Better a Microbenchmark on a Cluster than a User at the Office: Flink Cluster Benchmarking

David Georg Reichelt
Universität Leipzig
dg.reichelt@uni-leipzig.de

Lars-Peter Meyer
Universität Leipzig
lars-peter.meyer@uni-leipzig.de

Stefan Kühne
Universität Leipzig
kuehne@uni-leipzig.de

Abstract

When operating an Apache Flink cluster, performance problems may occur on all components of its setup. Reproducing those problems in different software or hardware components and on different nodes requires systematic experiments. We present an Apache Flink cluster benchmark set for server operators which is able to measure the performance of an Apache Flink cluster. This enables spotlighting irregularities in software or hardware behaviour.

1 Introduction

When operating an Apache Flink cluster, many aspects on different levels of hardware and software define the overall performance. Flink itself provides distributed batch and stream processing usable e.g. for event-driven, data analytics or data pipeline applications. Therefore, Flink is usually installed on a cluster with a set of job manager nodes and worker nodes running the `TaskManager`. Flink uses a scheduler, which may be Hadoops YARN, and the Hadoop File system for operation. All components are executed on Java VMs. These VMs are running on an operating system, which uses its hardware. The components of our cluster setup are visualized in figure 1.

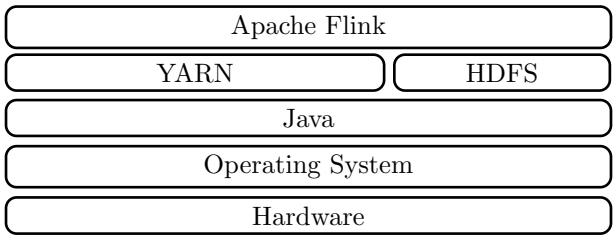

Figure 1: Tools and our Cluster Setup of Flink

The components visualized in figure 1 and their configuration all may contain problems affecting performance. From a server operators perspective, who has not a deep knowledge of all levels of a Flink setup, it is hard to reproduce performance problems. We define a microbenchmark suite[1] which aims for reproducing such problems. By making them reproducible, searching for reasons of irregularities is facilitated. The microbenchmark suite is built from an operators perspective, opposed to related work, which measures the performance by real world use cases mainly from a developers view. Since our workloads are very small, we call them microbenchmarks, even if the infrastructure for executing them is complex. By using our microbenchmark suite, we examined the clusters operated at Universität Leipzig. Thereby we were able to reproduce irregularities that users coming to our office reported.

In the remainder of this paper, we describe our microbenchmarks in section 2 and a case study using the microbenchmarks in section 3. Section 4 discusses related work. Finally, section 5 summarizes this paper.

2 Microbenchmark Suite

In order to define our microbenchmark suite, we first define the requirements of this suite. Afterwards, we describe the workloads we defined in order to benchmark the performance of Flink clusters and how we ensure statistic rigor of the measurements.

2.1 Requirements

The microbenchmark suite should be usable by operators in order to check whether a clusters performance fits its hardwares performance and whether the performance remains constant over time. We want to detect unusual or varying computing, RAM and HDFS performance. Besides a one-time usage, the microbenchmarks should be executable as short benchmarks on a regular basis in order to detect anomalies occurring due to temporary reasons, e.g. non-functional cooling or network load interference.

Furthermore, the microbenchmark suite should assure statistic rigor, i.e. that repeating the same measurements produces values we consider to stem from the same distribution. This is especially complex in Flink jobs and the tasks they consist of, since the execution time is influenced by (1) determination and optimization of scheduling, (2) the serialization, network communication and deserialization of tasks and (3) classloading, non-deterministic optimization and other non-deterministic effects of JVMs [1].

[1] Source Available: `https://git.sc.uni-leipzig.de/flink-benchmark/flink-benchmark`

2.2 Workloads

In order to check whether computing performance, RAM performance or HDFS access performance differs, we create a workload for each performance measure. Each workload is a Flink job which is executed on a cluster. For every job the execution time separately. Furthermore, each workload has a variable size and parallelism. The parallelism defines on how many workers the tasks of a job are distributed. The following benchmarks are implemented:

In order to benchmark **computing performance**, $size * factor$ additions are executed. These are grouped into N (*Default: 100*) map jobs. Those jobs contain $\frac{size*factor}{N}$ additions. Therefore, the network traffic is not increased with increasing benchmark size.

