Distributed Memory Task-Based Block Low Rank Direct Solver

Sameer Deshmukh [1] Rio Yokota [1]

[1]School of Computing, Tokyo Institute of Technology

Abstract

Dense LU factorization takes $O(N^3)$ time. The Block Low Rank matrix format reduces the computation time to $O(N^2)$ and storage cost to $O(N^{1.5})$. This work compares a task-based distributed Block Low Rank LU factorization against various distributed dense matrix LU factorization implementations.

Block Low Rank (BLR) matrix

The **Block Low Rank (BLR)** matrix allows expressing a large dense matrix as a **collection of low rank and dense blocks**. Fig. 1 shows a BLR matrix with diagonal dense blocks and off-diagonal low rank blocks.

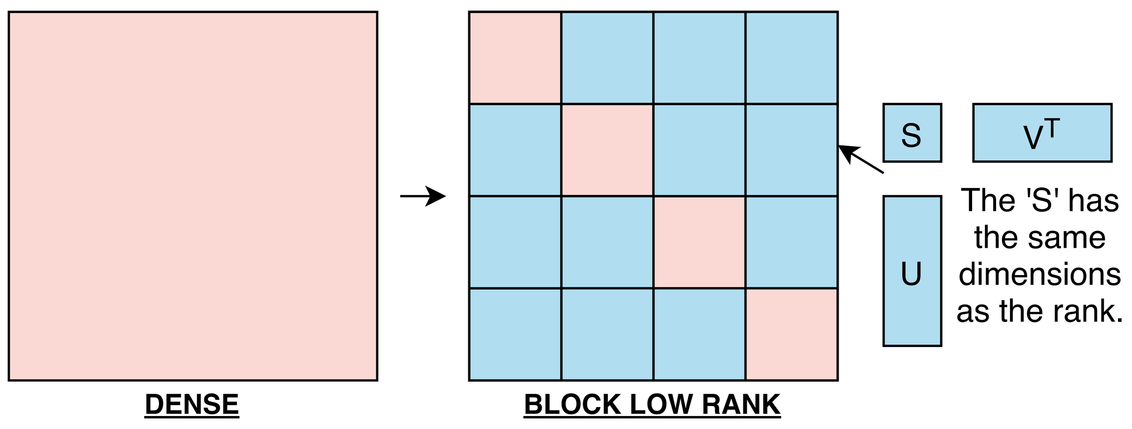

Figure 1: Large block dense matrix as a BLR matrix.

LU Factorization

The **LU factorization** splits a dense matrix into **lower and upper triangular** matrices.

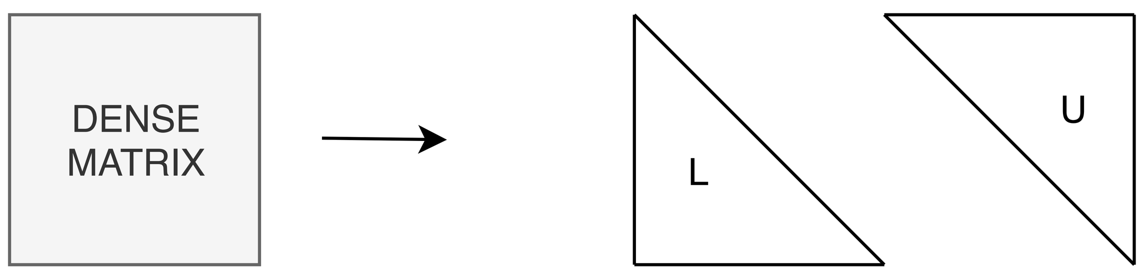

Figure 2: Dense matrix LU factorization.

