Evaluating PaRSEC through Matrix Computations in Scientific Applications

Qinglei Cao[1], Thomas Herault[2], Aurelien Bouteiller[2],

Joseph Schuchart[2], and George Bosilca[2,3]

[1] Saint Louis Universiity
[2] University of Tennesse, Knoxville
[3] NVIDIA

Abstract. Task-based runtime systems, characterized by their dynamic execution models and optimized resource management, contribute significantly to the computational revolution. They enable the development of more intricate and adaptable algorithms, essential in the field of computational science. This paper provides an in-depth exploration of the PaRSEC task-based runtime system, particularly focusing on its versatility in managing a variety of matrix computations. More specifically, we examine PaRSEC's role in enhancing efficiency when solving linear systems and processing dense, low-rank, mixed-precision, and sparse matrix operations, which are crucial in scientific applications, e.g., climate/weather prediction and 3D unstructured mesh deformation—the primary focus of this study. Through experimentation and analysis, we showcase PaRSEC's ability to significantly boost computational efficiency and scalability across a range of computationally intensive and less intensive tasks on various hardware architectures. Our findings not only underscore the potential of PaRSEC in advancing sustainable, efficient, and accurate domain modeling and simulation but also emphasize the growing necessity of task-based runtime systems in supporting the next generation of matrix computations.

Keywords: Task-based runtime · Matrix computations · Cholesky factorization · Low rank approximation · Mixed precision · Sparse operation.

1 Introduction

The world of High-Performance Computing (HPC) has undergone a remarkable evolution, transitioning from simple, single-processor systems to today's complex multi-core, multi-accelerator, and multi-node architectures [1]. This advancement has brought forth a new era of computing power, capable of performing quintillions of calculations per second. However, with this increase in power comes a corresponding rise in complexity, particularly in hardware design and architecture. HPC systems now feature a diverse array of components such as CPUs and GPUs, each with memory hierarchies adding layers to the computational puzzle. This complexity presents significant challenges in terms of pro-

gramming and optimization, demanding innovative approaches to fully harness the potential of these sophisticated machines [2].

Parallel to the evolution of hardware, the complexity of domain-specific applications in HPC has also escalated. These applications, ranging from molecular dynamics to large-scale astrophysics, now require the processing of enormous datasets and the execution of highly complex algorithms. The computational demands of these tasks have grown exponentially, often outpacing the advancements in hardware capabilities. As a result, there is an increasing need for specialized software that can effectively leverage the available hardware resources while managing the intricacies of these domain applications. This necessity is particularly acute in areas such as climate modeling and biomedical research, where accurate and timely results are critical.

In response to these challenges, task-based runtime systems have emerged as a pivotal solution [3–8]. These systems adopt a dynamic execution model, which allows for more efficient and adaptive management of computational tasks. Unlike traditional static execution models, task-based runtimes dynamically allocate resources based on the real-time demands of each task. This flexibility is crucial for optimizing performance across a diverse range of applications and hardware architectures [9–13]. Furthermore, these systems enable better load balancing, communication-computation overlap, and reduce idle time of computational resources, thereby enhancing overall efficiency and scalability.

In this study, we delve into the multifaceted capabilities of the PaRSEC task-based runtime system [8,14], with a special emphasis on its adept handling of diverse matrix operations, including dense, low-rank, mixed-precision, and sparse matrices. These matrix types are integral to a wide spectrum of scientific endeavors, notably in the realms of climate forecasting and 3D unstructured mesh deformation [15–21] – areas that form the cornerstone of our research. Through a series of experiments and analytical processes, we illuminate the profound impact of PaRSEC on elevating the performance and scalability of computational tasks. These tasks vary in their computational demands, yet consistently benefit from PaRSEC's robust architecture across diverse hardware platforms. This is particularly significant in an era where accuracy and efficiency are paramount.

The contributions of this paper are as follows. We go a step further to illustrate the indispensable role of task-based runtime systems like PaRSEC in the landscape of scientific computing. By adeptly balancing computational loads and optimizing resource utilization, PaRSEC not only excels in current computing environments but also paves the way for groundbreaking advancements in complex matrix computations. This study, therefore, not only showcases PaRSEC's current achievements but also sets the stage for task-based runtime's pivotal role in the evolution of scientific computing, with PaRSEC guiding the way beyond the traditional niche of dense, regular algorithms.

The remainder of this paper unfolds in the following manner. We present related work in Section 2 and introduce PaRSEC in Section 3. Section 4 details the scientific applications. Then, we present the performance analysis in Section 5, followed by conclusions and planned work in Section 6.

