Meeting the Real-Time Challenges of Ground-Based Telescopes Using Low-Rank Matrix Computations

Hatem Ltaief
Hatem.Ltaief@kaust.edu.sa
Extreme Computing Research Center
Computer, Electrical and Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

Jesse Cranney
Jesse.Cranney@anu.edu.au
Research School of Astronomy & Astrophysics
College of Science
Australian National University
Australia
School of Electrical Engineering & Computing
Faculty of Engineering and Built Environment
University of Newcastle
Australia

Damien Gratadour
Damien.Gratadour@anu.edu.au
Research School of Astronomy & Astrophysics
College of Science
Australian National University
Australia
LESIA, Observatoire de Paris
PSL University, Sorbonne University, Paris University
CNRS
France

Yuxi Hong
Yuxi.Hong@kaust.edu.sa
Extreme Computing Research Center
Computer, Electrical and Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

Laurent Gatineau
Laurent.Gatineau@emea.nec.com
NEC Deutschland GmbH
HPC Division
Germany

David Keyes
David.Keyes@kaust.edu.sa
Extreme Computing Research Center
Computer, Electrical and Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

ABSTRACT

Adaptive Optics (AO) is a technology that permits to measure and mitigate the distortion effects of atmospheric turbulence on optical beams. AO must operate in real-time by controlling thousands of actuators to shape the surface of deformable mirrors deployed on ground-based telescopes to compensate for these distortions. The command vectors that trigger how each individual actuator should act to bend a portion of the mirror are obtained from Matrix-Vector Multiplications (MVM). We identify and leverage the data sparsity structure of these control matrices coming from the MAVIS instruments for the European Southern Observatory's Very Large Telescope. We provide performance evaluation on x86 and accelerator-based systems. We present the impact of tile low-rank (TLR) matrix approximations on time-to-solution for the MVM and assess the produced image quality. We achieve performance improvement up to two orders of magnitude for TLR-MVM compared to regular dense MVM, while maintaining the image quality.

SC '21, November 14–19, 2021, St. Louis, MO, USA

ACM ISBN 978-1-4503-8442-1/21/11.
https://doi.org/10.1145/3458817.3476225

KEYWORDS

Ground-Based Telescopes, Real-Time Computational Astronomy, Tile Low-Rank Approximation, Matrix-Vector Multiplication.

ACM Reference Format:
Hatem Ltaief, Jesse Cranney, Damien Gratadour, Yuxi Hong, Laurent Gatineau, and David Keyes. 2021. Meeting the Real-Time Challenges of Ground-Based Telescopes Using Low-Rank Matrix Computations. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21), November 14–19, 2021, St. Louis, MO, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458817.3476225

1 INTRODUCTION

Our knowledge of the Universe will make a giant leap as the largest ground-based telescopes, with diameters of 25 to 40m [13, 41], see first light before the end of this decade. They will provide the angular resolution and collecting area required to detect the first stars and first galaxies as well as faint rocky exoplanets around other stars, possibly harboring life. But to reach the required resolution and contrast, they must overcome optical distortions induced by atmospheric turbulence. In order to compensate for such distortions, Adaptive Optics (AO) technologies were developed for astronomy more than 30 years ago, and are now essential components for most of the optical telescopes currently in operation [34]. In its simplest form, an AO system is composed of a Wavefront Sensor (WFS) used to measure atmospheric distortions at a high frame rate, which are then compensated with a Deformable Mirror (DM). The sub-system responsible for interpreting wavefront measurements into actual

commands to actuators is the Real-Time Controller (RTC). It must operate at high speed (i.e., kHz rate) to keep up with the rapidly changing optical turbulence. Most importantly, since AO is usually arranged in a closed loop setting, time-to-solution must be reliably at the level of a fraction of the operating rate (few tens of μs) to ensure stable behavior for this loop.

Wavefront sensing and actuator commands retrieval are based on modeling of the AO system error budget and solved with a linear control system, sometimes regularized with a given prior on the turbulence statistics. For instance, in present-day conventional AO systems, the RTC follows a scheme in which input measurements from sensors are reduced into a measurement vector which is multiplied by a control matrix to produce an output DM control vector of commands [40].

Other instruments are coupled with the AO module, as for example an imager or a spectrograph. Observations are usually carried out at wavelengths ranging from 1μm to 5μm, i.e., the Infrared (IR) window of the atmosphere, where the transmission is high, and for which AO systems deliver their best performance. Extending AO coverage to shorter optical wavelengths (i.e., below 1μm) is a challenge because it requires wavefront correction on very short spatial and temporal scales. While designing systems with a larger number of actuators and a faster response time is not that challenging conceptually, realizing these conceptual designs in practice is problematic because they demand exquisite stability, ultraprecise wavefront reconstruction and flawless subsystems calibration.

One of the limitations of classical AO is that the correction is only valid in a very small patch of sky, the size of which depends on the observing wavelength, from a few arcsec in the visible to a few tens of arcsec in the near IR. Multi-Conjugate Adaptive Optics (MCAO) solves this problem by using a series of DMs to compensate the turbulence in volume, enabling AO correction over a wide field of view (FoV) [9]. MCAO uses several guide stars and associated WFSs to probe the light wave aberrations in several directions, and an RTC using tomographic reconstruction determines the best commands to apply to the DMs. In this case, the control matrix is often called the tomographic reconstructor [45]. Figure 1 depicts the principle of MCAO in a very simple configuration with only 2 WFS and 2 DMs.

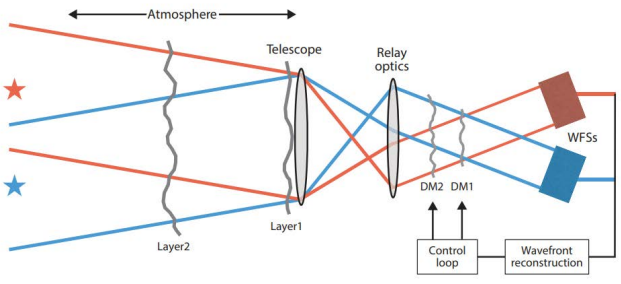


Figure 1: The concept of MCAO in which multi-directional measurements from several WFS are used to drive several DM, each optically conjugated to a given physical altitude of turbulence.

Indeed, while turbulence is a continuous stochastic process that develops in a volume (the whole atmosphere), current numerical models are based on a discrete set of infinitely thin layers at different altitudes (typically from 0 to 15km), each contributing to a fraction of the total turbulence, and each in a frozen flow with varying speed and orientation depending on their altitude. Usually, about 10 to 40 layers are enough to reproduce high resolution turbulence profiling data. These high resolution models are matched with actual hardware and only a few DMs are needed to achieve good performance.

A typical AO RTC is thus composed of two main sub-systems:

- a so-called Hard-RTC (HRTC), responsible for performing the main pipeline, dominated by the Matrix-Vector Multiply (MVM), with extremely tight constraints on time-to-solution, and,
- a so-called Soft-RTC (SRTC), responsible for leading a statistical analysis of the telemetry data from the AO system to identify the parameters of this turbulence model and compute the appropriate tomographic reconstructor.

In this paper, we leverage the performance of HRTC by exploiting the data sparsity of the tomographic reconstructor matrices coming from the MAVIS[1] AO instrument, currently built and scheduled for deployment at the European Southern Observatory's Very Large Telescope. In particular, we compute the Tile Low-Rank (TLR) approximations that consist in compressing individually each tile of the matrix operator up to an accuracy threshold – just enough to ensure the overall image quality, measured by the Strehl Ratio (SR), remains within an acceptable range to meet the science goals. We capture and retain the most significant information by using existing linear algebra compression algorithms (e.g., rank-revealing QR [27], regular/randomized singular value decomposition [32], etc). To generate a compressed data structure, composed of bases for each tile. The original dense MVM algorithm requires to be redesigned so that it can take into account this compressed tile data layout structure. This may engender a smaller number of flops and memory footprint, which can translate into a potential gain in terms of time-to-solution. However, the inherent memory-bound nature of MVM may be further exacerbated since TLR-MVM operations are fine-grained compared to dense MVM. We increase data locality by stacking the compressed bases in memory to ensure memory accesses are contiguous and cast the resulting matrix operations into standard BLAS calls. The latter permits to rely on vendor optimized numerical libraries for performance, while providing portability to our code. The tile size becomes a paramount tunable parameter, not only for performance, but also for numerical accuracy. We ultimately assess its impact along with the accuracy threshold on the numerical accuracy of the output image. Although we primarily focus on MAVIS, our TLR-MVM implementation can be used across a wide range of AO instruments [8, 13, 20].

We launch a thorough performance benchmarking campaign on a variety of x86 and accelerator-based vendor systems. We demonstrate the numerical robustness of our implementation using real datasets from MAVIS. We achieve performance improvement up to two orders of magnitude for TLR-MVM compared to the regular dense MVM, while maintaining an acceptable image quality.

[1] http://mavis-ao.org/mavis/

This further enables to meet the real-time challenges of capturing the turbulence evolution, which can potentially increase the AO performance by either reducing the overall latency of the AO system response to incoming perturbations or offering more room for additional fine grain processing in the overall HRTC pipeline. Finally, we highlight the importance of time reproducibility and predictability, which both ensure stable operations of this closed loop control system, as well as the need to rely on standardized libraries and programming models to build a hardware-software ecosystem when targeting complex multi-billion-Euro telescope campaigns sustainable over long hardware lifetimes.