Blockwise LU factorization

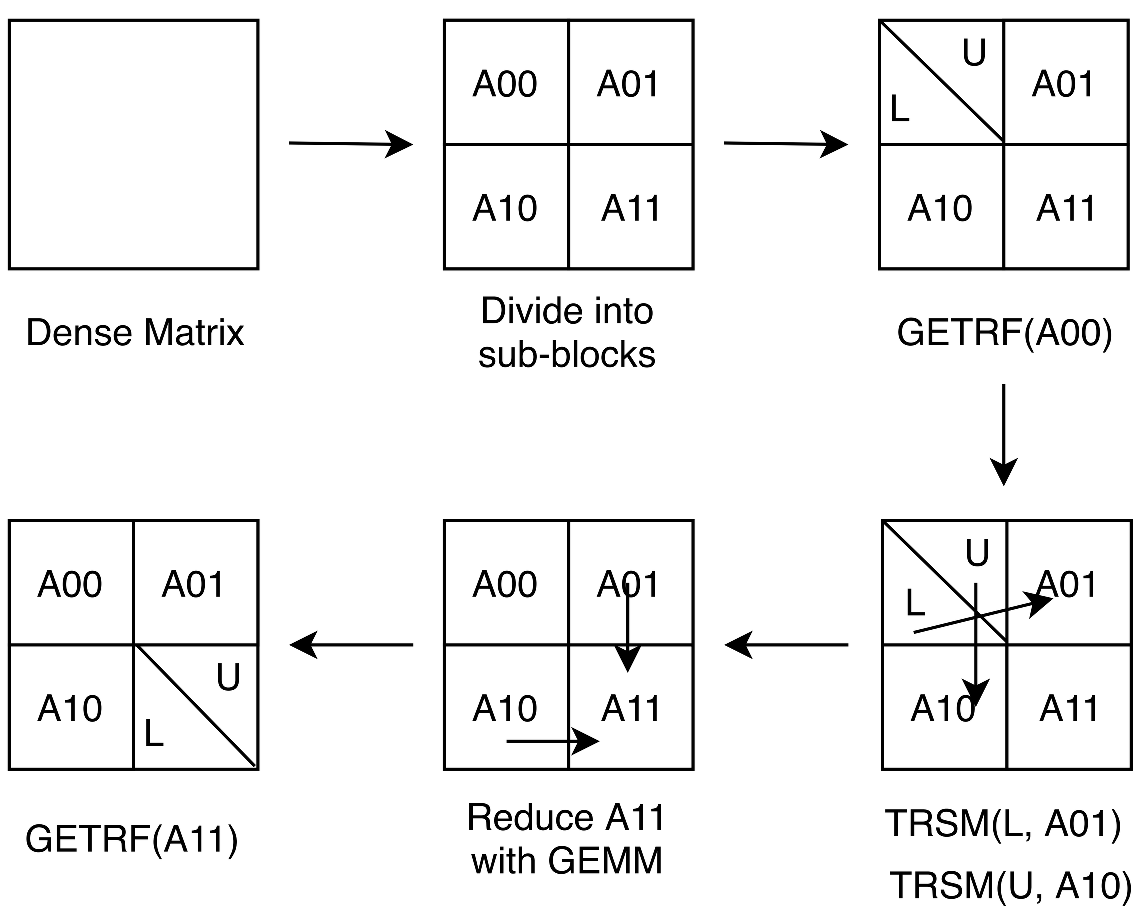

Figure 3: Block LU factorization.

Fig. 3 shows that an **LU factorization** can calculated by **dividing a large matrix into blocks** and performing **operations per block** (also called 'kernels').

- *Factor* $A_{00} = L_{00}U_{00}$.
- *Compute* $A_{10} = A_{10}U_{00}^{-1}$ and $A_{01} = L_{00}^{-1}A_{01}$.
- *Reduce* $A_{11} = A_{11} - A_{10}A_{01}$.
- *Factor* $A_{11} = L_{11}U_{11}$.

