

Memory-based Computing: First Results

Elsa Buchholz,¹ Lars Haupt,¹ and Hermann Heßling¹

¹*German Centre for Astrophysics (DZA), Görlitz, Saxony, Germany;*

elsa.buchholz@dzaastro.de

Abstract. The Square Kilometre Array Observatory (SKAO) generates high-resolution 3D images of the universe in the petabyte range, which cannot be processed very efficiently with current processor-centric architectures due to the memory wall problem. This work investigates memory-based computing as an alternative architecture. The performance of a simplistic model for an I/O-dominated workflow is examined for network-mounted storage and compared with memory-based computing. The result is that memory-based computing speeds up the processing time by about two orders of magnitude. This result illustrates the potential of memory-based computing over conventional architectures, which are often used in astronomical workflows.

1. Introduction

The Square Kilometre Observatory (SKAO) is able to capture very high-resolution images of the universe, where individual images can be up to petabytes in size ¹. However, processors often stall because they are waiting for data, which can cause performance issues. This is known as the memory wall problem (Wulf & McKee 1995). Its significance increases when petabytes of data need to be processed. In order to address this issue with regard to the upcoming data from the SKAO, a paradigm shift is needed: Current processor-centric computing should be replaced by memory-based computing. The German Centre for Astrophysics (DZA) ² is exploring this new type of computing architecture developed by Hewlett Packard Enterprise (HPE). This article presents performance results for a simplistic astronomical workflow analyzed on memory-based computing, using methods similar to those described in (Heßling et al. 2021). In astronomy, a common workflow is to read data from a hard disk during an imaging workflow. The performance of analyzing large amounts of data from hard disks mounted over Ethernet is compared with the performance of analyzing data using memory-based computing.

¹SKA Observatory, Science Processing Centers, accessed 2025-11-20, URL: <https://www.skao.int/en/explore/big-data/361/science-processing-centres>.

²German Centre for Astrophysics, accessed 2025-11-23, URL: <https://www.deutscheszentrumastrophysik.de/en>.

2. Architecture and Methods

This work uses a prototype for a memory-based computing system called 'Julia'. 'Julia' is housed in a 19" rack and has a modular architecture: There are 8 chassis, each equipped with 4 CPUs, and each CPU has 28 cores and 1.5 TB of main memory (DRAM). The four CPUs within a chassis are connected to each other in ring via Ultra Path Interconnect (UPI) and are also directly connected to each CPU in the other chassis via HPE Superdome Flex Grid cables (Hewlett Packard Enterprise 2021). The bandwidths for random read operations of any CPU node from memory of all other CPU were determined using the Intel Memory Latency Checker (Corporation 2025). As can be extracted from Fig. 1 (middle), reading from main memory directly attached to a CPU is fastest at 96 GB/s, while reading from the main memory of other CPUs in the same chassis is significantly slower at 17 GB/s. Reading from the main memories in a remote chassis via Flex Grid is the slowest, but at 12 GB/s it is significantly faster than accessing a remote hard disk storage via a 10 Gbps network connection.

For the analyses described below, two chassis with a configuration as shown in Fig. 1 (left), were used. The CPUs in one chassis were not used, only their main memories ('fabric') were used to create two shared memories, see Fig. 1 (left bottom). Each shared memory has 3 TB of storage space, which the CPUs in the other chassis ('compute') can access via a file mount and at 12 GB/s.

