IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 5 (Jul. - Aug. 2013), PP 34-38
www.iosrjournals.org

Ethopian Database Management system as a Cloud Service: Limitations and advantages

Shaik Saidhbi
M.Tech(CSE) Lecturer , Department of Information Systems, UNIVERSITY OF GONDAR ETHOPIA

***Abstract**: An Ethiopian Cloud database management system is a distributed database that delivers computing as a service instead of a product. It is the sharing of resources, software, and information between multiple devices over a network which is mostly the internet. It is expected that this number will grow significantly in the future. As a result, there is a growing interest in outsourcing database management tasks to third parties that can provide these tasks for much lower cost due to the economy of scale just like putting it into the cloud. In this paper, we discuss the recent trend in database management system and the possibilities of making it as one of the services offered in the cloud. We also Recently the cloud computing paradigm has been receiving significant excitement and attention in the media and blogosphere. To some, cloud computing seems to be little more than a marketing umbrella, encompassing topics such as distributed computing, grid computing, utility computing, and software as- a-service, that have already received significant research focus and commercial implementation. Nonetheless, there exist an increasing number of large companies that are offering cloud computing infrastructure products and services that do not entirely resemble the visions of these individual component topics.*

In this article we discuss the limitations and opportunities of deploying data management issues on these emerging cloud computing platforms. We speculate that large scale data analysis tasks, decision support systems, and application specific data marts are more likely to take advantage of cloud computing platforms than operational, transactional database systems (at least initially). We then discuss some currently available open source and commercial database options that can be used to perform such analysis tasks, and conclude that none of these options, as presently architected, match the requisite features. We thus express the need for a new DBMS, designed specifically for Ethiopian cloud computing environments.

I. Introduction

Data management applications are potential candidates for deployment in the cloud. This is because an on premises enterprise database system typically comes with a large, sometimes prohibitive up-front cost, both in hardware and in software. For many companies (especially for start-ups and medium-sized businesses), the payas-you-go cloud computing model, along with having someone else worrying about maintaining the hardware, is very attractive. In this way, cloud computing is reminiscent of the application service provider (ASP) and database-as-a-service (DaaS) paradigms. In practice, cloud computing platforms, like those offered by Amazon Web Services, AT&T's Synaptic Hosting, AppNexus, GoGrid, Rackspace Cloud Hosting, and to an extent, the HP/Yahoo/Intel Cloud Computing Testbed, and the IBM/Google cloud initiative, work differently than ASPs and DaaS. Instead of owning, installing, and maintaining the database software for you (often in a multi-tenancy architecture), cloud computing vendors typically maintain little more than the hardware, and give customers a set of virtual machines in which to install their own software. Resource availability is typically elastic, with a seemingly infinite amount compute power and storage available on demand, in a pay-only-for-what-you-use pricing model.

This article explores the advantages and disadvantages of deploying database systems in the cloud. We look at how the typical properties of commercially available cloud computing platforms affect the choice of data management applications to deploy in the cloud. Due to the ever-increasing need for more analysis over more data in today's corporate world, along with an architectural match in currently available deployment options, we conclude that read-mostly analytical data management applications are better suited for deployment in the cloud than transactional data management applications. We thus outline a research agenda for large scale data analysis in the cloud, showing why currently available systems are not ideally-suited for cloud deployment, and arguing that there is a need for a newly designed DBMS, architected specifically for cloud computing platforms.

II. Data Management in the Cloud

Our goal in this section is to decide which data management applications are best suited for deployment on top of cloud computing infrastructure. In order to do this, we first discuss three characteristics of a cloud computing environment that are most pertinent to the ensuing discussion.

2.1 Cloud Characteristics

Compute power is elastic, but only if workload is parallelizable. One of the oft-cited advantages of cloud computing is its elasticity in the face of changing conditions. For example, during seasonal or unexpected spikes in demand for a product retailed by an e-commerce company, or during an exponential growth phase for a social networking Website, additional computational resources can be allocated on the fly to handle the increased demand in mere. Similarly, in this environment, one only pays for what one needs, so increased resources can be obtained to handle spikes in load and then released once the spike has subsided. However, getting additional computational resources is not as simple as a magic upgrade to a bigger,

For example, Amazon's Elastic Compute Cloud (EC2) apportions computing resources in small, large, and extra large virtual private server instances, the largest of which contains no more than four cores. If an application is unable to take advantage of the additional server instances by offloading some of its required work to the new instances which run in parallel with the old instances, then having the additional server instances available will not be much help.

In general, applications designed to run on top of a shared-nothing architecture are well suited for such an environment. Some cloud computing products, such as Google's App Engine, provide not only a cloud computing infrastructure, but also a complete software stack with a restricted API so that software developers are forced to write programs that can run in a shared-nothing environment and thus facilitate elastic scaling.

