ALLUXIO

Using Alluxio to Improve the Performance and Consistency of HDFS Clusters

Calvin Jia
Software Engineer at Alluxio

What's Inside

ARTICLE

1 / Introduction

Alluxio is the world's first memory-speed virtual distributed storage system that bridges applications and underlying storage systems, providing unified data access orders of magnitudes faster than existing solutions. The Hadoop Distributed File System (HDFS) is a distributed file system for storing large volumes of data. HDFS popularized the paradigm of bringing computation to data and the co-located compute and storage architecture.

We used Spark 2.0 for computation and compared the performance of 2 stacks, one where Spark jobs was run directly on data in HDFS and another where Spark jobs were run on data in an Alluxio file system backed by HDFS.

2 / Why Alluxio

In a shared compute cluster, users are oen running jobs on similar datasets. For example, many data scientists may be trying to derive insights from the data gathered last week. At the same time, a few users may be accessing less frequently used datasets, such as data from the last month to generate a report.

In order to optimize the performance of the jobs, data can be stored in memory. However, a cluster's memory is limited and cannot store all the data, which naturally implies that a distributed memory management system is required. There are a few easily implemented choices.

Alluxio

By using Alluxio, you gain the ability to manage data based on characteristics such as access frequency. This enables the system to keep hot data in memory which greatly accelerates jobs that access this data. In addition, Alluxio provides predictable performance, allowing the system to guarantee a certain quality of service.

OS Buffer / Page Cache

The operating system will automatically try to utilize a machine's memory accelerate disk I/O. When repeatedly running jobs on the same dataset which fits in memory, this approach is eective and can provide similar performance benefits as Alluxio. However, with larger datasets or more varied workloads, the performance is highly variable and on average much less eicient than a data-aware system like Alluxio.

Spark Persist

Spark provides options to temporarily persist data for subsequent use without using any other systems. However, these mechanisms are limited to a single Spark Context, which prevents multiple users from gaining the benefits of one user's persisted data. As a result, each Spark Context consumes resources for its own in-memory or on-disk storage, which is ineicient in a shared environment, especially when large amounts of memory are consumed needlessly.

3 / Benefits in a Shared Environment

To emulate a multi-tenant environment with varying data temperatures, we set up the following experiment:

· At any given time, there are two jobs running on the cluster, a monthly and weekly job.

· Each job uses half of the available CPUs and compute memory resources.

· A new job is run immediately aer the previous job of the same type is complete.

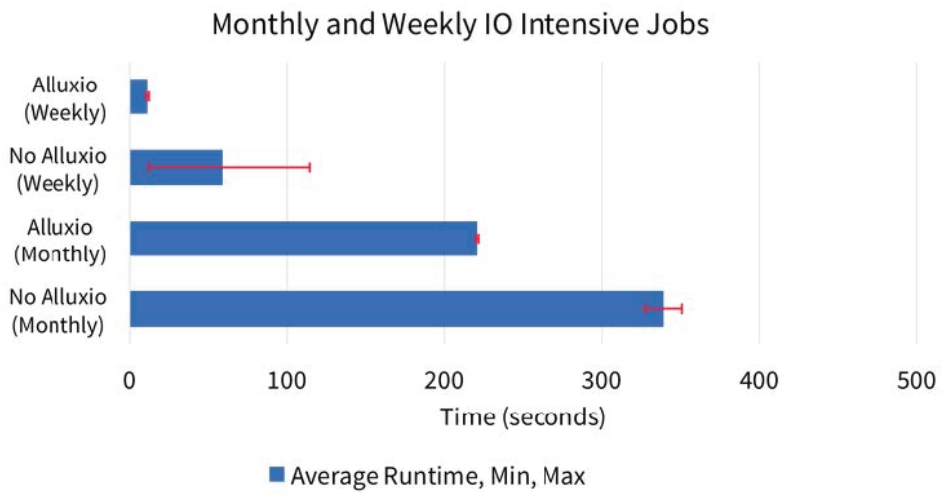
· A simple job is run on the weekly data beforehand to warm the OS cache or Alluxio in-memory storage.

