Profiling Platform Storage Using IO500 and Mistral

Nolan Monnier
Oral Roberts University
Sandia National Laboratories
monniern@oru.edu

Jay Lofstead
Sandia National Laboratories
gflofst@sandia.gov

Margaret Lawson
University of Illinois
Sandia National Laboratories
mlawso@sandia.gov

Matthew Curry
Sandia National Laboratories
mlcurry@sandia.gov

***Abstract*—This paper explores how we used IO500 and the Mistral tool from Ellexus to observe detailed performance characteristics to inform tuning IO performance on Astra, a ARM-based Sandia machine with an all flash, Lustre-based storage array. Through this case study, we demonstrate that IO500 serves as a meaningful storage benchmark, even for all flash storage. We also demonstrate that using fine-grained profiling tools, such as Mistral, is essential for revealing tuning requirement details. Overall, this paper demonstrates the value of a broad spectrum benchmark, like IO500, together with a fine grained performance analysis tool, such as Mistral, for understanding detailed storage system performance for better informed tuning.**

***Index Terms*—IO500, Mistral, Ellexus, System Monitoring, IO Performance**

I. Introduction

As HPC processing capabilities continue to rapidly improve, there is more and more pressure for the IO system to keep pace to prevent application bottlenecks. With the increased prevalence of solid state storage and the increased effectiveness of parallel file systems like Lustre, this pressure is shifting from hardware concerns toward software overheads and IO configuration. This paper concentrates on providing a methodology and set of tools that can be used to analyze and inform tuning IO system software and configuration.

Sandia National Laboratories' (SNL) new ARM-based supercomputer, Astra, is a testbed for proving a production capable open source HPC software stack on the ARM platform and for proving mission critical applications can run effectively on this architecture. Astra's platform storage is an all flash-based, Lustre parallel file system. Being one of the first ARM-based machines equipped with a Lustre file system, informing what needs to be fixed in the Lustre client and related software is an important contribution of this project. Particularly, the Astra team is exploring various information gathering tools, such as monitoring and/or benchmarking software, that can be used to optimize its configuration. This paper focuses on two of these tools and what they revealed that is informing the ongoing tuning work.

The IO500 benchmark [1], from Virtual Institute for IO [5], is a wide ranging benchmark ideal for IO tuning. It consists of a carefully designed set of twelve tests including IOR for write and read bandwidth; mdtest for file open, create, delete, stat; and a find operation to represent walking a file system for a purge or archive operation. These tests are divided into both "hard" and "easy" use cases. The hard cases represent what the community has experienced are the most difficult patterns a storage system may encounter. For example, for bandwidth, unaligned, 47KiB writes from each client process all to a single file is a significant challenge. For the easy configuration, the user is allowed to configure the patterns such that they achieves the best performance possible on the system. The hard and easy tests are carefully interleaved and timed to 5 minutes for create-style operations representing the typical 90% forward progress requirement used in platform purchases. To address potential cheating, all parameters and scripts used must be disclosed as part of the submission. The combined score is the geometric mean of all of the tests and offers individual results as well as an aggregate score offering a combined ranking for the "best" overall system. The 10 node challenge version of IO500 seeks to showcase the best that a standardized system size can achieve. Unlike the general list, the 10 node challenge requires that 10 client nodes are used against any size storage. This is a more comparable test of pure software overhead and interconnect performance. When combined with the full scale IO500 results, configuration trends for different scales can be realized. With more than 100 submissions already in its infancy, IO500 has become a standard performance measure for multiple storage vendors when offering bids for new machines. While the bids are confidential, this has been observed at multiple supercomputing sites anecdotally.

The challenge with getting a top score for IO500 is to tune the parameters to balance the hard and easy bandwidth tests against the hard and easy metadata tests. In many cases for Lustre, sacrificing a bit of bandwidth can offer higher metadata performance. While a top score does not indicate that all applications can achieve that performance, the range from the hard to easy on bandwidth and metadata give bounds for what applications should expect. Further, since all configurations are shared, insights can be gleaned into how to better configure an application's IO. Any storage system weaknesses or biases should be revealed in the individual benchmark results compared against theoretical peak performance and the balance between bandwidth and metadata and the easy against hard tests.

The "flaw" of IO500 as well as any other coarse-grained test is that the results are over a longer time period. In this case, 300 seconds. When trying to tune IO500, or any other IO system tuning for that matter, determining how the system performs on an instantaneous basis rather than just over the course of a long-running operation would yield insights critical

to overall system performance and how an individual operation changes over its lifetime. For example, timing a bandwidth operation only reveals the average bandwidth over a given time. Understanding what is happening during the operation is not possible. If we were able to see what bandwidth is achieved at many points during the test, the effects of caching and interference, as they come and go, can be measured and understood.

Using a tool like Mistral [3] from Ellexus can generate finer-grained profiling data. Mistral collects detailed time series data about all aspects of IO operations. It can also manage IO with the potential to throttle excessive resource usage from overly demanding processes or jobs. The Darshan [2] tool created by Argonne offers some similar monitoring features, but evaluation of the differences is part of the future work.

This paper uses the IO500 benchmark as a way to explore what monitoring tools can offer as a way to help understand and guide optimization for IO activities on a platform.

This paper is organized as follows: Section II contains a brief description of the environment of Astra system. Section III describes running the IO500 without any other monitoring tools. Section IV explores the use of Mistral in profiling IO500 performance, and how this information informed the tuning of Astra. Section V provides the measurements from the experiments. Finally, in Section VI, this paper generalizes the lessons learned from the Astra tuning process and makes recommendations for proper tuning practices.

II. Setup

The following section briefly describes the Astra environment. Unlike most deployed large scale systems, Astra's purpose as a proving ground rather than a production resource means that the computational load on the machine is relatively light. The overall bandwidth achievement is expected to be between 20% and 50% of theoretical peak.

A. Hardware

Astra is a new, highly power efficient, aarch64 HPC prototype. It uses 2.0 GHz ARM Cavium Thunder-X2 2S:28C nodes with 128 GB of RAM. Astra contains 2592 nodes and is networked with a QDR InfiniBand interconnect.

The /lustre file system tested consists of 40 OSSes, 2 MDSes, 21 x 1.6TB NVMe devices per OSS, spread across three ZFS zpools per node and using raidz. It has 240 GB/s peak bandwidth and 990TB usable storage running Lustre 2.12.1. The /oscratch file system is 8 OSSes, each serving 10 OSTs; 2 MDS nodes, 1 MDT each. This is connected with 3 LNET routers each with 1x10GigE to Astra. The storage is configured with ldiskfs and Lustre 2.12. The MDT devices are 800GB 12Gb SAS SSDs. The storage hardware is all FDR IB attached to the MDS/OSS servers. The file system is EDR attached to the Astra IB fabric through the lnet gateways. The OSTs are HGST HUH721010AL4200 10G 7200rpm. Each OST is one RAID 6 8+2 LUN. Total of 80 OSTs.