Our main contributions can be summarized as follows. For the first time, we identify the data sparsity of the command matrix in AO real-time controllers that has an impact on many instruments currently deployed or yet to be deployed on-sky, beyond the herein studied MAVIS instrument. We envision this study will help advance the research in computational astronomy for ground-based telescopes. We provide a portable software solution for the TLR approximation applied to MVM that optimizes data locality by stacking the compressed bases. We study its accuracy impact on real datasets using HPC systems from various vendors. We perform an extensive performance benchmarking campaign and assess our implementation by looking at sustained bandwidth on each system via roofline performance models. To our knowledge, this is the first TLR approximations are used to accelerate HRTC workloads in the context of computational astronomy.

The remainder of the paper is as follows. Section 2 presents related work. We describe the challenges and opportunities when operating an AO instrument in Section 3. We recall the TLR approximation in Section 4 and provide implementations details of TLR-MVM in Section 5. Section 6 assesses the obtained numerical accuracy of the output images. Section 7 outlines TLR-MVM performance results in time and bandwidth on shared and distributed-memory systems. Section 8 gives an AO perspective moving forward and emphasize on the need to come up with a hardware-software-application co-design for computational astronomy and we conclude in Section 9.

2 RELATED WORK

Low-rank approximations represent a major trend in the linear algebra community as they enable not only to tackle the curse of dimensionality for big data applications (e.g., tensors) [29] but also to solve a broad class of traditional HPC large-scale scientific applications [15]. Low-rank approximations appear, therefore, as one of the efficient algorithmic techniques that facilitate the convergence between HPC and Big Data problems [21]. By retaining the most critical information of the matrix operator up to an application's accuracy threshold, they can significantly reduce the algorithmic complexity and the memory footprint. There exist several low-rank matrix approximations, especially in the form of hierarchical matrices ($\mathcal{H}$-matrices) [28, 31, 37]. In fact, there are many state-of-the-art data compression formats for $\mathcal{H}$-matrix approximation supporting weak (e.g., Hierarchically Semi-Separable (HSS) [19, 44], Hierarchically Off-Diagonal Low-Rank (HODLR) [5, 7]) and strong admissibility (e.g., $\mathcal{H}^2$-matrix [11]). The former is often the method of choice to solve 2D problems that exhibit small ranks for the off-diagonal blocks, while the latter provides better complexity for 3D problems, characterized by high ranks. Their hierarchical data structure has slowed down their wide adoption in HPC applications since they often require new specific optimized kernel developments that are not available in vendor standard optimized numerical libraries, such as BLAS. To mitigate this productivity issue and sustainability concerns, the Tile Low-Rank (TLR) compression data layout format relies instead on a flat data structure that permits to express most of matrix operations in terms of BLAS calls. Since TLR is a compromise between optimality and implementation complexity, TLR has managed to penetrate many applications. In particular, TLR has been successful in solving dense/sparse linear algebra problems at scale [2–4, 6, 14, 16, 17, 36] on a broad range of hardware architectures (i.e., x86, accelerators, shared/distributed-memory systems). As far as leveraging Adaptive Optics (AO) workloads with HPC, there are mostly works in supporting Soft-RTC (SRTC) [18, 23, 26, 30, 38] based on dense linear algebra algorithms powered by dynamic runtime systems [1, 10, 12] in the context of the MOSAIC instrument for the European Extremely Large Telescopes [33] and the Subaru Coronagraphic Extreme-AO (SCExAO) system of the Japanese Subaru telescope [39].

The authors in [22] have accelerated for the first time SRTC workloads for tomographic AO using low-rank approximations based TLR and $\mathcal{H}$-matrices, which improves the generation and factorization of the large covariance matrix needed to compute the tomographic reconstructor. In this paper, we identify and demonstrate for the first time the feasibility of deploying TLR matrix approximation in the Hard-RTC (HRTC) pipeline. HRTC is mostly driven by the dense Matrix-Vector Multiplication (MVM) that operates on the command matrix coming from the SRTC computational phase. In fact, these matrices are not sparse but data-sparse. In other words, their original data structure is dense but they can be compressed using tile low-rank approximation. The resulting compressed tile data structure expressed with tall (left) or squat (right) bases. Though the block they represent is data-sparse, these objects are dense and the standard SpMV data structures (e.g., CSR, COO, ELL, SELL-C, etc.) do not apply. Instead, we have to stack the bases for contiguous memory access purposes and reformulate the original dense MVM using an algebraic formulation by means of batch dense MVMs with variable sizes. Last but not least, TLR-MVM relies on standard programming models and vendor optimized libraries since observatories impose strong constraints on using standard software tools and libraries for long-term maintenance cost reasons.

3 AO INSTRUMENTS: CHALLENGES AND OPPORTUNITIES

MAVIS [43] is a general-purpose instrument for exploiting the highest possible angular resolution of any single optical telescope available in the next decade with sensitivity comparable to or better than larger aperture facilities. The principal innovation of MAVIS is to deliver a large AO-corrected FoV with an image quality close to the diffraction limit at optical wavelengths, across the vast majority of the sky. By probing the frontier of angular resolution and sensitivity across a large portion of the observable sky, MAVIS will enable progress on an array of scientific topics, from our own planetary system to those around other stars, and from the physics

of star formation in the Milky Way to the first star clusters in the Universe. The MAVIS MCAO module will provide a beam, cleaned of atmospheric turbulence, to an imager and an Integral Field Unit spectrograph. To achieve that, it will use not only field stars for the various WFS but also artificial stars, so-called Laser Guide Stars (LGS), created in the mesosphere using high intensity lasers tuned to Sodium excitation lines.

These unique features of MAVIS come with a number of challenges. In particular, driving this extremely complex AO system, requires a powerful RTC able to cope with the very large amount of data streamed from the various WFS, the distributed nature of the AO correction over several conjugated layers of turbulence as well as the ultraprecise wavefront reconstruction required to meet the performance specifications together with non-functional requirements imposed by real-time operations.

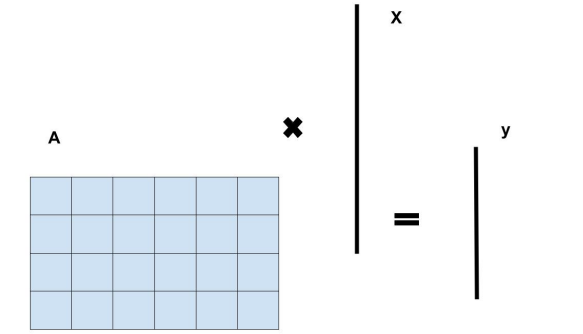
Due to the tight error budget of MAVIS, it is likely that the final design will include a predictive linear controller. One such controller is the so-called *Predictive Learn and Apply* strategy [26, 46**?**]. In this strategy, the RTC latency introduced into the overall AO pipeline is completely determined by the time-to-solution of the predictive wavefront reconstructor MVM. The dimensions of this matrix are determined by the AO system parameters, and the contents of this matrix are determined by the identified system parameters, including the AO optical design, but also the time-varying atmospheric parameters (turbulence strength and wind-velocities). The output of MVM corresponds to the command vector that triggers the movement of the actuators behind each deformable mirror in order to compensate in real-time for the atmospheric turbulence.

Assuming a typical WFS sampling time of 1ms (to keep a maximum up-time even during the low Sodium season) and in order to obtain an adequate AO rejection bandwidth, the overall AO loop delay should be of the order of 2 frames (i.e. 2ms) or lower. Following the typical definition of the RTC latency and assuming a readout time of 500μs for the WFS camera and an inevitable delay of 1 frame due to half of the WFS sampling time and half of the zero-order hold on the DM, leave us with less than half a frame (500μs) of RTC latency to remain below the 2 frames delay. In order to remain on the safe side, our goal is to reach less than 200μs of RTC latency, inline with our performance requirements.

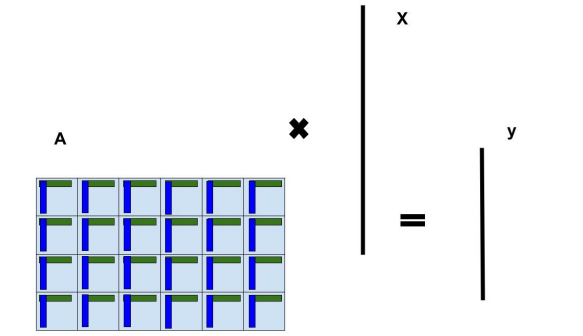
Since the state-of-the-art HRTC computational phase is currently driven by a dense MVM (i.e., Level-2 BLAS), this operation is supported today by all major vendor optimized numerical libraries. And perhaps the lack of existing work in improving further this area comes from the strong constraints imposed by observatories on using standard tools and libraries for sustainability purposes and/or for reducing cost associated with software maintenance.

4 TILE LOW-RANK MATRIX APPROXIMATIONS

The main idea behind tile low-rank (TLR) approximations is to perform a data compression on a tiled matrix using linear algebra matrix algorithms (e.g., rank-revealing QR [27], regular/randomized SVD [32], etc.) and to express the new MVM algorithms based on standard BLAS calls, as explained in Section 3. Fig. 2(a) shows a rectangular matrix split into 4×6 tiles. The rectangular shape (i.e, short and wide) of the matrix is typical for HRTC AO workloads. The MVM operation $A \times x = y$ needs to be performed successively thousands of times at a regular and constant pace to cope with the evolving wind and atmospheric conditions. We leverage the data sparsity of A by applying an SVD (or any other cheaper options) to compress each tile and create two bases, i.e., U an V, with size nb x k, nb being the tile size and k the rank. We have to compress up to a certain accuracy threshold that does not deteriorates the quality of the final image output. We filter out the singular values as follows. Assume a given accuracy ϵ, we compute for each tile $||A_{i,j} - U^{\epsilon}_{i,j}\Sigma^{\epsilon}_{i,j}V^{T\epsilon}_{i,j}||_F \leq \epsilon||A||_F$ with U the row bases, V^T the transposed column bases, and $||.||_F$ the Frobenius norm. We remove then the singular values lower than the accuracy threshold ϵ and the number of the remaining ones correspond the rank k of that specific tile. Theoretical details may be found in at [42]. Fig. 2(b) shows the U and V bases in blue and green, respectively. These two dense objects cannot be stored in existing sparse formats available in the literature since they are now decoupled from the global matrix index representation. For simplicity, we assume k is constant but during on-sky production simulations, k may vary from tile to tile. It is noteworthy to mention that this compression step happens only occasionally when the command matrix gets updated by the SRTC phase. It is therefore not part of the critical path.