Data is stored at an untrusted host. Although it may not seem to make business sense for a cloud computing host company to violate the privacy of its customers and access data without permission, such a possibility nevertheless makes some potential customers nervous. In general, moving data off premises increases the number of potential security risks, and appropriate precautions must be made. Furthermore, although the name "cloud computing" gives the impression that the computing and storage resources are being delivered from a celestial location, the fact is, of course, that the data is physically located in a particular country and is subject to local rules and regulations. For example, in the United States, the US Patriot Act allows the government to demand access to the data stored on any computer; if the data is being hosted by a third party, the data is to be handed over without the knowledge or permission of the company or person using the hosting service. Since most cloud computing vendors give the customer little control over where data is stored the customer has little choice but to assume the

Worst and that unless the data is encrypted using a key not located at the host, the data may be accessed by a third party without the customer's knowledge.

Data is replicated, often across large geographic distances Data availability and durability is paramount for cloud storage providers, as data loss or unavailability can be damaging both to the bottom line and to business reputation. Data availability and durability are typically achieved through under-the-covers replication. Large cloud computing providers with data centers spread throughout the world have the ability to provide high levels of fault tolerance by replicating data across large geographic distances. Amazon's S3 cloud storage service replicates data across "regions" and "availability zones" so that data and applications can persist even in the face of failures of an entire location. The customer should be careful to understand the details of the replication scheme however; for example, Amazon's EBS (elastic block store) will only replicate data within the same availability zone and is thus more prone to failures.

2.2 Data management applications in the cloud

The above described cloud characteristics have clear consequences on the choice of what data management applications to move into the cloud. In this section we describe the suitability of moving the two largest components of the data management market into the cloud: transactional data management and analytical data management.

2.2.1 Transactional data management

By "transactional data management", we refer to the bread-and-butter of the database industry, databases that back banking, airline reservation, online e-commerce, and supply chain management applications. These applications typically rely on the ACID guarantees that databases provide, and tend to be fairly write-intensive. We speculate that transactional data management applications are *not* likely to be deployed in the cloud, at least in the near future, for the following reasons:

Transactional data management systems do not typically use a shared-nothing architecture. The transactional database market is dominated by Oracle, IBM DB2, Microsoft SQL Server, and Sybase [29]. Of these four products, neither Microsoft SQL Server nor Sybase can be deployed using a shared-nothing architecture. IBM released a shared-nothing implementation of DB2 in the mid-1990s which is now available as

a "Database Partitioning Feature" (DPF) add-on to their flagship product [4], but is designed to help scale analytical applications running on data warehouses, not transactional data management [5]. Oracle had no

shared-nothing implementation until very recently but again, this implementation is designed only to be used for data warehouses [6].

Implementing a transactional database system using a shared-nothing architecture is non-trivial, since data is partitioned across sites and, in general, transactions can not be restricted to accessing data from a single site. This results in complex distributed locking and commit protocols, and in data being shipped over the network leading to increased latency and potential network bandwidth bottlenecks.

There are enormous risks in storing transactional data on an untrusted host. Transactional databases typically contain the complete set of operational data needed to power mission-critical business processes. This data includes detail at the lowest granularity, and often includes sensitive information such as customer data or credit card numbers. Any increase in potential security breaches or privacy violations is typically unacceptable. We thus conclude that transactional data management applications are not well suited for cloud deployment. Despite this, there are a couple of companies that will sell you a transactional database that can run in Amazon's cloud:

2.2.2 Analytical data management

By "analytical data management", we refer to applications that query a data store for use in business planning, problem solving, and decision support. Historical data along with data from multiple operational databases are all typically involved in the analysis. Consequently, the scale of analytical data management systems is generally larger than transactional systems (whereas 1TB is large for transactional systems, analytical systems are increasingly crossing the petabyte barrier [25, 7]). Furthermore, analytical systems tend to be read-mostly,with occasional batch inserts. Analytical data management consists of $3.99 billion [35] of the $15.6 billion database market [39] (37%) and is growing at a rate of 11.3% annually [35]. We speculate that analytical data management systems are well-suited to run in a cloud environment, and will be among the first data management applications to be deployed in the cloud, for the following reasons:

Shared-nothing architecture is a good match for analytical data management. Teradata, Netezza, Greenplum, DATAllegro (recently acquired byMicrosoft), Vertica, and Aster Data all use a shared-nothing architecture in their analytical DBMS products, with IBM DB2 and recently Oracle also adding shared-nothing analytical products. The ever increasing amount of data involved in data analysis workloads is the primary driver behind the choice of a shared-nothing architecture, as the architecture is widely believed to scale the best. Furthermore, data analysis workloads tend to consist of many large scan scans, multidimensional aggregations, and star schema joins, all of which are fairly easy to parallelize across nodes in a shared-nothing network. Finally, the infrequent writes in the workload eliminates the need for complex distributed locking and commit protocols.