2 Related Work

In the realm of HPC, the emergence of task-based runtime systems has been a pivotal development, particularly in their adept handling of the intricacies and concurrent nature of contemporary computing architectures. These systems are adept at decomposing computational processes into smaller, manageable tasks. This decomposition facilitates dynamic allocation and balancing of computational load, while concurrently mitigating the overheads associated with communication and synchronization processes.

OpenMP [5], a widely recognized standard for parallel programming in shared-memory systems, includes task-based features that have transformed it into a dynamic, task-oriented environment. Parallelism is achieved through a mix of directives and library routines, enabling efficient task management by the compiler and runtime system. OmpSs [4], an extension to OpenMP, introduces support for heterogeneous systems, asynchronous tasks, and data dependencies, enhancing flexibility. Similarly, COMP Superscalar (COMPSs) [22] aims to simplify development for distributed systems with a programming interface and a runtime system that leverages application parallelism. StarPU [3] provides a framework for environments with distributed, heterogeneous multicore systems, enabling task-specific kernel annotations and efficient task scheduling and data management by the runtime system. HPX (High-Performance ParalleX) [6], based on the ParalleX model, offers a C++ runtime system optimized for high-performance computing in parallel and distributed environments. Lastly, Legion [7], a runtime system with a unique approach, is designed for distributed and heterogeneous computing architectures. Legion's programming model distinctively separates the definition of tasks and data from their actual mapping onto hardware resources. This approach not only facilitates automatic parallelism detection but also optimizes data locality, thereby enhancing both performance and scalability.

In this work, we concentrate on PaRSEC [8], which offers a set of original programming paradigms that often enable higher scalability than competing approaches; our study particular emphasis is on assessing performance efficiency and scalability of these programming concepts within the context of real-world scientific applications, instead of relying on synthetic benchmarks like in [23].

3 The PaRSEC Runtime System

The core design of PaRSEC, akin to other similar systems, leverages the concepts of tasks and dependencies. These concepts are instrumental in defining computations and their data flows, allowing for the representation of algorithms as Directed Acyclic Graphs (DAGs) where tasks are nodes and dependencies are edges. PaRSEC distinguishes itself in its ability to dynamically map these DAGs across distributed resources, ensuring that data dependencies are meticulously managed. This involves adeptly moving data across various memory spaces—be it within a node, across nodes, or between different devices—and efficiently assigning tasks to diverse computational resources. The system's interaction with users is facilitated through a range of Domain-Specific Languages

(DSLs), offering users heightened flexibility and enabling scientists to express complex algorithms more intuitively. One such DSL that is used in this research, the Parameterized Task Graph (PTG) [24] provides a succinct yet expansive description of task dependencies through a format known as the Job Data Flow (JDF). This representation allows for the categorization of tasks into classes, each encompassing the necessary details for the instantiation and execution of task instances on various computational units. PaRSEC is further enhanced by integrating Template Task Graph (TTG) [25], a C++ API that expands on PTG by allowing dynamic task dependency selection with varied parameter types. Moreover, PaRSEC incorporates the Dynamic Task Discovery (DTD) [26], a DSL that focuses on sequential task insertion in DAG construction, though it shares the overhead challenges common to distributed task-insertion runtimes.

4 Applications as Testbed

Two scientific applications are utilized as testbeds, summarized in the following.