(a) Original dense MVM.



(b) Compressed MVM.

Figure 2: Dense vs TLR compressed data structure for MVM.

Once each tile is compressed, we stack the bases together to ensure a contiguous memory access during the computation, as pictured in Fig. 3. Once the TLR compressed data structure is formed,

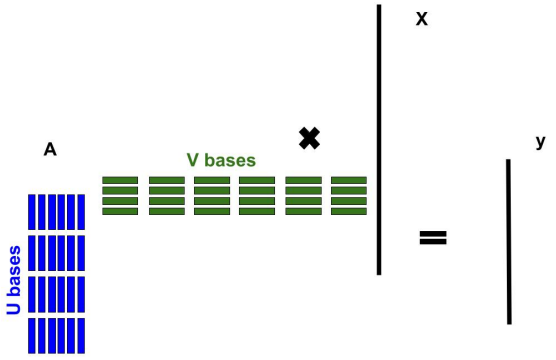
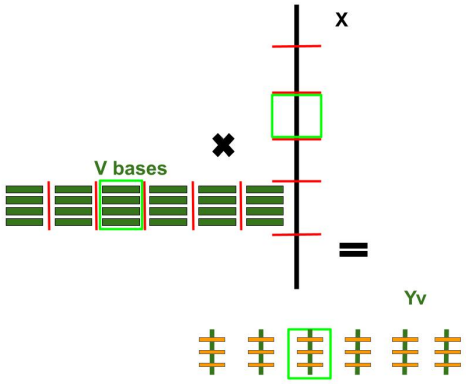
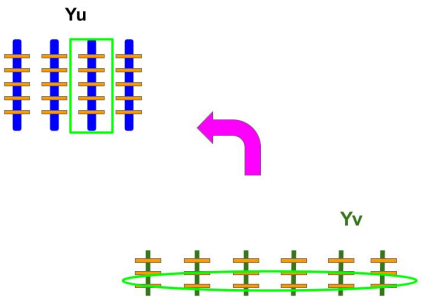


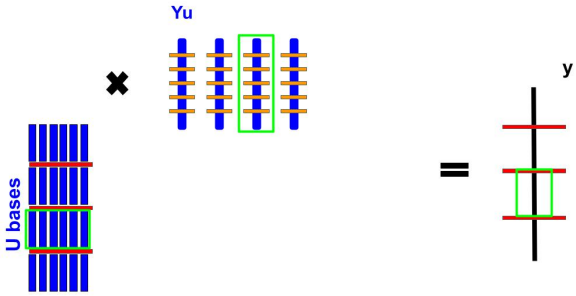
Figure 3: Stacked bases for TLR-compressed format.



(a) Batch of MVM w/V bases.



(b) Intermediate vector projected from V to U bases.



(c) Batch of MVM w/U bases and output command vector y.

Figure 4: Computational phases of TLR-MVM.

TLR-MVM is expressed in three phases: (1) a batch of regular MVM involving only the V bases and the input vector x resulting in the set of output vectors Yv (Fig. 4(a)), (2) a reshuffle phase that projects the ranks (identified by an orange delimiter) within each set of vectors Yv from phase 1 into the U bases resulting into the set of output vectors Yu (Fig. 4(b)), and (3) a batch of regular MVM involving only the U bases and the set of vectors Yu (Fig. 4(c)) that computes at the end the approximated vector command y.

A similar concept of stacking the bases has been applied for sparse direct solver to further increase the arithmetic complexity of the factorization [35]. In this paper, since MVM is inherently memory-bound, we stack the bases in order to increase data locality in the high level caches of the memory subsystem. This allows our implementation to decouple potentially from the main memory and occasionally, depending on the underlying hardware architecture, to benefit from cache bandwidth.

5 IMPLEMENTATION DETAILS

We describe in this section the implementation of our TLR-MVM algorithm, highlighted previously in Section 4.

5.1 Relying on Standard Programming Models

Our TLR-MVM implementation is written in C and uses the MPI + OpenMP programming model. Algorithm 1 shows the OpenMP-only pseudo-code of TLR-MVM with the three computational phases. The three phases are parallelized using standard OpenMP *for* loop pragma. We link our code against the corresponding sequential BLAS library provided by the vendor optimized numerical software. Nested parallelism in this context has not shown performance superiority.

Algorithm 1 OpenMP pseudo-code of TLR-MVM.

```
1: Compress A
2: for Each stacked tile column V_j do                    ▷ Phase 1: Batch MVM
3:     GEMV with the corresponding nb portion of vector x to get a set of output Yv
4: end for
                                                          ▷ Phase 2: Reshuffle
5: Project the set of output vectors Yv from V_j bases to U_j bases to get the set of output vector Yu
6: for Each concatenated tile row U_i do                  ▷ Phase 3: Batch MVM
7:     GEMV with the corresponding vector column of Yu
8: end for
9: Send the command vector to actuators
```

Algorithm 2 highlights the MPI+OpenMP version of the TLR-MVM implementation. We use a 1D cyclic block data distribution similar to ScaLAPACK [10] to mitigate the load imbalance that may appear with variable ranks. We split the U and V bases vertically among the MPI processes. While the splitting for the U bases eventually engenders workloads that may be operated in an embarrassingly parallel fashion, the vertical splitting for the V bases requires an MPI reduce operation to sum the partial results to the

root process. However, the three computational phases powered by OpenMP (i.e., Algorithm 1) can run independently on each MPI process. The two pseudo-codes assume constant ranks for read-

Algorithm 2 MPI+OpenMP pseudo-code of TLR-MVM.

```
1: Compress A
2: Get mpirank, mpisize                                   ▷ rank and worldsize
3: nt = 0                                                 ▷ number of tiles on each process
4: for each tile column index j do                        ▷ get number of tiles
5:     opnode = j mod mpisize
6:     if I am opnode then
7:         nt = nt + 1
8:     end if
9: end for
10: for each tile row index i do
11:     for each tile column index j do
12:         opnode = j mod mpisize
13:         if root then
14:             send tile U_{i,j}, V_{i,j} and x_j to opnode
15:         else
16:             opnode receives U_{i,j}, V_{i,j} and x_j from root process
17:         end if
18:     end for
19: end for                                               ▷ 1D cyclic data splitting
20: Call OpenMP TLR-MVM from Algorithm 1
21: Reduce the partial results from each MPI process to root
22: Send the command vector to actuators
```

ability purposes. The generalized code for variable ranks contains additional pointer arithmetics to ensure a proper off-setting when traversing the U and V bases and the intermediate sets of vectors Yv and Yu.

5.2 Arithmetic Complexity

The floating-point operations (FLOPS) and the memory bandwidth of the dense GEMV are $2mn$ and $B(mn + n + m)/t$, where B is the number of bytes per element and t is the execution time. In TLR-MVM, the calculation of FLOPS is as follows. In the computation of phase 1, the column size of each GEMV is nb and the row size of each GEMV is the sum of the ranks along this tile column. So, the FLOPS for computation of V is $2Rn_b$, where R is the sum of the ranks across all tiles of the matrix. In the reshuffle phase 2, there are no computations but only data movement. In phase 3, the FLOPS rate is calculated now with the row size of each GEMV fixed to nb and the column size is the sum of the ranks along the tile row. Therefore, the total FLOPS for phase 3 is $2Rn_b$. All in all, the overall FLOPS of TLR-MVM is thus $\text{FLOPS}_{\text{TLR-MVM}} = 4Rn_b$.

In order to compute the memory bandwidth for TLR-MVM, one needs to count the bytes read in and written back to the main memory. During the phase 1, the total bytes read in and out are $B(Rn_b + n + R)$. In the phase 2, $2BR$ bytes are read in and out through memory. In the phase 3, the total bytes read in and out are $B(Rn_b + R + m)$. So the memory bandwidth is $\text{BW}_{\text{TLR-MVM}} = \frac{B(2Rn_b+4R+n+m)}{t}$.

6 NUMERICAL ACCURACY

In order to verify the numerical accuracy of the TLR-MVM method in the context of the AO application, the compressed control matrix (reconstructor) is used in the end-to-end AO simulator, COMPASS [24]. By doing this, the resulting control matrix is tested under realistic conditions, and it is clear if the numerical accuracy lost by compressing the matrix is impactful on the AO system performance.

In AO, the main performance metric is the so-called *Strehl Ratio* (SR) which relates the imaging performance of a given optical system, with realistic optical aberrations, to the ideal performance of that same system without aberrations. For a given atmospheric condition and a given AO controller, a maximum SR is determined through end-to-end simulations. When using a TLR-MVM compressed control matrix, there is naturally some loss in numerical accuracy, which corresponds to a decrease in SR compared to using the original control matrix in the same simulations.

