Distributed Memory Task-Based Block Low Rank Direct Solver

Sameer Deshmukh
deshmukh.s.aa@m.titech.ac.jp

Rio Yokota
rioyokota@gsic.titech.ac.jp

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 direct solver against other distributed dense matrix libraries.

Block Low Rank Matrix

The block low rank matrix allows us to express a large dense matrix as a collection of low rank and full rank blocks. The Low Rank Matrix is represented as a product of the Singular Value Decomposition (SVD) of the corresponding dense matrix.

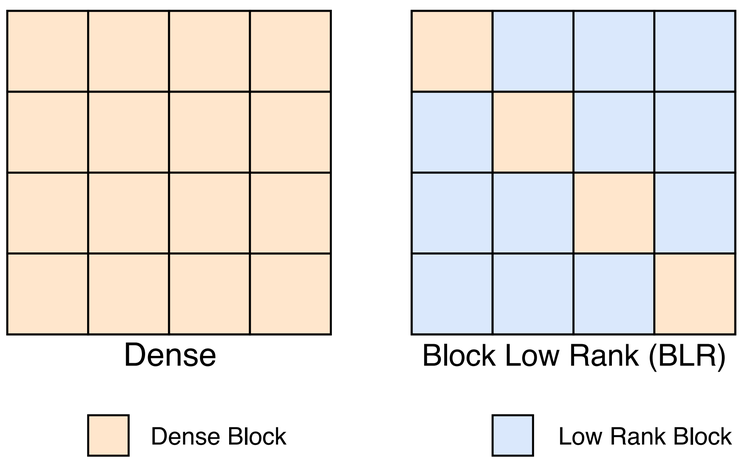

Figure 1: A block dense matrix can be expressed as a block low rank matrix.

LU Factorization

The LU factorization works by splitting a dense matrix into a lower and upper diagonal matrix.

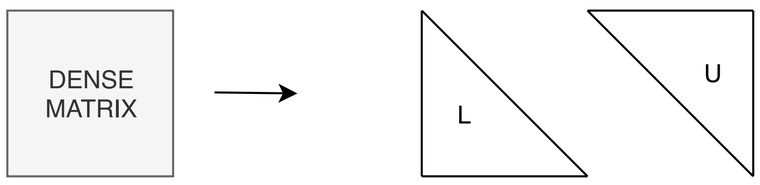

Figure 2: LU factorization of a simple dense matrix.

Blockwise LU decomposition

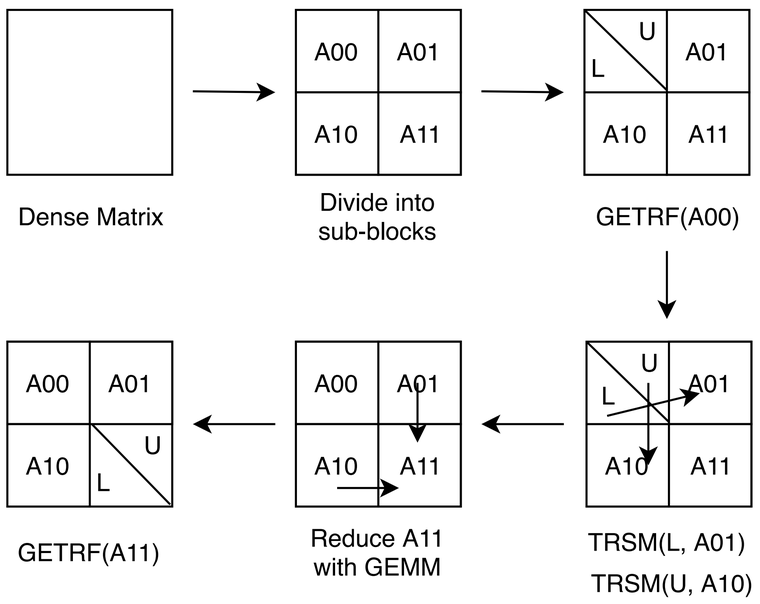

Figure 3: Block LU decomposition using Schur's compliment.