The compute nodes have a 2:1 taper from rack-level switches into the next level of the network, whereas storage servers enjoy full bandwidth into the core (4xEDR, 100Gbps).

B. Software

Astra uses the RHEL7 OS. Testing is done on the /lustre mount point. Job submissions are handled through Slurm, in interactive mode. IO500 v1.1 is used for benchmarking. The version of Mistral used is 2.13.3, while Mistral visualization is done through Grafana 6.2.5.

III. IO500 Configuration

Recall that IO500 consists of fixed parameter hard tests and partially user configurable easy tests. Finding the right balance between these is what gives the best overall score.

Configuring Astra's flash-based storage array to achieve the best hard and easy results requires testing the ARM/Lustre/Flash combination. A recent paper about NERSC's Perlmutter storage system [6], [7] describes the process of specifying and buying a large scale flash-based storage array along with system trends. Astra's storage array is a second attempt to explore this space. With Astra's combination of both flash and ARM, this study offers a critical view into the viability of this combination for future platforms.

As a control, we ran the IO500 benchmark untuned. That is Astra was first configured without any additional system profiling tools. The program itself was also configured untuned, optimizing it through theory and experimentation.

While the IO500 benchmarks themselves do support a few configuration options, most of these options are specified with optimum settings for Lustre as the default. For example, the easy metadata test is already configured in the test script to run with files only at leaves. As another example, the easy bandwidth test writes a single file per process. While this leaves little room to see how best to optimize an application for the Astra architecture, it does provide any easy and consistent measurement for judging the IO performance of the architecture as a whole. Overall, the performance tuning required was more process and node counts rather than more detailed configurations.

As expected, achieving the best result on the IO500 requires adjusting the system setting. Some experimentation developed a solid approach. A series of shorter tests were run on /lustre to determine the best process. For example, the short test dealing only with node configuration yielded a score of 22.0241 for 121 nodes and 2 cores per node. Introducing specific striping patterns to optimize IO raised the score to 42.7811. Further adjustments brought the short score to 46.92. From these milestones, we generalized a process for configuring the full scale tests:

1) Obtain system information and theoretical characteristics.
2) Set test directories' stripe size based on test files' size and number of storage targets.
3) Determine number of nodes to use.
4) Increase the cores per node to maximize bandwidth, until the bandwidth for ior_easy reasonably approaches a theoretical limit.
5) Adjust the cores per node to balance bandwidth and metadata results

The theoretical maximum bandwidth of our system is 240 GB/sec. The stripe setting were set to maximize individual test performance:

- lfs setstripe -c 1 "$io500_ior_easy"
- lfs setstripe -c {# of storage tar.} -S 32064k "$io500_ior_hard"
- lfs setstripe -L mdt -E 1M "$io500_mdt_easy"
- lfs setstripe -L mdt -E 1M "$io500_mdt_hard"

To maximize bandwidth, the number of nodes allocated for each IO500 run equaled the number of storage targets in the /lustre mount point. For /lustre, the best performance was achieved with full load (58 cores) on 121 nodes. For the 10 node challenge, we ran full loads as well. Since the metadata tests do not rely on file storage, using the same striping is adequate.

IV. Mistral

Mistral is an IO profiling and system telemetry tool from Ellexus. It is designed to run in production to give a live feed of per-application IO patterns, IO performance, and host health metrics. It is therefore provides lightweight and scalable IO profiling capabilities ideal for characterising a benchmark such as the IO500. By capturing reads, writes, meta data patterns and IO performance alongside CPU and memory usage Mistral can identify rogue applications, bad IO patterns and system bottlenecks across high-performance computing clusters and distributed applications.

Mistral's preferred logging solution, Elasticsearch, was not used. We were unable to get permission for the software install during the time we had to perform our experiments. The impact of this is detailed in the results.

A. Performance Effects

Mistral did not appear to affect the IO500 performance significantly. While our platform is lightly loaded, determining an exact impact range would require considerable testing and is beyond the scope of this paper. Darshan has documented the overheads to less than 0.05% [2]. We expect Mistral to have similar overheads once detailed measurements are performed.

B. Data Gathering and Visualization

By default, Mistral gathers data at a 1 second interval on only a per-node basis aiming to monitor the system as a whole. It can be used to monitor a particular application run as well. This is how Mistral was used for these experiments. Individual per-process statistics would require a less scalable solution, like Ellexus' Breeze product. Breeze is aimed at detailed reporting to identify good and poor behaviors per process.

The logging approach is either to text files or via a plugin to a database like Elasticsearch [4]. Given contention with the text file logs, we had approximately a 0.2%-2.0% data loss. With the database plugin, this loss would have been avoided.

With the collected data, the recommended Grafana visualization workflow quickly revealed both expected performance graphs as well as the unexpected slow decay and performance variability. Example graphs are in Figures 1- 6. Given the space limitations, larger versions are placed in the appendix.

V. Results

Mistral's biggest strength is its ability to capture fine-grained data about systems performance with both the breadth and detail needed for good visualization. Mistral excels at not just monitoring applications with low overhead, but also at helping developers and system administrators identify problems and their root causes through enabling them to see interactions between data feeds. While Figures 1- 8 are difficult to read, what is most important is the shape of the curves and the highlighted areas. Full scale images are in the appendix with a table listing the mapping to make finding the corresponding figure easier.

One clarifying note. Metadata operations are broken into two parts. The MD part represents finding and modifying metadata (access, create, and rm). The STREAM part represents metadata operations for dealing with actual data. In particular, it is file stream operations associated with functions like fprintf and sscanf (i.e., seek and open).

Mistral identified various noteworthy performance features that warrant further investigation. These features take various

