Report about rucio-fs tests

N. Avdeev

September 19, 2025

1 Introduction

RucioFS is a FUSE-based filesystem interface for the Rucio data management system. It allows users to access and interact with datasets stored in Rucio as if they were part of a local filesystem. By providing a familiar POSIX-like interface, RucioFS facilitates data analysis.

At the current stage, RucioFS is only a prototype. The main objective of this work was to test the application, since further development requires a clear understanding of its strengths and limitations.

In this study, we evaluated the performance of the **ls** command for both single and multiple users. We also conducted brief tests with the **cat** and **cp** commands.

2 Evaluation of the ls command for a single user

2.1 Objective

The objective is to estimate the execution time of the **ls** command in two scenarios: with the caching system enabled and with it disabled.

2.2 Methodology

We measured the execution time of the **ls** command for the following numbers of files: 500, 1,000, 5,000, 10,000, and 50,000. Each file had a size of 4 KB. The files were created by a simple Bash script and contained only the file number. To measure the execution time of the **ls** command, we used the **time** Bash command. For each file count, 100 independent measurements were performed.

2.3 Results

The results of this test are presented in Figure 1. As can be seen, even with 50,000 files, the execution time remains relatively low when the caching system is enabled.

In contrast, when caching is disabled, the execution time is significantly higher. This case corresponds to the situation when a user opens a directory for the first time.

3 Evaluation of the ls command for multiple users

3.1 Objective

The objective is to estimate the execution time of the **ls** command executed simultaneously by multiple users in two scenarios: with the caching system enabled and with it disabled.

3.2 Methodology

We measured the execution time of the **ls** command for 10,000 files, with each user executing the command simultaneously. To measure execution time, we used the same **time** Bash command as in the previous test. It should be noted that by "users" we mean separate containers running on the same machine as the server. Therefore, in a real scenario, where actual users run **ls** for the first time simultaneously (similar to the case with caching disabled), the execution time might be longer.

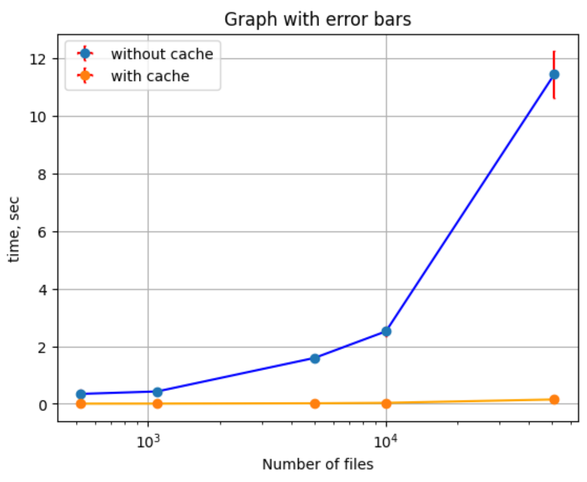

Figure 1: This graph corresponds execution time of **ls** for different file counts for case with and without cache

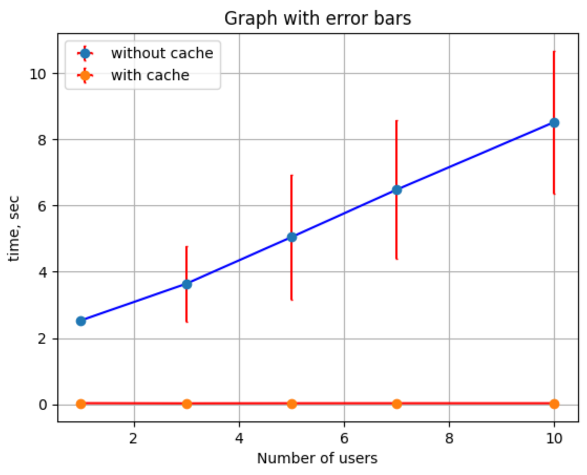

Figure 2: This graph corresponds execution time of **ls** for 10000 files, which run by many users simultaneously for case with and without cache

3.3 Results

The results of this test are presented in Figure 2. As can be seen, with caching enabled the execution time remains low regardless of the number of users. This result is expected, since in this case the users interact only with the cache.

In contrast, when caching is disabled, the execution time increases with the number of users. We attribute this effect to the increasing number of requests sent to the server. It should also be noted that the variance in execution times for multiple users is large, as the number of concurrent requests to the server may differ, given that it is practically impossible to run the command truly simultaneously. This scenario corresponds to the case when many users run **ls** for the first time at the same moment.

From these results, it can be concluded that there is no significant issue when a small number of users execute the command simultaneously. However, the situation may become problematic when 100 or more users attempt to run the command for the first time at the same time.

4 Brief tests of cat and cp

4.1 Test of the cat command

There is no strong motivation to use the **cat** command for gravitational wave data files. Therefore, for more meaningful and precise testing, it is necessary to determine what types of files should be read with the **cat** command in Rucio.

For small `.txt` files (approximately 80 rows), the average execution time of the **cat** command was 2.2 seconds when run for the first time (without caching) and 0.02 seconds for subsequent runs (with caching).

4.2 Test of the cp command

This command serves as a simple analogue of the `rucio download` functionality in RucioFS. However, at the current stage it does not perform well with large files. For example, when copying a 4.2 GB raw Virgo file, only 205 MB of the file could be transferred. Therefore, at present, there is no practical reason to use this command for large data files.

5 Conclusion

At present, the performance of the **ls** command is generally satisfactory. Only in extreme situations could performance become problematic, such as when handling hundreds of thousands of files in a single directory or hundreds of first-time simultaneous requests.

For testing the **cat** command, it is necessary to determine which types of files should be read.

The **cp** command does not perform reliably at this stage; therefore, there is no strong justification for conducting detailed performance tests for this command.