Calculating terrain parameters from Digital Elevation Models on multicore processors

Grethell Castillo Reyes and Dirk Roose

UCI, Geoinformatics and Digital Signals Center, Havana, Cuba and KU Leuven, Department of Computer Science, Scientific Computing Research Group, Belgium

Background

- The calculation of terrain parameters from Digital Elevation Models (DEMs) is an important task in Geographic Information Systems and Digital Terrain Analysis.
- Large size models may condition the cost of storage in terms of memory used, the latency of the network transfer time and the execution time of the algorithms used for its analysis.
- In several of those algorithms, stencil kernels are the dominant part of the computation, and an efficient parallel implementation of the kernel is therefore crucial in order to reduce the time to solution.

Objective

To evaluate the performance of an OpenMP algorithm and a code generated by PATUS, an autotuning framework for stencil computations.

Parallel Algorithms

1. We implement on a multicore processor two variants of the algorithm to calculate the slope of the terrain represented in a DEM. The first one uses OpenMP and the second one uses PATUS to efficiently use cache memory on stencil operations.
2. We used row decomposition to implement the parallel algorithm using OpenMP. We assume decomposition of the data matrix in horizontal sub-domains.
3. Each core $P_i, i = 0, 1, 2, ..., p-1$, receives a block with $\frac{n-2}{p} \times m$ grid points. Hence we assume that $n-2$ is a multiple of p. If this not the case, the assignment of rows to cores has to ensure a minimal work load unbalance.
4. The second variant was implemented using PATUS, a framework for stencil computation. Using PATUS, we define an stencil kernel using a C-like syntax to perform the stencil operation.

Experiments and Results

Hardware configuration

Cluster	Thinking
Type of node	haswell
CPU type (Xeon)	E5-2680v3
Interconnect	IB-FDR 6
# nodes	48/96
# sockets/node	2
# cores/socket	12
# cores/node	24

1. The slope was calculated using single precision floating point numbers on 1 to 12 cores.
2. Grids of size 16000×16000 were used.
3. The Basic Threading strategy of PATUS parallelizes the stencil computation without doing any blocking optimization. The consecutive blocks are assigned to threads with consecutive threads IDs.
4. In order for the autotuner (Basic Threading + SSE + Autotuner) to optimize the performance, the code generator also creates a benchmark harness.
5. The Basic Threading strategy of PATUS was used.
6. In case of PATUS, row decomposition leads to better performance, measured in aggregate number of operations per second Figures 1 and 2.
7. In all cases the PATUS variant shows much better performance. Figure 3 shows the elapsed time for grids of different sizes in function of the number of threads (= cores).

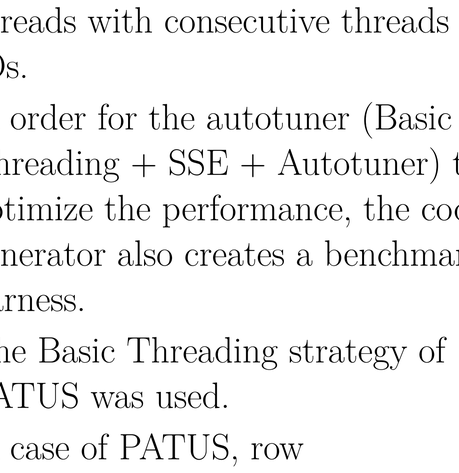

Figure 1: Performance results for the slope calculation with 4000×4000 grid points using PATUS with different blocksizes and one threads per core.

Expriments and Results

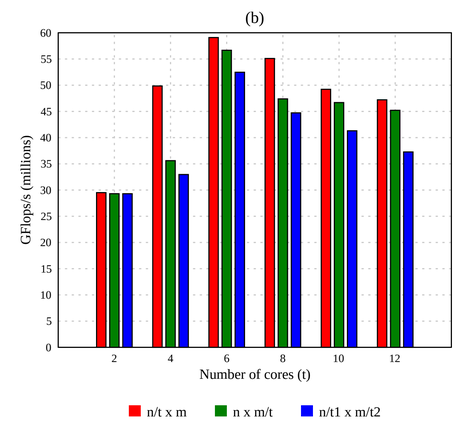

Figure 2: Performance results for the slope calculation with 4000×4000 grid points using PATUS with different blocksizes and two threads per core.