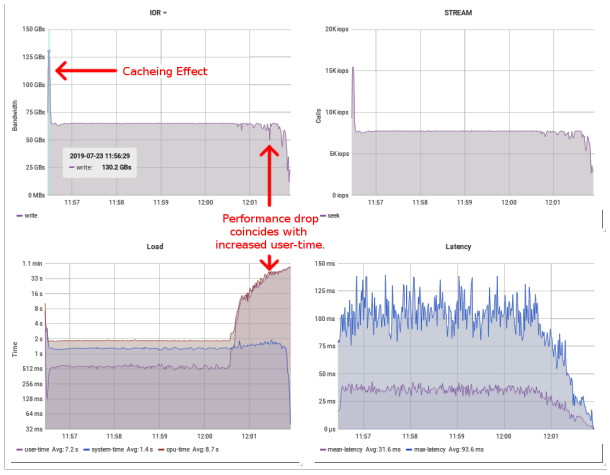

Fig. 1. **Mistral Results for ior_write_easy**

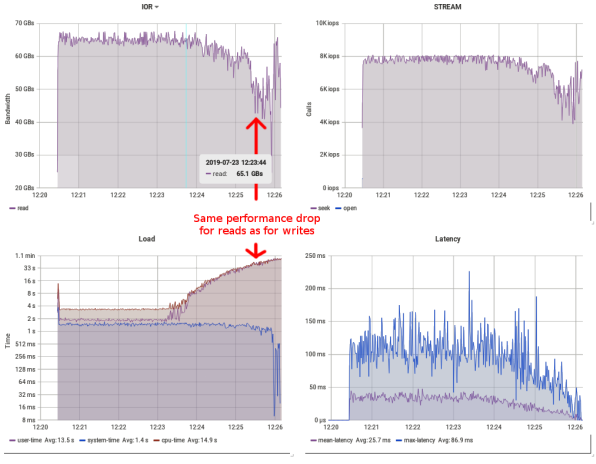

Fig. 2. **Mistral Results for ior_read_easy**

forms and have varying effects on performance. However, each feature described below was significant to the Astra team and showcases the importance of fine-grained profiling.

For example, IOR performance loss was occurring as a

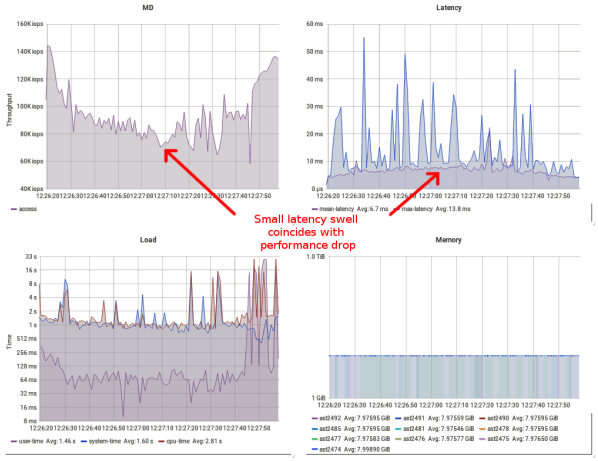

Fig. 3. **Section of md_stat_easy**

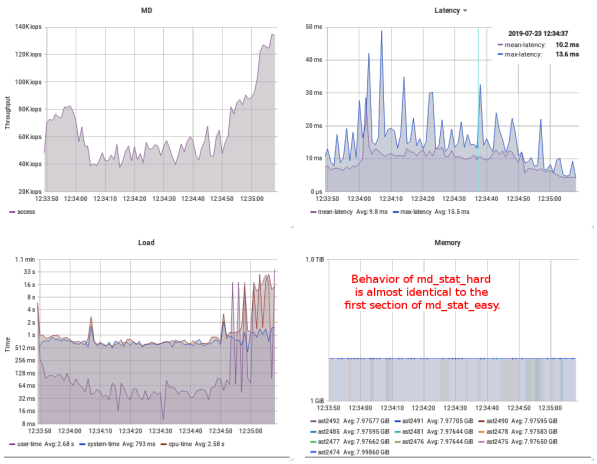

Fig. 4. **Mistral Results for md_stat_hard**

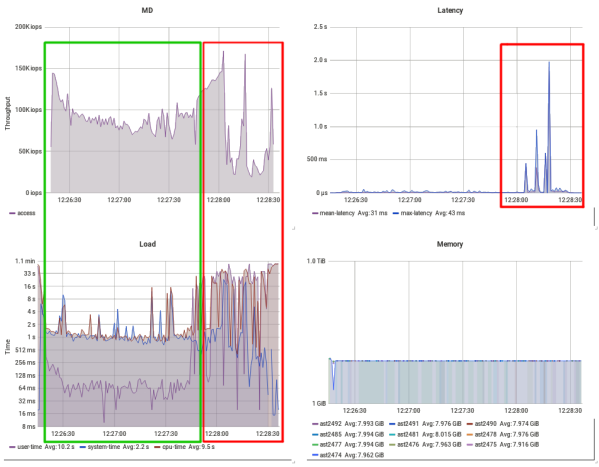

Fig. 5. **Mistral Results for md_stat_easy**

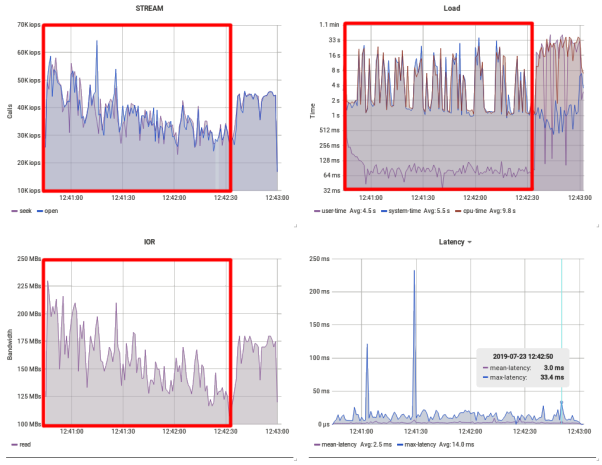

Fig. 6. **Mistral Results for md_read_hard**

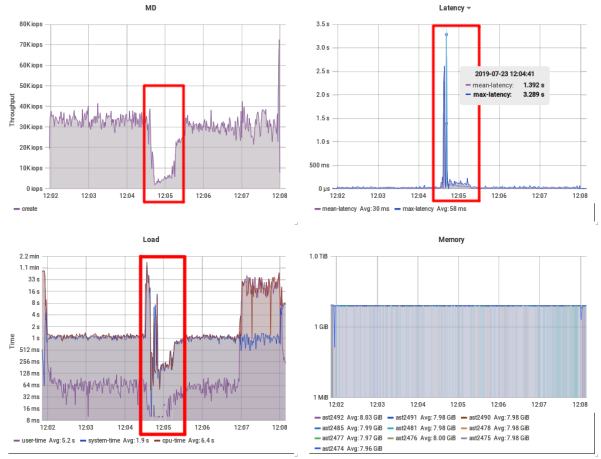

Fig. 7. **Mistral Results for md_write_easy**

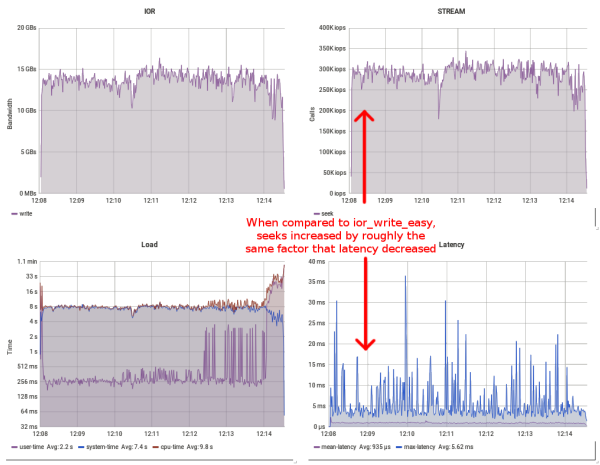

Fig. 8. **Mistral Results for ior_write_hard**

result of increased CPU Load (Figures 1 and 2). The loss is clearly different from a caching effect clearly labeled at the start of Figure 1. As load increases, bandwidth performance instability occurs. While the loss for ior_write_easy was estimated through hand calculation to be minor ($\sim$1%), the effect of the increased load on performance grew by a factor of six for ior_read_easy (at 6%-7%). Thus clearly, this bug in either the lustre system or the Astra hardware can have varying effects on performance and could exceed the estimated performance loss. As such, this finding warrants further investigation.

