COMP90025 Parallel and Multicore Computing

Project 1A Report

Hanbin Li *hanbinl1*
Wenqing Xue *wenqingx*

August 2019

Approach

The purpose of this project is to implement an OpenMP[1] program in order to achieve better performance in a parallel manner. In general, the sequential Floyd-Warshall algorithm which is provided as skeleton code, runs in $O(N^3)$ time, where N is the number of vertices in the graph.

Classic Floyd-Warshall Algorithm

Initially, the basic approach is quite straightforward, by applying the OpenMP compiler directives to certain for-loops in order to be executed in parallel. In the original Floyd-Warshall algorithm, the nested i and j for-loops are relatively independent but both depend on k, therefore the i for-loop can be parallelized for all inner iterations. In this case, the iterations can be distributed to a number of threads by one master thread and the variables *Distance* and k will be declared as shared variables for each thread since they all use the same copy of those two variables. In this case, this approach has a runtime of $O(\frac{N^3}{P})$, where P indicates the number of processors.

Tiled Floyd-Warshall Algorithm

The second approach is a tiled version Floyd-Warshall algorithm implemented based on the classic Floyd-Warshall but applied partition to split the computations between the processes, so that each process is assigned to a specific tile block of the matrix [1]. The details of the tiled Floyd-Warshall algorithm are described as follow: Initially, the input matrix is divided into tiles of size B (block-size). During the k^{th} iteration, the k^{th} independent diagonal tile is updated first (while tile in Figure 1). After the algorithm updates the diagonal tile, the row and column tile blocks of k^{th} will be updated simultaneously (dark grey tiles in Figure 1). When all the row and column tiles are updated, the remaining tiles in other directions will be eventually updated (light grey tiles in Figure 1). By computing the matrix this way, all dependencies are satisfied accordingly, and the pseudo code is shown below.

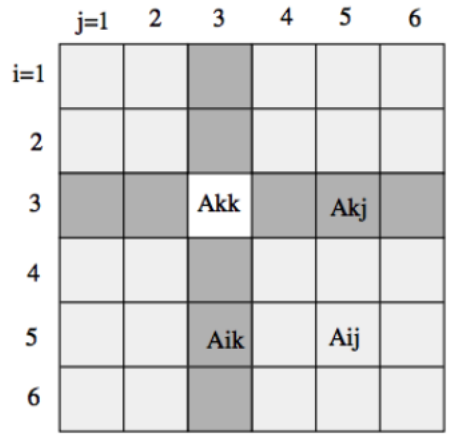

Figure 1: Three Phases of Tiled Floyd-Warshall

Algorithm 1 Tiled Floyd-Warshall Algorithm

```
1:  procedure TiledFW(A, N)
2:      for k ← 0 to N − 1 step B do
3:          FW(A_kk)                          ▷ Diagonal
4:          for i ← 0 to N − 1 step B do
5:              FW(A_kj)                      ▷ Row
6:              FW(A_ik)                      ▷ Column
7:          end for
8:          for i ← 0 to N − 1 step B do
9:              for j ← 0 to N − 1 step B do
10:                 FW(A_ij)                  ▷ Rest
11:             end for
12:         end for
13:     end for
14: end procedure
```

As the input matrix is divided into tiles of size B, there are $\frac{N}{B}^2$ tiles and $\frac{N}{B}$ iterations in total. All the blocks are updated through each iteration and that requires to hold 3 tiles in memory, which is a total of $(3 * B^2)$ nodes. The total amount of memory is calculated by total number of iterations times the total number of tiles accessed in each iteration and then times the number of memory access in each

[1] https://www.openmp.org

tile update, which is a function of $O(\frac{N^3}{B})$. Therefore, the total amount of processor memory access is reduced by a factor of B, which is optimal for cache locality of the processor. The performance is increased as reads and writes to memory became faster.

Conclusion

To sum up, the performance of speedup between two algorithms is shown in Figure 2. With the increase of threads, tiled approach can perform much better than classic approach due to the fact that total work is essentially distributed into each thread, and each thread is responsible for their own blocks.

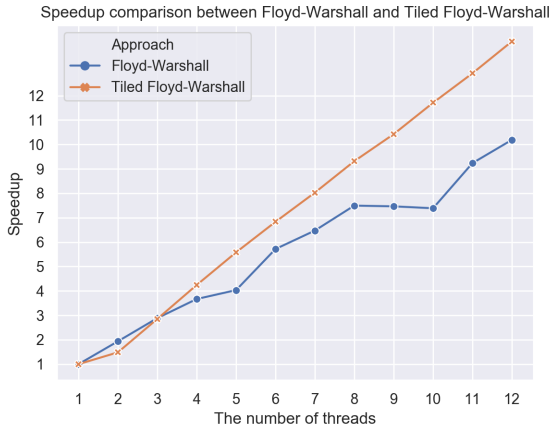

Figure 2: Speedup comparison with $N = 5000$

ILLIAC

The ILLIAC systems, a series of supercomputers represent a historical step forward in computer system architecture using parallelism. Trace back to 1952, the first ILLIAC generation was built with von Neumann architecture that data operation and instruction fetch could not perform at the same time. The ILLIAC II was the first pipelined supercomputer with implementation of instruction level parallelism within a single processor. Unlike previous systems, The ILLIAC IV achieved data level parallelism by using SIMD pattern and became the world's first parallel supercomputer. Beyond using instruction-level parallelism to divide incoming instructions to process different parts of instructions in parallel, the ILLIAC IV worked well on large data sets by using a single control unit to broadcast instructions to many processing elements. For this project, our current program uses gcc -O3, which allows auto-vectorization by default and we uses OpenMP pragmas to achieve thread-level parallelism. Therefore, it might not be suitable for running on any of the ILLIAC computers. However, if we use SIMD to launch user-mandated vectorisation of the loops when the auto-verctorization could not be done, it will be suitable for running on ILLIAC IV to achieve data-level parallelism.

References

[1] Parallelizing the floyd-warshall algorithm on modern multicore platforms: Lessons learned students of the parallel processing systems course.