Runtime-Based Systems: starPU

A **runtime system** such as starPU expresses a **computation as a flow of tasks** and overlaps computation and communication. **Each kernel** of the blockwise LU can be **expressed as a task**. StarPU **executes** this graph, inserts **asynchronous communication** between tasks and **runs independent tasks** in parallel.

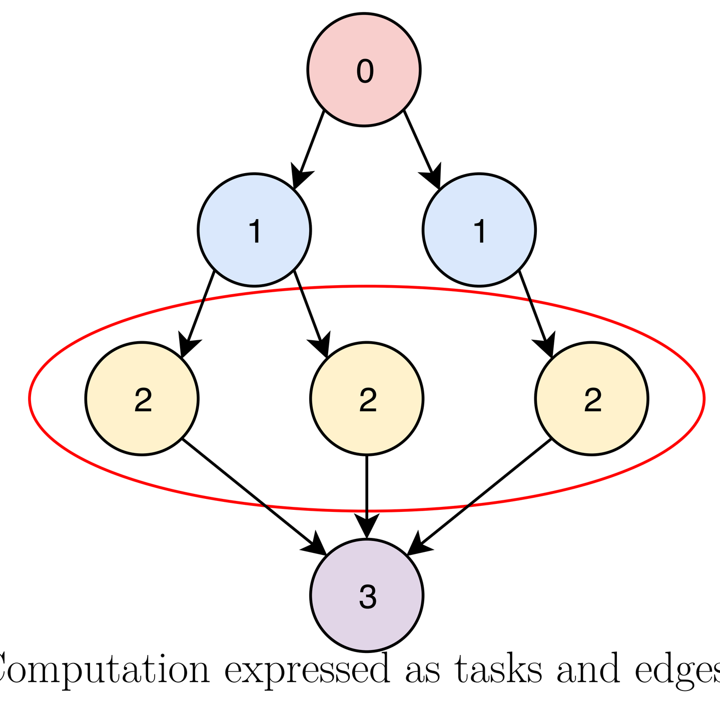

Figure 4: Computation expressed as tasks and edges in a run time system. Same colored nodes are executed together.

Distributed low rank LU problems

Lack of computational intensity combined with **irregular size of tasks** makes it hard to correctly to optimize distributed low rank LU factorization. This leads to several problems, as shown in Fig. 5.