Since each job is independent, the experiments are not applicable to using the Spark persist strategy. Therefore, we compare Alluxio versus the operating system's memory management.

Job Name	Description
Monthly IO	I/O intensive job over last month's data
Monthly CPU	CPU intensive job over last month's data
Weekly IO	I/O intensive job over last week's data
Weekly CPU	CPU intensive job over last week's data

The same experiment was conducted with two dierent stacks, one with Alluxio (Spark + Alluxio + HDFS) and one without (Spark + HDFS). The experiment was ran on Amazon EC2 using c4.2xlarge instances. The total dataset size was three times the available memory of the cluster.

SCENARIO 1

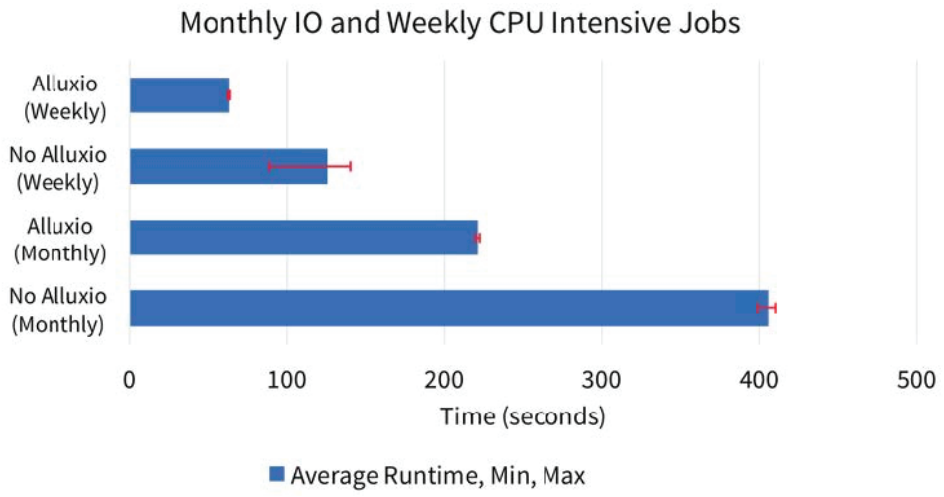


In the first scenario, both the monthly job and the weekly jobs are running an I/O intensive workload. Alluxio greatly improves the performance of both workloads.

For the weekly task, the hot data is guaranteed to be in Alluxio, and we enjoy memory speed reads which substantially accelerate the workload. In fact, you will notice that the previously I/O bound workload will now be compute bound. Without Alluxio, the performance is greatly varied (see red min-max ranges in the chart above), and can be more than 10x worse than with Alluxio. This is due to the unpredictable nature of what data will be served from the OS page cache.

The advantages for the monthly job are more subtle. Although we are able to statically partition CPU and memory resources (used by the Spark task, not to be confused with Alluxio memory) through Spark, we do not have fine grain control over I/O resources, such as disk. Since both weekly and monthly jobs are I/O bound in the case without Alluxio, the resource bottleneck is shared when the data is not available from the OS cache, reducing performance. With Alluxio, the monthly job has full use of the disk bandwidth because the weekly job is consistently reading from memory.