Mistral also reveals IOP latency swells in the middle of md_stat tests (Figures 3 and 4). This swell appears to correspond with metadata performance loss. Unfortunately, the actual percentage of performance loss is difficult to estimate because there is no consistent performance in either figure to form a basis for comparison. Nevertheless, it should be clear that this correlation provides inconsistent and less than optimal performance. Exploring this bug is part of our ongoing work.

There are other observed features that we have yet to explain. Most of these phenomenon deal with metadata. We have not determined the cause of the performance degradation shown in red on Figure 5. Estimates for performance loss in the red section of md_stat_easy are again difficult to accurately calculate. However, since the degradation is not present in md_stat_hard, it is possible that this feature might be eliminated all together. Metadata performance degrades more gradually in md_read_hard (Figure 6) but has the greatest performance impact of all estimated losses ($\sim$18%). There is also a sudden, unexplained drop in metadata performance in md_write_easy (Figure 7). This performance loss is estimated to be roughly 10%. While the IO500 scores show we are achieving reasonably expected performance, these details indicate that we can further optimize the ARM/Lustre system.

Mistral also captured more expected phenomenon. For example, ior_write_hard challenges performance by decreasing the transfer size. Mistral captured how the decreased transfer size increases file system seeks while decreasing write latency (Figure 8). As discussed earlier, Mistral can also display when and how often we cache (Figure 1). And all though this information is not particularly useful in our context, it could be useful for developers.

A. Astra Tuning

An immediately useful tuning feature of Mistral is actually not the information it gives us, but its ability to throttle performance. While this has tons of application in software development and system administrations, we found it useful to throttle the md_write_hard portion of the IO500. This kept us from breaking the files per directory limit and allowed us to obtain a potentially valid IO500 score on /oscratch without an incredible reduction in performance. This exploration then led to using the `-I` parameter to slow file creation so that it reached near the peak possible without crashing from exceeding this threshold. We investigated our ability to adjust these limits and determined that, for our installation, we were unable to adjust the maximum making our work on /oscratch incomplete compared to /lustre.

Other tuning required include further root cause analysis of the drivers and network to isolate the sources of the intermittent performance anomalies. Some of the coarse-grained results have been noticed, but this testing has revealed details enabling more detailed analysis and debugging the near first of its kind architecture.

B. IO500 Results

Table I shows the best results achieved in tuning IO500 for Astra. The results for the /lustre mount point are high enough to take the 14^{th} spot on the official IO500 board and the 13^{th} sport on the 10 node challenge as of the June 2019 list [1].

TABLE I
OPTIMIZED IO500 RESULTS

Nodes	Mount Point	Bandwidth	IOPS	Score
121	/lustre	84.8118 GB/s	35.9847 kiops	55.2443
10	/lustre	28.4097 GB/s	45.7227 kiops	36.0412

As strong as this initial showing for Astra is, the detailed IO500 results in the graphs more easily read in the appendix, shows some area for improvement. Further tuning of the Lustre client on ARM is still required to get better performance.

VI. Conclusion and Future Work

Overall, the IO500 benchmark proved an effective approach to test storage system performance. Our measured 33% of peak performance hit the middle of our fairly broad expected range of 20-50%. Having these hard and easy tests offers a way to experiment without using production applications while still generating values that can be compared against other machines of similar architecture to see if the achieved results are reasonable. Otherwise, a user would be left guessing if what they achieved was reasonable performance.

The addition of a monitoring tool, in this case Mistral, offered insights into *how* the IO operations actually performed. The periodic snapshot of the various system characteristics and instantaneous performance has offered avenues to attempt to tune the system. The standardized benchmarks IO500 employs offers a consistent testing approach for use with the monitoring tool. The combination is shown to reveal system performance details that would otherwise be hidden behind coarse numbers.

For future work, first we plan to compare what Mistral offers against Darshan. With many new technologies becoming viable, Sandia's testbed system collection is expanding. We plan to use this testing approach on these new architectures as the machines come online. Finally, measuring impact on Astra from this approach is being investigated.

Acknowledgements

Thank you to Ellexus for supporting this work.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] J. Bent, J. Kunkel, J. Lofstead, and G. Markomanolis, "IO500 Full Ranked List, Supercomputing 2018 (Corrected)," Nov. 2018, http://io500.org/list/19-01/start. [Online]. Available: https://doi.org/10.5281/zenodo.2602025

[2] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, "Understanding and improving computational science storage access through continuous characterization," *ACM Trans. Storage*, vol. 7, no. 3, pp. 8:1–8:26, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2027066.2027068

[3] R. Francis, W. Frings, R. Laifer, and S. Mendez, "Tools for analyzing parallel i/o," in *High Performance Computing: ISC High Performance 2018 International Workshops, Frankfurt/Main, Germany, June 28, 2018, Revised Selected Papers*, vol. 11203. Springer, 2018, p. 49.

[4] C. Gormley and Z. Tong, *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015.

[5] J. Kunkel and et al., "The Virtual Institue for IO," Jul. 2019. [Online]. Available: https://www.vi4io.org

[6] G. K. Lockwood, K. Lozinskiy, R. Cheema, L. Gerhardt, D. Hazen, and N. J. Wright, "NERSC all-flash Lustre file system design data and analysis," in *HPC-IO in the Data Center Workshop at ISC 2019*, Jun. 2019.

