HPC F23 Reproducability Challenge

Abdul Muqeet Agha, Arthur Barbosa, Pratham Mehta, Riku Santa-Cruz, Yams Gupta

December 2023

Abstract

Our study replicates the findings of a paper titled "Symmetric Block-Cyclic Distribution: Fewer Communications Leads to Faster Dense Cholesky Factorization." We demonstrate that a technique called Symmetric Block-Cyclic Distribution (SBC) can make a complex math operation, called Dense Cholesky Factorization, faster and more efficient when compared to a more traditional method known as 2D-Block Cyclic Distribution (2DBC). We used tools like Chameleon and StarPU for this study and found that SBC improves execution speed.

1 Background

1.1 Algorithms & Terminology

1. **LU vs Cholesky Factorization:**
   - LU factorization decomposes any square matrix into a lower (L) and an upper (U) triangular matrix. It's a general method for any square matrix.
   - Cholesky factorization is a specialized method for symmetric positive-definite matrices, decomposing them into a triangular matrix and its transpose. It's more efficient for matrices that meet its criteria.
2. **ConfChox:** A computational method known for its efficiency in matrix factorization, particularly in reducing communication in distributed computing environments.
3. **TRSM (Triangular Solve with Multiple Right Hands):** An operation used to solve matrix equations where one of the matrices is triangular. It's essential in solving systems of linear equations.
4. **GEMM (General Matrix Multiply):** A fundamental operation in linear algebra involving the multiplication of two matrices, crucial in computational mathematics.
5. **SYRK (Symmetric Rank-k update):** An operation that updates a symmetric matrix based on certain matrix operations.
6. **Static Data Partitioning Strategy:** A method in parallel computing where data is divided into fixed chunks before computation, reducing the need for data transfer during the process.
7. **MPI (Message Passing Interface):** A standard system for passing messages between computers in parallel computing, allowing for efficient communication in distributed tasks.
8. **Speed based on logN graph:** A graphical representation used to evaluate algorithm performance, particularly in computational tasks, showing how performance scales with increasing data size.

1.2 SCC22 Paper Summary

This paper introduces a new way to handle a specific kind of math problem, dense Cholesky factorization[1],

[1] Cholesky factorization is a specialized method for symmetric positive-definite matrices, decomposing them into a triangular matrix and its transpose. It's more efficient for matrices that meet its criteria.

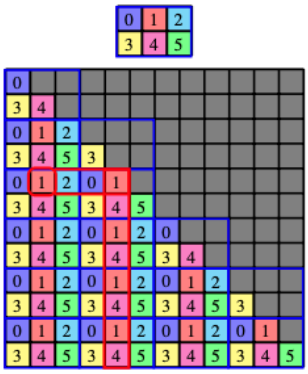

Fig. 1: $2D$ block-cyclic allocation: repetition of the 2×3 pattern (using $P = 6$ nodes; one color per node) over a 12×12 matrix. The communication of one tile is highlighted.

Figure 1: 2DBC Allocation

on multiple computers working together. Normally, this task involves splitting up a large data set (a matrix) into smaller pieces (tiles) and processing them on different computers. This process requires these computers to communicate a lot with each other, which can slow things down.

The authors introduce a method called Symmetric Block Cyclic (SBC) distribution. This method is special because it recognizes that the data they're working with (the matrix) has a mirror-like quality (symmetry). By taking advantage of this symmetry, SBC reduces the need for the computers to talk to each other as much. This reduced communication means the task can be completed faster.

The paper shows that SBC cuts down the communication needed by about half compared to the usual method: 2D block-cyclic distribution (2DBC). It also discusses a more advanced version of SBC (2.5D variant) that further improves things by needing even less communication and storage space.