ACID guarantees are typically not needed. The infrequent writes in analytical database workloads, along with the fact that it is usually sufficient to perform the analysis on a recent snapshot of the data makes the 'A', 'C', and 'I' of ACID easy to obtain. Hence the consistency tradeoffs that need to be made as a result of the distributed replication of data in transactional databases are not problematic for analytical databases.

III. Data Analysis in the Cloud

Now that we have settled on analytic database systems as a likely segment of the DBMS market to move into the cloud, we explore various currently available software solutions to perform the data analysis. We focus on two classes of software solutions: MapReduce-like software, and commercially available shared-nothing parallel databases. Before looking at these classes of solutions in detail, we first list some desired properties and features that these solutions should ideally have.

3.1 Cloud DBMS Wish List

Efficiency. Given that cloud computing pricing is structured in a way so that you pay for only what you use,the price increases linearly with the requisite storage, network bandwidth, and compute power. Hence, if data analysis software product A requires an order of magnitude more compute units than data analysis software product B to perform the same task, then product A will cost (approximately) an order of magnitude more than B. Efficient software has a direct effect on the bottom line.

Fault Tolerance. Fault tolerance in the context of analytical data workloads is measured differently than fault tolerance in the context of transactional workloads. For transactional workloads, a fault tolerant DBMS can recover from a failure without losing any data or updates from recently committed transactions, and in the context of distributed databases, can successfully commit transactions and make progress on a workload even in the face of worker node failure. For read-only queries in analytical workloads, there are no write transactions to commit, nor updates to lose upon node failure. Hence, a fault tolerant analytical DBMS is simply one that does not have to restart a query if one of the nodes involved in query processing fails. Given the large

of amount of data that needs to be accessed for deep analytical queries, combined with the relatively weak compute capacity of a typical cloud compute server instance, complex queries can involve hundreds of server instances and can take hours to complete. Furthermore, clouds are typically built on top of cheap, commodity hardware, for which failure is not uncommon. Consequently, the probability of a failure occurring during a long-running data analysis task is relatively high; Google, for example, reports an average of 1.2 failures per analysis job [16]. If a query must restart each time a node fails, then long, complex queries are difficult to complete.

Ability to run in a heterogeneous environment. The performance of cloud compute nodes is often not consistent, with some nodes attaining orders of magnitude worse performance than other nodes. There are a variety of reasons why this could occur, ranging from hardware failure causing degraded performance on a node , to an instance being unable to access the second core on a dual-core machine [8], to contention for non-virtualized resources. If the amount of work needed to execute a query is equally divided amongst the cloud compute nodes, then there is a danger that the time to complete the query will be approximately equal to time for the slowest compute node to complete its assigned task. A node observing degraded performance would thus have a disproportionate affect on total query latency. A system designed to run in a heterogeneous environment would take appropriate measures to prevent this from occurring.

3.2. MapReduce-like software

MapReduce and related software such as the open source Hadoop , useful extensions , and Microsoft's Dryad/SCOPE stack are all designed to automate the parallelization of large scale data analysis workloads.

Although DeWitt and Stonebraker took a lot of criticism for comparing MapReduce to database systems in their recent controversial blog posting,a comparison is warranted since MapReduce is in fact a useful tool for performing data analysis in the cloud . The MapReduce programming model and framework implementation satisfies many of the previously stated desired properties:

Fault Tolerance. MapReduce is designed with fault tolerance as a high priority. A data analysis job is divided into many small tasks and upon a failure, tasks assigned to a failed machine are transparently reassigned to another machine. Care is taken to make sure that partially executed tasks are not doubly accounted for in the final query result. In a set of experiments in the original MapReduce paper, it was shown that explicitly killing 200 out of 1746 worker processes involved in a MapReduce job resulted in only a 5% degradation in query performance .

Ability to run in a heterogeneous environment. MapReduce is also carefully designed to run in a heterogeneous environment. Towards the end of a MapReduce job, tasks that are still in progress get redundantly executed on other machines, and a task is marked as completed as soon as either the primary or the backup execution has completed. This limits the effect that "straggler" machines can have on total query time, as backup executions of the tasks assigned to these machines will complete first. In a set of experiments in the original MapReduce paper, it was shown that backup task execution improves query performance by 44% by alleviating the adverse affect caused by slower machines.

Ability to operate on encrypted data. Neither MapReduce, nor its derivatives, come with a native ability to operate on encrypted data. Such an ability would have to be provided using user-defined code.