[7] ——, "NERSC all-flash Lustre file system design data and analysis," Jun. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3244453

APPENDIX

The figure mapping to the main paper text is in Table II:

TABLE II
MAIN TEXT FIGURE MAPPING TO APPENDIX FIGURE

Content Type	Main Text	Appendix
IOR Write Easy	Figure 1	Figure 9
IOR Read Easy	Figure 2	Figure 10
MD Section	Figure 3	Figure 11
MD Stat Hard	Figure 4	Figure 12
MD Stat Easy	Figure 5	Figure 13
MD Read Hard	Figure 6	Figure 14
MD Write Easy	Figure 7	Figure 15
IOR Write Hard	Figure 8	Figure 16

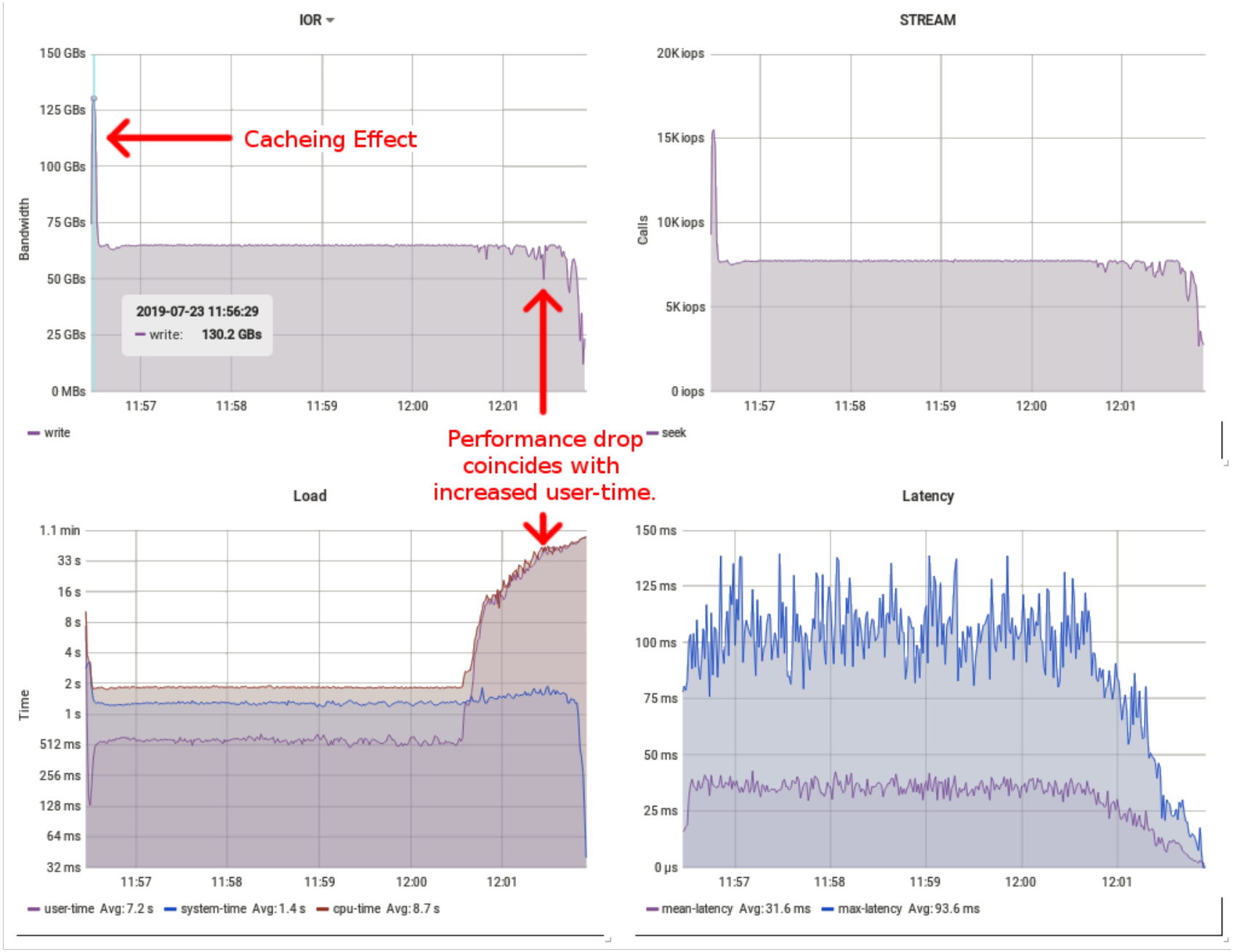