Geospatial Modeling Towards Climate and Weather Prediction [16, 20,21]. Gaussian processes (GPs) serve as a cornerstone in machine learning and Bayesian statistics, particularly for their versatility in modeling and predicting complex behaviors. They are especially prevalent in spatial data analysis. One of the key methods in employing GPs for spatial data is through Maximum Likelihood Estimation (MLE). This involves defining a GP model with specific mean and covariance functions for a set of spatial points. The MLE method then seeks to optimize the model parameters to best fit the observed data. Consider a set of spatial observations $\mathbf{Z}$, where $\mathbf{Z} = \{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n)\}^\top$ corresponds to the observed values at n different locations $\mathbf{s}_1, \ldots, \mathbf{s}_n$ in a d-dimensional space $\mathbb{R}^d$. We assume these observations come from a stationary and isotropic Gaussian random field with a mean of zero. The covariance between any two points is defined by the function $C(\mathbf{h}; \boldsymbol{\theta}) = \mathrm{cov}\{Z(\mathbf{s}), Z(\mathbf{s}+\mathbf{h})\}$, where $\mathbf{h}$ is the lag vector in $\mathbb{R}^d$, and $\boldsymbol{\theta}$ is a vector of unknown parameters. The covariance matrix for these points, denoted by $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, has entries $\boldsymbol{\Sigma}_{ij} = C(\mathbf{s}_i - \mathbf{s}_j; \boldsymbol{\theta})$ for $i, j = 1, \ldots, n$, and is both symmetric and positive definite. The statistical inference of $\boldsymbol{\theta}$ is based on the Gaussian log-likelihood function: $\ell(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\mathbf{Z}^\top\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{Z}$. The goal in GP modeling is to determine the optimal parameter vector $\widehat{\boldsymbol{\theta}}$ that maximizes this log-likelihood function. We consider the squared exponential covariance function for 2D/3D spaces, which is given by: $C(\mathbf{h}; \boldsymbol{\theta}) = \sigma^2 \exp\left(-\frac{h^2}{\beta}\right)$, where $h = \|\mathbf{h}\|$ is the Euclidean distance between spatial points, σ^2 is the variance, β represents the correlation range, and $\boldsymbol{\theta} = (\sigma^2, \beta)^\top$.

3D Unstructured Mesh Deformation [19, 27]. In simulations involving fluid-structure interactions with 3D moving bodies, handling large mesh deformations is a significant challenge. The Radial Basis Function (RBF) method offers a solution for generating high-quality adaptive meshes, which is particularly useful for determining the movement of internal nodes within a volume based on the movement of nodes on the boundary. Following the approach outlined in [28],

we can express the displacement d within the entire domain as a sum of radial basis functions: $d(\mathbf{x}) = \sum_{i=1}^{n_b} \alpha_i \phi(\|\mathbf{x} - \mathbf{x}_{b_i}\|) + p(\mathbf{x})$, where $\mathbf{x}_{b_i} = [x_{b_i}, y_{b_i}, z_{b_i}]$ represents the known boundary nodes, p is a polynomial, n_b is the number of boundary nodes, and ϕ is a chosen basis function. The coefficients α_i and the polynomial p are determined by the boundary conditions $d(\mathbf{x}_{b_i}) = d_{b_i}$, with d_b containing the known displacement values at the boundary. The unknown coefficients α must satisfy the constraint $\sum_{i=1}^{n_b} \alpha_i p(\mathbf{x}_{b_i}) = 0$. We consider the Gaussian RBF, which is defined as $\phi(r) = \exp(-r^2)$, where r is the Euclidean distance. To manage the condition numbers of the RBF matrices, global support functions are scaled by a shape parameter δ, leading to the scaled RBF function $\phi_\delta(r) := \phi(r/\delta)$, with δ typically set to half the minimum distance between nodes: $\delta = \frac{1}{2} \times \min \|\mathbf{x} - \mathbf{x}_{b_i}\|$.

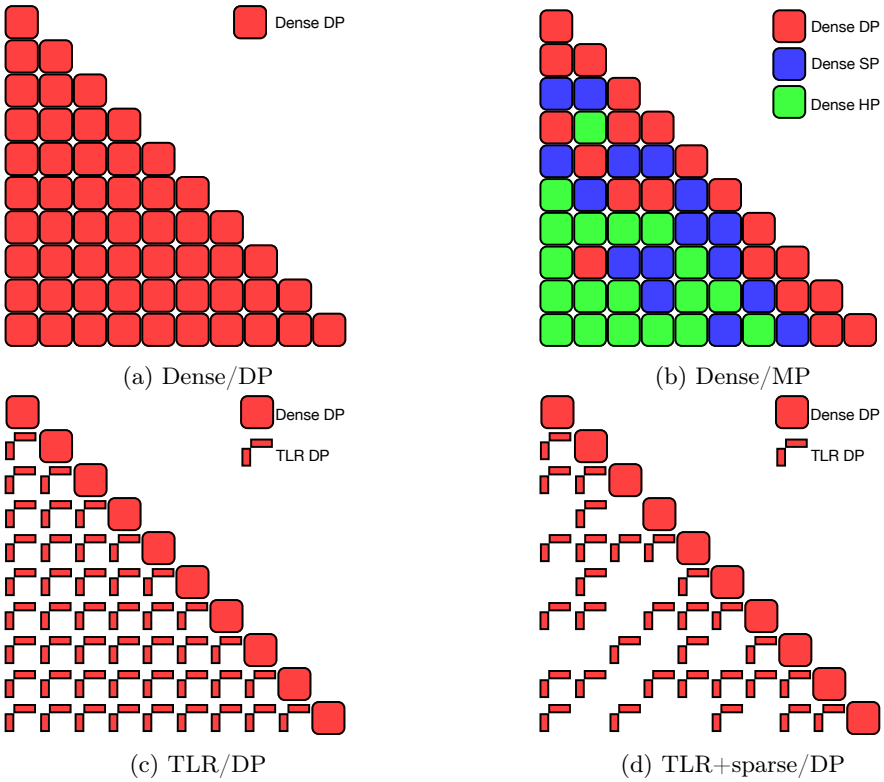
These two scientific applications both necessitate the solution of large-scale dense linear systems, which involve conducting a Cholesky factorization on a symmetric positive-definite covariance matrix, a process that is intensive in terms of both memory usage and computational load. HiCMA [15–21], a leading-edge solution for these applications, powered by PaRSEC, explores the data sparsity and/or sparsity in these applications and adopts tile low-rank (TLR) and mixed-precision (MP) technologies to address these challenges. In this study, we assess PaRSEC via HiCMA, focusing on several matrix computations (employing dense/MP and TLR/DP for geospatial modeling, and TLR+sparse/DP for 3D unstructured mesh deformation), as illustrated in Fig. 1.

