Data Engineering 1 Project

Anirudh A Inginshetty, Joel Sikström, Max Norberg, Nils Carlberg
Group 12

Mars 2023

Contents

1 Introduction **3**

2 Data Format **4**

3 Computation Experiments **5**

3.1 Cluster Configuration 5

3.1.1 Structure 5

3.1.2 Motivation 6

3.2 Scalability Experiments 7

3.2.1 Subreddits With Common Comment Authors 7

3.2.2 Type of Scalability Experiment 7

3.3 Results 8

4 Discussion and Conclusion **9**

1 Introduction

Reddit is a website with thousands of communities discussing anything. It can be anything from sports, gaming, news, movies, or just sharing pictures. Each community is called a subreddit and features a specific theme or subject that users can discuss in that subreddit. In a subreddit a user can submit text, videos, pictures, and links and other users can react to these submissions with either an upvote or a downvote as well as comment [1]. Subreddits are moderated by their creator and users of their choosing and they decide what the rules are for posting in the subreddit. When a user posts or comments something they receive *karma*. They also receive karma when what they submitted gets voted on, and having more karma points means that you can contribute more on the website [2]. In conclusion, the Reddit framework is a complex and sophisticated system that has evolved over time to become one of the most popular and engaging social platforms on the internet. Its use of subreddits, voting, comments, moderation, and bots all work together to create a dynamic platform.

The dataset we have chosen to work with is from the website `https://files.pushshift.io/reddit/comments/` which is a project aimed at distributing Reddit comment data for research and analysis purposes. The datasets on pushshift is compiled of two sets of files, one for submissions and one for comments that are made on any of the submissions [3]. In our computational experiment we will only focus on the comment data.

The data is stored in Hadoop HDFS due to its performance benefits over a local filesystem, and also replicated on all VMs in the cluster for maximized resiliency and performance. For the computational experiment we will use Spark, and mainly its provided RDD functionality, to perform preprocessing and analysis of the dataset.

The report has been divided into three sections where the data format, computational experiments and results and conclusions are discussed.

2 Data Format

The Reddit comment data from Pushshift is available in JSON format and compressed using Zstandard (zstd). It works by replacing repetitive strings of data with shorter codes, which reduces the overall file size without losing any information. Zstandard is a lossless compression format, meaning that the compressed file can be decompressed back to its original form without any data loss. In our experiment we have used the dataset "RC_2011-01", which is about 600MB compressed and about 3.6GB uncompressed. As indicated by the name of the dataset, the comments are collected from comments made in January 2011.

As discussed in a study by A. Abdel-raouf and I. Hanafi [4], HDFS has a lower degree of parallelism when it processed compress data and as a result, the performance is less when processing compressed data as opposed to uncompressed data. To fully utilize the parallel processing capabilities of both Spark and HDFS, we have chosen to decompress the data before storing it on our HDFS cluster.

The data format that Pushshift provides the comment data in is JavaScript Object Notation (JSON). JSON is a lightweight data interchange format that is easy to read and write, making it ideal for exchanging data between different applications and systems. Furthermore, JSON is very similarly structured to the dictionary data type in Python, making conversion and seralization between JSON and dictionaries easy and straight-forward.

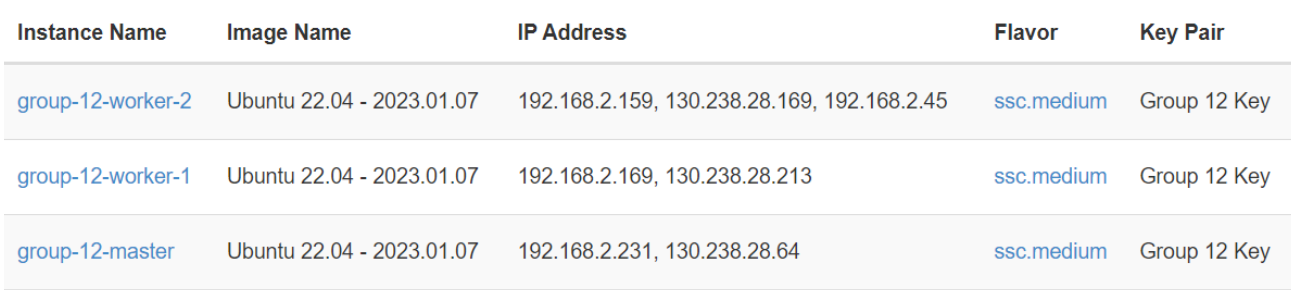
A negative aspect of storing data in JSON, which works in a key/value pair approach, is that the key for each value needs to be present on every line in the data file. Had the data instead been stored in a comma-separated values (CSV) file, the name for the keys would only need to be present on the very first line in the file and not on every line, resulting in a smaller file size. However, given the relatively small size of the dataset, this is considered not to be a problem. If one were to process larger datasets, this might require significantly more storage which also could be utilised more efficiently by converting the dataset to a CSV file.