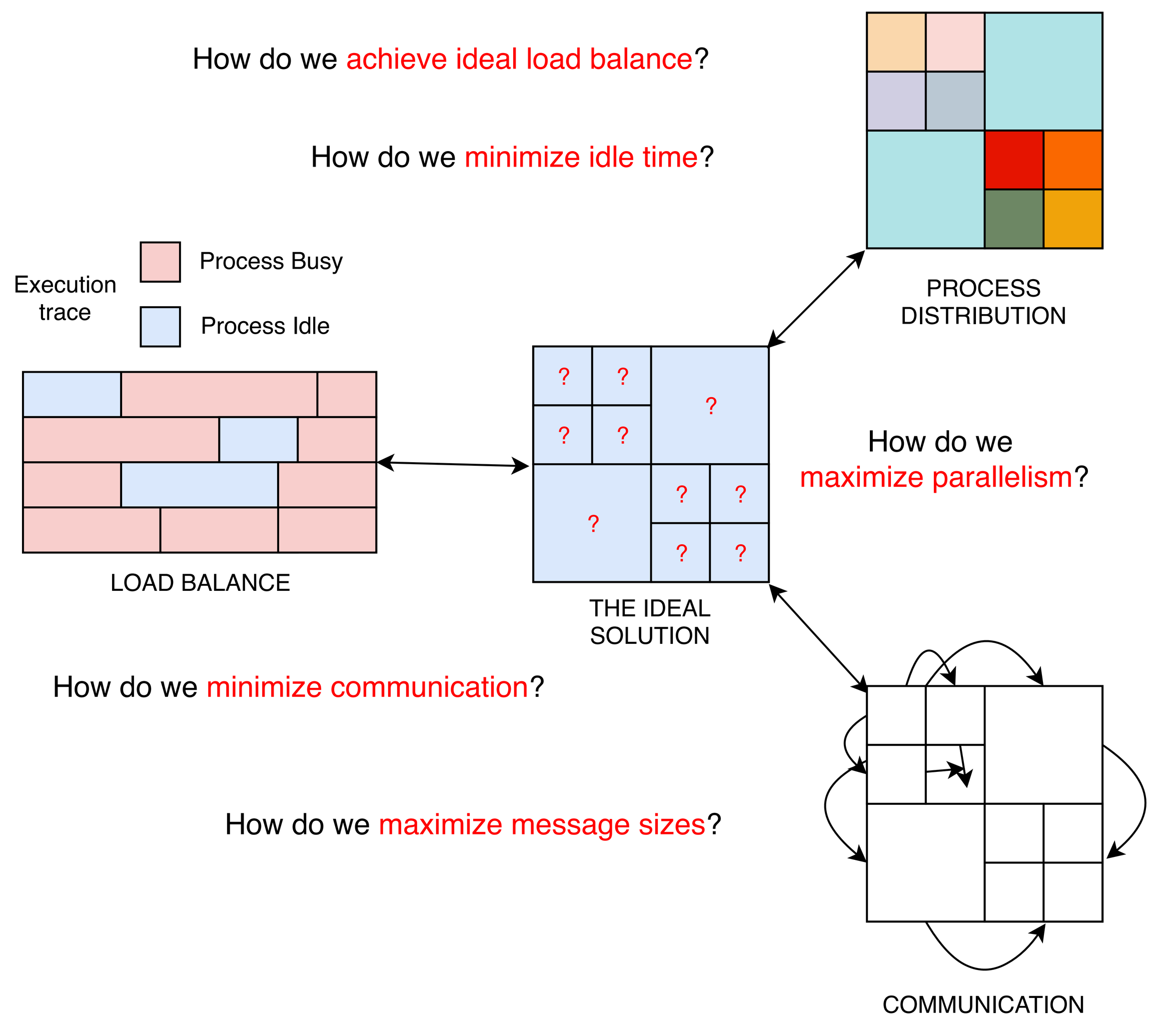

Figure 5: Trade-offs for low rank distributed LU factorization.

The problem of **load balance** can be solved with the use of **run time systems**.

Multi-Node Strong Scaling

Comparison of various dense LU factorization implementations against task-based BLR LU factorization. Executed using **single thread** on a **varying number of nodes**. `N=32k` and `NLEAF=1024`.

