Journal of Advances in Science and Technology

Vol. IV, No. VIII, February-2013, ISSN 2230-9659

REVIEW ARTICLE

REVIEW OF CLOUD COMPUTING & EVALUATION OF DATABASE FOR THEIR COMPATIBILITY WITH CLOUD COMPUTING

www.ignited.in

Journal of Advances in Science and Technology
Vol. IV, No. VIII, February-2013, ISSN 2230-9659

Review of Cloud Computing & Evaluation of Database for Their Compatibility with Cloud Computing

Surbhi Agarwal

Research Scholar, CMJ University, Shillong, Meghalaya, India

----------♦----------

INTRODUCTION

The potential benefits of cloud computing are overwhelming. However, attaining these benefits requires that each aspect of the cloud platform support the key design principles of the cloud model. One of the core design principles is dynamic scalability, or the ability to provision and decommission servers on demand. Unfortunately, the majority of today's database servers are incapable of satisfying this requirement. This paper reviews the benefits of cloud computing and then evaluates two database architectures—shared-disk and shared-nothing—for their compatibility with cloud computing.

Cloud computing is the latest evolution of Internet-based computing. The Internet provided a common infrastructure for applications. Soon, static web pages began to add interactivity. This was followed by hosted applications like Hotmail. As these web applications added more user-configuration, they were renamed Software-as-a-Service (SaaS). Companies like Salesforce.com have led this wave.

With a growing number of companies looking to get in on the SaaS opportunity, Amazon released Amazon Web Services (AWS) that enables companies to operate their own SaaS applications. In effect, Amazon hosted the LAMP stack, which they have since expanded to include Windows as well. Soon others followed suit. Then, large companies began to realize that they could create their own cloud platform for internal use, a sort of private cloud.

So, just as the public Internet spawned private corporate intranets, cloud computing is now spawning private cloud platforms. Both public and private cloud platforms are looking to deliver the benefits of cloud computing to their customers. Whether yours is a private or public cloud, the database is a critical part of that platform. Therefore it is imperative that your cloud database be compatible with cloud computing. In order to understand cloud computing requirements, we must first understand the benefits that drive these requirements.

The shared-disk database architecture is ideally suited to cloud computing. The shared-disk architecture requires fewer and lower-cost servers, it provides high-availability, it reduces maintenance costs by eliminating partitioning, and it delivers dynamic scalability.

THE BENEFITS OF CLOUD COMPUTING

Cloud computing is not a fad, it is driven by some tangible and very powerful benefits. Whether the cloud is provided as an internal corporate resource, as a service hosted by a third-party, or as a hybrid of these two models, there are some very real advantages to this model. These advantages derive from specialization and economies of scale:

Specialization: There is a great deal of specialized knowledge required to set-up and operate systems to address security, scalability, platform maintenance (patches, updates), data maintenance (backups) and more. In a traditional model, each development effort had to include this expertise on staff. Cloud computing enables these capabilities to be staffed by experts who are shared across many customers. Instead of hiring that one person who does a decent job across all of these elements, cloud computing entities can hire individuals with deep expertise in each area, and then amortize this expense across a large number of customers. This degree of specialization enables a variety of benefits that are driving cloud computing.

Economies of Scale: This is also a powerful driver for cloud computing. The ideal platform is very expensive to build. The servers, networking equipment, data storage/backup, power, redundant high-speed connectivity, etc. can result in a huge start-up cost for a single product or project. Add to this the fact that most development efforts fail, and the economics simply don't make sense for investment of this level in each project. Cloud computing enjoys economies of scale, because that same investment can be amortized over a large number of projects. If one project fails, it can be replaced by a number of new

projects that continue to amortize the initial investment.

Economies of scale also apply to IT tasks. For example, let us use backup as an example of a standard IT task. In a standalone environment, an IT person might schedule and manage the backup process. In a cloud environment, backup is highly automated, whereby that same IT person can oversee simultaneous backups for hundreds or thousands of customers.

