ALLUXIO

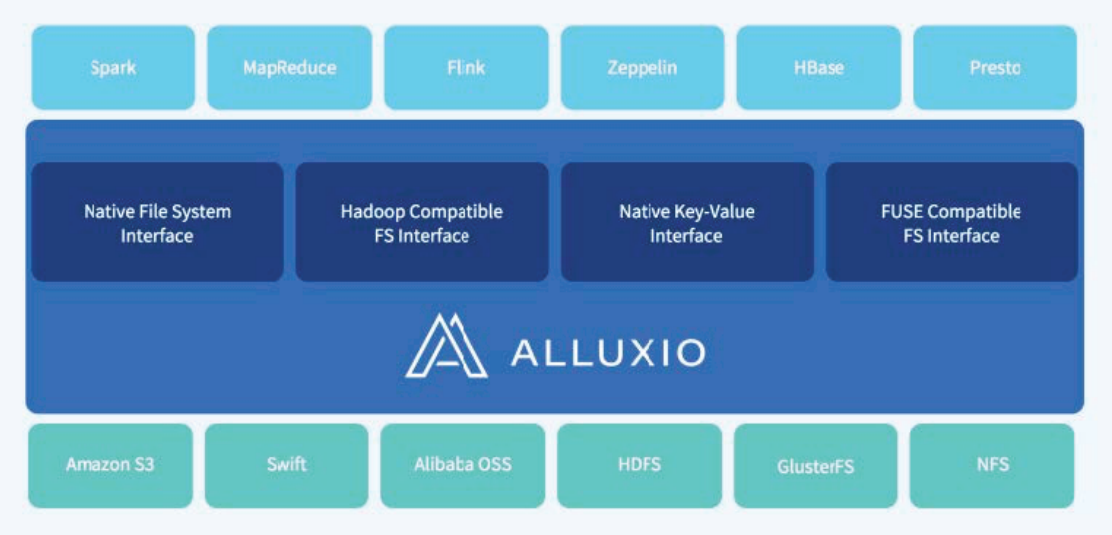
Qunar Performs Real-Time Data Analytics up to 300x Faster with Alluxio

What's Inside

CASE STUDY

1 / Introduction

Real-time data analytics is becoming increasingly important for Internet companies like Qunar, the leading travel search engine in China. Alluxio, a memory speed virtual distributed storage system, plays an important role in the big data ecosystem, and brings great performance improvements to applications. In this article, we present how Alluxio is used as the storage management layer in Qunar's stream processing platform, and how we use Alluxio to improve performance. At Qunar, we have been running Alluxio in production for over 9 months, and have observed 15x performance improvement on average, and 300x improvement at peak service times. At Qunar, the streaming platform processes around 6 billion system log entries (4.5 TB) daily. Many jobs running on the platform are business critical, and therefore impose strict requirements on both stability and low latency. For example, our real-time user recommendations are primarily generated based on the log analysis of user's click and search behavior. Faster analysis delivers more accurate feedback to the users. Therefore low latency and high stability are the top priorities of our system.



Our real-time analytics architecture includes many technologies including Mesos, Spark Streaming, Flink, HDFS, and now Alluxio. The architecture is described in more detail in the following section. The benefits that Alluxio brings to our system include:

1. Alluxio's tiered storage feature manages various storage resources including memory, SSD and disk.
2. Multiple computing frameworks can share data at memory speed with Alluxio.
3. Alluxio's unified namespace manages remote storage systems and provides a unified interface for applications, making it possible to access dierent storage systems from dierent applications and frameworks.