In order to benchmark **RAM performance**, the benchmark reserves $size * factor$ bytes of RAM. In every job, a random count of indices of the array is initialized with a random value. Afterwards, the same count of random indices are added. The sum of the random indices is the result of the job. Therefore, the array is not erased by optimization in OpenJDK 8.

In order to benchmark IO performance, we implemented a **HDFS write** and a **HDFS read benchmark**. These benchmarks use a mapping function as main building block and write or read $size * factor$ bytes of data to or from HDFS and return the throughput as map result. As HDFS supports only append operations, the write benchmark creates sequential output per mapping thread. The read benchmark consists of sequential read operations with random offsets.

2.3 Statistic Rigor

In order to make our measurements comparable, they need to be executed until the measurement values reach its steady state. We assume that the measurements are gaussian distributed. Therefore, we consider the steady state as being reached if the two-sample t-test with confidence level 99% does not reject the null hypothesis that the means of both distributions are equal.

In order to determine when repeating the measurements produces equal results, we started 100 jobs with 25 iterations of every workload with sizes 10000, 20000 and 40000 each on our Flink cluster. We found that the t-test does not reject the null hypothesis for two samples build from measurement 5 to 15 and the last 10 measurements. Therefore, we execute every measurement 25 times and use the last 10 measurements.

3 Case Study

At Universität Leipzig, we are operating two Hadoop clusters. We used the microbenchmark suite in order to examine whether the Flink performance of the clusters fits their hardware and whether irregularities over time occur. At first, we compared the performance of Flink to the local execution of the same workload and afterwards, we compared multiple Flink executions on both clusters. We used Flink 1.5 on top of YARN together with Java 1.8. Both clusters consist of 16 workers and use no virtualization, for the hardware see table 1.

	Worker@Cluster1	Worker@Cluster2
CPUs	2 Xeon®, 2.4GHz	1 Xeon®, 2.5GHz
Model	E5-2620v3	E5-2430v2
RAM	128 GByte	48 GByte
HDDs	5x4TB SATA	2x4TB SATA
Ethernet	10 GBit/s	1 GBit/s
OS	CentOS 7.6	openSUSE 13.2

Table 1: Hardware of the Clusters Used

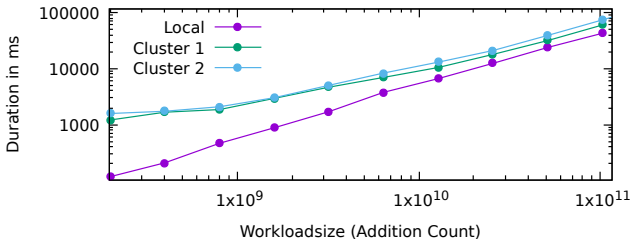

Figure 2: Add-Benchmark Execution Time

3.1 Comparison with Local Execution

In order to find out how much overhead an execution in Flink produces compared to the execution in a local Java VM, we run the workloads in both environments. We used the same hardware for this, i.e. we used a cluster node with deactivated Flink for the local execution. For Flink, the initialization time, `ExecutionGraph` creation and sending input data via the network to the `TaskManager` is expected to create overhead. In order to determine the amount of overhead, we executed the workloads with exponential computation size growth on the local server and in both clusters. The average measurement results of the add-benchmark are visualized in figure 2. While in the beginning the computation time is considerably higher for Flink execution, the computation time does not grow faster with Flink. We deduce that our clusters use their full CPU power as expected and are slown down by the overhead of Flink.

3.2 Comparison between Flink Executions

We examined whether there are performance irregularities during long-term operation of the clusters. Therefore, we executed our microbenchmarks over 5 days every hour on both clusters. Figure 3 shows the frequency of execution durations of our clusters with an exponential scale to improve the outliers visibility.