KEY BENEFITS OF CLOUD COMPUTING:

- Lower costs: All resources, including expensive networking equipment, servers, IT personnel, etc. are shared, resulting in reduced costs, especially for small to mid-sized applications and prototypes.

- Shifting CapEx to OpEx: Cloud computing enables companies to shift money from capital expenses (CapEx) to operating expenses (OpEx), enabling the customer to focus on adding value in their areas of core competence, such as business and process insight, instead of building and maintaining IT infrastructure. In short, cloud computing allows you to focus your money and resources on innovating.

- Agility: Provisioning-on-demand enables faster set-up and tear-down of resources on an as-needed basis. When a project is funded, you initiate service, then if the project is killed, you simply terminate the cloud contract.

- Dynamic scalability: Most applications experience spikes in traffic. Instead of over-buying your own equipment to accommodate these spikes, many cloud services can smoothly and efficiently scale to handle these spikes with a more cost-effective pay-as-you-go model. This is also known as elasticity and is behind Amazon's name Elastic Computing Cloud (EC2).

- Simplified maintenance: Patches and upgrades are rapidly deployed across the shared infrastructure, as are backups.

- Large scale prototyping/load testing: Cloud computing makes large scale prototyping and load testing much easier. You can easily spawn 1,000 servers in the cloud to load test your application and then release them as soon as you are done, try doing that with owned or corporate servers.

- Diverse platform support: Many cloud computing services offer built-in support for a rich collection of client platforms, including browsers, mobile, and more. This diverse platform support enables applications to reach a broader base of users right out of the gate.

- Faster management approval: This is closely aligned with cost savings. Since cloud computing has very low upfront costs, the management approval process is greatly accelerated, causing faster innovation. In fact, costs are so low, that individuals can easily fund the expense personally to demonstrate the benefits of their solution, while avoiding organizational inertia.

- Faster development: Cloud computing platforms provide many of the core services that, under traditional development models, would normally be built in house. These services, plus templates and other tools can significantly accelerate the development cycle.

The combination of these benefits is driving cloud computing from mere buzzword to disruptive and transformational tsunami.

With corporate adoption of cloud computing, we are seeing an explosion of cloud options. One of those options is the provisioning of database services in the form of cloud databases or Database-as-a-Service (DaaS). For the remainder of this paper, we focus on the requirements of cloud databases and the various options available to you.

EVOLVING CLOUD DATABASE REQUIREMENTS

Cloud database usage patterns are evolving, and business adoption of these technologies accelerates that evolution. Initially, cloud databases serviced consumer applications. These early applications put a priority on read access, because the ratio of reads to writes was very high. Delivering high-performance read access was the primary purchase criteria. However, this is changing.

Consumer-centric cloud database applications have been evolving with the adoption of Web 2.0 technologies. User generated content, particularly in the form of social networking, have placed somewhat more emphasis on updates. Reads still outnumber writes in terms of the ratio, but the gap is narrowing. With support for transactional business applications, this gap between database updates and reads is further shrinking. Business applications also demand that the cloud database be ACID compliant: providing Atomicity, Consistency, Isolation and Durability.

Perhaps it will be beneficial to consider two examples to better understand the differing cloud database requirements.

EXAMPLE 1: CONSUMER CLOUD DATABASE

Consider a database powering a consumer-centric cosmetics website. If the user does a search for a certain shade of lipstick, it is important that the results be delivered instantaneously to keep the user

engaged, so she doesn't click on another cosmetics site. If the site said that the chosen lipstick is in inventory and completed the sale, it wouldn't be the end of the world to later find out that, as a result of inconsistent data, that lipstick wasn't really in inventory. In this case, the consumer receives an email explaining that it is on backorder and will be shipped soon…no problem.