3 Computation Experiments

3.1 Cluster Configuration

3.1.1 Structure

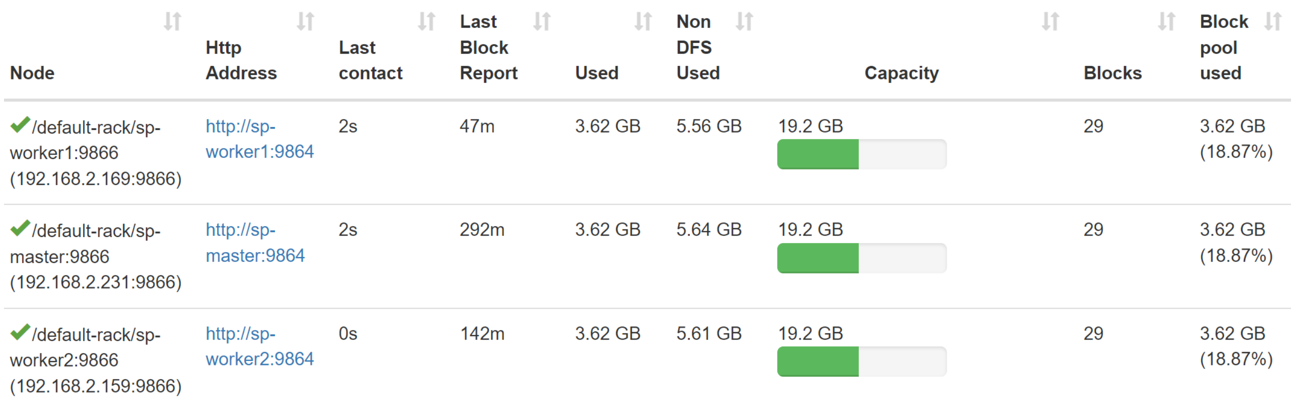
In our cluster, which is housed in the SNIC cloud, we have started with three virtual machines (VMs). Of the three VMs, one node is dedicated as the master node and the other two as worker nodes. We have built a snapshot picture of one of the workers to streamline the process of adding more worker nodes to the cluster, which eliminates the need to manually setup a new worker. Every VM uses the ”ssc.medium” flavor, which includes 2 CPU cores, 4 GB of Memory, and 20 GB of storage. Figure 1 shows the VMs as displayed in the SNIC cloud.



Instance Name	Image Name	IP Address	Flavor	Key Pair
group-12-worker-2	Ubuntu 22.04 - 2023.01.07	192.168.2.159, 130.238.28.169, 192.168.2.45	ssc.medium	Group 12 Key
group-12-worker-1	Ubuntu 22.04 - 2023.01.07	192.168.2.169, 130.238.28.213	ssc.medium	Group 12 Key
group-12-master	Ubuntu 22.04 - 2023.01.07	192.168.2.231, 130.238.28.64	ssc.medium	Group 12 Key