Fig. 9. **Mistral Results for ior_write_easy**

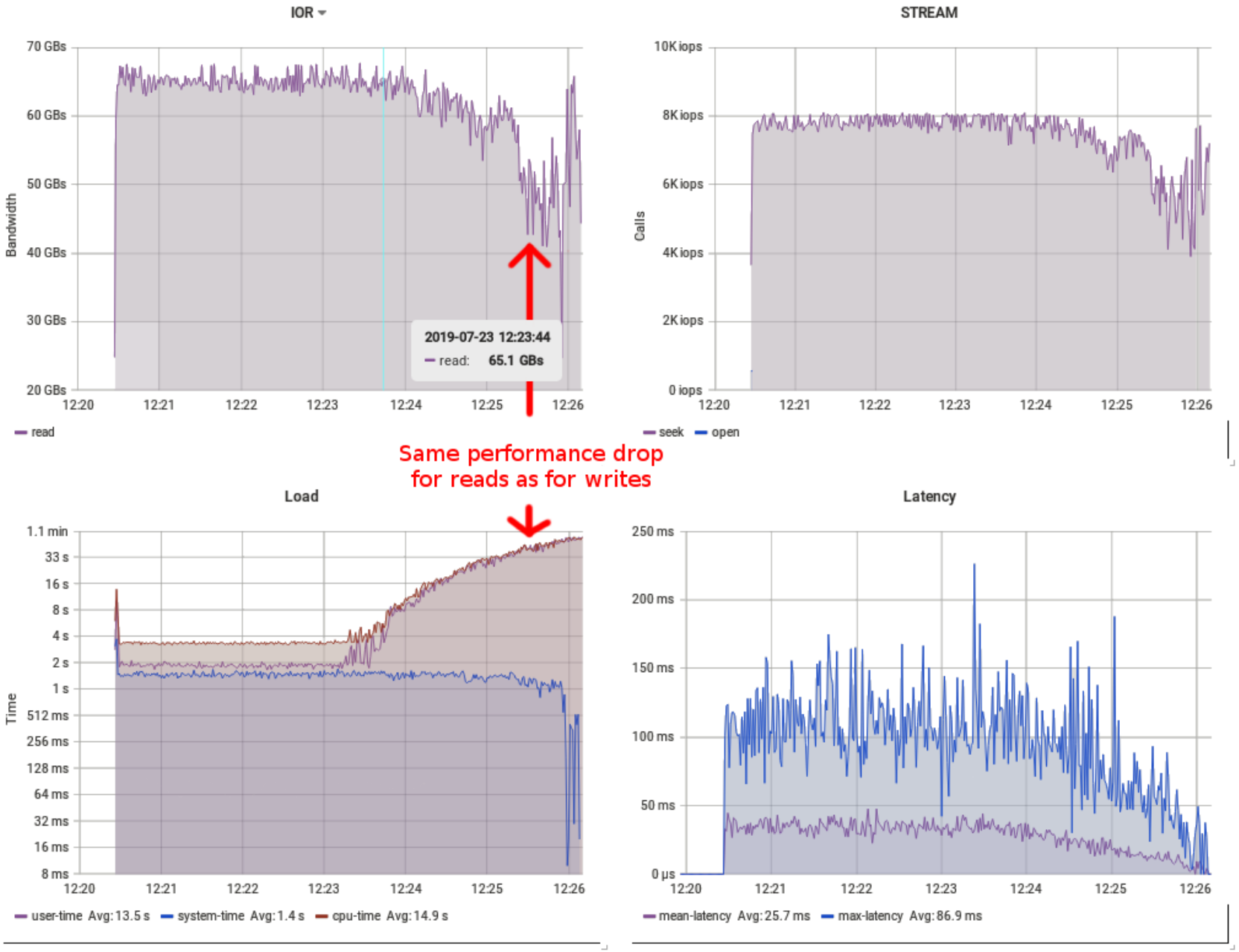

Fig. 10. **Mistral Results for ior_read_easy**

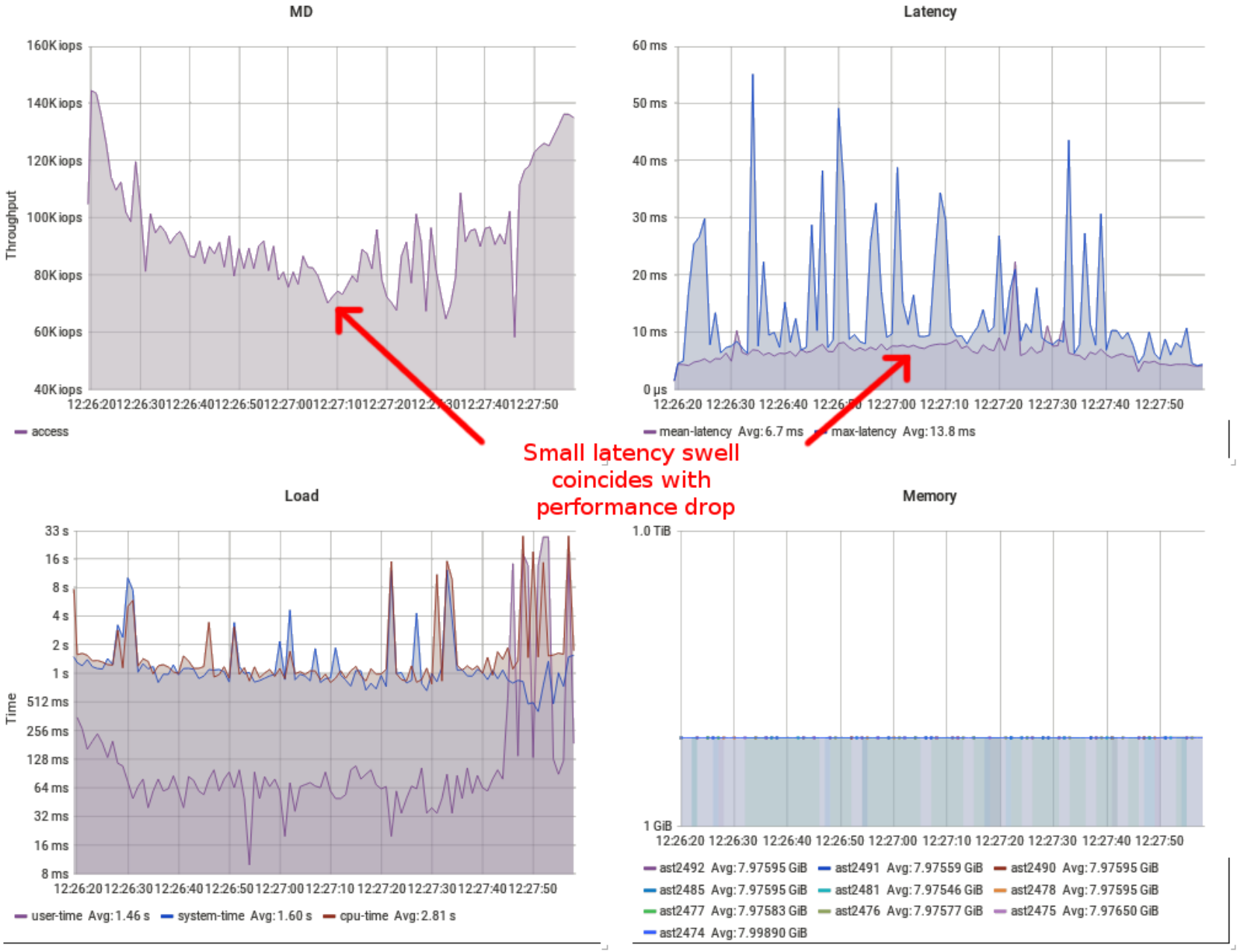

Fig. 11. **Section of md_stat_easy**

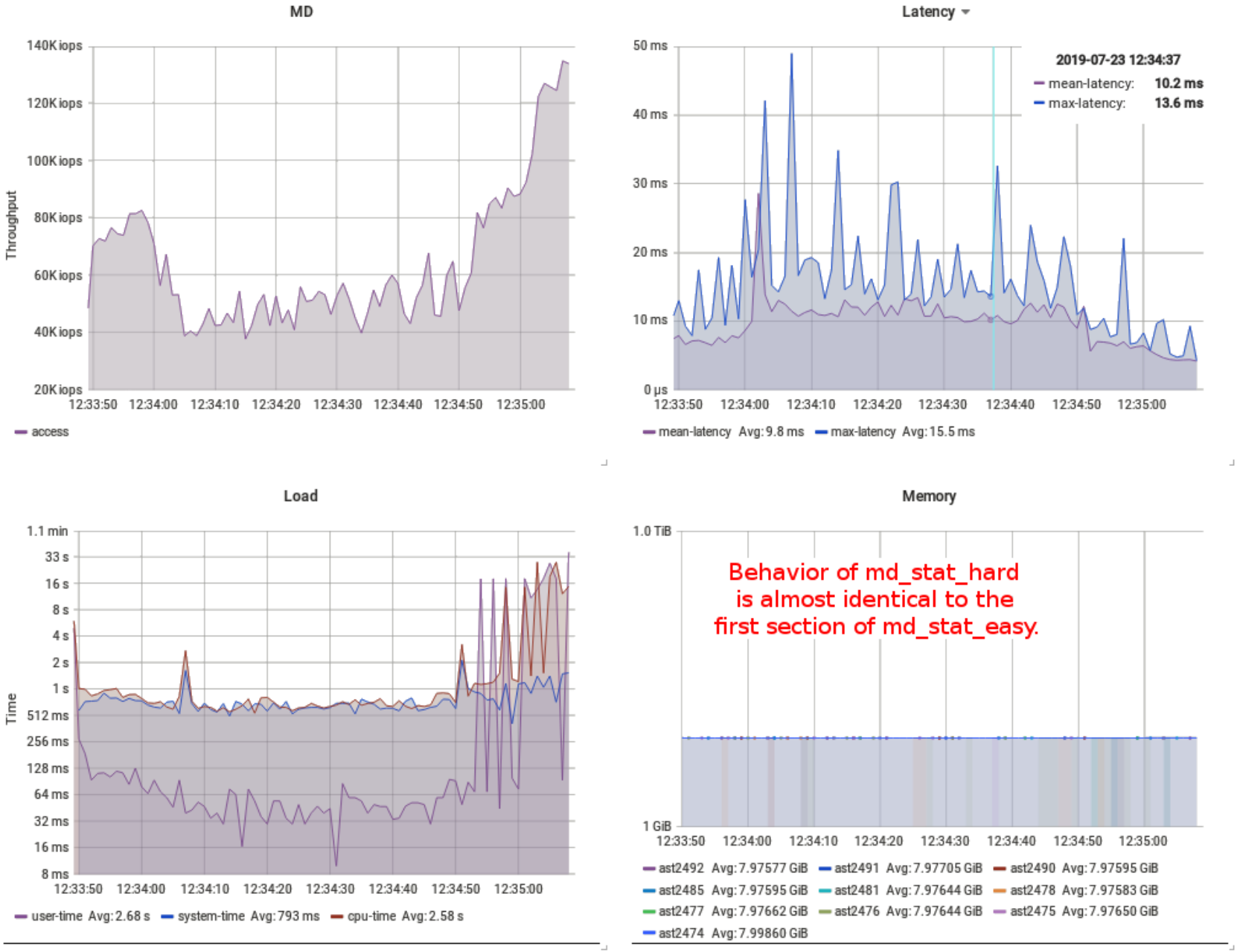

Fig. 12. **Mistral Results for md_stat_hard**