EXAMPLE 2: CORPORATE CLOUD DATABASE

Consider a company that sells widgets to manufacturers. A large company purchases a load of widgets necessary to keep its production line running. In this example, if the inventory was incorrect, due to inconsistent data, and the shipment is delayed, the company who purchased the widgets may be forced to shut down a production line at a cost of $1,000,000 per day…big problem!

With this understanding of the different stakes involved, it is easy to understand how corporate adoption of cloud databases are changing the game considerably.

THE ACHILLES HEEL OF CLOUD DATABASES

Dynamic scalability—one of the core principles of cloud computing—has proven to be a particularly vexing problem for databases. The reason is simple; most databases use a shared-nothing architecture. The shared-nothing architecture relies on splitting (partitioning) the data into separate silos of data, one per server.

You might think that dynamically adding another database server is as simple as splitting the data across one more server. For example, if you have two servers, each with 50% of the total data, and you add a third server, you just take a third of the data from each server and now you have three servers each owning 33% of the data. Unfortunately, it isn't that simple.

Many user requests involve related information. For example, you might want to find all customers who placed an order in the last month. You need to go to the invoices table and find the invoices dated for last month. Then you follow a database key to the customer table to collect their contact information. If this is spread across multiple servers, you end-up processing information on one machine and then passing that data to the second machine for processing. This passing of information, called data shipping, will kill your database performance. For this reason, the partitioning of the data must be done very carefully to minimize data shipping. Partitioning data, a time-consuming process, is referred to as a black art because of the level of skill required. The ability to partition data in an efficient and high-performance manner really separates the men from the boys in the world of DBAs. Automating this process remains an elusive goal.

Sure you can use middleware to automatically repartition the data on the fly to accommodate a changing number of database servers, but your performance can quickly go down the toilet. If we use the example above, let's say that you have two servers with partitioned data and a query is taking .5 seconds. Then you add a third database server, dynamically repartition the data with some middleware, and now that same query takes 1.0 seconds, because of the data shipping between nodes. Yes, the performance can actually decrease with the addition of more servers. This is the Achilles Heel of deploying a shared-nothing database in the cloud.

ARE REPLICATED TABLES THE ANSWER?

Since data partitioning and cloud databases are inherently incompatible, Amazon, Facebook and Google have taken another approach to solve the cloud database challenge. They have created a persistence engine—technically not a database—that abandons typical ACID compliance in favor replicated tables of data that store and retrieve information while supporting dynamic or elastic scalability. Facebook offers BigTable, Amazon has SimpleDB and Facebook is working on Cassandra. These solutions are ideal for the needs defined in the consumer example #1 above. However, they are not a replacement for a real database, and they do not address corporate cloud computing requirements.

THE SHARED-DISK DATABASE ARCHITECTURE IS IDEAL FOR CLOUD DATABASES

The database architecture called shared-disk, which eliminates the need to partition data, is ideal for cloud databases. Shared-disk databases allow clusters of low-cost servers to use a single collection of data, typically served up by a Storage Area Network (SAN) or Network Attached Storage (NAS). All of the data is available to all of the servers, there is no partitioning of the data. As a result, if you are using two servers, and your query takes .5 seconds, you can dynamically add another server and the same query might now take .35 seconds. In other words, shared-disk databases support elastic scalability.

The shared-disk DBMS architecture has other important advantages—in addition to elastic scalability—that make it very appealing for

deployment in the cloud. The following are some of these advantages:

Fewer servers required: Since shared-nothing databases break the data into distinct pieces, it is not sufficient to have a single server for each data set, you need a back-up in case the first one fails. This is called a master-slave configuration. In other words, you must duplicate your server infrastructure. Shared-disk is a master-master configuration, so each node provides fail-over for the other nodes. This reduces the number of servers required by half when using a shared-disk database.

Lower cost servers (extend the life of your current servers): In a shared-nothing database, each server must be run at low CPU utilization in order to be able to accommodate spikes in usage for that server's data. This means that you are buying large (expensive) servers to handle the peaks. Shared-disk, on the other hand, spreads these usage spikes across the entire cluster. As a result, each system can be run at a higher CPU utilization. This means that with a shared-disk database you can purchase lower-cost commodity servers instead of paying a large premium for high-end computers. This also extends the lifespan of existing servers, since they needn't deliver cutting-edge performance.