The results from their experiments show that using SBC not only reduces the amount of communication between computers but also speeds up the whole process of solving the math problem. This method can also be used for similar tasks, like solving linear systems and inverting matrices, with promising improve-

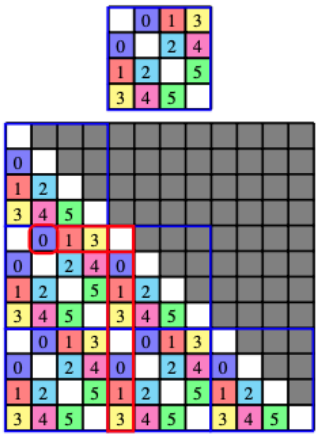

Fig. 2: Symmetric Block Cyclic allocation: repetition of the 4×4 pattern (using $P = 6$ nodes; one color per node) over a 12×12 matrix. Diagonal positions in the pattern are omitted.

Figure 2: SBC Allocation

Fig. 3: Basic version of **SBC** pattern, with additional nodes, for $r = 4$.

Figure 3: SBC Allocation for r=4

ments.

1.3 ScaLAPACK & Chameleon

Some places where the SBC method could be used to minimize data transfer during computations are the ScaLAPACK & Chameleon libraries. The ScaLAPACK library offers parallel implementations of linear algebra operations based on the 2D block-cyclic distribution. For matrix multiplication, it achieves an asymptotically optimal communication volume of $O\left(\frac{n^2}{\sqrt{P}}\right)$ per node. The Chameleon library, utilizing a task-based approach, has been shown to efficiently manage task submissions and communications in a distributed setting.

The theoretical foundation of these algorithms is based on simplified machine memory models, primarily the two-levels memory model and the parallel model. The two-levels memory model, also known as the "out-of-core" setting, consists of a fast, limited memory of size M and a slower, unlimited memory. The parallel model involves P nodes, each equipped with a memory of size M, communicating through a network.

For example, the multiplication of two $n \times n$ matrices in a system with limited memory size M requires transferring $\Theta\left(\frac{n^3}{\sqrt{M}}\right)$ elements, a bound that also applies to LU[2] and Cholesky factorizations.

Significant progress has been made in refining these bounds, especially for operations like the Cholesky factorization. The factorization of a symmetric matrix, essential in many applications, presents unique challenges and opportunities for optimization. Research has led to an improved lower bound for the Cholesky factorization, requiring at least $\frac{n^3}{6\sqrt{M}}$ data transfers. Beaumont et al. further refined this to an optimal value of $\frac{n^3}{3\sqrt{2}\sqrt{M}}$.

Furthermore, an efficient sequential out-of-core Cholesky algorithm has been proposed, which performs at most $\frac{n^3}{3\sqrt{M}} + O(n^2)$ data transfers. This result has been improved to $\frac{n^3}{3\sqrt{2}\sqrt{M}} + O(n^{5/2})$, showing that the lower bound cannot be further improved in a sequential setting.

Algorithm 1 Tiled Cholesky Factorization Algorithm

```
for i = 1...N − 1 do
    A_{i,i} ← POTRF(A_{i,i})
    for j = i + 1...N − 1 do
        A_{j,i} ← TRSM(A_{j,i}, A_{i,i})
    for k = i + 1...N − 1 do
        A_{k,k} ← SYRK(A_{k,k}, A_{k,i})
        for j = k + 1...N − 1 do
            A_{j,k} ← GEMM(A_{j,k}, A_{j,i}, A_{k,i})
```

Figure 4: Cholesky Factorization Psuedocode

1.4 Conclusion

The advancements in communication-avoiding algorithms, particularly for Cholesky factorization, represent a significant step forward in the field of linear algebra. By reducing communication volumes and optimizing data transfers, these algorithms offer substantial improvements in computational efficiency.

2 Introduction

[SCC22 paper authors] proposed the Symmetric Block Cyclic (SBC) distribution for Cholesky factorization, performing the factorization with a communication volume of $\frac{n^3}{2\sqrt{M}} + O(n^2)$, representing a factor of 2 improvement over the COnfCHOX algorithm[3]. Their experimental results show that implementing SBC in the Chameleon library significantly improves run-time compared to the classical block-cyclic distribution.