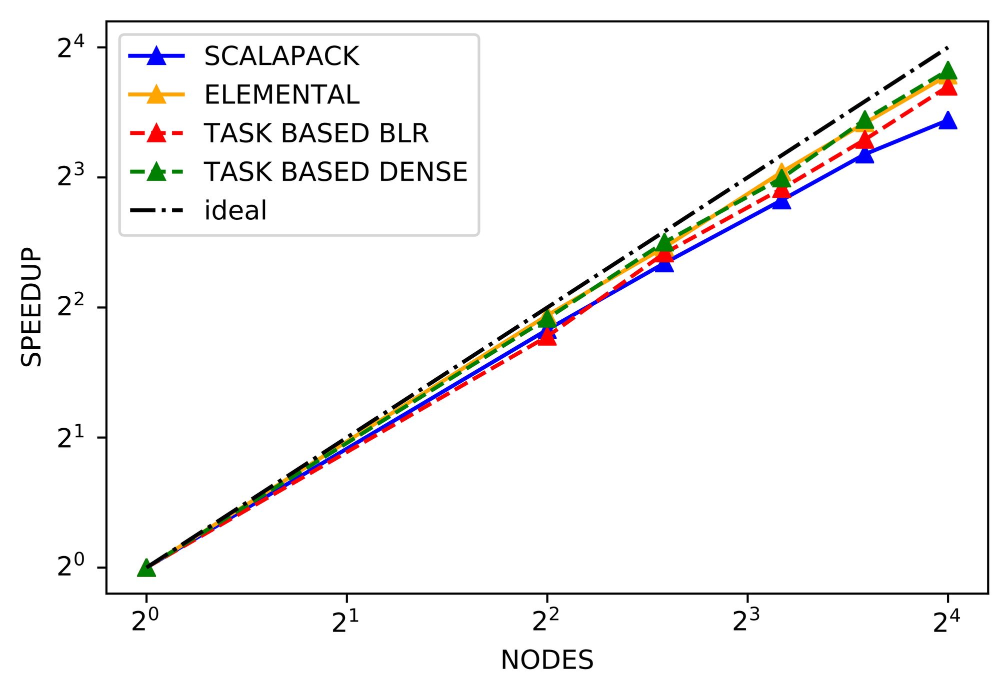

Figure 6: Speedup.

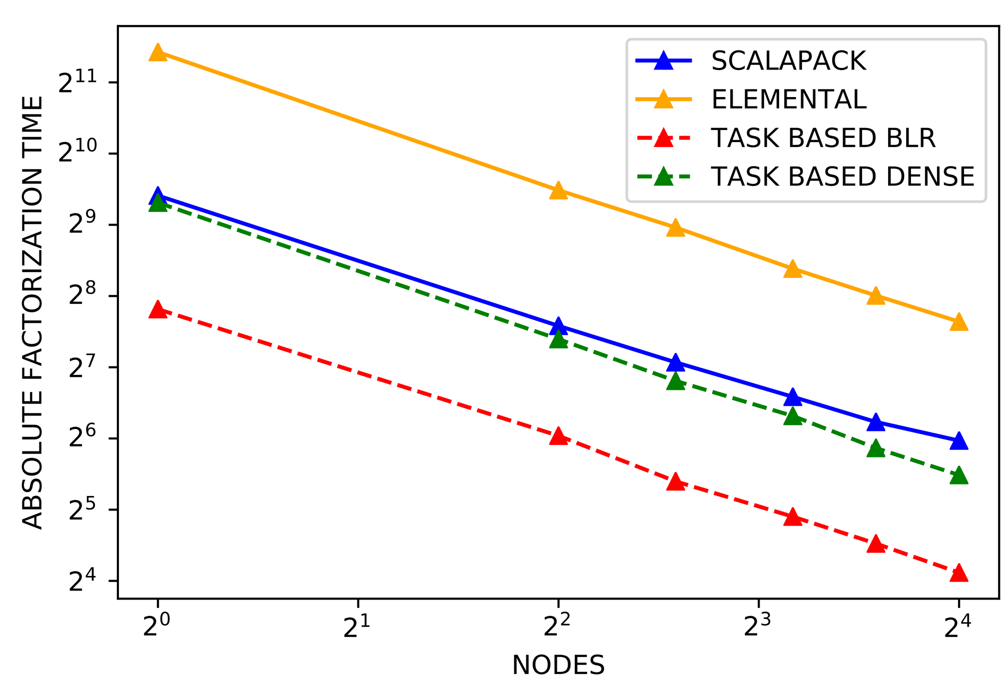

Figure 7: Absolute time.

Increasing number of nodes shows **good scaling** as a result of **more time** spent in **computation** than in communication and waiting (Figures 8 to 11).