Scale-in: The scale-in1 model enables cloud providers to allocate and bill customers on the basis of how many instances of a database are being run on a multi-core machine. Scale-in enables you to launch one instance of MySQL per CPU core. For example, a 32-core machine could support a cluster-in-a-box of 32 instances of MySQL. Simplified maintenance/upgrade process: Servers that are part of a shared-disk database can be upgraded individually, while the cluster remains online. You can selectively take nodes out of service, upgrade them, and put them back in service while the other nodes continue to operate. You cannot do this with a shared-nothing database because each individual node owns a specific piece of data. Take out one server in a shared-nothing database and the entire cluster must be shut down. High-availability: Because the nodes in a shared-disk database are completely interchangeable, you can lose nodes and your performance may degrade, but the system keeps operating. If a shared-nothing database loses a server the system goes down until you manually promote a slave to the master role. In addition, each time you (re)partition the database, you must take the system down. In other words, shared-nothing involves more scheduled *and* unscheduled downtime than shared-disk systems. Reduced partitioning and tuning services: In a shared-nothing cloud database, the data must be partitioned. While it is fairly straightforward to simply split the data across servers, thoughtfully partitioning the data to minimize the traffic between nodes in the cluster—also known as function or data shipping—requires a great deal of ongoing analysis and tuning. Attempting to accomplish this in a static shared-nothing cluster is a significant challenge, but attempting to do so with a dynamically scaling database cluster is a Sysiphian task. Reduced support costs: One of the benefits of cloud databases is that they shift much of the low-level DBA functions to experts who are managing the databases in a centralized manner for all of the users. However, tuning a shared-nothing database requires the coordinated involvement of both the DBA and the application programmer. This significantly increases support costs. Shared-disk databases cleanly separate the functions of the DBA and the application developer, which is ideal for cloud databases. Shared-disk databases also provide seamless load-balancing, further reducing support costs in a cloud environment.

CONCLUSION

Whether you are assembling, managing or developing on a cloud computing platform, you need a cloud-compatible database. Shared-nothing databases require data partitioning, which is structurally incompatible with dynamic scalability, a core foundation of cloud computing. The shared-disk database architecture, on the other hand, does support elastic scalability. It also supports other cloud objectives such as lower costs for hardware, maintenance, tuning and support. It delivers high-availability in support of Service Level Agreements (SLAs). As with every tectonic shift in technology, there is a Darwinian ripple effect as we realize which technologies support these changes and which are relegated to legacy systems. Because of their compatibility, cloud computing will usher in an ascendance of the shared-disk database.

REFERENCES:-

- Cisco Systems. *Cisco Catalyst 3750-E Series Switches Data Sheet*, June 2008.
- J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. MAD Skills: New Analysis Practices for Big Data. *Under Submission*, March 2009.
- J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI '04*, pages 10–10, 2004.
- D. J. DeWitt and R. H. Gerber. Multiprocessor Hash-based Join Algorithms. In *VLDB '85*, pages 151–164, 1985.
- D. J. DeWitt, R. H. Gerber, G. Graefe, M. L. Heytens, K. B. Kumar, and M. Muralikrishna. GAMMA - A High Performance Dataflow Database Machine. In *VLDB '86*, pages 228–237, 1986.
- S. Fushimi, M. Kitsuregawa, and H. Tanaka. An Overview of The System Software of A Parallel

Relational Database Machine. In *VLDB '86*, pages 209–219, 1986.

- S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, 2003.

- M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *EuroSys '07*, pages 59–72, 2007.

- E. Meijer, B. Beckman, and G. Bierman. LINQ: reconciling object, relations and XML in the .NET framework. In *SIGMOD '06*, pages 706–706, 2006.