SCENARIO 2



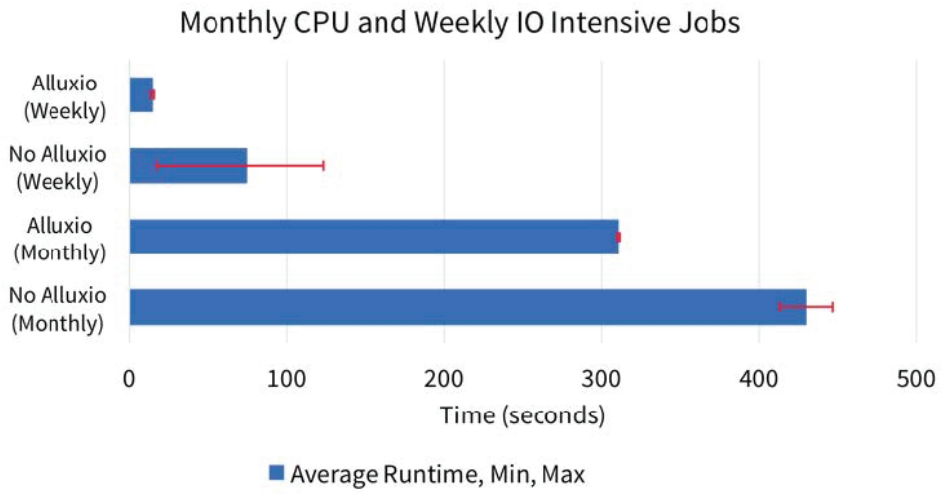
In the second scenario, the monthly job remains I/O intensive, but the weekly job becomes CPU intensive. In this scenario, Alluxio somewhat surprisingly still improves performance for both workloads.

The weekly task benefits from Alluxio's memory speed I/O, but to a much less noticable extent compared to previous I/O intensive workload. The performance improvement will be directly related to the CPU throughput the machine can handle.

However, the monthly job still performs much better with Alluxio because all the factors improving the monthly job's performance in scenario 1 still hold true.

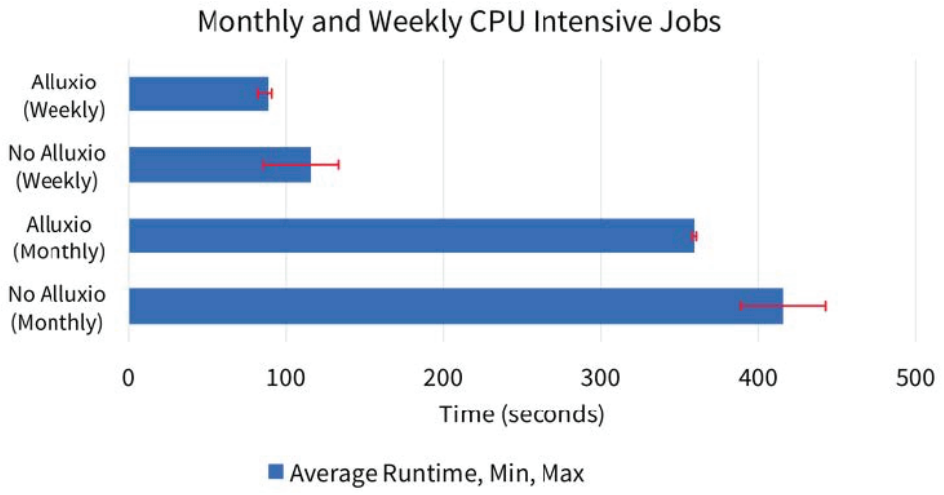
In addition, the weekly CPU job utilizes the OS cache less eiciently which leads to more contention on the disk resource, slowing down the monthly job even further.

SCENARIO 3



In the third scenario, the monthly job is CPU intensive and the weekly job is I/O intensive. Alluxio provides significant benefits to the weekly job because the data is completely in memory. The previously I/O bound workload is accelerated to the extent it becomes CPU bound. We also see benefits in the CPU intensive monthly job because Alluxio prevents the weekly job from competing with the monthly job for disk resources.

SCENARIO 4



3 /

In the last scenario, both the monthly and weekly jobs are CPU intensive. In this situation, Alluxio cannot provide significant benefits because the performance of both jobs are unrelated to the I/O throughput. However, Alluxio still provides performance stability by consistently managing the in-memory portions of the dataset.

4 / Conclusion

Alluxio provides predictable resource partitioning and utilization which enables system administrators to provide performance guarantees. Moreover, Alluxio brings significant performance gains even for compute clusters co-located with storage. The advantages of using Alluxio are amplified by the number of jobs accessing data in the cluster. Overall, users can expect two key performance benefits when using Alluxio in environments where compute and storage and co-located.

· Performance predictability allowing SLAs to be met more easily.

· Up to 10x improved performance.