CMSC416: Introduction to Parallel Computing

Topic: Parallel Algorithms
Date: March 7, 2024

Cache and Memory Management

In the realm of parallel algorithms, the constraints imposed by the finite number of cache lines in caches play a crucial role. This limitation necessitates thoughtful memory management, as cache filling requires the replacement of existing parts, potentially resulting in cache misses.

To address these challenges and improve cache performance, the concept of blocking is introduced. By creating smaller blocks that fit into the cache, it becomes possible to promote cache reuse. For instance, a matrix multiplication example in the slide illustrates the idea of blocking, emphasizing the reuse of specific columns or rows, such as A_{10}, throughout multiple computations.

Parallel Matrix Multiplication

Data and Work Distribution

When delving into parallel matrix multiplication, a critical decision revolves around how to distribute data and work effectively. The overarching strategy involves assigning different subblocks of the resulting matrix C to various computational nodes. This entails dividing matrices A and B, storing them in a distributed manner, and establishing communication channels between processes. Each process is then responsible for computing a specific portion of the resulting matrix C.

Cannon’s 2D Matrix Multiply

Cannon’s algorithm introduces a 2D virtual grid of processes and assigns sub-blocks of matrices A and B to each process. Each process takes charge of computing a distinct sub-block of the final matrix C. This method necessitates communication with other processes in both its row and column. There are two important steps in this method, initial skew and shifting by one. Initial skew involves displacing blocks in row i by i on matrix A and column j by j on matrix B. Shift by 1 involves moving all rows of matrix A one position to the left and all columns of matrix B one position up. These techniques help in organizing the communication between processes effectively. The time

complexity of Cannon’s algorithm is influenced by additional computations, emphasizing the impact of communication overhead.

Agarwal’s 3D Matrix Multiply

Agarwal's approach takes it a step further by arranging processes in a 3D virtual grid. Sub-blocks of matrices A and B are assigned to each process, and each process computes a partial sub-block of the resulting matrix C. Notably, data movement occurs only once before and after computation. This involves copying A to all i-k planes and B to all j-k planes, minimizing communication frequency. A single matrix multiplication is performed to calculate partial C. This stage focuses on local computations, reducing the need for inter-process communication during the main computation. The final result is calculated through an all-reduce operation along i-j planes. While the one-time data movement minimizes communication overhead during the computation, it comes at the cost of increased memory requirements.

Communication Algorithms

Reduction

Communication algorithms like reduction come into play, involving scalar and vector reductions. The MPI_Reduce function is employed to implement point-to-point operations. Reduction strategies include the naive approach, where each process sends data to a central root, and the spanning tree approach, which organizes processes in a k-ary tree, thereby optimizing communication efficiency.

All-to-All

In the context of all-to-all communication, each process sends a distinct message to every other process. The naive algorithm for this communication involves pairwise data exchange between processes.