We found that execution times differed across the cluster and that Cluster 1 had a higher variance and some outliers. This matches with users com-

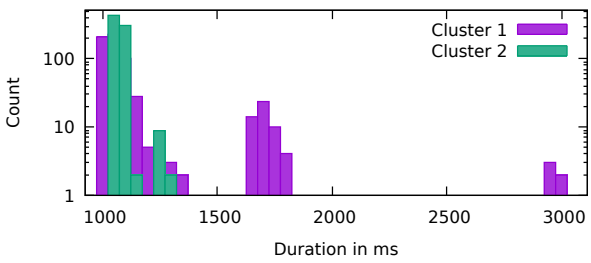

Figure 3: Histogram of Execution Durations

plains about performance irregularities on Cluster 1. Other measurements have shown equal performance for RAM benchmarks and worse performance of Cluster 1 for HDFS performance. Unfortunately, we are not finished finding the root cause of the outliers or the higher variance in computing performance. Nevertheless, the microbenchmark suite gives us a starting point for reproducing and debugging this problem.

4 Related Work

There exists related work (1) benchmarking Apache Flink from users perspective and (2) benchmarking HDFS from an operators perspective.

Benchmarks for Flink from users perspective (1) are included in the HiBench Suite [2]. HiBench consists of different user-driven workloads, e.g. using k-means for clustering randomly generated data. HiBench can benchmark different Big Data Frameworks, including Flink and Hadoop. Another benchmark for Flink is BigBench [3] which got standardized by the Transaction Processing Performance Council as TPCx-BB [5]. While those benchmarks only use Flink alone, other benchmarks use whole real world production scenarios for benchmarking, including the use of other tools, like Kafka and Redis [4] [6]. While those benchmarks are built from a usage perspective and define use cases similar to real world applications, our rationale is to provide a comparison from a server operation perspective focussing on small benchmarks.

Benchmarks focussing on HDFS include TestDFSIO, included in Hadoops source code[2], and the enhanced TestDFSIO, included in the HiBenchSuite. These benchmarks do not include Apache Flink but use HDFS in a direct manner.

Acknowledgements

This work was funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B and BMBF 01IS18026B) and a PhD scholarship of Hanns Seidel Foundation. Computations for this work were done with resources of Leipzig University Computing Centre.

[2]Source location: TestDFSIO.java in `http://www-eu.apache.org/dist/hadoop/common/hadoop-2.7.7/hadoop-2.7.7-src.tar.gz`

5 Summary

In order to benchmark the performance of Apache Flink clusters, we defined a microbenchmark suite which is capable of finding irregularities in Flink cluster performance. The microbenchmark suite consists of different benchmarks using CPU, RAM and HDFS-I/O. We used them on our clusters and found that, while they are basically operating as expected, performance variation is higher on one cluster. Since users complained about this cluster, we will further investigate this issue.

In order to address this issue, we will search the logs of Flink, of HDFS and of the clusters operation systems and examine reasons for this behavior. Furthermore, we will add more workloads, e.g. focusing on network communication. Another possible extension of this work is implementing the workloads separately for all components, i.e. for Hadoop, HDFS, Java and as binary for the OS, in order to experimentally find out which component is causing problems.

References

[1] A. Georges, D. Buytaert, and L. Eeckhout. “Statistically rigorous java performance evaluation”. In: *ACM SIGPLAN Notices* 42.10 (2007), pp. 57–76.

[2] S. Huang et al. “The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis”. In: *New Frontiers in Information and Software as Services: Service and Application Design Challenges in the Cloud*. Ed. by D. Agrawal, K. S. Candan, and W.-S. Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 209–228.

[3] A. Ghazal et al. “BigBench: towards an industry standard benchmark for big data analytics”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. New York, New York, USA: ACM, 2013, pp. 1197–1208.

[4] S. Chintapalli et al. “Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming”. In: *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2016, pp. 1789–1792.

[5] P. Cao et al. “From BigBench to TPCx-BB: Standardization of a Big Data Benchmark”. In: *Performance Evaluation and Benchmarking. Traditional - Big Data - Internet of Things*. Ed. by R. Nambiar and M. Poess. Cham: Springer International Publishing, 2017, pp. 24–44.

[6] S. Yang et al. “Scalability and State: A Critical Assessment of Throughput Obtainable on Big Data Streaming Frameworks for Applications With and Without State Information”. In: *Parallel Processing Workshops*. Ed. by D. B. Heras et al. LNCS. 2018, pp. 141–152.