Fig. 13. **Mistral Results for md_stat_easy**

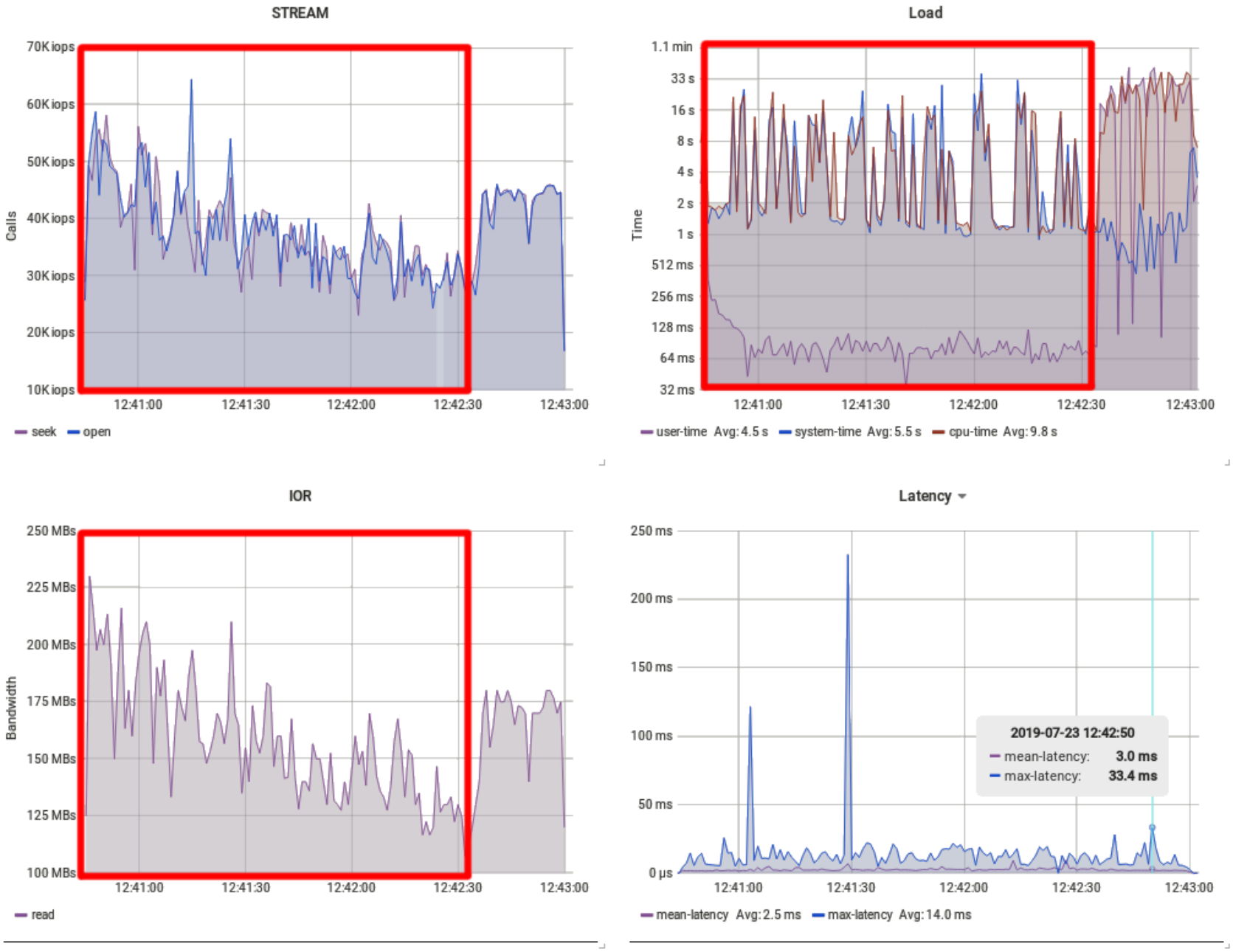

Fig. 14. **Mistral Results for md_read_hard**

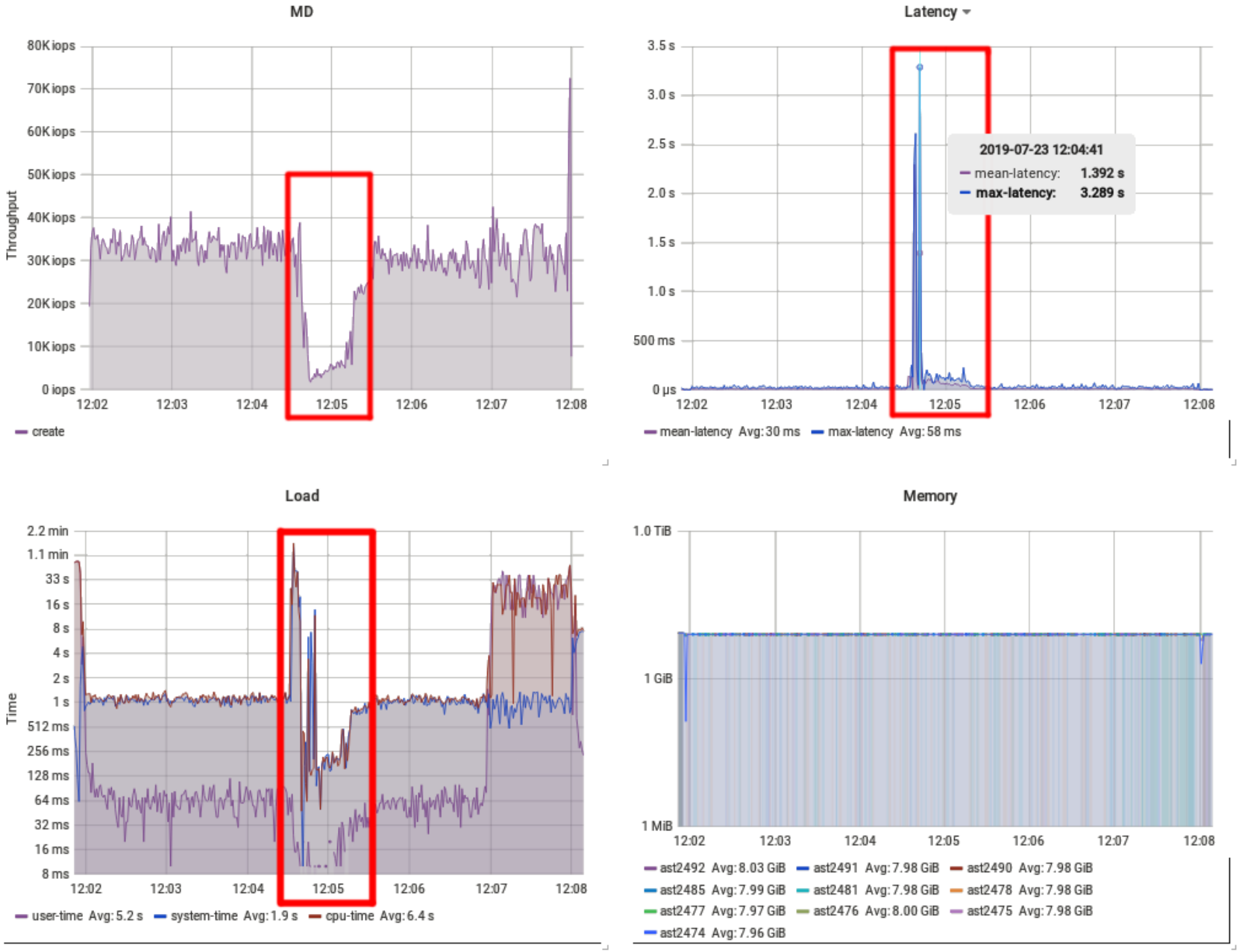

Fig. 15. **Mistral Results for md_write_easy**

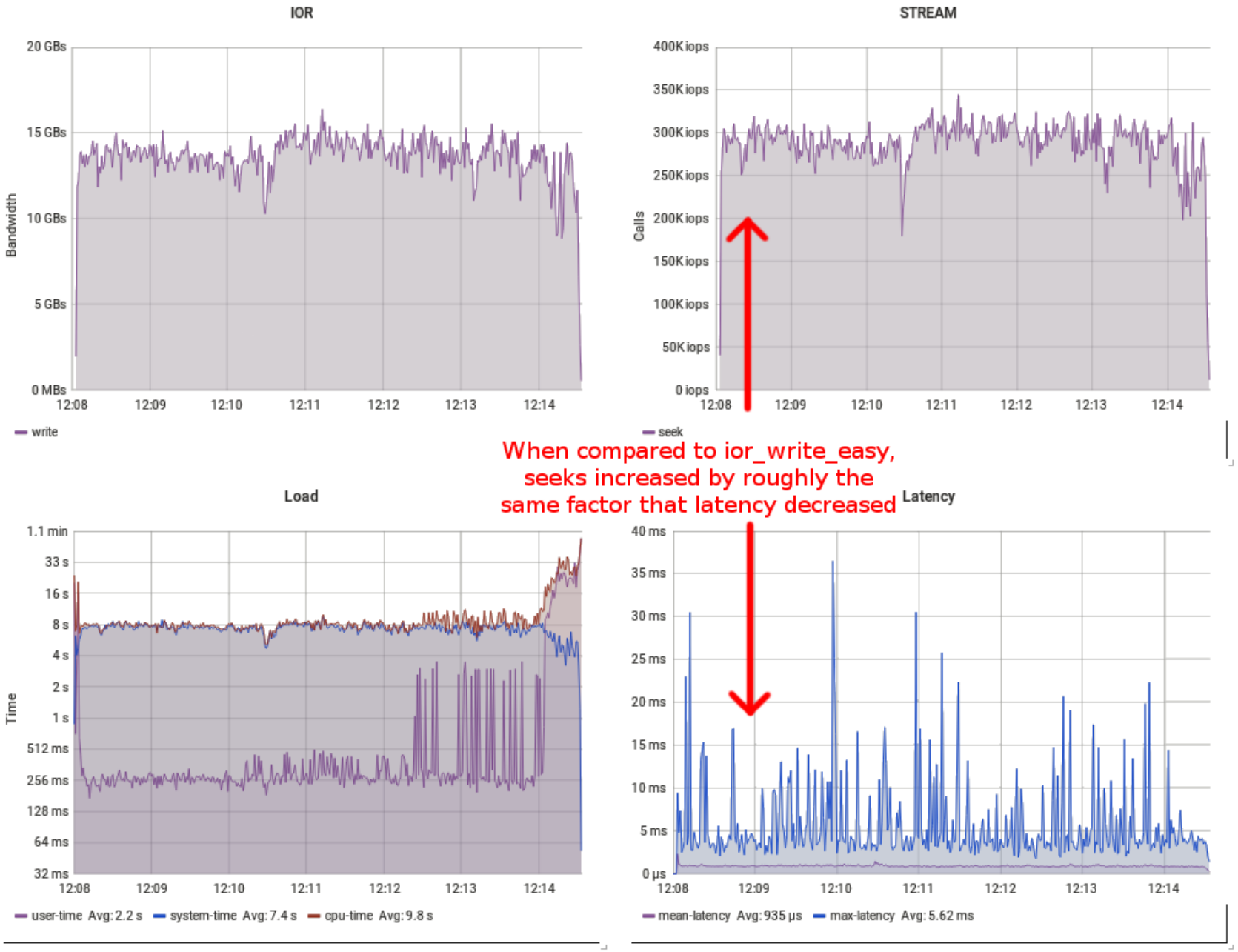

Fig. 16. **Mistral Results for ior_write_hard**