Distributed Memory Task-Based Block Low Rank Direct Solver

Sameer Deshmukh
School of Computing,
Tokyo Institute of Technology
Tokyo, Japan
deshmukh.s.aa@m.titech.ac.jp

Rio Yokota
Global Scientific Information Center,
Tokyo Institute of Technology
Tokyo, Japan
rioyokota@gsic.titech.ac.jp

1 INTRODUCTION

Large dense matrices with low rank structures appear in problems such as maximum likelihood estimation and boundary integral methods. The Block Low Rank matrix format reduces the time complexity of the LU factorization from $O(N^3)$ to ~$O(N^2)$ and the storage complexity from $O(N^2)$ to $O(N^{1.5})$. Increasing problem sizes demand efficient distributed memory distributions. In this work, we implement and benchmark a distributed asynchronous runtime-system based Block Low Rank direct solver and compare it against a fully dense task-based distributed solver, Scalapack and Elemental [2].

Dense matrices whose eigenvalues decay rapidly can be approximated by means of a low rank approximation by using an SVD decomposition and keeping only the k most significant eigenvalues of the matrix. If however, we have a large dense matrix whose sub-blocks consist of low rank and full rank blocks, we can approximate parts of the large matrix and get a considerable gain in terms of performance and storage costs with tunable accuracy. The Block Low Rank matrix format (Fig. 1) consists of Low Rank blocks in the off-diagonal parts and full rank blocks along the diagonal. We use the MPI interface of the starPU runtime system [1] for building a distributed memory task-based LU factorization routine.

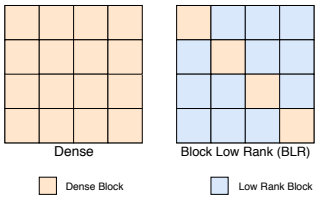


Figure 1: **Block dense matrix (left) vs. block low rank matrix (right). Notice that the diagonal elements are dense whereas off-diagonals are low-rank.**

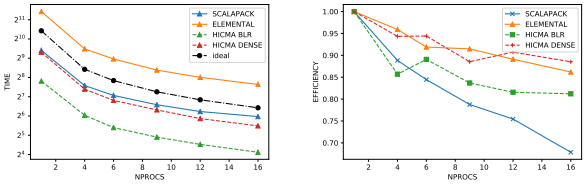
2 RESULTS

Experiments were conducted on the TSUBAME 3.0 supercomputer. Each node houses two Intel Xeon E5-2680 v4 (2.4 GHz, 14 core) processors. Each node runs a single process and single thread. We compare our dense and BLR distributed LU factorization implementations against Scalapack and Elemental. We make comparisons for a matrix generated from a gaussian distribution of total size 32768 and block size 1024. We use the *DMDA* scheduler from starPU.

2.1 Absolute time and process efficiency

Fig. 2a shows the comparison of absolute time of execution. Elemental is the slowest while task-based BLR is the fastest.

The process efficiency is a measure of how well the processes are utilized. In this case we model the process efficiency as $E(N, P) = \frac{1}{P} \times \frac{T_1(N)}{T(N,P)}$, 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. Fig. 2b shows comparison between the process efficiecies of various implementations.

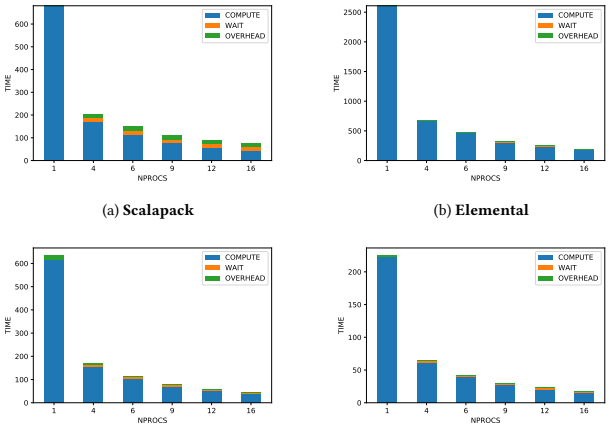


(a) **Absolute time.** (b) **Process efficiency.**

Figure 2: **Comparison of absolute time and process efficiency.**

2.2 Execution Profile

The execution profile of an application shows a breakdown of the proportion of time that the computation spends in actual computation, waiting for data and data transmission or runtime overhead. Obtaining such values helps in precisely targeting the most non-optimum parts of the application.



(a) **Scalapack** (b) **Elemental**

(c) **Task-based all dense** (d) **Task-based block low rank**

Figure 3: **Comparison of execution profiles.**

ACKNOWLEDGEMENTS

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

REFERENCES

[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 Tr ff and Jack Dongarra, editors, *EuroMPI 2012*, volume 7490 of *LNCS*. Springer.
[2] Jack Poulson, Bryan Marker, Robert A. van de Geijn, Jeff R. Hammond, and Nichols A. Romero. Elemental: A New Framework for Distributed Memory Dense Matrix Computations. 39(2):13:1–13:24.