MapReduce and object stores – How car better?

Gil Vernik
Published on May 19, 2017
0

In my previous post, I covered general aspects of Hadoop and object stores. In this post, I'll offer s MapReduce works with object stores, highlight potential issues related to object stores and Hadoop make this integration work better.

As I described in my previous post, Hadoop can work with object stores without having to copy data File System (HDFS) cluster. Inspired by this, my team did a simple experiment with Apache Spark 2 persisted a collection of 9 elements as a data object in the object store.

```
val data = Array(1,2,3,4,5,6,7,8,9)
val myData = sc.parallelize(data,9)
myData.saveAsTextFile("s3a://mybucket/data.txt")
```

The following table shows the summary of HTTP requests that were sent.

	API	GET	HEAD	PUT	DELETE
Hadoop (s3a)	S3	158	361	26	16

We were surprised to see such a high number of requests to the object store. Because we observed other Hadoop connectors, we needed to better understand what caused the problem. After studyin concluded that Hadoop connectors are not optimized to work with object stores. The issues we disc Spark as well as the broader Hadoop ecosystem, including Hadoop MapReduce and Hive.

What causes the inefficiency?

The primary reasons why Hadoop connectors don't work with object stores are that the standard ou adapted for object stores, the cost of supporting FS shell operations is excessive, and the complexit as file systems.

Standard output committers are not adapted for object stores

Hadoop uses OutputCommitters to write data into underlying storage. To achieve fault tolerance, ea writes data into a temporary location. During the task commit phase, data is moved from the tempo destination name. As previously explained, rename operations are not native object store operation temporary files seriously reduces the overall performance. For more information, check out the "St Connecting Object Stores to Spark" blog post and the "Exabytes, elephants, objects and spark" blo

The excessive cost of supporting shell file systems operations

File System (FS) shell operations in Hadoop are crucial for the work with HDFS. Object stores, howe shell operations and provide their own CLI tools, which are adapted to work with object stores.

Let's look back to the example explaining that object stores have different semantics for directories

```
./bin/hadoop fs-mkdirs hdfs://myhdfs/a/b/c/
```

But this time, let's see how it works.

The actual flow performs two loops. The first one to loop over "a/b/c/" for each entry verifies that th file. Then it will loop over "a/b/c/" again and create all three directories recursively. While this appro systems, it's inefficient for object stores and will create too many RESTfull calls that basically achie existing Hadoop object store connectors, we observed that all of them had complex logic and used shell operations.

How does all this affect analytic flows? This time, let's go back to our example of persisting a collec how it works behind the scenes.

Each task is responsible to persist only its relevant data part. Thus, for example, task 0 is responsib an object. However, it will initially create a temporary object.

```
/data.txt/_temporary/0/_temporary/attempt_201705040936_0000_m_000000_0/part-0000
```

During the task commit phase, this temporary object will be renamed to the final name.

```
/data.txt/part-0000
```

The main issue here is that Hadoop connectors will not distinguish between FS shell operations and treat the temporary object URI consisting of directory `_temporary/0/_temporary/attempt_20170504` an object with name `part-0000`.

Creating `_temporary/0/_temporary/attempt_201705040936_0000_m_000000_0` recursively will generat GET, and PUT requests. And this will happen for each task. Next, it will also remove them recursivel large number of DELETE requests. Without supporting FS shell operations, existing connectors coul with a single PUT request.

The important observation here is that analytic flows don't need shell-like operations at all, and the significantly reduces the overall performance of the Hadoop connectors.

Complexity of treating object stores as HDFS

To move from HDFS to object stores, you might need to brush up some of your skills and perhaps ch scripts that were designed for file systems. For example, consider a script that creates a directory, u HDFS, runs a word count application, and deletes the data from HDFS after the job is complete. This for HDFS, but doesn't make sense for object stores.

First, there is no need to create a directory to upload an object. And if the object is already uploade deleting it, since object storage is not a temporary storage solution. Object store is not a file system some changes to existing scripts to use object store efficiently.

Our solution: Stocator

So, what can be done? The good news is that we have a solution. Stocator is a connector for object s developed in IBM Research and released to the open source under Apache license 2.0.

Stocator is explicitly designed for object stores and has very a different architecture from the existi doesn't depend on the Hadoop modules and interacts directly with object stores. Stocator addresse described here. For more information about Stocator, visit the project page.

Where to find Stocator?

IBM Data Science Experience is based on Stocator. Stocator enables Spark in IBM's Data Science Ex IBM Cloud Object Storage more efficiently than native Hadoop connectors. Running the same exam following results comparing Stocator with Hadoop connector:

	API	GET	HEAD	PUT	DELETE
Hadoop (s3a)	S3	158	361	26	16
Stocator	S3	1	2	11	0

Stocator in DSX can be used to analyze large amounts of data. In fact, the SETI is using it to process be discussing the SETI use case with Graham Mackintosh in our presentation "Very large data files, learning – lessons learned while looking for signs of extra-terrestrial life" at the Spark Summit, SF 2

I also invite you to attend the talk I will present with Trent-Gray Donald on "Hadoop and object stor at Hadoop Strata 2017, 22-25 May, London.

And of course, I invite you to join IBM DSX and be part of that exciting experience of using Stocator and data stored in IBM Cloud Object Storage.

TAGS MAPREDUCE, OBJECT STORES, STOCATOR

Gil Vernik

Twitter

I am a researcher in IBM, storage clouds, security and analytics group. I received my PhD degree at Mathematics from the University of Haifa and completed post doctoral position in Germany. In IBM i work with Apache Spark, Hadoop, Object Stores, no-SQL databases. I have more than 25 years of experience as a code developer, both server side and client side, know Java, Python, Scala, C/C++, Erlang

---

Join The Discussion

Your email address will not be published. Required fields are marked *

Enter your comments...

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.