# Stocator: A High Performance Object Store Connector for Spark*

Gil Vernik, Michael Factor, Elliot K. Kolodner, Effi Ofer
IBM Research – Haifa
{gilv, factor, kolodner, effio}@il.ibm.com

Pietro Michiardi, Francesco Pace
Eurecom
{pietro.michiardi, francesco.pace}@eurecom.fr

## CCS Concepts

•**Information systems → MapReduce-based systems; Cloud based storage;**

Data is the natural resource of the 21st century. It is being produced at dizzying rates, e.g., for genomics by sequencers, for Media and Entertainment with very high resolution formats, and for Internet of Things (IoT) by multitudes of sensors. Object Stores such as AWS S3, Azure Blob storage, and IBM Cloud Object Storage, are highly scalable distributed storage systems that offer high capacity, cost effective storage for this data. But it is not enough just to store data; we also need to derive value from it. Apache Spark is the leading big data analytics processing engine. It runs up to one hundred times faster than Hadoop MapReduce and combines SQL, streaming and complex analytics. In this poster we present Stocator, a high performance storage connector, that enables Spark to work directly on data stored in object storage systems.

Current connectors to object stores for Spark, e.g., S3a [1] and the Hadoop Swift Connector [2] are notorious for their poor performance for write workloads. The poor performance of these connectors follows from their assumption of file system semantics, which is natural given that their model of operation is based on the way that Hadoop interacts with its original storage system, HDFS. In particular, Spark and Hadoop achieve fault tolerance and enable speculative execution by creating temporary files and then renaming these files. This paradigm avoids interference between threads doing the same work and thus writing output with the same name. Notice, however, that rename is not a native object store operation; not only is it not atomic, but it must be implemented using a costly copy operation, followed by delete.

Others have tried to improve the performance of object store connectors, e.g., the DirectOutputCommitter [4] for S3a introduced by Databricks, but have failed to preserve the fault tolerance and speculation properties of the temporary file/rename paradigm. There are also recommendations in the Hadoop open source community to abandon speculation and employ an optimization [5] that renames files when tasks complete instead of waiting for the completion (commit) of the entire job. However, incorrect executions, though rare, can still occur even with speculation turned off.

Stocator takes advantage of object store semantics to achieve both high performance and fault tolerance. It eliminates the rename paradigm by writing each output object to the object's final name. The name includes both the part and attempt numbers, so that multiple attempts to write the same part use different object names. By leveraging the inherent atomicity of object creation we obtain fault tolerance and enable speculative execution; by avoiding the rename paradigm we greatly decrease the complexity of the connector and the number of operations on the object store. Our connector also takes advantage of HTTP Chunked Transfer Encoding, streaming output data to the object store as it is produced, thereby avoiding the need to write objects to local storage prior to writing them to the object store.

We have implemented our connector for the OpenStack Swift API and shared it in open source [3]. We have compared its performance with the S3a and Hadoop Swift connectors over a ranges of workloads and found that it executes far less operations on the object store, in some cases as little as one thirtieth of the operations. Since the price for an object store service typically includes charges based on the number of operations executed, this reduction in operations lowers the costs for clients in addition to reducing the load on client software. It also reduces costs and load for the object store provider since it can serve more clients with the same amount of processing power. Stocator also substantially increases performance for Spark workloads running over object storage, especially for write intensive workloads, where it is as much as 18 times faster.

*The research leading to these results has received funding from the European Union Horizon 2020 Research and Innovation Programme (grant agreement 644182 – IOStack).



*SYSTOR '17 Haifa, Israel*



ACM ISBN 978-1-4503-5035-8/17/05.

DOI: http://dx.doi.org/10.1145/3078468.3078496

## REFERENCES

[1] Hadoop-AWS Module: Integration with Amazon Web Services. https://hadoop.apache.org/docs/current/hadoop-aws/tools/hadoop-aws/index.html.

[2] Hadoop OpenStack Support: Swift Object Store. http://hadoop.apache.org/docs/current/hadoop-openstack/index.html.

[3] IBM Stocator Source Code. https://github.com/SparkTC/stocator.

[4] [SPARK-10063][SQL] Remove DirectParquetOutputCommitter. https://github.com/apache/spark/pull/12229.

[5] Using Apache Spark with Amazon S3. https://docs.hortonworks.com/HDPDocuments/HDCloudAWS/HDCloudAWS-1.11.0/bk_hdcloud-aws/content/s3-spark/index.html.