The two parameters which determine the numerical accuracy of the TLR-MVM compressed matrix are the tile size nb and the accuracy threshold ϵ (see Section 4). In order to determine an appropriate choice for these parameters, a set of AO systems and atmospheric conditions are investigated for a range of tile size nb and accuracy threshold ϵ. Figure 5 shows the resulting SR for one

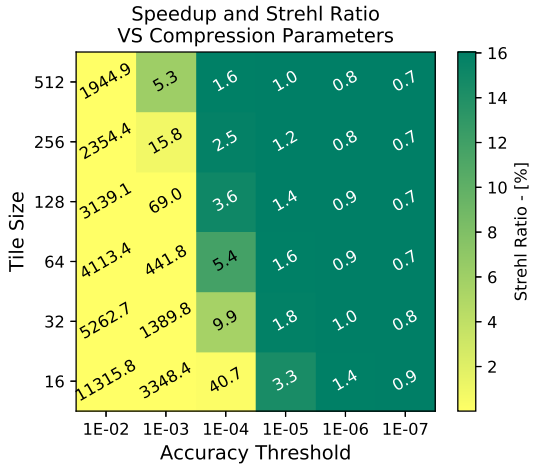


Figure 5: Strehl Ratio (at $\lambda = 550$ nm) and speedup for the MAVIS system under varying compression parameters. Colour indicates the SR and text entries indicate the expected speedup factor based on the actual FLOPS (see Section 5.2).

such system based on the MAVIS design (see Section 7.3 for the system description). Note here that the speedup reported (the text value in the cells) is based solely on the reduced number of FLOPS owing to the TLR-MVM compression compared to the original dense MVM, as detailed in Section 5.2). That is, this analysis does not consider any hardware utilisation differences that occur when using different tile sizes or a dense MVM. One can observe that if a very high accuracy is required operating in a reduced basis with high rank can cause speeddown, as reported with speedup factors less than one.

Additionally, what is considered a *negligible* loss in SR is somewhat subjective, so for a given AO instrument a trade-off study would be taken. Indeed, AO systems that can afford a larger drop in SR and are rewarded with a larger TLR-MVM speedup as a consequence. For the purpose of this investigation, any SR larger than 15% can be considered as *lossless*, with any SR less than 10% being unacceptably *lossy*.

From these results, there is clearly a range of parameters that provides a significant speedup with negligible loss in SR. For example, a tile size of $nb = 128$ and an accuracy of $\epsilon = 10^{-4}$ provide a speedup of 3.6 (number of FLOPS is 3.6 times less) with an absolute drop in SR of only 0.93%. To further explore this parameter space, a fine-grain search of the effect of accuracy on SR is performed for a variety of given atmospheric conditions, with a fixed $nb = 128$ and a varying accuracy between $10^{-6} \leq \epsilon \leq 10^{-3}$. This analysis is performed on four different atmospheric conditions (described in Section 7.3) and the results given in Figure 6.

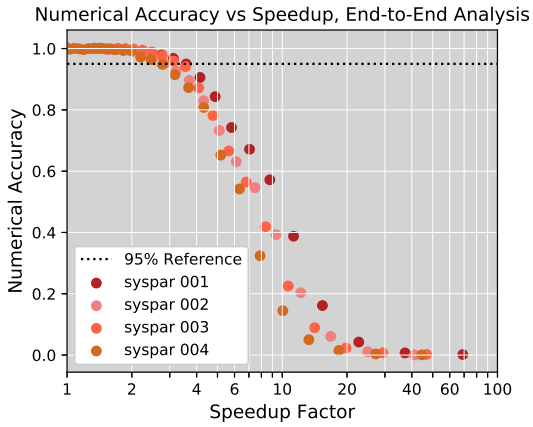


Figure 6: Numerical accuracy loss for increasing speedup. Results obtained from end-to-end simulations using the compressed control matrix for four different sets of atmospheric conditions.

Since the atmospheric conditions are different in each simulation, the obtained SR (even from the original control matrices before compression) may evolve over time. Instead, the numerical accuracy is assessed by comparing the SR obtained for a compressed matrix to the SR obtained for the original control matrix (so that if there is no compression, the resulting numerical accuracy is 1.0).

From Figure 6, it is clear that there is an unavoidable, albeit predictable trade-off between numerical accuracy and speedup factor. For the variety of atmospheric conditions tested, a speedup factor of around 3.0 comes with very little loss in SR. As the compression becomes more aggressive, the SR drops further, with most systems becoming unusable at speedup factors greater than 10.0.

7 EXPERIMENTAL RESULTS

This section reports on the accuracy impact and performance of our TLR-MVM implementation.

7.1 Environment Settings

The experiments are carried on several x86 and accelerator-based systems from the major vendors, i.e., Intel Cascade Lake (codenamed CSL), AMD Epyc Rome (codenamed Rome), AMD Instinct GPU MI100 (codename MI100), Fujitsu A64FX (codenamed A64FX), NVIDIA A100 GPU (codenamed A100), and NEC SX-Aurora Tsubasa (codenamed Aurora). A detailed hardware and software description is illustrated in Table 1. While most of our experiments are done on shared-memory systems, we also report performance scalability on a small number of Fujitsu A64FX nodes linked by the TOFU interconnect and multiple NEC Vector Engines connected via Infiniband. All computations are performed in single precision arithmetic and we report performance jitter out of 5000 runs.

7.2 Synthetic Datasets

We first assess our TLR-MVM implementation on randomly generated U and V with constant rank k. Although this may not be realistic to have exact same ranks everywhere for computational astronomy (but can be useful if minimum padding is an option), it still provides some insights on how each hardware architecture responds to a memory-bound workload with batch executions supported via the OpenMP *for* loop pragma for all systems except the NVIDIA A100 GPUs. For the latter, we replace the OpenMP loop with a single call of batch cuBLAS GEMM kernel, since the batch cuBLAS GEMV kernel is not supported. Of course, we set the number of columns of the resulting matrix to 1 to fall back to a batch cuBLAS GEMV kernel. Figure 7 highlights the impact of the tile sizes on the sustained bandwidth, as calculated in Section 5.2. We can see that nb has an impact for some hardware and less for others, depending on the underlying hardware architectures. For instance, A64FX is oblivious to nb, while Rome benefits significantly as nb decreases due to its large LLC capacity. All in all, $nb = 100$ seems to deliver decent performance on all systems. For the following graphs, we will report performance with $nb = 100$.

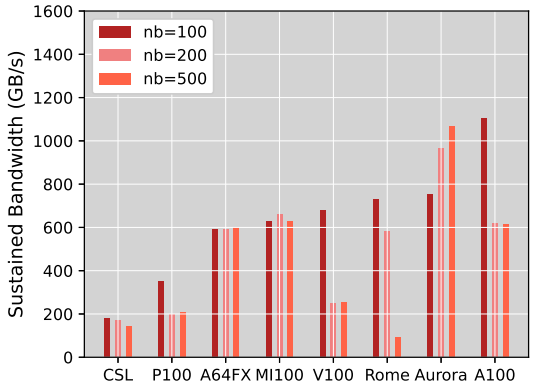


Figure 7: Performance impact of tile sizes.

Figure 8 shows performance comparisons of our TLR-MVM implementation on all hardware systems. In particular, we can see the performance of TLR-MVM across three generations of NVIDIA GPUs, i.e., P100/V100/A100. Moreover, we can see that hardware supporting high bandwith memory technology (HBM) delivers the most performance for such memory-bound kernel as opposed to DDR4 technology. While this statement applies for Intel CSL, we see a different behavior for AMD Rome. This can be explained by the physically partitioned last-level cache (L3) across the core complex (CCX) groups composed of four cores. A single AMD Rome core only sees its own CCX's L3 cache before going into main memory.

Table 1: Hardware/software specifications.

Vendor	Intel	AMD		Fujitsu	NVIDIA	NEC
Family	Cascade Lake	EPYC Rome	Instinct	Primergy A64FX	Ampere GPU	SX-Aurora TSUBASA
Model	6248	7702	MI100	FX1000	A100	B300-8
Node(s)/Card(s)	1	1	1	16	1	8
Socket(s)	2	2	N/A	4	N/A	N/A
Cores	40	128	7680	48	6912	8
GHz	2.5	2.2	1.5	2.2	2.6	1.6
Memory Sustained BW	384GB DDR4 232GB/s	512GB DDR4 330GB/s	32GB HBM2 1.2TB/s	32GB HBM2 800GB/s	40GB HBM2e 1.5TB/s	48GB HBM2 1.5TB/s
LLC Sustained BW	27.5MB 1.1TB/s	**512MB** 4TB/s	8MB 3TB/s	32MB 3.6TB/s	40MB 4.8TB/s	16MB 2.1TB/s
Compiler	Intel compiler 19.1.0	GCC compiler 8.2.0		Fujitsu compiler 4.5.0	NVCC 11.0	NEC compiler 3.1.1
BLAS library	Intel MKL 2020	BLIS 3.0.0		Fujitsu SSL II	cuBLAS 11.0	NEC NLC 2.1.0
MPI library	OpenMPI 4.0.3	OpenMPI 3.1.2		Fujitsu MPI 4.0.1	NCCL 2.0	NEC MPI 2.13.0

In case of a cache miss, the core only needs to scan its local L3 cache, which provides in return a much higher bandwidth.

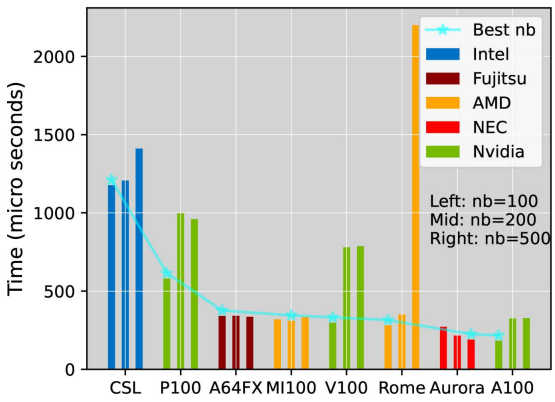


Figure 8: Best time to solution on different architectures.