Figure 1: VMs running in the SNIC cloud

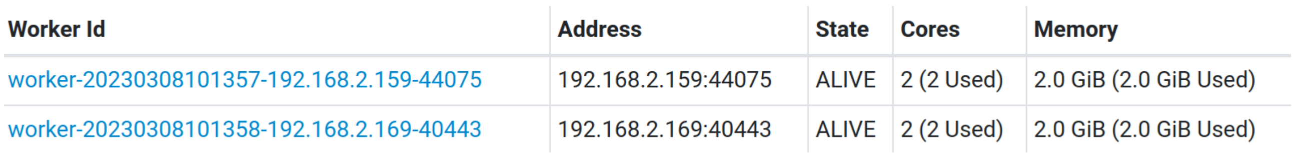
All of the VMs in our cluster are running Hadoop Distributed File System (HDFS), which is where we store the data for the computational experiment. There are several advantages of using a distributed filesystem like HDFS rather than a local filesystem since it operates on numerous separate devices. As data is written on several devices rather than just one, HDFS offers high throughput access to the data that is stored and resiliency in case of failure. We have decided to replicate data across all available DataNodes, meaning that if one of the VMs crash or becomes unresponsive, the data can still be accessed from the other DataNodes. Figure 2 shows the DataNode setup and usage that we have in our HDFS cluster. If the data that is processed becomes too large for the cluster to handle or if we want further resiliency in our cluster we can add more DataNodes or storage to existing DataNodes in the future. To utilize all available storage on the VMs we also ran a DataNode next to the NameNode on the master VM.



Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used
✓/default-rack/sp-worker1:9866 (192.168.2.169:9866)	http://sp-worker1:9864	2s	47m	3.62 GB	5.56 GB	19.2 GB	29	3.62 GB (18.87%)
✓/default-rack/sp-master:9866 (192.168.2.231:9866)	http://sp-master:9864	2s	292m	3.62 GB	5.64 GB	19.2 GB	29	3.62 GB (18.87%)
✓/default-rack/sp-worker2:9866 (192.168.2.159:9866)	http://sp-worker2:9864	0s	142m	3.62 GB	5.61 GB	19.2 GB	29	3.62 GB (18.87%)

Figure 2: DataNodes in HDFS cluster

Apache Spark is a data processing framework that can effectively handle jobs on huge datasets using distributed processing on numerous servers, and we are utilizing it to undertake computational experiments. Figure 3 shows the workers running in our Spark cluster, where each worker will contribute to processing jobs together, which enables faster computations than if the same operation had been performed on a single machine. Also, we have set up each worker to only use 2GB of the machine's total 4GB of RAM. In practice you might want to make use of as much RAM as you can on each VM, but we have chosen to limit the RAM to ensure that each VM behaves identically in our computational experiment.



Worker Id	Address	State	Cores	Memory
worker-20230308101357-192.168.2.159-44075	192.168.2.159:44075	ALIVE	2 (2 Used)	2.0 GiB (2.0 GiB Used)
worker-20230308101358-192.168.2.169-40443	192.168.2.169:40443	ALIVE	2 (2 Used)	2.0 GiB (2.0 GiB Used)

Figure 3: Workers in Spark cluster

3.1.2 Motivation

We have decided to use Apache Spark for our computational experiments mainly becuase of its Superior execution speed compared to Hadoop MapReduce, as described by A. Verma et al [5]. Spark is able to provide its speed advantage becuase it processes data in RAM while still able to provide fault tolerancy through Resilient Distributed Datasets (RDD). Futhermore, writing instructions/code for Spark is designed to be compact and similar to SQL queries and provides functionality for streaming data, machine learning and graph processing. This will enable us to focus on the scalability part of the experiment rather than spending too much time on writing code for processing.

3.2 Scalability Experiments

3.2.1 Subreddits With Common Comment Authors

The goal of this experiment is to design a single data processing workflow that can be solved with different scales of resources in order to find out how scalable it is. The aim of the data processing workflow is to find out what subreddits have the most authors of comments in common. To do this, we analyze comments from multiple subreddits to find out what authors have commented on what subreddits and later on try to match the number of authors that subreddits have in common. The flow of the preprocessing and analysis that is done in this experiment is shown in Figure 4, where the input is a text file containing un-serialized JSON objects and the final output is a list of subreddit pairs along with the number of authors they have in common, structured in a 'triple'.

All preprocessing and analysis in the experiment is performed via Spark's RDD and its accompanying transformation functionality, such as `map`, `filter`, `cartesian`, `distinct` and `sortBy`. In between some processing/analysis steps we have used actions to make sure that the format of the data is correct, but in a final or production workflow, the only action that is needed is when extracting/printing the final result.