As can be seen in Fig. 3, the LU decomposition of a block dense matrix can be calculated as follows:

- Factor $A_{00} = L_{00}U_{00}$.
- Compute $A_{10} = A_{10}U_{00}^{-1}$ and $A_{01} = L_{00}^{-1}A_{01}$.
- Form the Schur's compliment $A_{11} = A_{11} - A_{10}A_{01}$.
- Factor $A_{11} = L_{11}U_{11}$

Problems with distributed LU

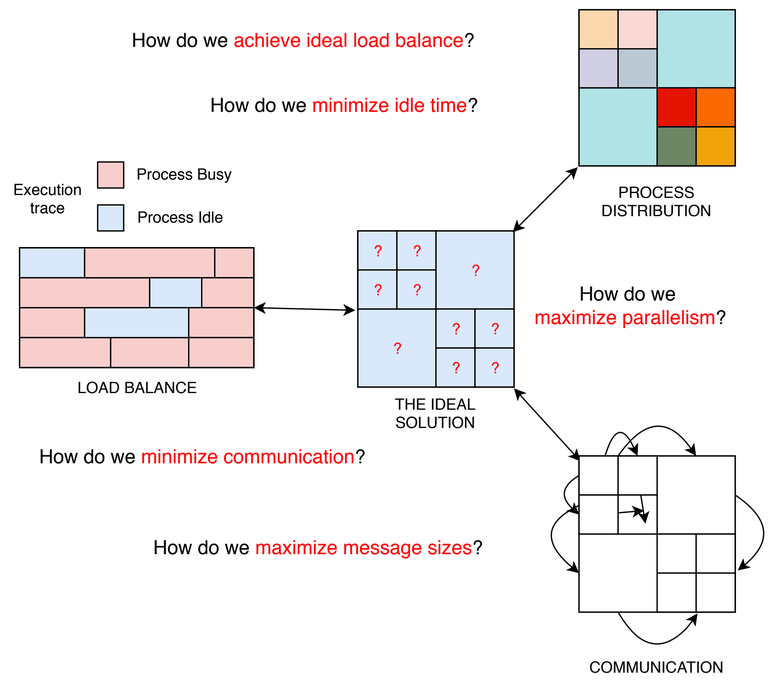

Figure 4: Illustration of the 5 trade-offs that are necessary to balance in order to effectively create a distributed HLU algorithm.

Runtime-Based Systems: starPU

A 'DAG engine' such as starPU [1] allows us to express a computation as a flow of data between nodes of a DAG and easily overlap computation and communication. StarPU executes this graph and has the ability to insert dependencies between tasks and run independent tasks in parallel.

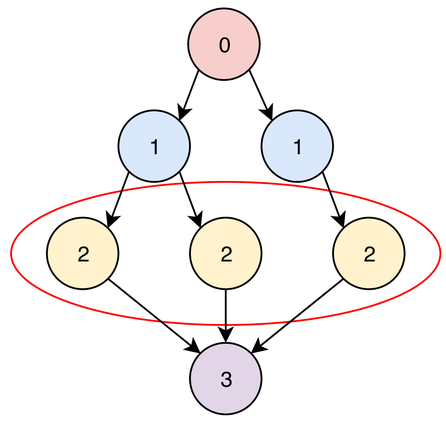

Figure 5: Computation expressed as a Directed Acyclic Graph. Same colored nodes are executed together.

Absolute Time

Elemental is the slowest while BLR is the fastest in terms of execution time.