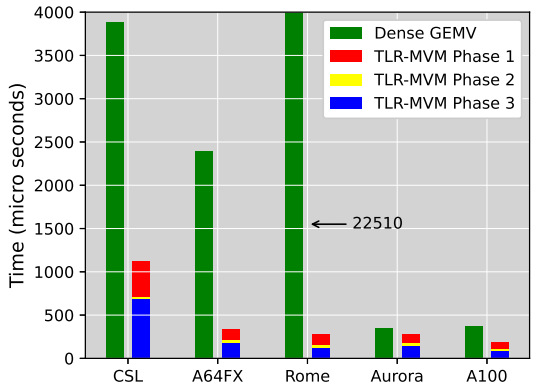


Figure 9: Dense GEMV Vs TLR-MVM.

Figure 9 compares the dense MVM with TLR-MVM (based on constant ranks with randomly generated bases). TLR-MVM achieves up to two orders of performance improvements against its counterpart dense MVM.

7.3 MAVIS Datasets

In this section, we now run TLR-MVM against the MAVIS AO system. In the presented simulations, it has 19078 measurements and 4092 actuators, resulting in a matrix reconstructor of dimensions $M = 4092$, $N = 19078$ (i.e., the dimensions of the matrix which undergoes TLR compression). For the full description of the MAVIS system, we refer to [43].

For the end-to-end AO simulations presented in Section 6, the MAVIS system is assessed due to its moderate dimensionality, tight error budget, capacity for predictive control, and latency demand. The performance of a predictive controller in AO depends on the wind-velocity profile and the turbulence strength profile (both as a function of height). To verify the stability of the TLR-MVM compression method for a variety of observing conditions, the simulations were performed with varying atmospheric parameters. These parameters are summarized in Table 2.

The performance of the AO loop depends on the magnitude of the speed of each layer, as well as the strength of those layers, and the distance between the strong layers and the post-focal DMs in the AO system. As such, the systems described represent vastly different AO performance, allowing to test the TLR-MVM for a wide range of conditions.

7.4 Rank Statistics

Figure 10 shows the rank distribution for MAVIS reference profile measurements using $nb = 128$ and $\epsilon = 1e{-}4$. The red vertical dotted line shows the rank limit $k = nb/2 = 64$ below which TLR-MVM becomes competitive (i.e, left side of the red vertical dotted line). One can clearly see the data sparsity of the command matrix. Since NVIDIA does not support variable batch sizes, we are not able to run experiments on NVIDIA GPUs using MAVIS AO system in the

Table 2: Atmospheric parameters used for MAVIS end-to-end simulations. Table entries show fractional turbulence strength (top), wind speed in m/s, and bearing in degrees (bottom) for a given atmospheric layer.

	Layer [km]									
	0.03	0.14	0.28	0.56	1.13	2.25	4.50	7.75	11.00	14.00
syspar 001	0.59 31.7∠352°	0.02 21.2∠288°	0.04 22.7∠166°	0.06 37.0∠281°	0.01 2.8∠43°	0.05 3.5∠230°	0.09 0.8∠52°	0.04 33.3∠340°	0.05 31.1∠188°	0.05 34.8∠149°
syspar 002	0.24 4.5∠48°	0.12 5.7∠13°	0.05 17.8∠30°	0.06 29.3∠77°	0.10 18.4∠196°	0.06 23.7∠236°	0.14 13.5∠212°	0.07 18.2∠207°	0.09 7.5∠120°	0.06 16.4∠137°
syspar 003	0.25 39.9∠241°	0.11 3.2∠105°	0.05 11.4∠116°	0.12 21.4∠150°	0.14 33.8∠175°	0.12 8.0∠339°	0.06 32.5∠264°	0.06 14.9∠351°	0.06 32.4∠208°	0.03 0.5∠185°
syspar 004	0.16 0.1∠136°	0.09 39.2∠283°	0.13 13.7∠31°	0.02 3.8∠197°	0.10 15.8∠58°	0.12 0.2∠104°	0.02 29.5∠16°	0.12 38.2∠120°	0.13 32.8∠265°	0.11 13.8∠302°

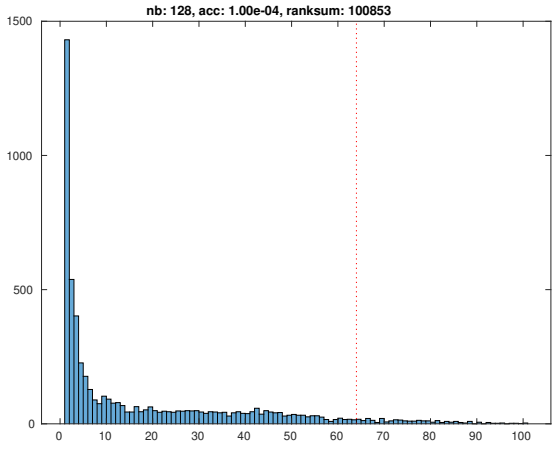


Figure 10: Rank distributions for MAVIS reference profile measurements using $nb = 128$ and $\epsilon = 1e-4$.

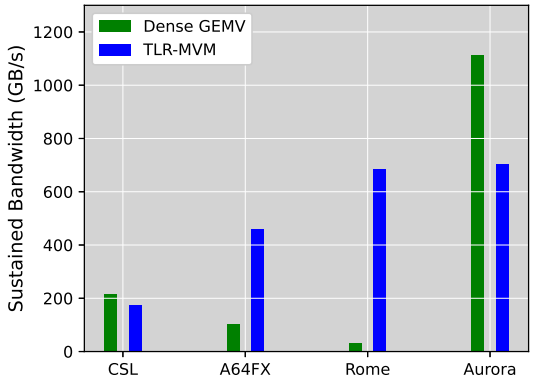


Figure 11: Performance bandwidth for MAVIS system.

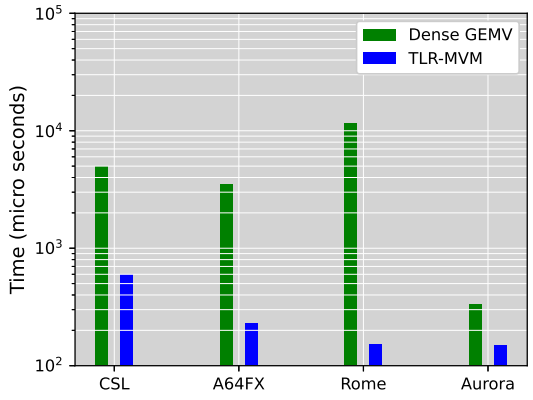


Figure 12: Time to solution for MAVIS system.

subsequent graphs, due to variable ranks. We run using MAGMA batch kernels but performance obtained is very low.

7.5 Performance Results

Figure 11 shows the sustained bandwidth achieved with the dimension and dataset from MAVIS AO system. We can see that NEC Aurora and AMD Rome achieves almost similar bandwidth with different memory technologies. The tiny GEMV kernels in phase 1 and phase 3 of TLR-MVM are able to fit in LLC and greatly benefit from higher cache memory bandwidth.

Figure 12 pictures time to solution for TLR-MVM using MAVIS system. AMD Rome and NEC Aurora are below 200 microseconds for a single TLR-MVM call, which open new opportunities moving forward. On real datasets, our TLR-MVM achieves up to 8.2X/15.5X/2.2X performance speedups compared to vendor optimized multithreaded dense SGEMV kernel on Intel CSL / A64FX / NEC SX-Aurora, respectively. On AMD Epyc/Rome, we obtain up to 76.2X performance speedup against the vendor supported BLIS library. These actual speedup factors are much higher than the theoretical speedup factors shown in Figure 5. Our TLR-MVM implementation is capable of better utilizing the underlying hardware resources, thanks to data locality considerations and optimized memory accesses.

Figure 13 highlights the performance jitter of TLR-MVM, which is critical to maintain it low for HRTC. We can see that NEC Aurora reproduces the same time to solution for most of the iteration runs. However, Intel CSL and Fujitsu A64FX suffer the most.

Figure 14 reports bandwidth jitter for MAVIS, which represents the same trend in Figure 13, with Intel CSL and Fujitsu A64FX showing a large pyramid base, as opposed to NEC Aurora.

Figure 15 reports time to solution for all various MAVIS profiles, as described in Table 2. Fujitsu A64FX and NEC Aurora are oblivious

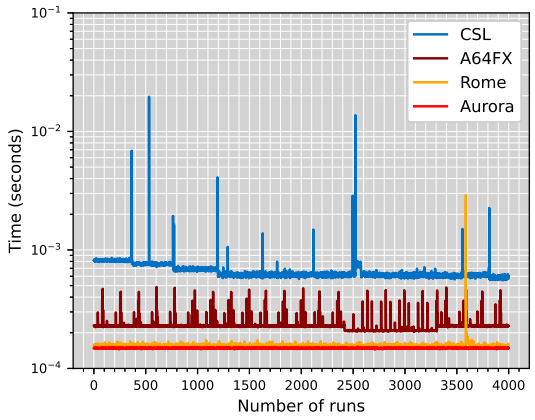


Figure 13: Performance jitter for MAVIS.