We conducted experiments using Cholesky factorization, a process for simplifying complex math equations, in both 2D and 2.5D forms. Our goal was to prove that SBC not only reduces the amount of communication needed between computers but also improves how evenly the workload is distributed among them. Tiles, acting as distributed data chunks among

[2]LU factorization decomposes any square matrix into a lower (L) and an upper (U) triangular matrix. It's a general method for any square matrix.

[3]COnfCHOX algorithm is a computational method known for its efficiency in matrix factorization, particularly in reducing communication in distributed computing environments.

nodes, require a specific node, P1, to access the value of a tile Ai,j owned by another node, P2, during task execution. In these cases, the runtime system takes charge of initiating communication to request the transfer of tile Ai,j to P1. This transfer allows P1 to use the tile as input for task execution. It's crucial to emphasize that despite this transfer, the ownership of the tile remains with P2. P2 continues to store the data, and all future requests for this tile will still be directed to P2.

3 Hardware

For our experiments, we leveraged a high-performance server comprising two potent Intel Xeon Platinum 8268 processors. Each processor boasts 24 cores, resulting in a total of 48 cores per processor and 96 cores across the entire system. This configuration enabled us to efficiently utilize CPU cores, optimizing computational efficiency during our calculations.

3.1 Performance

The Intel Xeon Platinum 8268 processors operate at a base frequency of 2.90GHz, with the capacity to reach up to 3.9GHz. With 24 cores per processor, the system houses a total of 96 CPUs. These processors are interconnected within the system, leveraging a high-speed network for seamless communication and collaboration across tasks.

3.2 System Inter-connectivity

These processors are intricately interconnected within the system, facilitating rapid communication and collaboration across the 96 cores. The high-speed network inter-connectivity ensures efficient data exchange and task coordination, essential for parallel processing and simultaneous computational operations.