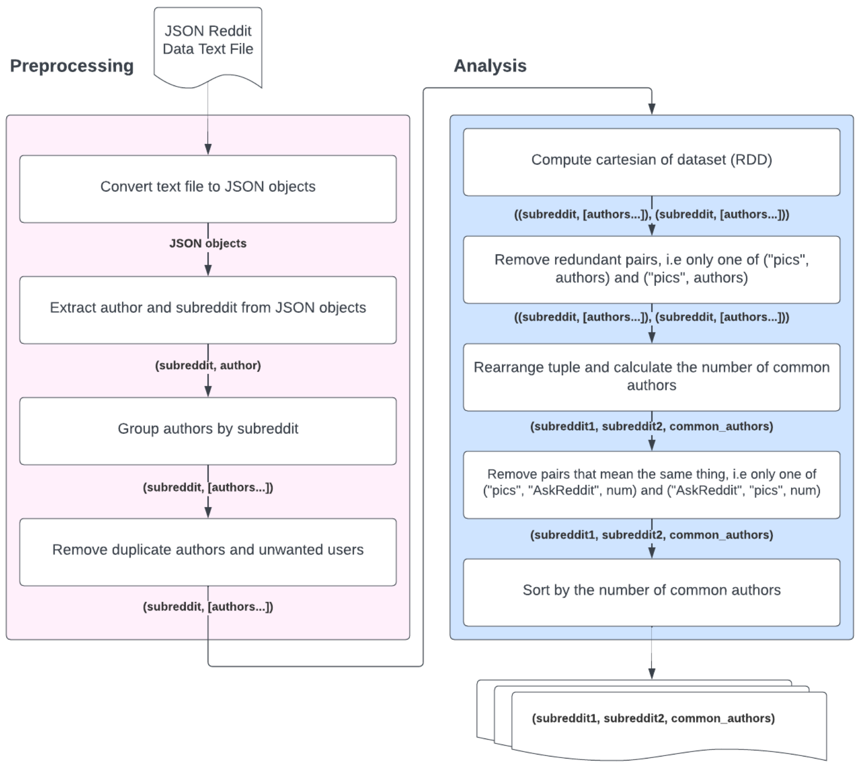

JSON Reddit Data Text File
Preprocessing
Convert text file to JSON objects
JSON objects
Extract author and subreddit from JSON objects
(subreddit, author)
Group authors by subreddit
(subreddit, [authors...])
Remove duplicate authors and unwanted users
(subreddit, [authors...])
Analysis
Compute cartesian of dataset (RDD)
((subreddit, [authors...]), (subreddit, [authors...]))
Remove redundant pairs, i.e only one of ("pics", authors) and ("pics", authors)
((subreddit, [authors...]), (subreddit, [authors...]))
Rearrange tuple and calculate the number of common authors
(subreddit1, subreddit2, common_authors)
Remove pairs that mean the same thing, i.e only one of ("pics", "AskReddit", num) and ("AskReddit", "pics", num)
(subreddit1, subreddit2, common_authors)
Sort by the number of common authors
(subreddit1, subreddit2, common_authors)

Figure 4: Flow of data in experiment

3.2.2 Type of Scalability Experiment

We intend to explore how performing horizontal scaling affects the time it takes to perform our data processing workflow. We will do this by adding more workers/nodes to our Spark cluster, effectively adding more cores and RAM to perform calculations. In order to maximize the resources available to us we will also try adding the master as a worker node as well. We will discuss how having the master node both as a Spark master and worker might impact performance later on in the report.

The new workers that will be added to the group of workers will have the same configuration as the existing workers, 2 CPU cores and 2 GB of ram.

3.3 Results

Figure 5 shows the top 100 results of the data processing workflow, where the subreddits “AskReddit” and “pics” have the most comment authors in common.