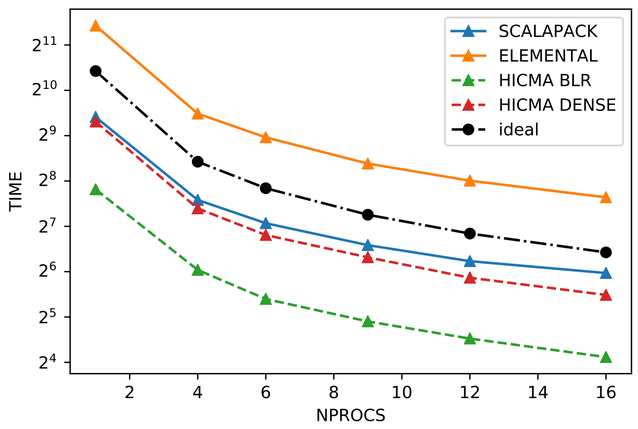

Figure 6: Scaling w.r.t nprocs of LU factorization for matrix of size 32k and nleaf 1024.

Process Efficiency

The process efficiency is a measure of how well the processes are utilized as we keep the problem size constant and scale the number of processes used. In this case we model the process efficiency using Eq. 1, where T_1 is the time taken by a single process, $T(N, P)$ is the time taken by P processes for a problem size N, and $E(N, P)$ is the efficiency. Fig. 7 shows comparison between the process efficiecies of various implementations.

$$E(N, P) = \frac{1}{P} \times \frac{T_1(N)}{T(N, P)} \quad (1)$$

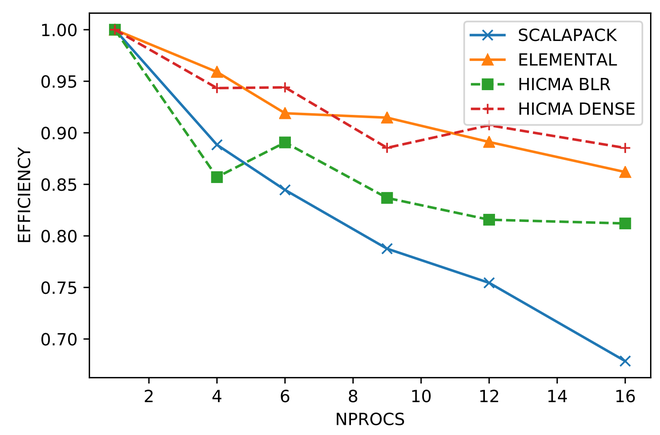

Figure 7: Process efficiency of LU factorization for contant leaf size.

Execution Profile

Execution profiles show a breakdown of the time spent in communication, computation and waiting for data to arrive.

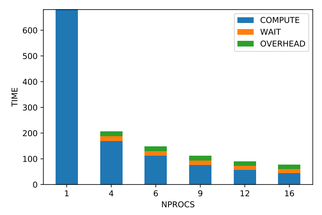

Figure 8: Scalapack

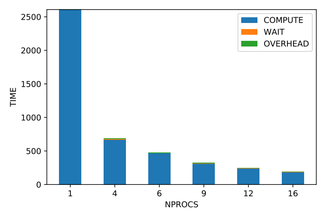

Figure 9: Elemental

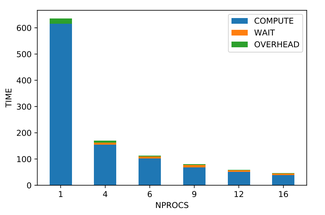

Figure 10: Task-based all dense

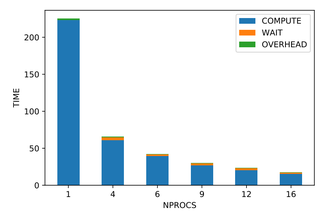

Figure 11: Task-based block low rank

Acknowledgement

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

[1] Cedric Augonnet, Olivier Aumage, Nathalie Furmento, Raymond Namyst, and Samuel Thibault. StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators. In Siegfried Benkner Jesper Larsson Trï£¡ff and Jack Dongarra, editors, *EuroMPI 2012*, volume 7490 of *LNCS*. Springer.