???

Otho José Sirtoli Marcondes*, Lucas Mello Schnorr*
* Institute of Informatics/PPGC/UFRGS, Porto Alegre, Brazil

***Abstract*—**

I. Introduction

High-Performance Computing (HPC) systems, particularly computing clusters, are essential for solving large-scale scientific and engineering problems. These clusters consist of multiple interconnected nodes, each with its own processor units and memory. In order to maximize application performance on clusters, it is essential to consider both inter-node communication efficiency and workload balance across the computing nodes.

A crucial aspect of achieving efficient parallel performance is data partitioning, which determines how data is divided and distributed across the computing nodes. Among various strategies, static data partitioning is commonly used due to its simplicity and low runtime overhead. One of the examples of static data distribution is the block-cyclic (BC) distribution, a method that was popularized by the ScaLAPACK [**?**] library.

This paper focuses on a scenario that combines static data partitioning with dynamic task scheduling. By leveraging task-based runtimes, we aim to dynamically schedule tasks at runtime while maintaining a static block layout of data. This approach enables better adaptability to runtime variations, such as load imbalance and communication delays, while preserving the advantages of a static data map.

As a case study, we explore the LU factorization, a fundamental operation in linear algebra widely used in scientific computing. We adopt a block cyclic distribution scheme for the input matrix, a method that balances the computational load and spreads data evenly across processes. Our goal is to evaluate how dynamic scheduling of tasks can improve the performance of LU factorization in clusters.

Throughout the development of this work, several challenges were encountered related to the use of MPI for executing applications across multiple nodes. These included: configuration challenges with Guix for package management across distributed nodes; issues related to the TCP interface in the MPI NewMadeleine implementation; and errors when using StarVZ [**?**] visualization framework with the traces collected from the executions (still not resolved).

The paper is structured as follows. Section~II presents some related work on matrix distribution and modern task-based runtimes. Section~III details our methodology and explains how we conducted the experiments in our investigation. Section~IV presents the experiments and their results. Section~V concludes this work with some considerations.

II. Related Work

III. Experimental Methodology

IV. Results

V. Conclusion

Acknowledgements

The experiments in this work used the PCAD infrastructure, http://gppd-hpc.inf.ufrgs.br, at INF/UFRGS.