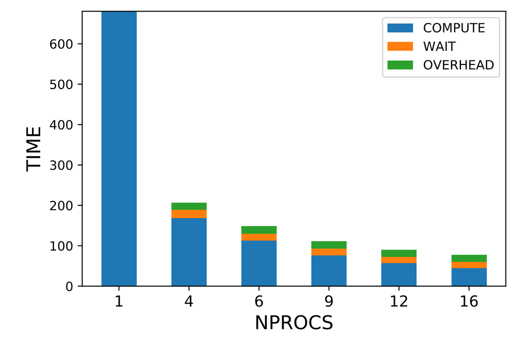

Figure 8: Scalapack

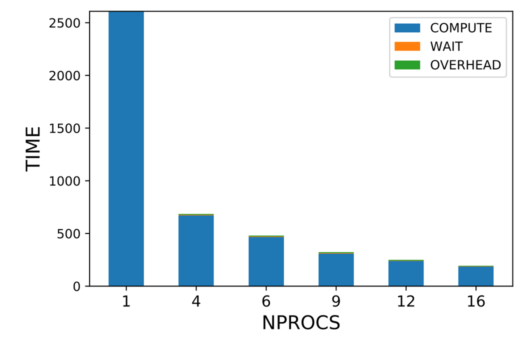

Figure 9: Elemental

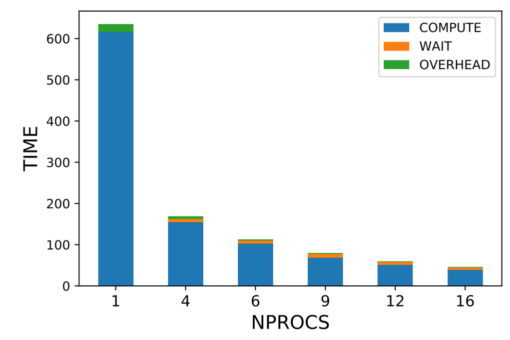

Figure 10: Task-based dense

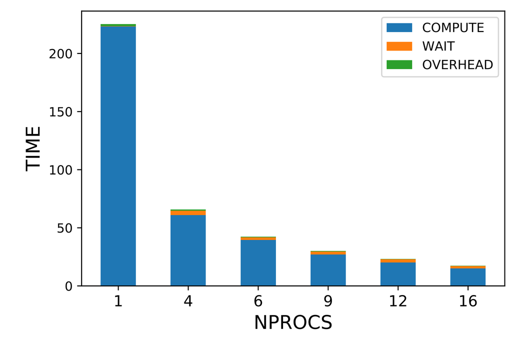

Figure 11: Task-based BLR

Single-Node Strong Scaling

Strong scaling for `N=131k` continues positively **until 25 threads** as can be seen in Figure 13. The scaling is **worse than dense** factorization due to **lack of compute intensity**.