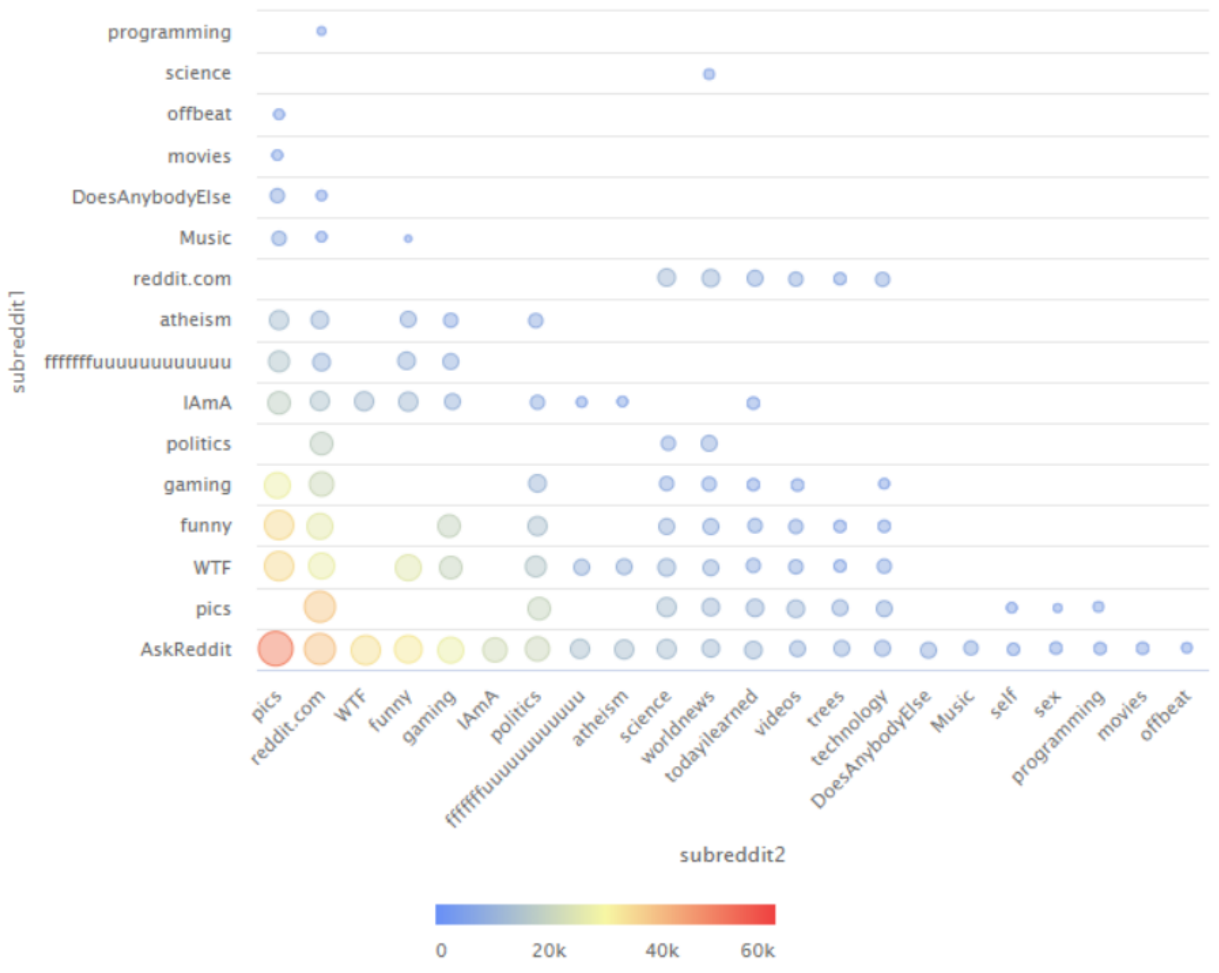


Figure 5: The number of common comment authors between subreddits. The color of each circle indicates the number of common authors the corresponding two subreddits on each axis have.

Figure 6 shows the execution time when using different number of worker nodes. Each worker node adds 2 CPU cores and 2 GB RAM, which means that having 5 worker nodes results in a total of 10 CPU cores and 10 GB RAM.