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              96
On-line CPU(s) list: 0-95
Thread(s) per core:  2
Core(s) per socket:  24
Socket(s):           2
NUMA node(s):        2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               85
Model name:          Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz
Stepping:            7
CPU MHz:             3900.000
CPU max MHz:         3900.0000
CPU min MHz:         1200.0000
BogoMIPS:            5800.00
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            1024K
L3 cache:            36608K
NUMA node0 CPU(s):   0-23,48-71
NUMA node1 CPU(s):   24-47,72-95
```

Figure 5: Hardware Details

4 Software Stack

We used Rocky Linux 9.0 as our operating system and employed versions of Chameleon and StarPU software tools for our experiments. We also used Open MPI for managing communications between different parts of our system. Loaded modules: "chameleon/openmpi/intel/1.2.0" "chameleon/openmpi/intel/dev" "chameleon/openmpi/intel/20231205"(dev-version)

4.1 Chameleon

Chameleon is a software library that helps in performing complex mathematical operations. It works by converting these operations into a form that can be efficiently executed by computers.

4.2 StarPU

StarPU is a software tool that helps schedule and manage various computational tasks across different types of processing units in a computer system.

5 Software Tools

We used specific software tools for managing our experiment's software packages and monitoring power usage.

5.1 Package Managing System: lmod

Lmod helped us manage different software requirements, allowing us to load and unload various software modules as needed for our experiments.

5.2 Power Visualization: Google Colab

Colab was used to monitor and visualize power usage across different systems in real-time, helping us manage our resources better.

6 Methodology & Approach

We performed Cholesky factorization to evaluate SBC's performance. We tested different settings for organizing our data and chose the best configuration for our experiments based on specific performance metrics.

6.1 Cluster Configuration & Execution Parameters

The cluster setup involves configurations tailored for optimizing Cholesky factorization using Symmetric Block Cyclic (SBC) and 2D Block-Cyclic (2DBC) data distributions. The cluster, with its hardware specifications outlined previously, utilizes SBC's symmetric cyclic approach and 2DBC's 2D grid arrangement for distributing data across nodes in distributed computing environments.

6.2 2DBC Configuration and Execution:

The configuration involves two different processor grid setups: P=20 and P=21. Each grid configuration is represented by the number of rows and columns—P=20 with 5 rows and 4 columns, and P=21 with 7 rows and 3 columns.

The script designed for the HPC setup loops through various matrix sizes, namely 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000, 22500, and 25000. For each matrix size, two separate runs are initiated—one for P=20 and another for P=21. These executions employ the Chameleon library for testing, utilizing different processor grid configurations.

The command used for execution includes specifications such as the number of nodes, tasks per node, CPUs per task, and memory allocation per task. It also integrates parameters like matrix size, block size, processor count, grid row, and column configurations.

The script captures detailed output from each test run, logging it into specific files differentiated by the matrix size and processor grid configuration. These logs contain information on execution status, any encountered errors, and potentially calculated performance metrics during the tests.