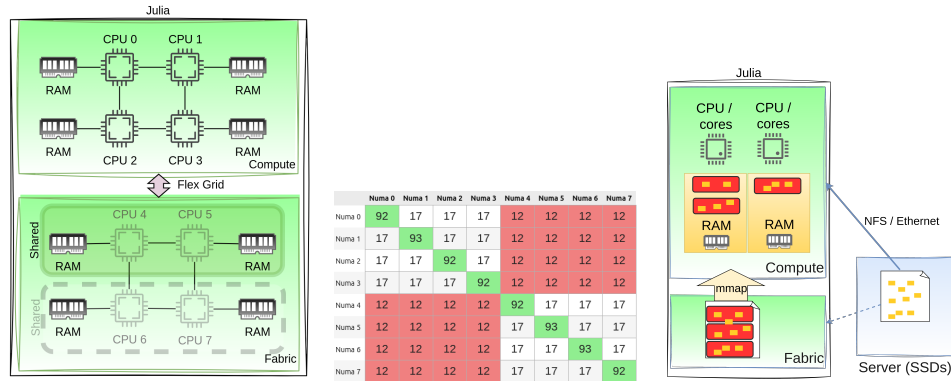


Figure 1. Left: The CPUs on the top are used for compute and the main memories of the CPUs at the bottom are combined to two shared memories ('fabric'). Middle: Performance of connections between CPUs in the memory-based system: The average bandwidth [GB/s] for random read operations from each NUMA node to all other NUMA nodes are determined using Intel's Memory Latency Checker. Right: Workflow based on a simplistic model of an SQL-based analysis. The data is read in small chunks [size 1KB] that are randomly distributed in a file, filled with random digits [see yellow boxes], and - to imitate an analysis - the sum of the digits in the chunks is calculated. First, according to the traditional workflow pattern, the file stored on a remote server is accessed via NFS (over Ethernet). Alternatively, the file is copied to the 'fabric' of Julia, and large subsets of the file ('windows' see red boxes) are provided from there for in-memory computation (via `mmap`), where the 'windows' are analyzed in parallel by threads. Illustration by the authors.

A simplistic SQL-like workflow is used to compare the performance between a traditional file read and a memory-mapped file read; see Fig. 1 (right). To ensure that reads are performed directly from the physical disk and fabric, without the effects of

cached data affecting performance, the page cache of the operating system (OS) is dropped. The `fread` system call is commonly used for reading files from disk. This benchmark operates in a single-thread mode and reads each chunk sequentially. The second benchmark, which involves reading a memory-mapped file, uses the `mmap` system call. In addition, the file is divided into 'windows' for `mmap`, which are locked into memory using the `mlock2` method. This ensures that the mapping stays in DRAM and cannot be read from the underlying 'fabric' (Kerrisk 2025).

3. Results and Discussion

In astronomy, data are often read from a remote disk storage that is mounted over a network. Fig. 2 (left) shows the results of analyzing the simplistic workflow for a file accessed via Network File System (NFS) over Ethernet. Only one thread was used for the analysis, but with repetitions at different times. The error bars show the standard deviations around the mean values. The strong fluctuations of the mean values could be caused by temporary high network utilization.

Fig. 2 (right) shows the results of the corresponding analyses of the same simplistic workflow on memory-based computing. First of all, it should be noted that the times are reduced by about two orders of magnitude, i.e. I/O-dominated workflows can benefit significantly from memory-based computing. Furthermore, the relatively small error bars show that no temporary strong fluctuations are to be expected in 'Julia's' closed system. In addition, the number of threads was varied, resulting in a further increase in performance. It is important to note in this context that the speedup is less than linear when the number of threads is increased. However there is no noticeable drop in performance with more than 8 threads, as was observed, for example when running the astronomy tool CASA (Common Astronomy Software Applications) in parallel (Buchholz 2020).