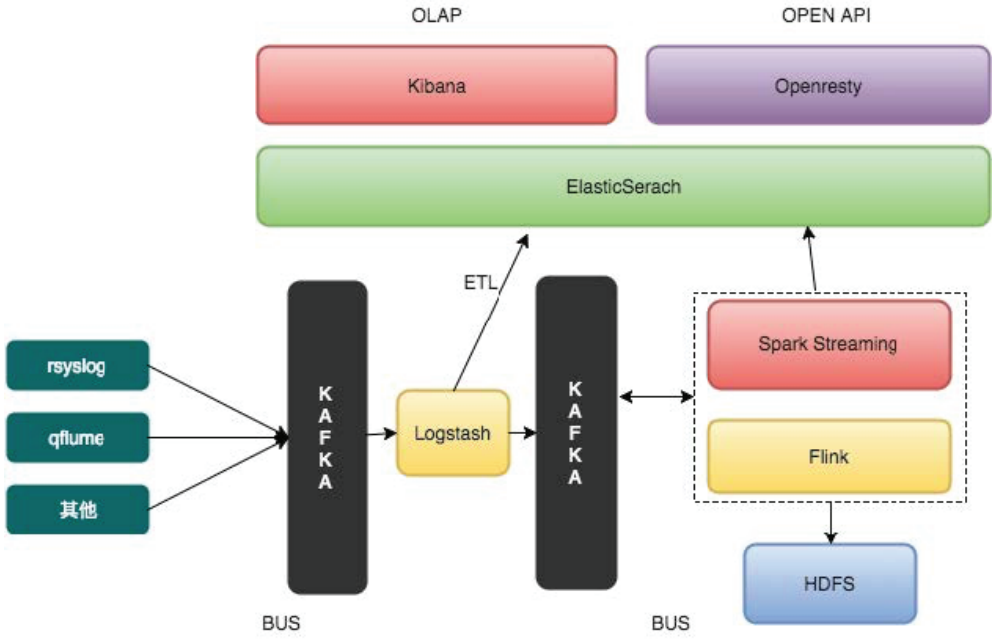
4. Alluxio provides various user-friendly APIs, which simplifies the transition to Alluxio.

5. Alluxio enables some large jobs to finish, by storing data in Alluxio memory instead of the application memory.

The remaining part of this blog will compare the stream processing architecture before and aer deploying Alluxio. At the end, we will briefly mention our future plans with Alluxio.

2 / Original Architecture and its Problems

Our real-time stream processing system uses Mesos for cluster management, and uses it to manage Spark, Flink, Logstash, and Kibana.



As shown in the figure above, logs are from multiple sources and consolidated by Kafka. Everyday, the business application generates about 6 billion log entries. The main computing frameworks, Spark streaming and Flink, subscribe to the data in Kafka, process the data, and persist results to HDFS.

This architecture has the following bottlenecks:

1. The HDFS storage system storing the input and output data is located in a remote cluster, introducing large network latency. Data exchange becomes one of the bottlenecks in the stream processing system.

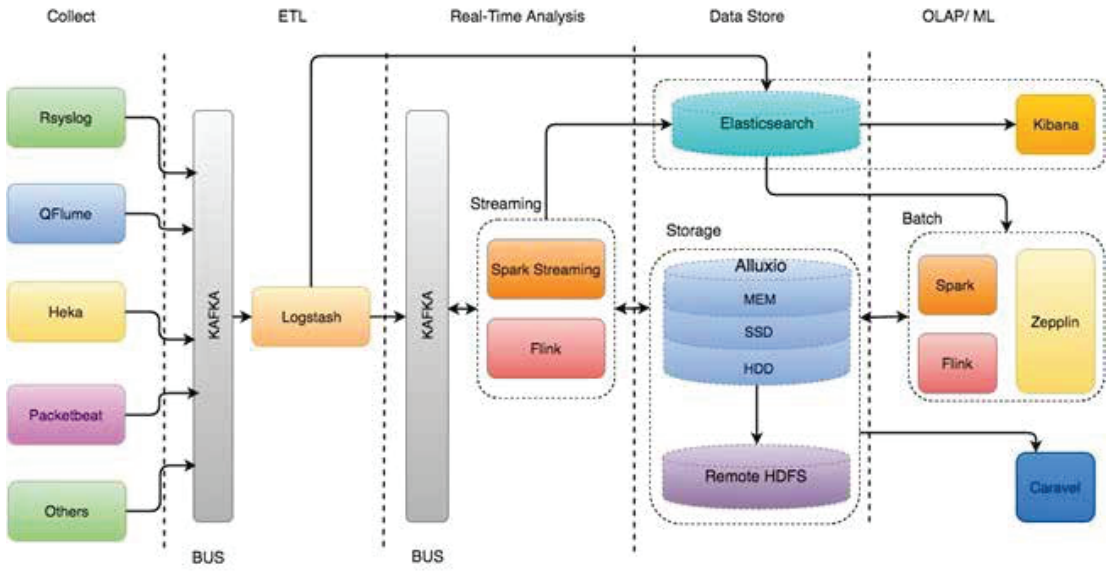
2. HDFS uses spinning disks, so I/O operations, especially write operations, have high latency. Spark streaming executors need to read from HDFS and the repetitive cross-cluster read operations further decreases the overall performance.

3. When a Spark streaming executor runs out of memory and fails, it will be restarted on a dierent node, and won't be able to reuse the checkpoint information from the previous node. Therefore, certain jobs can never finish.