- Fig. 1(a) demonstrates the traditional approach, where the entire matrix is handled in double-precision (DP or FP64).
- Fig. 1(b) presents a tile-based precision-aware approach for matrix computations. Here, the precision of each tile (DP, single-precision (SP or FP32), or half-precision (HP or FP16)) is determined by its norm relative to the overall matrix norm [20, 29]. The choice of precision affects computational load, memory requirements, and communication volume.
- Fig. 1(c) illustrates the use of TLR compression by condensing the off-diagonal tiles of the dense covariance matrix to a certain accuracy, specific to the application (see the heatmap in [15] for 2D/3D kernels). The varying rank of each tile, along with the rank discrepancy between on- and off-diagonal tiles, poses a new challenge in balancing computation, memory, and communication loads.
- Fig. 1(d) also shows the TLR compression format, but combined with sparsity, meaning the ranks of some tiles are reduced to zero after compression. The level of sparsity is application-specific [19], further exacerbating the load imbalance challenges inherent in the TLR format.

5 Performance Results and Analysis

5.1 Experimental Settings

Experiments were carried out on four systems, each with a unique architecture.



(a) Dense/DP (b) Dense/MP

(c) TLR/DP (d) TLR+sparse/DP

Fig. 1: Matrix computations.

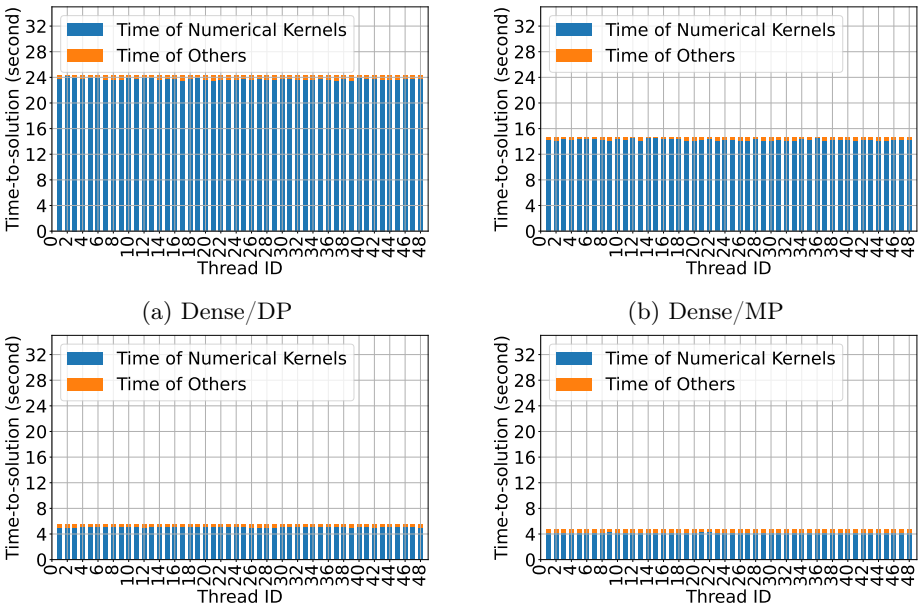
- Shaheen II: This is a Cray XC40 system comprising 6,174 compute nodes. Each node is equipped with two 16-core Intel Haswell CPUs operating at 2.30 GHz and has 128 GB of DDR4 main memory.
- Fugaku: This system is based on ARM architecture and includes over 150,000 compute nodes. Each node features a 48-core A64FX CPU running at a maximal 2.2 GHz (boost mode), coupled with 32 GB of HBM2 main memory.
- Haxane: An Intel-based GPU compute node, incorporating two 8-core Xeon(R) Silver 4309Y CPUs at 2.80 GHz, 63 GB of main memory, and a single NVIDIA H100 PCIe GPU.
- Frontier: This is a GPU-based AMD cluster with 9,408 compute nodes. Each node is composed of a 64-core AMD Optimized 3rd Gen EPYC CPU and four AMD MI250X GPUs, with access to 512 GB of DDR4 memory.

