Hierarchical Matrix Techniques on Massively Parallel Computers

Abstract

Mohammad Izadi

Hierarchical matrix ($\mathcal{H}$-matrix) techniques can be used to efficiently treat dense matrices. In fact, for a dense matrix $A \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$, the storage requirements and performing all fundamental operations, namely matrix-vector multiplication, matrix-matrix multiplication and matrix inversion can be done in almost linear complexity $\mathcal{O}(n \log^c n)$, where $n = |\mathcal{I}|$ and c is some moderate constant.

In this work our aim is to reduce even further these almost linear costs for $\mathcal{H}$-matrix arithmetic by exploiting parallelism. For large-scale computations, the platform of choice will be large-scale distributed-memory systems.

Prior to devising any parallel algorithm for $\mathcal{H}$-matrices one needs to determine $\mathcal{H}$-distribution schemes by which the underlying matrix will be distributed across the involved processors. Our approach towards an $\mathcal{H}$-matrix distribution relies on the splitting of the *index set* $\mathcal{I}$. We specify a block size NB as the number of contiguous indices from $\mathcal{I}$ to be assigned to a single processor.

Based on the index-wise $\mathcal{H}$-distribution and different block sizes NB, namely `BLOCK`, `CYCLIC` and `BLOCK-CYCLIC` strategies, we devise parallel algorithms for $\mathcal{H}$-matrix truncation, matrix-vector multiplication, matrix-matrix multiplication, and finally $\mathcal{H}$-matrix inversion. For the $\mathcal{H}$-matrix truncation, the parallel QR factorization is implemented in three different settings, namely Householder, Givens, and tall and skinny QR (TSQR). All three QR algorithms are scalable for a large matrix size, however the TSQR is also scalable for small matrices.

For the matrix-matrix multiplication *scheduling* is required on distributed-memory systems to minimize idling. We introduce three scheduling algorithms for the list of block sub-multiplications. These scheduling algorithms based on

- The cost of each individual block multiplication,
- The size of involved processor sets,
- The total cost of all block multiplication per processor set.

The numerical experiments for both $\mathcal{H}$-matrix truncation and matrix-vector multiplication show that they are highly scalable especially for the `BLOCK` distribution. Furthermore, for this kind of distribution, the $\mathcal{H}$-matrix matrix multiplication

is scalable with respect to increasing the number of processors and matrix sizes simultaneously.

For the $\mathcal{H}$-matrix inversion algorithm we have that it does not scale in the case of `BLOCK` distribution at all, partly due to a sequential inversion part corresponding to the diagonal and due to higher communication costs in the block Gauß elimination algorithm. Therefore, we consider the `BLOCK-CYCLIC` distribution which is a trade-off between idle time and total sequential work done by all individual processors along the diagonal. Our numerical experiments show that the inversion algorithm has limited scalability for a small block size.

We summarize the main results achieved in this work based on the index-wise $\mathcal{H}$-distribution:

- A highly scalable algorithm for the $\mathcal{H}$-matrix truncation and matrix-vector multiplication,
- A scalable algorithm for the $\mathcal{H}$-matrix matrix multiplication,
- A limited scalable algorithm for the $\mathcal{H}$-matrix inversion for a large number of processors.