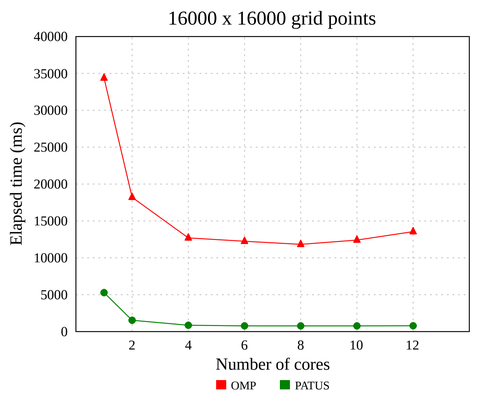

Figure 3: Performance results comparison for the slope calculation using OpenMP and PATUS with a DEM of size 16000×16000.

We performed performance benchmarks using the Basic Threading, the SSE (Basic Threading + SSE) and the autotuned code (Basic Threading + SSE + Autotuned) applied to the stencil kernel describe above. Results given in Figures 4 show that:

a) The aggregate number of operations per second increases with increasing number of threads (= cores), until an optimal number of threads is used. Using more threads leads to a lower performance and an increasing wall clock time.

b) For this stencil operation, the PATUS Basic Threading + SSE strategy does not lead to a better performance compared with the Basic Threading strategy.

c) In all cases the autotuner leads to the best results, by finding the best configuration of blocks with minimal running time. The differences with the performance of the PATUS Basic Threading strategy is the largest for grids of size 8000×8000.

Experiments and Results

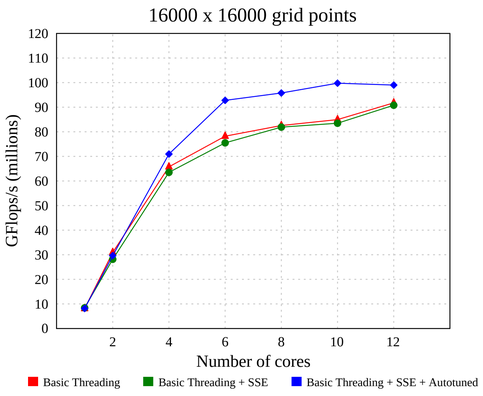

Figure 4: Performance results for the slope calculation with a DEM of size 16000×16000 using PATUS with different strategies.

Important results

Results on Figure 5 show that:

1. Using the Basic Threading option, results in 5 to 19 times faster execution than the program parallelized using OpenMP.
2. The larger the number of cores, the larger the gain obtained by using PATUS.
3. Hypertreading results in a performance gain of 15 to 50%, when compared with a single thread per core.

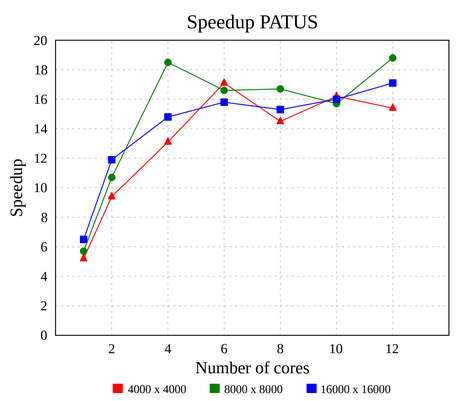

Figure 5: Performance results for the slope calculation with a DEM of size 16000×16000 using PATUS with different strategies.

Conclusion

The results show that the parallel software generated by PATUS, results in much lower execution times than the algorithm implemented using OpenMP. Using autotuning in PATUS leads to better performance than using the Basic Threading option in PATUS, especially for DEMs of medium size (8000×8000).

Contact Information

- Email: gcreyes@uci.cu
- Phone: +32 493707719