Accuracy thresholds are set according to the requirements of each application: 10^{-9} for geospatial modeling and 10^{-5} for 3D unstructured mesh deformation. The "band distribution" strategy [16] for TLR formats is employed, complemented by a two dimensional block cyclic data distribution (2DBCDD) strategy for tiles outside the band, using a process grid of $P \times Q$ (as square as possible), where $P \leq Q$. A 2D squared exponential covariance function is used in geospatial

modeling unless otherwise specified. Experiments are repeated multiple times to ensure consistency; as minimal variation is observed, the highest performance achieved is reported.

5.2 Load Balancing

Figure 2 illustrates the distribution of computational load across the 48 cores within a single Fugaku node for the four matrix computations discussed in Figure 1. The size of the matrices is consistent across these four evaluations. In each one, the cost for each core is represented by two components: the blue bars indicate the time spent on executing numerical kernels (including POTRF for Cholesky decomposition on diagonal tiles, TRSM for solving triangular matrix equations, SYRK for symmetric rank-k updates, and GEMM for general matrix multiplication), whereas the orange bars represent all additional costs incurred by that core, encompassing CPU idle periods, runtime overheads, communication delays, etc. The analysis of these figures reveals several key insights: (1) a progressive reduction in the time-to-solution transitioning from dense/DP, to dense/MP, to TLR/DP, and finally to TLR+sparse/DP formats; (2) an exceptionally balanced workload distribution, high CPU utilization, and minimal overhead across all cores, particularly notable in the TLR+sparse/DP format where variations in rank and the presence of sparsity are factors, as detailed in



(a) Dense/DP (b) Dense/MP

(c) TLR/DP (d) TLR+sparse/DP

Fig. 2: Load balancing across shared memory on Fugaku.

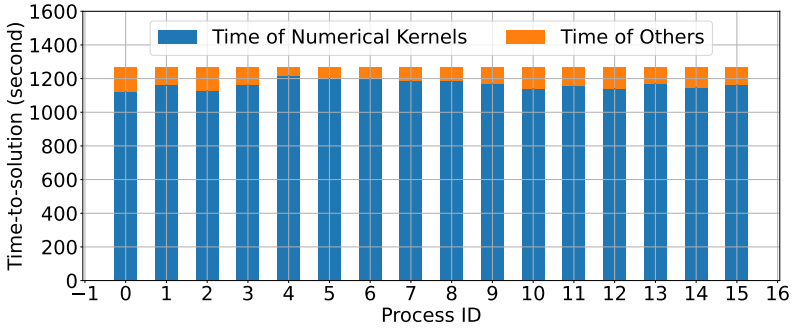


Fig. 3: Load balancing across distributed memory on Shaheen II.

Section 4. Moreover, the sparsity level in Figure 2(d) is approximately 2% for this smaller matrix size, explaining the modest improvement when moving from TLR/DP to TLR+sparse/DP. Similarly, Figure 3 assesses the balance of workload distribution across nodes/processes for TLR/DP on 16 Shaheen II nodes.

5.3 GPU Efficiency

HiCMA currently lacks support for TLR computations on GPUs. Consequently, we present an analysis of GPU utilization for computations in dense formats, as illustrated in Figure 4. Each point in these figures corresponds to the actual GPU occupancy, determined through periodic measurements using Nvidia's suite of diagnostic tools. The analysis reveals that attaining full (100%) GPU utilization is achievable during operations in both dense/DP and dense/MP formats. This observation underscores that the computational workflows are not hindered by data transfer operations (D2H and H2D), confirming that data transfers can be effectively overlapped with computational operations in PaRSEC.