Ability to interface with business intelligence products. Since MapReduce is not intended to be a database system, it is not SQL compliant and thus it does not easily interface with existing business intelligence products.

Efficiency. The efficiency and raw performance of MapReduce is a matter of debate. A close inspection of the experimental results presented in the MapReduce paper would seem to indicate that there is room for performance improvement.

3.3 Shared-Nothing Parallel Databases

Amore obvious fit for data analysis in the cloud are the commercially available shared-nothing parallel databases, such as Teradata, Netezza, IBM DB2, Greenplum, DATAllegro, Vertica, and Aster Data, that already hold a reasonable market share for on-premises large scale data analysis [35]. DB2, Greenplum, Vertica, and Aster Data are perhaps the most natural fit since they sell software-only products that could theoretically run in the data centers hosted by cloud computing providers. Vertica already markets a version of its product designed to run in Amazon's cloud .

Parallel databases implement a largely complimentary set of properties from our wish list relative toMapReducelike software:

Ability to interface with business intelligence products. Given that the business intelligence products are designed to work on top of databases, this property essentially comes for free. More mature databases, such as DB2, tend to have carefully optimized and certified interfaces with a multitude of BI products.

Efficiency At the cost of the additional complexity in the loading phase discussed in Section 3.2, parallel databases implement indexes, materialized views, and compression to improve query performance.

Fault Tolerance. Most parallel database systems restart a query upon a failure. This is because they are generally designed for environments where queries take no more than a few hours and run on no more than a few hundred machines. Failures are relatively rare in such an environment, so an occasional query restart is not problematic. In contrast, in a cloud computing environment, where machines tend to be cheaper, less reliable, less powerful, and more numerous, failures are more common. Not all parallel databases, however, restart a query upon a failure; Aster Data reportedly has a demo showing a query continuing to make progress as worker nodes involved in the query are killed .

Ability to run in a heterogeneous environment. Parallel databases are generally designed to run on homogeneous equipment and are susceptible to significantly degraded performance if a small subset of nodes in the parallel cluster are performing particularly poorly.

Ability to operate on encrypted data. Commercially available parallel databases have not caught up to the recent research results on operating directly on encrypted data. In some cases simple operations are supported, but advanced operations, such as performing aggregations on encrypted data, is not directly supported. It should be noted, however, that it is possible to hand-code encryption support using user defined functions.

3.4 A Call For A Hybrid Solution

It is now clear that neither MapReduce-like software, nor parallel databases are ideal solutions for data analysis in the cloud. While neither option satisfactorily meets all five of our desired properties, each property is met by at least one of the two options. Hence, a hybrid solution that combines the fault tolerance, heterogeneous cluster, and ease of use out-of-the-box capabilities of MapReduce with the efficiency, performance, and tool plugability of shared-nothing parallel database systems could have a significant impact on the cloud database market.

SCOPE project at Microsoft] aim to integrate declarative query constructs from the database community into MapReduce-like software to allow greater data independence, code reusability, and automatic query optimization.

Greenplum and Aster Data have added the ability to write MapReduce functions over data stored in their parallel database products . Although these four projects are without question an important step in the direction of a hybrid solution, there remains a need for a hybrid solution at the systems level in addition to at the language level.

One interesting research question that would stem from such a hybrid integration project would be how to combine the ease-of-use out-of-the-box advantages of MapReduce-like software with the efficiency and sharedwork advantages that come with loading data and creating performance enhancing data structures. Incremental algorithms are called for, where data can initially be read directly off of the file system out-of-the-box, but each time data is accessed, progress is made towards the many activities surrounding a a DBMS load.

Another interesting research question is how to balance the tradeoffs between fault tolerance and performance. Maximizing fault tolerance typically means carefully checkpointing intermediate results, but this usually comes at a performance cost (e.g., the rate which data can be read off disk in the sort benchmark from the original MapReduce paper is half of full capacity since the same disks are being used to write out intermediate Map output). A system that can adjust its levels of fault tolerance on the fly given an observed failure rate could be one way to handle the tradeoff.

The bottom line is that there is both interesting research and engineering work to be done in creating a hybrid MapReduce/parallel database system.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proc. of SIGMOD*, pages 563–574, 2004.

[2] AmazonWeb Services. SimpleDB. Web Page. http://aws.amazon.com/simpledb/.

[3] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska. Building a Database on S3. In *Proc. of SIGMOD* pages 251–264, 2008.

[4] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: Easy and efficient parallel processing of massive data sets. In *Proc. of VLDB*, 2008.

[5] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[6] H. Hacig¨um¨us¸, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proc. of SIGMOD*, pages 216–227, 2002.

[7] Hadoop Project. Welcome to Hadoop! Web Page. http://hadoop.apache.org/core/.