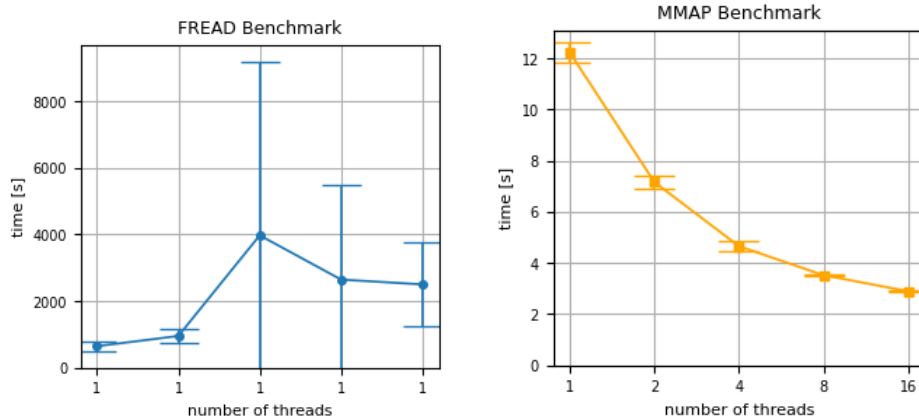


Figure 2. Benchmark results of the simplistic model: The time for analysing a 1.5 TB file with 1 millionen chunks (y-achsis) versus the number of threads (x-achsis), the error bars show the standard-deviation. Left: Conventional analysis where the file is stored on a mounted remote disk space. Right: Memory-based analysis where the file is stored in a fabric. Memory-based computing leads to a speedup of the time by nearly two orders of magnitude.

To summarize, a comparison between a traditional file read and memory-mapped file read shows that the latter achieves a speedup of nearly two orders of magnitude. Memory-based computing enables faster access to large files than the traditional `fread` method. This performance improvement is relevant for the efficient processing of petabyte-scale data volumes produced by the SKAO. The impact of memory-based computing on an astronomical workflow will be explored next. Large files must be divided into smaller 'windows' so that they fit into the main memory of individual CPUs in the 'compute' area, which could lead to data loss. The impact of this partitioning on multithreaded analysis will be examined next, as well as the impact of using Zarr files (Miles et al. 2023).

Acknowledgments. We are grateful for stimulating discussions with Sagar Borra (DZA Görlitz), Marc Drobek (DZA Görlitz), Andrei Kazantsev (MPIfR Bonn), Yurii Pidopryhora (MPIfR Bonn), Tanumoy Saha (HTW Berlin), Marcel Trattner (HTW Berlin), Yanik Yiannakis (MPIfR Bonn). We would like to thank the Center for Information Services and High Performance Computing (ZIH) of TU Dresden for providing computational resources and support. We would also like to express our sincere gratitude to Clarete Riana Crasta from HPE for her invaluable assistance in enhancing our understanding of the memory-based computing architecture.

References

- Buchholz, E. 2020, Master's thesis, University of Applied Sciences (HTW) Berlin, Berlin. Submitted in fulfillment of the requirements for the degree of Master of Science (M.Sc.) in Applied Computer Science, Department of Computer Science, Communication, and Business
- Corporation, I. 2025, Intel Memory Latency Checker (Intel MLC), Intel Corporation. Accessed: 2025-11-23, URL www.intel.com/content/www/us/en/download/736633/intel-memory-latency-checker-intel-mlc.html
- Heßling, H., Strutz, M., Buchholz, E. I., & Hufnagl, P. 2021, Big Data Res., 25. URL <https://doi.org/10.1016/j.bdr.2021.100214>
- Hewlett Packard Enterprise 2021, HPE Superdome Flex Server Architecture and RAS, Tech. rep., Hewlett Packard Enterprise. Accessed: 2025-11-23, URL <https://hpe.metroconnect.co.th/wp-content/uploads/2023/11/HPE-Superdome-Flex-Server-Architecture-and-RAS-technical-white-paper-a00036491enw.pdf>
- Kerrisk, M. 2025, mlock2 Linux manual page, man7.org Training and Consulting. Accessed: 2025-11-23, URL www.man7.org/linux/man-pages/man2/mlock.2.html
- Miles, A., Striebel, J., Rzepka, N., Maitin-Shepard, J., & Moore, J. 2023, Zarr Core Specification, Version 3.1. Accessed: 2025-12-13, URL <https://zarr-specs.readthedocs.io/en/latest/v3/core/index.html>
- Wulf, W. A., & McKee, S. A. 1995, ACM SIGARCH computer architecture news, 23, 20