5.4 Scalability

Fig. 5 illustrates the performance scalability of PaRSEC on homogeneous CPU architectures. Specifically, Fig. 5(a) presents the weak scaling of dense/DP Cholesky factorization via HiCMA on Shaheen II, benchmarked against ConfCHOX [30].

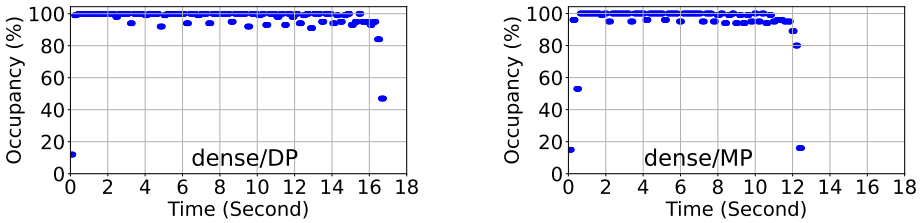
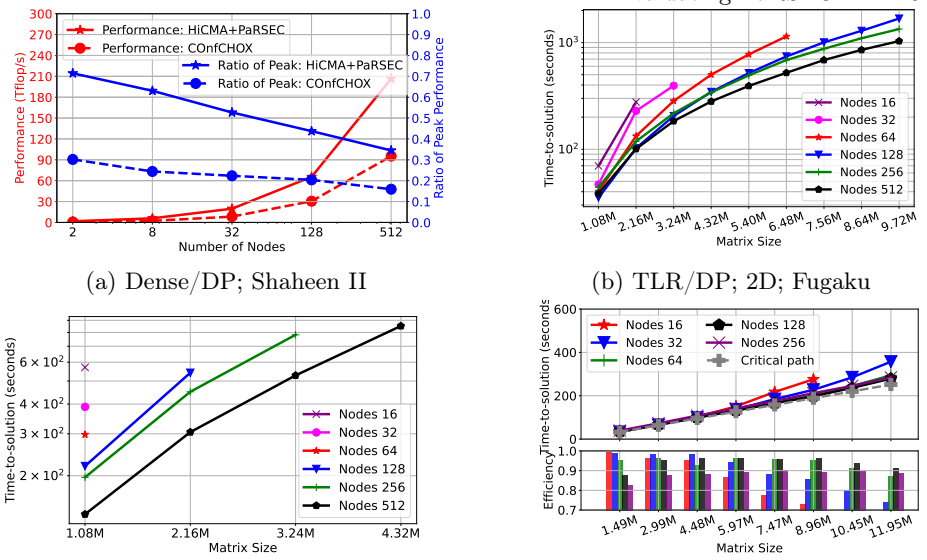


Fig. 4: GPU occupancy on H100.



(a) Dense/DP; Shaheen II

(b) TLR/DP; 2D; Fugaku

(c) TLR/DP; 3D; Fugaku

(d) TLR+Sparse/DP; Shaheen II

Fig. 5: Performance scalability on homogenous CPU architectures.

The settings follow the specifications in [30], e.g., matrix size of $8192 \times \sqrt{P}$ where P is the number of processes. In this figure, the left y-axis quantifies the achieved performance, whereas the right y-axis compares this performance to the theoretical peak. Notably, HiCMA consistently outperforms ConfCHOX, frequently securing doubled performance gains. Fig. 5(b) and 5(c) detail the performance on Fugaku for TLR/DP in 2D and 3D respectively. These two figures highlight strong scalability for each matrix size, with individual graphs indicating weak scalability per node. The transition from 2D to 3D kernels represents a shift to more computationally intensive problems, thereby enhancing scalability along with smaller solvable matrices on identical resources. This effect is particularly pronounced for TLR+sparse/DP, as depicted in Fig. 5(d). Here, the upper part displays performance across varying node counts, while the lower part evaluates this performance relative to the bounded critical path in Cholesky factorization. This critical path includes the serial and incompressible steps: POTRF, the initial TRSM, and the first SYRK for each panel factorization. This analysis considers only the computational costs of these three kernels, with attainable performance reaching 90% across all matrix sizes.

Furthermore, Fig. 6 presents the performance for dense/DP and dense/MP across varying node counts on Frontier. Annotations related to the "MP effect" highlight the observed maximum performance enhancement achieved by MP over DP under different node configurations, consistently achieving a speedup exceeding 3×. Additionally, in terms of parallel efficiency, there is a notable improvement in performance, with a 13.6× and 10.6× enhancement observed when scaling from 4 nodes to 64 nodes for dense/DP and dense/MP, respectively.