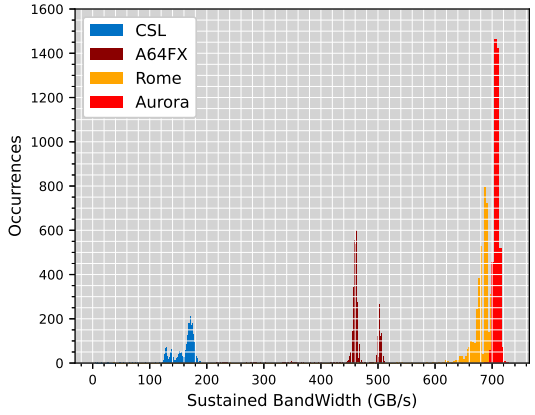


Figure 14: Bandwith jitter for MAVIS.

to the profile characteristic and are able to deliver same time to solution, while the x86 systems show some variable timings.

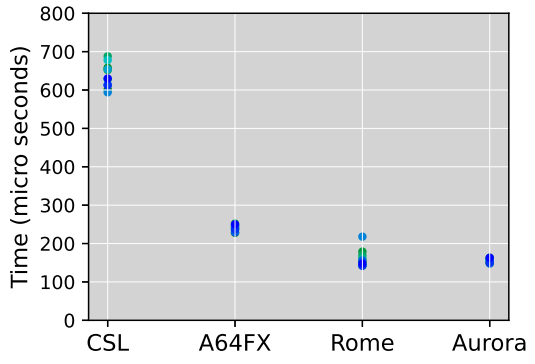


Figure 15: MAVIS configuration time to solution from 000 (green) to 070 (blue).

Figures 16 and 17 show performance scalability on multiple A64FX nodes and NEC Aurora cards, respectively. As we increase the number of processing units, the workload per node/cards decreases and may not saturate the bandwidth anymore, as seen when running on single processing units. We consider additionally larger matrix sizes that are representative of other instruments under consideration for the European Extremely Large Telescope. We synthetically generate their rank distributions and show our software capabilities when increasing the number of nodes/cards. For the EPICS instrument, we can saturate the bandwidth and achieves a decent performance scalability on both systems.

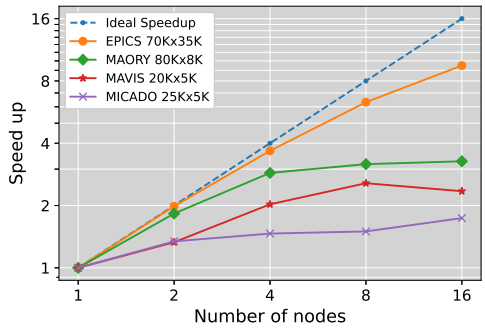


Figure 16: Performance scalability on A64FX.

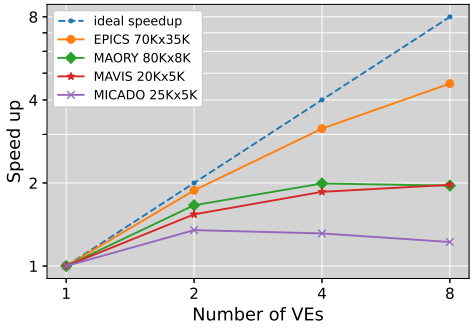


Figure 17: Performance scalability on Aurora.

Figures 18 and 19 show the roofline performance models of our TLR-MVM implementation on AMD Epyc Rome and Fujitsu A64FX systems on Mavis dataset. We can see that the sustained bandwith on the AMD Epyc Rome system is decoupled from main memory and is bound by LLC bandwidth. This is an interesting outcome, after tuning the tile size and mapping threads to physical cores with a proper binding. On the Fujitsu A64FX system, our TLR-MVM implementation is limited by HBM2 bandwidth since the LLC capacity is too small to avoid data movement with main memory.

Systems equipped with large LLC capacity, physically partitioned for data locality with high bandwidth, and stacked on top of HBM technology appear to be an interesting hardware landscape for supporting real-time processing. Such systems become heavily NUMA

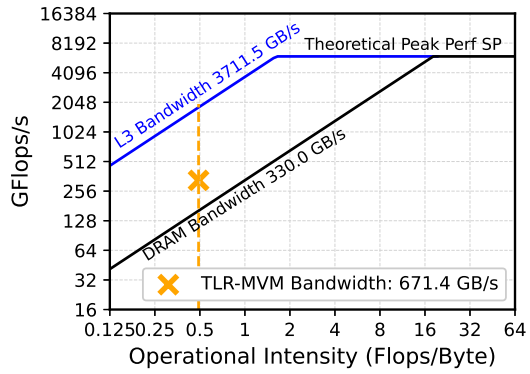


Figure 18: AMD Roofline Performance Model on Mavis dataset.

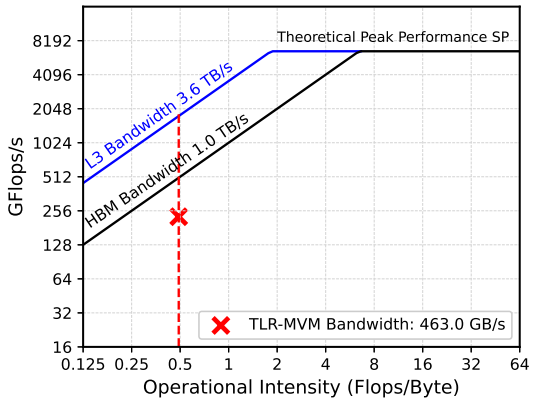


Figure 19: A64FX Roofline Performance Model on Mavis dataset.

nodes. Developers may end up programming these shared-memory systems as if they are actual distributed-memory nodes.

8 DISCUSSION

As seen from the RTC subsystem, AO performance and our ability to reach the science goals with our instrument are driven by two parameters: predictability and reproducibility. The former ensures us to stay within the target latency budget for the AO loop (hence the AO system rejection bandwidth) and the latter is mandatory to ensure stable operations.

This new TLR-MVM approach provides a clear performance boost, in terms of time-to-solution, as compared to the dense MVM, as demonstrated in Figure 12. This can eventually translate into lower delay in the AO loop with potential benefits on AO performance. Indeed, the so-called servo-lag error in AO is the result of a delay between the time measurements are made on the WFS and the time commands are applied on the DM. This error is the combination of several effects: integration time on the WFS camera, frame transfer and read-out on the detector, pixel data transfer to the RTC, RTC pure delay, DM rise time and DM zero-order hold. Lowering time-to-solution for part of the HRTC leads to a reduced RTC pure delay and thus reduced servo-lag error. This can be exploited to enhance AO performance (and in that case Figure 4 provides an upper bound of the possible Strehl loss versus accuracy). Alternatively, one could choose to keep the same time envelope for the whole RTC pipeline and use the margin provided by this TLR-MVM to add additional tasks in this pipeline such as more efficient denoising of the WFS frames or additional filtering at the output of the MVM computation. In particular, more complex control schemes, also based on MVM but deemed unfeasible today due to the major increase in compute demand they represent could become realistic. This is discussed in the next section. Dedicated trade-offs, depending on the main science cases and performance goals for each instrument shall lead to choose between one or the other strategy.

Jitter on measured time-to-solution varies a lot across the various vendors. While the NEC Aurora performance seems to be extremely stable out of the box and compatible with typical instruments specifications as is (i.e. in the range of 10-20μs), outliers (AMD, NVIDIA) and even regular peak patterns (CSL) are observed for other vendors. We note here that this may not prevent these solutions to be used for AO RTC systems, eventually, since these tests have been performed without any particular tuning of the host system, nor particular features in the software to make it real-time. Proven solutions exist to embed these BLAS calls into a hard real-time software framework [25], enabling complex heterogeneous pipelines involving MVM and additional kernels. This is fully applicable to our TLR-MVM approach and should provide the required low jitter for any of the systems tested in this paper and make it usable in practice.

Finally, applications that are extremely sensitive to latency, such as the AO HRTC, will benefit the most from a densely populated system involving PCIe only (dense cluster of co-processors) rather than a setup where the nodes are distributed over a network fabric. Network interconnect may introduce a significant amount of latency (at best of the order of 10 µs per transaction in case of Ethernet). For that reason, our baseline design for MAVIS HRTC relies on a fat node rather a cluster of distributed-memory nodes.

9 CONCLUSION AND FUTURE WORK

In this paper, we introduce a new TLR-MVM implementation that leverages the data sparsity of the command matrix in a real-time controller, which operates as a replacement to the traditional state-of-the- art dense MVM for computational astronomy in ground-based telescope. We demonstrate software portability across several vendor systems. We achieve performance improvement up to two orders of magnitude compared to the regular dense MVM, while maintaining the required image quality.

This significant improvement in time-to-solution can potentially unlock major breakthroughs in ground based astronomy, since the next generation of giant telescopes capabilities are almost entirely driven by AO performance. While the AO results shown in this paper utilize a Predictive Learn and Apply control scheme, more advanced approaches, such as Linear Quadratic Gaussian (LQG) [21, 46], can potentially bring a significant performance boost in terms of Strehl Ratio at the cost of significantly larger control matrices.

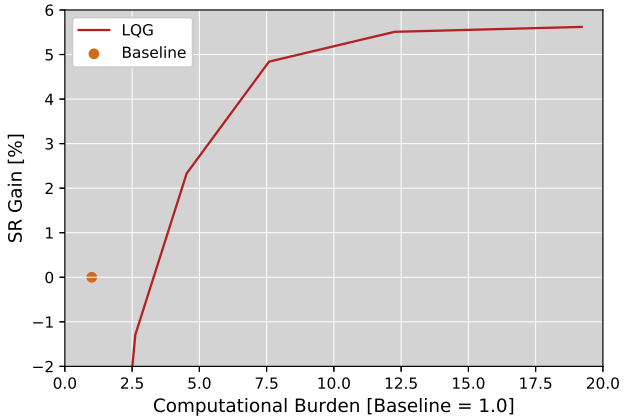


Figure 20: Performance gained by LQG in MAVIS for an increased computational load.