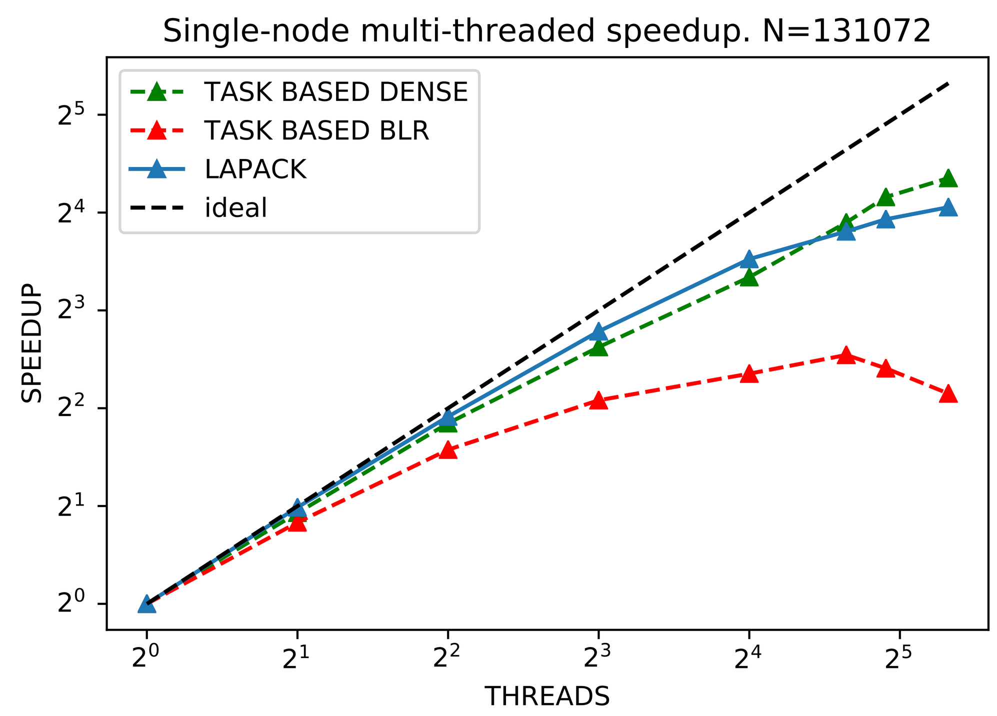

Figure 12: Speedup.

Single-Node Bandwidth Utilization

Bad scaling post 25 threads can be attributed to lack of computational intensity for a BLR matrix, leading to **lesser bandwidth utilization**.

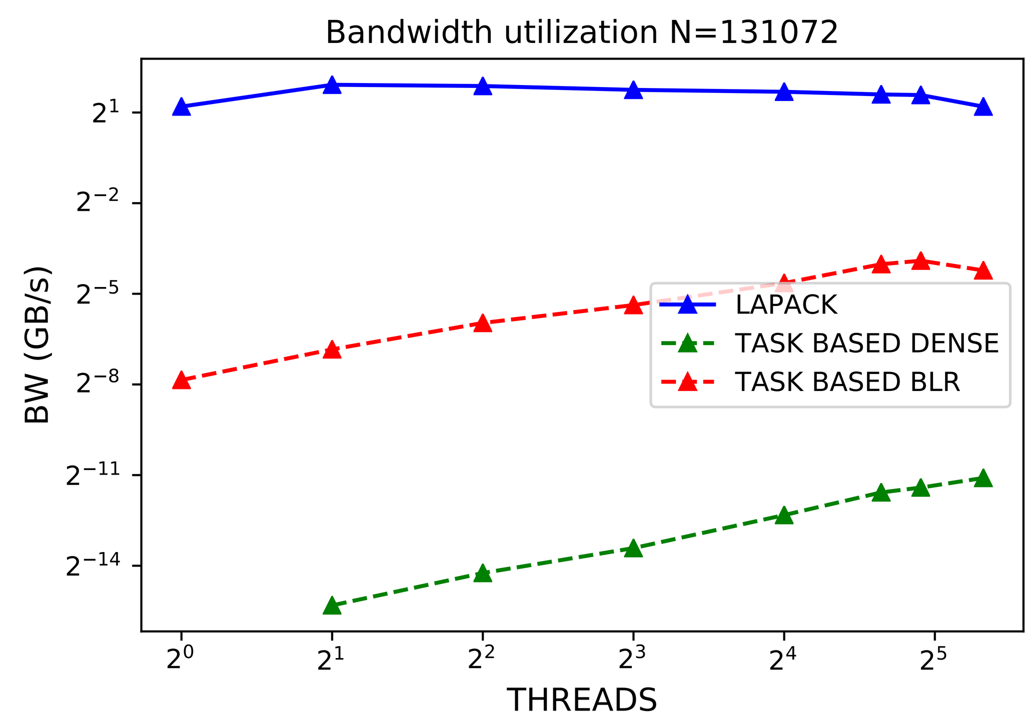

Figure 13: Single node bandwidth utilization.

Conclusion

Almost ideal speedup and **better time to solution** of BLR factorization is acheived on **single threaded, multi node execution** due to better scheduling by the run time system. However, the **speedup** of BLR factorization is **worse than dense** for single node, multi-threaded execution due to lack of compute intensity. This is **proven by poor utilization** of available **bandwidth**.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP18H03248, JP17H01749.