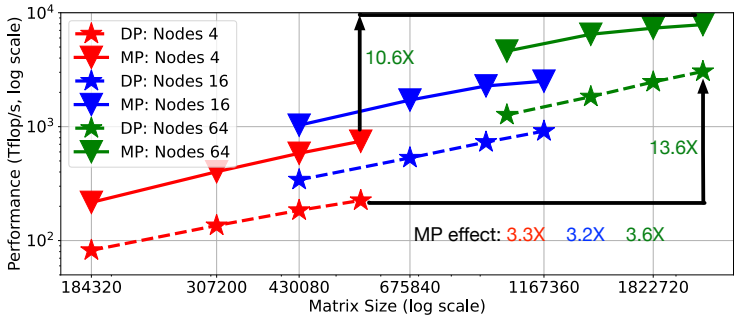


Fig. 6: Performance scalability on heterogeneous GPU system Frontier.

All in all, these findings highlight PaRSEC's versatility in handling complex real-world scientific applications, particularly regarding load balancing, efficiency, and scalability.

6 Conclusion and Future Work

This study focuses on PaRSEC's versatility in handling operations introduced by scientific applications, which involve dense matrices, low-rank matrices, matrices in mixed-precision formats, and sparse matrices, each presenting unique challenges. Through experiments and analysis, we demonstrate PaRSEC's remarkable capabilities in enhancing load balancing, computational efficiency, and scalability across a variety of hardware architectures. Looking forward, we aim to delve deeper into task-level behaviors and hardware counter metrics to uncover more granular insights, especially on Frontier. Furthermore, we envisage expanding our investigation to encompass additional task-based runtime systems, thereby broadening the scope of our research.

Acknowledgments. This research was supported in part by internal awards from Saint Louis University (Grant-0001651 and PROJ-000498) and the Exascale Computing Project (17-SC-20-SC). For computer time, this research used the compute node at Innovative Computing Laboratory, Shaheen II supercomputer at King Abdullah University of Science & Technology, Fugaku supercomputer at RIKEN (Group ID: ra010008), and Frontier supercomputer at Oak Ridge National Laboratory (Project ID: CSC612).

References

1. Hans Meuer, Erich Strohmaier, Jack Dongarra, and Horst Simon. The Top500 List. June 2020. `http://www.top500.org`.
2. David E Keyes, Hatem Ltaief, and George Turkiyyah. Hierarchical algorithms on hierarchical architectures. *Philosophical Transactions of the Royal Society A*, 378(2166):20190055, 2020.

3. C. Augonnet, S. Thibault, R. Namyst, and P. Wacrenier. StarPU: A unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency Computat. Pract. Exper.*, 23:187–198, 2011.
4. A. Duran, R. Ferrer, E. Ayguadé, R.M. Badia, and J. Labarta. A Proposal to Extend the OpenMP Tasking Model with Dependent Tasks. *International Journal of Parallel Programming*, 37(3):292–305, 2009.
5. OpenMP. OpenMP 5.2 Complete Specifications, 2021.
6. T. Heller, H. Kaiser, and K. Iglberger. Application of the ParalleX Execution Model to Stencil-based Problems. *Computer Science - Research and Development*, 28(2-3):253–261, 2013.
7. M. Bauer, S. Treichler, E. Slaughter, and A. Aiken. Legion: Expressing Locality and Independence with Logical Regions. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, 2012.
8. G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Herault, and J. Dongarra. PaRSEC: A Programming Paradigm Exploiting Heterogeneity for Enhancing Scalability. *Computing in Science and Engineering*, 99:1, 2013.
9. G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Hérault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaief, P. Luszczek, A. YarKhan, and J. Dongarra. Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA. In *IPDPS Workshops*. IEEE, 2011.
10. Kadir Akbudak, Hatem Ltaief, Aleksandr Mikhalev, Ali Charara, Aniello Esposito, and David Keyes. Exploiting data sparsity for large-scale matrix computations. In *European Conference on Parallel Processing*. Springer, 2018.
11. Noha Al-Harthi, Rabab Alomairy, Kadir Akbudak, Rui Chen, Hatem Ltaief, Hakan Bagci, and David Keyes. Solving Acoustic Boundary Integral Equations Using High Performance Tile Low-Rank LU Factorization. In *35th International Conference on High Performance*. Springer, 2020.
12. H. Jagode, A. Danalis, and J. Dongarra. Accelerating nwchem coupled cluster through dataflow-based execution. *The International Journal of High Performance Computing Applications*, page 1–13, 01-2017 2017.
13. M. Tillenius, E. Larsson, E. Lehto, and N. Flyer. A task parallel implementation of a scattered node stencil-based solver for the shallow water equations. In *Proc. 6th Swedish Workshop on Multi-Core Computing :*. Halmstad University, 2013.
14. G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Herault, and J.J. Dongarra. PaRSEC: Exploiting heterogeneity to enhance scalability. *Computing in Science Engineering*, 15(6):36–45, Nov 2013.
15. Qinglei Cao, Yu Pei, Thomas Herauldt, Kadir Akbudak, Aleksandr Mikhalev, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. Performance analysis of tile low-rank cholesky factorization using parsec instrumentation tools. In *2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools)*, pages 25–32. IEEE, 2019.
16. Qinglei Cao, Yu Pei, Kadir Akbudak, Aleksandr Mikhalev, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. Extreme-scale task-based cholesky factorization toward climate and weather prediction applications. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11, 2020.
17. Qinglei Cao, Yu Pei, Kadir Akbudak, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. Leveraging parsec runtime support to tackle challenging 3d data-sparse matrix problems. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2021.