4. When Spark streaming tasks persist data using MEMORY_ONLY there is replicated data in JVM memory, causing garbage collection issues and sometimes failures. However, when MEMORY_TO_DISK or DISK_ONLY is used, the perfor- mance of the whole stream processing architecture is limited to the speed of disk I/O.

3 / Improving the Architecture with Alluxio

We solved the problems mentioned above by introducing Alluxio into our architecture. The stream processing logic remains unchanged in the new architecture. The only dierence to the architecture is that we deployed Alluxio between the computation frameworks and HDFS. The computing frameworks exchange data eiciently via Alluxio because Alluxio provides memory speed data access. In this architecture, only the final results need to be persisted to the remote HDFS cluster. Below shows the figure representing the new architecture with Alluxio



Much of the architecture is similar as before. The main dierence is that much of the data used and generated by Spark streaming is stored in Alluxio, which avoids communication with the slow remote HDFS cluster. Meanwhile, the data stored in Alluxio can be conveniently shared by all the computing frameworks like Flink and Zeppelin. The following important Alluxio features played critical roles in improving the system's performance:

1. Tiered storage: Alluxio workers are deployed on every compute node to manage the local memory, SSD and disk. The data required by the stream processing is saved locally as much as possible to save network bandwidth. Meanwhile, Alluxio provides various eviction policies to keep most used data in memory. This greatly enhances data access eicien-cy.

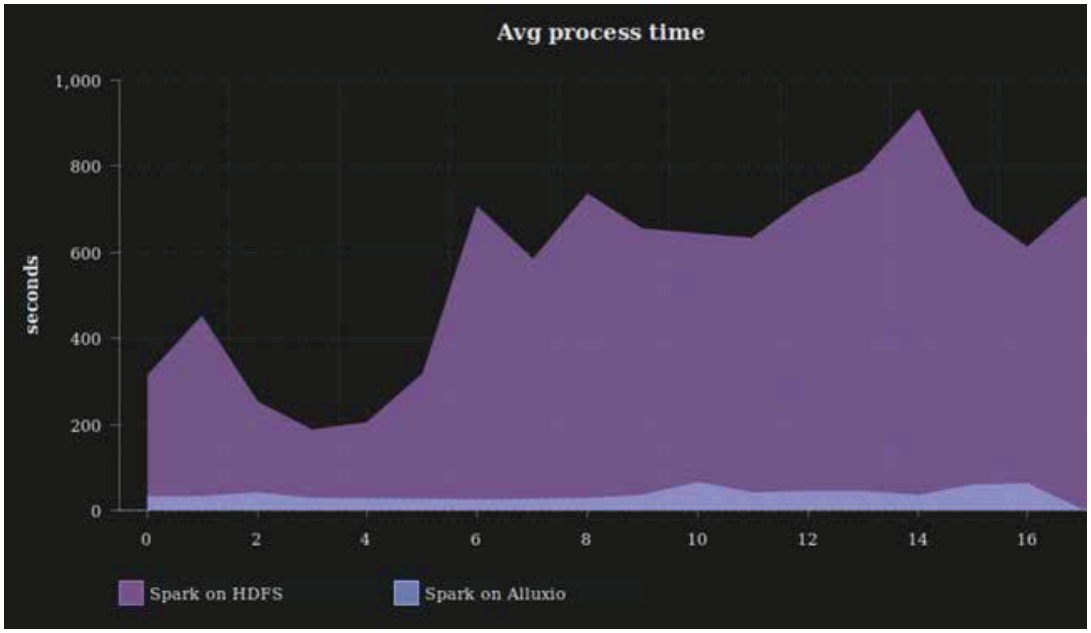
2. Data sharing among computing frameworks: Apart from Spark streaming, other components read data from Alluxio. Spark streaming and Spark batch jobs can share data at memory speed with Alluxio. We are planning to move Flink related services to Alluxio to achieve eicient data sharing among computing frameworks.

3. Unified namespace: Alluxio unifies the HDFS management and provides a global, unified namespace for computation frameworks. This greatly simplifies input and output logic.