LGQ is deemed infeasible today to meet the real time constraint; see Figure 20 for example, which shows the performance gained by using LQG in the MAVIS context. This gain has a large impact on the MAVIS error budget, hence on science return for the instrument, and the switch to LQG comes only at the cost of HRTC burden, which can be addressed using the TLR-MVM approach. This has to be compared to the cost of an additional high power Laser or deformable mirror which each represent millions of Euros, to obtain a similar performance boost.

With the ability to utilize data sparsity within the RTC control matrices, it is expected that schemes such as LQG will become feasible in systems where they were previously prohibitively expensive.

We demonstrate the impact of TLR-MVM on the MAVIS AO instrument. We believe that this work will impact many other AO instruments currently or to be deployed on-sky. The new TLR-MVM algorithm is designed so that its software stack and programming models remain standard for software sustainability purposes.

ACKNOWLEDGMENTS

The authors would like thank Fujitsu limited for their support on the evaluation environment and AMD/NVIDIA for the remote accesses to their respective GPU-based systems.

REFERENCES

[1] E. Agullo, J. Demmel, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, H. Ltaief, P. Luszczek, and S. Tomov. 2009. Numerical Linear Algebra on Emerging Architectures: the PLASMA and MAGMA Projects. In *Journal of Physics: Conference Series*, Vol. 180. IOP Pub., 012037.

[2] K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, A. Esposito, and D. Keyes. 2018. Exploiting Data Sparsity for Large-Scale Matrix Computations. In *European Conference on Parallel Processing*. Springer, 721–734.

[3] K. Akbudak, H. Ltaief, A. Mikhalev, and D. Keyes. 2017. Tile Low-Rank Cholesky Factorization for Climate/Weather Modeling Applications on Manycore Architectures. In *32nd International Conference on High Performance, Frankfurt, Germany*. Springer, 22–40.

[4] N. Al-Harthi, R. Alomairy, K. Akbudak, R. Chen, H. Ltaief, H. Bagci, and D. Keyes. 2020. Solving Acoustic Boundary Integral Equations Using High Performance Tile Low-Rank LU Factorization. In *35th International Conference on High Performance, Frankfurt, Germany*. Springer.

[5] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil. 2015. Fast Direct Methods for Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2015), 252–265.

[6] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. 2015. Improving Multifrontal Methods by Means of Block Low-Rank Representations. *SIAM Journal on Scientific Computing* 37, 3 (2015), A1451–A1474.

[7] A. Aminfar, S. Ambikasaran, and E. Darve. 2016. A Fast Block Low-rank Dense Solver With Applications to Finite-element Matrices. *J. Comput. Phys.* 304 (2016), 170–188.

[8] A. Baruffolo, I. Foppiani, G. Agapito, C. Plantet, L. Busoni, G. Cosentino, I. Baronchelli, C. Arcidiacono, P. Feautrier, P. Ciliegi, S. Esposito, R. Ragazzoni, R. Biasi, M. Manetti, J.-P. Véran, D. Kerley, M. Smith, and J. Dunn. 2020. MAORY RTC, a Status Update. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11448)*. Article 1144839, 1144839 pages. https://doi.org/10.1117/12.2561585

[9] J. M. Beckers. 1989. Detailed Compensation of Atmospheric Seeing Using Multiconjugate Adaptive Optics. In *Active telescope systems (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 1114)*, F. J. Roddier (Ed.). 215–217. https://doi.org/10.1117/12.960826

[10] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, et al. 1997. *ScaLAPACK Users' Guide*. Vol. 4. SIAM.

[11] S. Börm. 2010. *Efficient Numerical Methods for Non-local Operators: H2-matrix Compression, Algorithms and Analysis*. Vol. 14. European Mathematical Society.

[12] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Herault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaief, P. Luszczek, A. YarKhan, and J. Dongarra. 2011. Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. 1432–1441.

[13] Antonin H. Bouchez, George Z. Angeli, David S. Ashby, Robert Bernier, Rodolphe Conan, Brian A. McLeod, Fernando Quirós-Pacheco, and Marcos A. van Dam. 2018. An Overview and Status of GMT Active and Adaptive Optics. In *Adaptive Optics Systems VI (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 10703)*, Laird M. Close, Laura Schreiber, and Dirk Schmidt (Eds.). Article 107030W, 107030W pages. https://doi.org/10.1117/12.2314255

[14] W. H. Boukaram, G. Turkiyyah, H. Ltaief, and D. E. Keyes. 2018. Batched QR and SVD Algorithms on GPUs with Applications in Hierarchical Matrix Compression. *Parallel Comput.* 74, C (May 2018), 19–33. https://doi.org/10.1016/j.parco.2017.09.001

[15] S. Börm, L. Grasedyck, and W. Hackbusch. 2003. Introduction to Hierarchical Matrices With Applications. *Engineering Analysis with Boundary Elements* 27, 5 (2003), 405 – 422. https://doi.org/10.1016/S0955-7997(02)00152-2 Large scale problems using BEM.

[16] Q. Cao, Y. Pei, K. Akbudak, A. Mikhalev, G. Bosilca, H. Ltaief, D. Keyes, and J. Dongarra. 2020. Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. ACM.

[17] A. Charara, D. Keyes, and H. Ltaief. 2018. Tile Low-Rank GEMM Using Batched Operations on GPUs. In *Euro-Par 2018: Parallel Processing*, Marco Aldinucci, Luca Padovani, and Massimo Torquati (Eds.). Springer, 811–825.

[18] A. Charara, H. Ltaief, D. Gratadour, D. Keyes, A. Sevin, A. Abdelfattah, E. Gendron, C. Morel, and F. Vidal. 2014. Pipelining Computational Stages of the Tomographic Reconstructor for Multi-Object Adaptive Optics on a Multi-GPU System. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 262–273. https://doi.org/10.1109/SC.2014.27

[19] E. Corona, P.-G. Martinsson, and D. Zorin. 2015. An $o(n)$ Direct Solver for Integral Equations on the Plane. *Applied and Computational Harmonic Analysis* 38, 2 (2015), 284–317.

[20] J. Crane, G. Herriot, D. Andersen, J. Atwood, P. Byrnes, A. Densmore, J. Dunn, J. Fitzsimmons, T. Hardy, B. Hoff, K. Jackson, D. Kerley, O. Lardière, M. Smith, J. Stocks, J.-P. Véran, C. Boyer, L. Wang, G. Trancho, and M. Trubey. 2018. NFIRAOS Adaptive Optics for the Thirty Meter Telescope. In *Adaptive Optics Systems VI (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 10703)*, Laird M. Close, Laura Schreiber, and Dirk Schmidt (Eds.). Article 107033V, 107033V pages. https://doi.org/10.1117/12.2314341

[21] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, X. Chi, A. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, A. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, A. Hoisie, K. Hotta, Z. Jin, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, A. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, A. Trefethen, M. Valero, A. van der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick. 2011. The International Exascale Software Project Roadmap. *The International Journal of High Performance Computing Applications* 25, 1 (2011), 3–60. https://doi.org/10.1177/1094342010391989

[22] N. Doucet, R. Kriemann, E. Gendron, D. Gratadour, H. Ltaief, and D. Keyes. 2018. Scalable Soft Real-Time Supervisor for Tomographic AO. In *Adaptive Optics Systems VI (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*

Series, Vol. 10703). Article 107034L, 107034L pages. https://doi.org/10.1117/12.2313273

[23] N. Doucet, H. Ltaief, D. Gratadour, and D. Keyes. 2019. Mixed-Precision Tomographic Reconstructor Computations on Hardware Accelerators. In *2019 IEEE/ACM 9th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. 31–38.

[24] F. Ferreira, D. Gratadour, A. Sevin, and N. Doucet. 2018. COMPASS: An Efficient GPU-based Simulation Software for Adaptive Optics Systems. In *2018 International Conference on High Performance Computing Simulation (HPCS)*. 180–187. https://doi.org/10.1109/HPCS.2018.00043

[25] F. Ferreira, A. Sevin, J. Bernard, O. Guyon, A. Bertrou-Cantou, J. Raffard, F. Vidal, E. Gendron, and D. Gratadour. 2020. Hard Real-Time Core Software of the AO RTC COSMIC Platform: Architecture and Performance. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11448)*. Article 1144815, 1144815 pages. https://doi.org/10.1117/12.2561244

[26] É. Gendron, A. Charara, A. Abdelfattah, D. Gratadour, D. Keyes, H. Ltaief, C. Morel, F. Vidal, A. Sevin, and G. Rousset. 2014. A Novel Fast and Accurate Pseudo-Analytical Simulation Approach for MOAO. In *Adaptive Optics Systems IV (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 9148)*. Article 91486L, 91486L pages. https://doi.org/10.1117/12.2055911

[27] G. H. Golub and C. F. Van. 2012. *Matrix Computations*. Vol. 3. Third Edition, Johns Hopkins University Press 2012.

[28] S. A. Goreinov, E. E. Tyrtyshnikov, and A. Y. Yeremin. 1997. Matrix-Free Iterative Solution Strategies for Large Dense Linear Systems. *Numerical Linear Algebra with Applications* 4, 4 (1997), 273–294.

