AWS Big Data Blog

How do smart businesses achieve success with their great startup information? Carrying big information (and machine learning) during the day provides many benefits. AWS enables large-scale services, such as AWS as a broad range of needs and adapting resources, stability and affordability, and managing the services provided by today, a highly developed and operational environment. data analysis. Information engineers, construction, information scientists, and IT employees can focus their efforts on providing information and extracting valuable insights.

Services like Amazon EMR, AWS Glue, and Amazon S3 allow you to decouple and scale your classification and storage independently, while providing a unified, well-managed, highly resilient environment, quickly reducing many of the problems of on-site access . This method leads to faster, easier, easier to use, and more cost-effective data and lake data plans.

However, the common experience on Apache Hadoop and Apache Spark sites is not always the best guide to cloud-based cloud computing. An easy to move and modify method in running cluster nodes in the cloud is easy but suboptimal in the model. Different options go a long way in increasing your revenue as you travel a lot of information in a structured cloud.

The main advantage of using Amazon EBS in deploying Cassandra is that it significantly reduces data traffic when a node fails or needs to be replaced. The alternate node joins the cluster much faster. However, Amazon EBS could be more expensive, depending on your data storage needs.

Cassandra has built-in fault tolerance by copying data to partitions via an adjustable number of nodes. Not only can it pass node errors, but if a node fails, it can also be recovered by copying data from other replicas to the new node. This can mean copying tens of gigabytes of data, depending on your application. This adds additional delay to the recovery process, increases network traffic, and can affect the performance of the Cassandra cluster during recovery.

Data stored on Amazon EBS is retained in the event of a failure or termination of the instance. The node data stored on the EBS remains intact and the EBS space can be installed on the new EC2 instance. Most duplicate data for an alternate node is already available in the EBS notebook and will not need to be copied over the network from another node. Only changes made after the failed original node need to be downloaded over the network. This makes this process much faster.

EBS quantities are recorded periodically. So if the volume fails, you can create a new volume from the last known good shot and attach it to a new instance. This is faster than creating a new notebook and mastering all the data.

Most Cassandra introductions use a recurrence factor of three. However, Amazon EBS mimics its own copy under the covers for fault tolerance. In practice, EBS quantities are about 20 times more reliable than conventional disk drives. So a repetition factor of two can be used. This not only saves costs, but also allows for deployments in a region that has two availability zones.

By default, multi-level Amazon Redshift configurations automatically store unnecessary copies of table data on other nodes in the same cluster, so we can tolerate multiple disk and node failures. As with all synchronous data replications, some overhead is associated with the process. In the case of temporary tables, this replication process does not occur because the system assumes that the data is truly temporary. In addition, blocks associated with temporary tables are not included in the threshold that activates automatic backups. So by using temporary tables for really short-term data, we can also avoid starting backup processes and improve the speed of completing automatic backup processes - just as we did with the BACKUP NO option.

Because temporary table data is not mirrored, operations that perform temporary table data are less expensive than those that provide permanent table data. This feature means you can significantly reduce cluster overheads by using temporary tables where they are appropriate. Examples include scenarios where the table data needs to be available in only one session and is really temporary.

Sources:

1. https://aws.amazon.com/blogs/big-data/amazon-emr-migration-guide/

2. https://aws.amazon.com/blogs/big-data/best-practices-for-running-apache-cassandra-on-amazon-ec2/

3. https://aws.amazon.com/blogs/big-data/amazon-redshift-engineerings-advanced-table-design-playbook-table-data-durability/