18. Sameh Abdulah, Qinglei Cao, Yu Pei, George Bosilca, Jack Dongarra, Marc G Genton, David E Keyes, Hatem Ltaief, and Ying Sun. Accelerating geostatistical modeling and prediction with mixed-precision computations: A high-productivity approach with parsec. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):964–976, 2021.
19. Qinglei Cao, Rabab Alomairy, Yu Pei, George Bosilca, Hatem Ltaief, David Keyes, and Jack Dongarra. A framework to exploit data sparsity in tile low-rank cholesky factorization. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 414–424. IEEE, 2022.
20. Qinglei Cao, Sameh Abdulah, Rabab Alomairy, Yu Pei, Pratik Nag, George Bosilca, Jack Dongarra, Marc G Genton, David E Keyes, Hatem Ltaief, et al. Reshaping geostatistical modeling and prediction for extreme-scale environmental applications. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 2022.
21. Qinglei Cao, Sameh Abdulah, Hatem Ltaief, Marc G Genton, David Keyes, and George Bosilca. Reducing data motion and energy consumption of geospatial modeling applications using automated precision conversion. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 330–342. IEEE, 2023.
22. Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Alvarez, Fabrizio Marozzo, Daniele Lezzi, Raül Sirvent, Domenico Talia, and Rosa M Badia. Servicess: An interoperable programming framework for the cloud. *Journal of grid computing*, 12(1):67–91, 2014.
23. Elliott Slaughter, Wei Wu, Yuankun Fu, Legend Brandenburg, Nicolai Garcia, Wilhem Kautz, Emily Marx, Kaleb S Morris, Qinglei Cao, George Bosilca, et al. Task Bench: A parameterized Benchmark for Evaluating Parallel Runtime Performance. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.
24. A. Danalis, G. Bosilca, A. Bouteiller, T. Herault, and J. Dongarra. PTG: An Abstraction for Unhindered Parallelism. pages 21–30, 2014.
25. George Bosilca, Robert J Harrison, Thomas Herault, Mohammad Mahdi Javanmard, P Nookala, and Edward F Valeev. The Template Task Graph (TTG)-an Emerging Practical Dataflow Programming Paradigm for Scientific Simulation at Extreme Scale. In *IEEE/ACM 5th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*. IEEE, 2020.
26. R. Hoque, T. Herault, G. Bosilca, and J. Dongarra. Dynamic Task Discovery in PaRSEC: A Data-flow Task-based Runtime. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ScalA '17, 2017.
27. Rabab Alomairy, Wael Bader, Hatem Ltaief, Youssef Mesri, and David Keyes. High-performance 3d unstructured mesh deformation using rank structured matrix computations. *ACM Transactions on Parallel Computing*, 9(1):1–23, 2022.
28. A. De Boer, M. S. Van der Schoot, and Hester Bijl. Mesh Deformation Based on Radial Basis Function Interpolation. *Computers and Structures*, 2007.
29. Nicholas J Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022.
30. Grzegorz Kwasniewski, Marko Kabic, Tal Ben-Nun, Alexandros Nikolaos Ziogas, Jens Eirik Saethre, André Gaillard, Timo Schneider, Maciej Besta, Anton Kozhevnikov, Joost VandeVondele, et al. On the parallel i/o optimality of linear algebra kernels: Near-optimal matrix factorizations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.