[29] L. Grasedyck, D. Kressner, and C. Tobler. 2013. A Literature Survey of Low-Rank Tensor Approximation Techniques. *GAMM-Mitteilungen* 36, 1 (2013), 53–78. https://doi.org/10.1002/gamm.201310004 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/gamm.201310004

[30] D. Gratadour, A. Sevin, J. Brulé, É. Gendron, and G. Rousset. 2012. GPUs for Adaptive Optics: Simulations and Real-Time Control. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 8447)*. Article 84475R. https://doi.org/10.1117/12.925723

[31] W. Hackbusch. 1999. A Sparse Matrix Arithmetic Based on $\mathcal{H}$-Matrices. Part I: Introduction to h-Matrices. *Computing* 62, 2 (1999), 89–108.

[32] N. Halko, P.-G. Martinsson, and J. A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.* 53, 2 (2011), 217–288. https://doi.org/10.1137/090771806 arXiv:https://doi.org/10.1137/090771806

[33] F. Hammer, B. Barbuy, J. G. Cuby, L. Kaper, S. Morris, C. J. Evans, P. Jagourel, G. Dalton, P. Rees, M. Puech, M. Rodrigues, D. Pearson, and K. Disseau. 2014. MOSAIC at the E-ELT: a Multi-object Spectrograph for Astrophysics, IGM and Cosmology. In *Ground-based and Airborne Instrumentation for Astronomy V (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 9147)*. Article 914727, 914727 pages. https://doi.org/10.1117/12.2055148

[34] J. W. Hardy, J. E. Lefebvre, and C. L. Koliopoulos. 1977. Real-Time Atmospheric Compensation. *Journal of the Optical Society of America (1917-1983)* 67 (March 1977), 360–369.

[35] C.-P. Jeannerod, T. Mary, C. Pernet, and D. S. Roche. 2019. Improving the Complexity of Block Low-Rank Factorizations with Fast Matrix Arithmetic. *SIAM J. Matrix Anal. Appl.* 40, 4 (2019), 1478–1496. https://doi.org/10.1137/19M1255628 arXiv:https://doi.org/10.1137/19M1255628

[36] D. E. Keyes, H. Ltaief, and G. Turkiyyah. 2020. Hierarchical Algorithms on Hierarchical Architectures. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 378, 2166 (2020), 20190055. https://doi.org/10.1098/rsta.2019.0055 arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2019.0055

[37] R. Kriemann. 2013. H-LU Factorization on Many-Core Systems. *Comput. Vis. Sci.* 16, 3 (June 2013), 105–117. https://doi.org/10.1007/s00791-014-0226-7

[38] H. Ltaief, A. Charara, D. Gratadour, N. Doucet, B. Hadri, E. Gendron, S. Feki, and D. Keyes. 2018. Real-Time Massively Distributed Multi-Object Adaptive Optics Simulations for the European Extremely Large Telescope. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 75–84. https://doi.org/10.1109/IPDPS.2018.00018

[39] H. Ltaief, D. Sukkari, O. Guyon, and D. Keyes. 2018. Extreme Computing for Extreme Adaptive Optics: The Key to Finding Life Outside Our Solar System. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (Basel, Switzerland) *(PASC '18)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3218176.3218225

[40] P.-Y. Madec. 1999. Control Techniques. In *Adaptive Optics in Astronomy*, F. Roddier (Ed.).

[41] Gianpietro Marchiori, Francesco Rampini, Leonardo Ghedin, and Riccardo Bressan. 2018. ELT Design Status: the Most Powerful Ground Telescope. In *Ground-based and Airborne Telescopes VII (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 10700)*, Heather K. Marshall and Jason Spyromilio (Eds.). Article 1070021, 1070021 pages. https://doi.org/10.1117/12.2314947

[42] T. Mary. 2017. Block Low-rank Multifrontal Solvers: Complexity, Performance, and Scalability. *Phd Thesis, Université Paul Sabatier - Toulouse III, 2017. English. NNT : 2017TOU30305. tel-01929478* (2017).

[43] F. Rigaut, R. McDermid, G. Cresci, V. Viotto, S. Ellis, D. Brodrick, G. Agapito, T. Fusco, B. Neichel, P. Haguenauer, C. Plantet, B. Salasnich, M. Aliverti, S. Antoniucci, A. Balestra, A. Baruffolo, O. Beltramo-Martin, M. Bergomi, A. Blanco, M. Bonaglia, G. Bono, L. Busoni, E. Carolo, S. Chinellato, R. Content, J. Cranney, G. de Silva, S. Esposito, D. Fantinel, J. Farinato, D. Haynes, A. Horton, G. Gausachs, J. Gilbert, D. Gratadour, D. Greggio, M. Gullieuszik, V Korkiakoski, D. Magrin, L. Magrini, L. Marafatto, H. McGregor, T. Mendel, S. Monty, F. Pedichini, E. Pinna, E. Portaluri, K. Radhakrishnan, R. Ragazzoni, D. Robertson, C. Schwab, S. Ströbele, E. Thorn, A. Vaccarella, D. Vassallo, L. Waller, F. Zamkotsian, A. Zanutta, and H. Zhang. 2020. MAVIS Conceptual Design. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11447)*. Article 114471R, 114471R pages. https://doi.org/10.1117/12.2561886

[44] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov. 2016. A Distributed-memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization. *ACM Transactions on Mathematical Software (TOMS)* 42, 4 (2016), 27.

[45] F. Vidal, E. Gendron, and G. Rousset. 2010. Tomography Approach for Multi-Object Adaptive Optics. *Journal of the Optical Society of America A* 27, 27 (Oct. 2010), A253. https://doi.org/10.1364/JOSAA.27.00A253

[46] H. Zhang, J. Cranney, N. Doucet, Y. Hong, D. Gratadour, H. Ltaief, D. Keyes, and F. Rigaut. 2020. Predictive Learn and Apply: MAVIS Application - Learn. In *(submitted to) Adaptive Optics Systems VII*. International Society for Optics and Photonics, SPIE.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

Intel

(1) Model: Nmae: Cascade Lake
(2) 1 Intel(R) Xeon(R) Gold 6248 40 cores CPU
(3) 2 socktes
(4) Memory: 384 GB DDR4
(5) Memory Bandwidth: 232GB/s

AMD

(1) Model Name: EPYC Rome 7702 64-Core Processor
(2) 2 AMD EPYC 7702 64-Core Processor
(3) 2 sockets
(4) Memory: 512 GB DDR4
(5) Memory Bandwidth: 330 GB/s

Fujitsu

(1) Model Name: Primergy A64FX FX1000
(2) 2 sockets
(3) Memory: 32 GB DDR4
(4) Memory Bandwidth: 800 GB/s

NEC

(1) Model Name: SX-Aurora TSUBASA B300-8
(2) CUDA cores: 6912
(3) Memory: 40 GB HBM2
(4) Memory Bandwidth: 1.5 TB/s

NVIDIA P100

(1) Model Name: P100 GPU
(2) CUDA cores: 3584
(3) Memory: 16 GB
(4) Memory Bandwidth: 720 GB/s

NVIDIA V100

(1) Model Name: V100 GPU
(2) CUDA cores: 5120
(3) Memory: 32 GB
(4) Memory Bandwidth: 900 GB/s

NVIDIA A100

(1) Model Name: A100 GPU
(2) 6912 CUDA cores
(3) Memory: 40 GB HBM2e
(4) Memory Bandwidth: 1.5 TB/s

The software stack that is needed to run experiments is listed in the Table-1 in the paper.

How to run the code: The Makefile in the root directory contains the commands one need to compile all the codes.

Below list the binary files you will generate from the Makefile and their jobs. Detail explaination of usage is described in the README.md file.

- intel-tlrmvm: tlrmvm on Intel Cascade Lake
- amd-tlrmvm: tlrmvm on AMD EPYC Rome
- nec-tlrmvm: tlrmvm on NEC system
- cuda-tlrmvm: tlrmvm on NVIDIA GPU
- fx1000-tlrmvm: tlrmvm on A64FX FX1000
- intel-sgemv: single precision GEMV on Intel Cascade Lake
- amd-sgemv: single precision GEMV on AMD EPYC Rome
- nec-sgemv: single precision GEMV on NEC system
- cuda-sgemv: single precision GEMV on NVIDIA GPU
- fx1000-sgemv: single precision GEMV on A64FX FX1000

Be sure to set the environment variable properly to link the right libraries.

Below is description to run code on NEC.

To compile the program, we used NEC C compiler 3.1.1, NEC MPI 2.13.0 and NEC NLC 2.10 for CBLAS library. The bash script bash/build_nec.sh can be used to compile the program using this compilation environment. The bash script bash/run_nec.sh can be used to run all configurations for the MAVIS case.

If you can't open link below, use this link https://www.dropbox.com/s/io5l4bm22ocvd53/codesc21_edited.zip

Author-Created or Modified Artifacts:

```
Persistent ID: https://www.dropbox.com/s/io5l4bm22oc⌋
↪ vd53/codesc21\_edited.zip
Artifact name: TLR-MVM code and experiment results
↪ (dropbox)

Persistent ID: https://drive.google.com/file/d/16WH8⌋
↪ MZN\_z\_oCQY7gPrNgsONAnYY8u7nX/view?usp=sharing
Artifact name: TLR-MVM code and experiment results
↪ (google driver)
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: NEC B300-8 server with 8 Vector Engines 20B

Operating systems and versions: Linux RedHat 7.7 running kernel 3.10.0-1062.4.1.el7.x86_64

Compilers and versions: NEC Compilers 3.1.1

Libraries and versions: MPI 2.13.0 and NEC NLC 2.10

Input datasets and versions: Random Dataset