relatively few works have considered the elasticity aspect. Therefore, further studies are suggested to more investigate this aspect.

In the context of the security aspect, proposed query optimization approaches addressing the security aspect have neither considered elasticity and nor economic aspect. Moreover, such studies used encryption techniques which use complex calculation and communication to achieve encryption and, thus high-query processing cost. Thus, we highlight that the trade-off between security and query efficiency must be carefully considered. It is clear that most of the examined techniques lack in considering the volume, elasticity, security, monetary, and heterogeneity aspects at the same time. Consequently, future work needs to consider all these challenges for providing enhanced query optimization methods. Furthermore, we reiterated the importance to examine the case of query optimization techniques over compressed data and how to get accurate statistics from autonomous data stores. Finally, some new challenges and factors have appeared in the cloud context such as energy efficiency. Indeed, the increase in energy consumption in cloud computing poses a severe threat to the environment. Thus, it is important to consider energy efficiency when developing new query optimization techniques.

Conclusion

In cloud environments, query optimization is an essential and crucial stage for massive data processing since their performances are directly felt by customers. However, in order to address the query optimization problem, one has to face several challenges.

In this study, we investigated the most important challenges related to query optimization in cloud environments. Following the proposed taxonomy, we have analyzed the literature regarding each challenge. We have presented a summary of this analysis considering all the described challenges. Our summary includes a comparison between studied works on the basis of the classical challenges of query optimization and those related to the cloud environments. This survey made it possible to suggest openings and opportunities as well as recommendations. Our study of existing research on query optimization techniques in cloud environments is an important and essential step for future work. It will help researchers either to adapt these techniques or to propose new techniques based on the constructive criticisms proposed in this study.

References

1. Curino C, Jones EPC, Popa RA, Malviya N, Madden E, Wu S, Balakrishnan H, Zeldovich N (2011) Relational cloud: a database-as-a-service for the cloud. In: Proceedings of the 5th Biennial Conference on Innovative Data Systems Research. Pacific Grove, CA, pp 235–241
2. Mansouri Y, Toosi AN, Buyya R (2018) Data storage management in cloud environments: taxonomy, survey, and future directions. ACM Comput Surv (CSUR) 50(6):91. https://doi.org/10.1145/3136623
3. Ioannidis YE (1996) Query optimization. ACM Comput Surv (CSUR) 28(1):121–123. https://doi.org/10.1145/234313.234367
4. DeWitt D, Gray J (1992) Parallel database systems: the future of high performance database systems. Commun ACM 35(6):85–98
5. Graefe G (1993) Query evaluation techniques for large databases. ACM Comput Surv (CSUR) 25(2):73–169. https://doi.org/10.1145/152610.152611