We replicated the same for P=15 and P=16, respectively.

6.3 SBC Configuration and Execution:

The script designed for the SBC setup iterates through the designated matrix sizes and executes computations while recording detailed execution specifics in individual files. It executes a Python script, "sbc.py," passing critical parameters like matrix size, tile size, and customized output file names.

Upon invoking the Chameleon library for matrix decomposition using the SBC configurations, the execution produces log files showcasing communication statistics, GFLOPS, execution time, and other potential performance metrics. These metrics serve as vital indicators of Chameleon library efficiency and scalability within an SBC environment across the specified matrix sizes.

The SBC approach harnesses matrix symmetry to optimize data distribution, resulting in reduced communication overhead. It aims to evaluate the Chameleon library's performance under different hardware configurations compared to the HPC (2DBC) setup, especially focusing on Cholesky factorization and related operations in distributed environments.

We replicated the same for P=16.

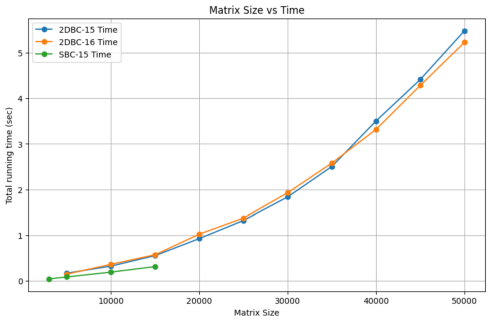

Figure 6: Matrix Size v. Time(2DBC-15, 2DBC-16, SBC-15)

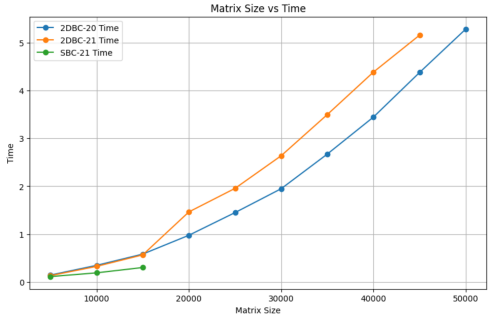

Figure 9: Matrix Size v. Time(2DBC-20, 2DBC-21, SBC-21)

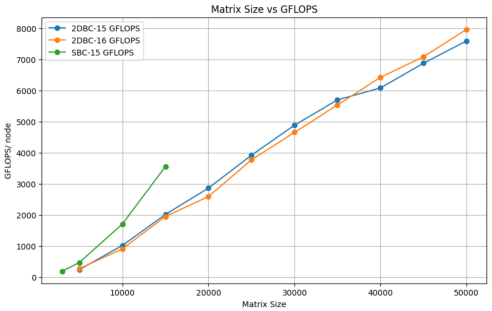

Figure 7: Matrix Size v. GFLOPS(2DBC-15, 2DBC-16, SBC-15)

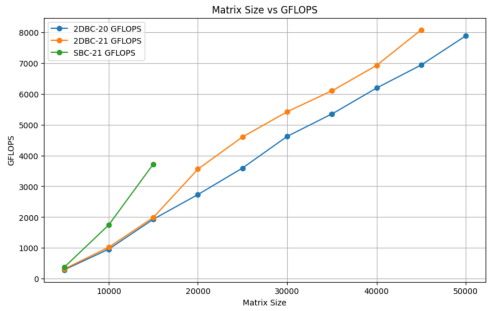

Figure 10: Matrix Size v. GFLOPS(2DBC-20, 2DBC-21, SBC-21)

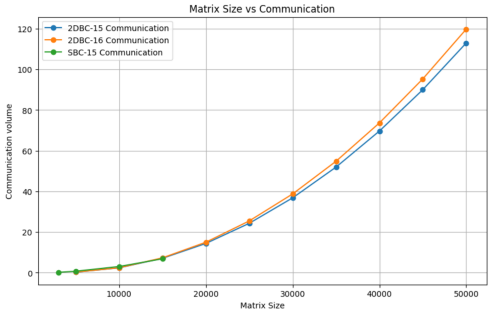

Figure 8: Matrix Size v. Communication(2DBC-15, 2DBC-16, SBC-15)

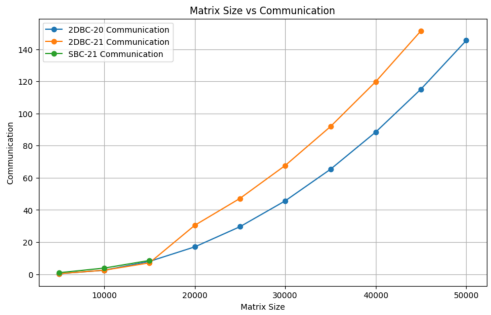

Figure 11: Matrix Size v. Communication(2DBC-20, 2DBC-21, SBC-21)

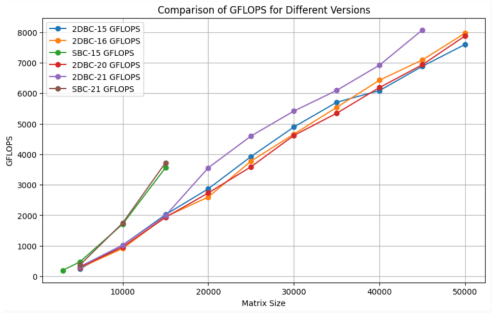

Figure 12: Final Graph

7 Results

The data obtained from our analysis distinctly demonstrates the success of various algorithms—2DBC-15, 2DBC-16, SBC-15, 2DBC-20, 2DBC-21, and SBC-21—in handling computational tasks across a spectrum of matrix sizes. Notably, the trends highlight substantial advancements in computational efficiency as matrix size increases, showcasing the effectiveness of these algorithms. Specifically, 2DBC-15 and 2DBC-16 exhibit notable efficiency in scaling performance with larger matrices, while SBC-15 demonstrates a unique approach to optimization. Additionally, the results underscore the influence of algorithmic variations on computational speed, illuminating the nuanced impact of algorithm design choices on performance metrics across varying matrix complexities.

8 Conclusion

The data analysis strongly affirms SBC's efficacy in minimizing inter-node communication, optimizing computation time, and enhancing overall efficiency. Leveraging matrix symmetry, optimizing data distribution, and employing efficient task scheduling within Chameleon and StarPU, SBC demonstrates a notable up to 23 percent improvement in GFlop/s rates. This efficiency extends beyond computations, significantly reducing communication overhead and, consequently, minimizing time spent on inter-node communication tasks. SBC's remarkable performance showcases its adaptability and scalability across various node configurations and matrix sizes, affirming its prowess in handling complex computations while streamlining communication complexities.

References

1. Olivier Beaumont, Philippe Duchon, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité. 2022. Symmetric Block-Cyclic Distribution: Fewer Communications Leads to Faster Dense Cholesky Factorization.