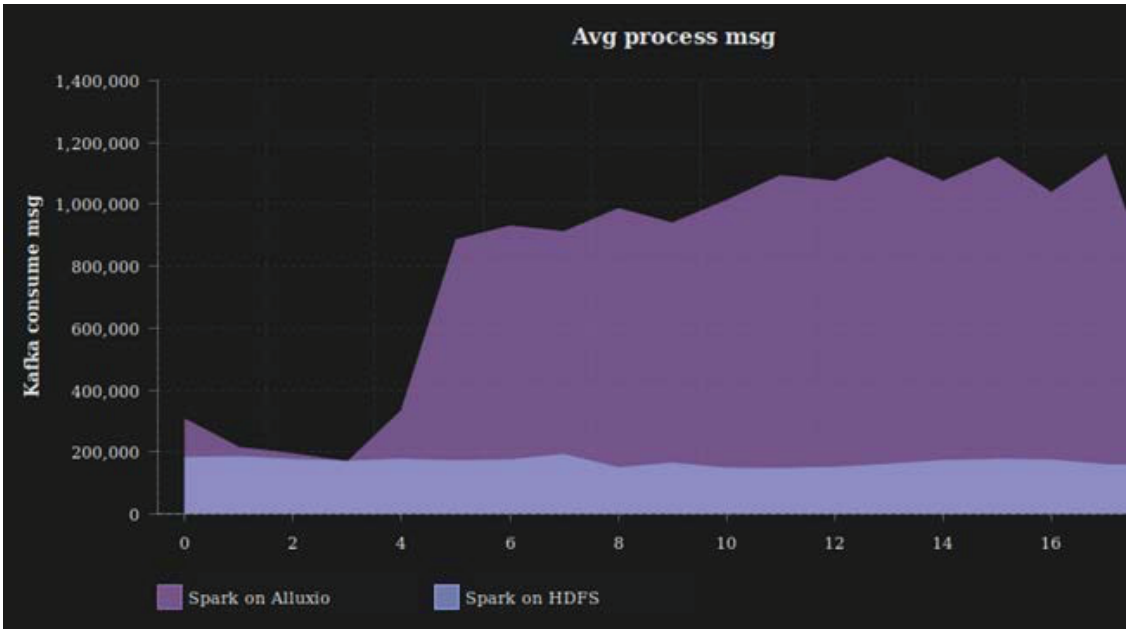
4. Simple API and easy integration: Alluxio provides multiple easy-to-use API, including an HDFS-compatible API. This makes it very easy to convert applications running on HDFS to Alluxio. The only modification was to change “hdfs://” URIs to “alluxio://” URIs.

5. Alluxio o-heap memory storage: Spark streaming jobs store data in Alluxio instead of Spark executor’s JVM. This maintains in-memory performance while greatly reducing Java garbage collection cost. This also prevents out-of-memory errors caused by redundant in-memory data blocks in Spark executors.

By sharing data between jobs with Alluxio memory, the throughput of the system increases and the latency decreases. Based upon our real-time monitoring, and using a Spark micro-batch size of 10 minutes, the throughput increases from about 20-300 events per second, to a stable throughput of around 7800 events per second. The average running time decreased from 8 minutes to about 30 seconds, about 15x faster! During periods of high load in the system and network, we see performance improvements of up to 300x faster with Alluxio! The figure below shows the average processing time of jobs, and it is obvious that using Alluxio greatly speeds up the Spark jobs.



The overall throughput of the system also improves significantly. Below is a figure showing the throughput of the events consumed from Kafka. The previous architecture would process around 200K events/second, but with Alluxio the throughput reaches 1,200K events/second.


Avg process msg
Kafka consume msg
1,400,000
1,200,000
1,000,000
800,000
600,000
400,000
200,000
0
0
2
4
6
8
10
12
14
16
Spark on Alluxio
Spark on HDFS

4 / Future Work

As previously shown, adding Alluxio to Qunar's real-time analytics platform greatly improved the performance of processing data. However, there are some areas for further improvement. Here is some future work:

1. Our production cluster uses Alluxio 0.8.2, which is almost a year old. We have already tested Alluxio version 1.0.1 which has tremendous improvements to various aspects and we are planning to put it in production.
2. We are planning to migrate our Flink jobs to use Alluxio. Meanwhile, we are planning to modify Presto so that it can take advantage of Alluxio as well.
3. We are planning to promote Alluxio to other internal services such as logs batch processing jobs.

5 / Conclusion

At Qunar, performance of our real-time analytics platform is very important for the business. Faster data processing leads to more immediate response to user behavior. Deploying Alluxio, a memory-speed virtual distributed storage system, improved the performance of our system with 15x – 300x speedups. Using Alluxio also helped some large jobs to complete. Alluxio has improved our production cluster for over 9 months, and we look forward to further usage and integrations of Alluxio.