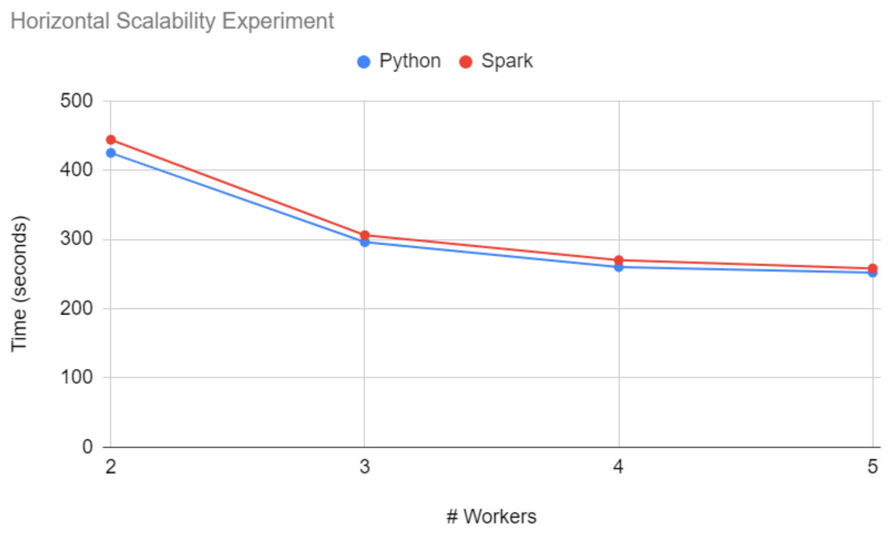


Figure 6: Execution time of data processing workflow relative to number of workers. The blue line is the time measured by Python's `time.time()` and the red line is measured by Spark job run time.

4 Discussion and Conclusion

In this project we have built both a Hadoop and Spark cluster to perform a data processing workflow and measure the scalability of it using horizontal scaling. We started with a VM designated as a master, running both a Hadoop NameNode and Spark Master and two worker VMs, running Hadoop DataNode and Spark Worker. At the end of our scalability experiment we had a total of five worker nodes in addition to the master, all running with the same configuration.

Although our data processing workflow does not take especially long to compute with our chosen data set, it is time consuming enough for us to measure noticeable performance changes in our scalability experiment. When increasing the number of Spark worker nodes used in processing, we have measured a performance increase for every added worker. However, the largest performance increase was measured when going from two to three workers. After three workers, the performance increase per added worker node was not significant in terms of execution time.

Despite performance in terms of execution time being the main aspect in our testing, other added benefits in terms of resiliency and speed of recovery is an added benefit when adding more workers to the cluster. More workers means that if (and probably when) a failure occurs, the RDD and cluster can recover its operations faster, as described in a study by M. Zaharia et al [6]. This is also in line with Sparks' core design, which is focused on making an entire system robust and resilient to possible failures. Another reason that the speed does not increase as much with more workers might be due to overhead in Spark when transferring data between workers. However this needs to be further investigated in the future.

In developing our data processing workflow we have tested it using a smaller subset of the dataset we are using, and have thus confirmed that the workflow can adapt to larger datasets without problem. The only limitation we think would hinder processing of an even larger dataset would be the total available RAM, but in our experiments, the size of the data has been smaller than this and thus not posed a problem.

All in all, we feel that we have successfully managed to prove the scalability of our workflow by adding more worker nodes to Spark, which increases performance through reduced execution time.

References

[1] Reddit, "Dive into anything," Accessed 2023-03-08. [Online]. Available: https://www.redditinc.com/

[2] J. H. Tim Weninger, Xihao Avi Zhu, "An exploration of discussion threads in social news sites: A case study of the reddit community," Ph.D. dissertation, University of Illinois Urbana-Champaign, n.d. [Online]. Available: https://dejanmarketing.com/wp-content/uploads/2013/10/reddit-paper.pdf

[3] B. K. M. S. J. B. Jason Baumgartner, Savvas Zannettou, "The pushshift reddit dataset," Ph.D. dissertation, Max-Planck-Institut fur Informatik, University of Colorado Boulder, Elon University, Binghamton University, Network Contagion Research Institute, 2020.

[4] A. Abdel-raouf and I. Hanafi, "P-codec: Parallel compressed file decompression algorithm for hadoop," *INTERNATIONAL JOURNAL OF COMPUTERS TECHNOLOGY*, vol. Vol.15 , No. 8, pp. PP. 6991–6998, 09 2016.

[5] A. Verma, A. H. Mansuri, and N. Jain, "Big data management processing with hadoop mapreduce and spark technology: A comparison," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1–4.

[6] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. USA: USENIX Association, 2012, p. 2.