Maximum Subarray Problem 2D

Dmitri Gribenko <gribozavr@gmail.com>
Alexander Zinenko <ftynse@gmail.com>
National Technical University of Ukraine "Kiev Polytechnic Institute"

November 2011

Code description

We implemented an adaptive solution that uses Takaoka algorithm [1] to recursively divide a large problem into smaller problems that can be solved in parallel. Small problems ($N, M < 256$) are handled with Kadane 2D algorithm (described in [2]) which is faster for small inputs.

At the heart of Takaoka algorithm lies a modified matrix multiplication algorithm that constitutes the biggest part of the execution time, thus being a good candidate for optimization:

- we derived a block variant of this algorithm to improve cache hit ratio (approx. 13% time reduction);
- we fused this algorithm with the next step thus reducing memory writes (approx. 47% time reduction);
- we vectorized the algorithm with SSE2 and SSE4.2 (approx. 51% time reduction);
- we implemented 32 and 64-bit variants of the algorithm (32-bit variant could be used when upper bound on the sum — the sum of all positive array elements — is less than `INT32_MAX`), but 32-bit variant performance turned out to be comparable to the 64-bit variant.

Takaoka algorithm was parallelized with `tbb::parallel_invoke`. On a 4-core Intel i5-750 CPU the speedup was 3.6 – 3.8, but on a 80-core MTL machine speedup was around 5 and many cores were not loaded at all. Tracing has shown that Takaoka algorithm's natural task parallelism didn't create enough subtasks for all cores. After we parallelized matrix multiplication with `tbb::parallel_for` speedup increased to around 78.

Figure 1 shows speedup that was measured on a 40-core MTL worker node. Speedup is close to linear.

References

[1] Tadao Takaoka. 2002. Efficient Algorithms for the Maximum Subarray Problem by Distance Matrix Multiplication. *Electronic Notes in Theoretical Computer Science 61.*

[2] Kyoko Fukuda and Tadao Takaoka. 2007. Analysis of air pollution (PM10) and respiratory morbidity rate using K-maximum sub-array (2-D) algorithm. In *Proceedings of the 2007 ACM symposium on Applied computing* (SAC '07). ACM, New York, NY, USA, 153-157. DOI=10.1145/1244002.1244041 `http://doi.acm.org/10.1145/1244002.1244041`

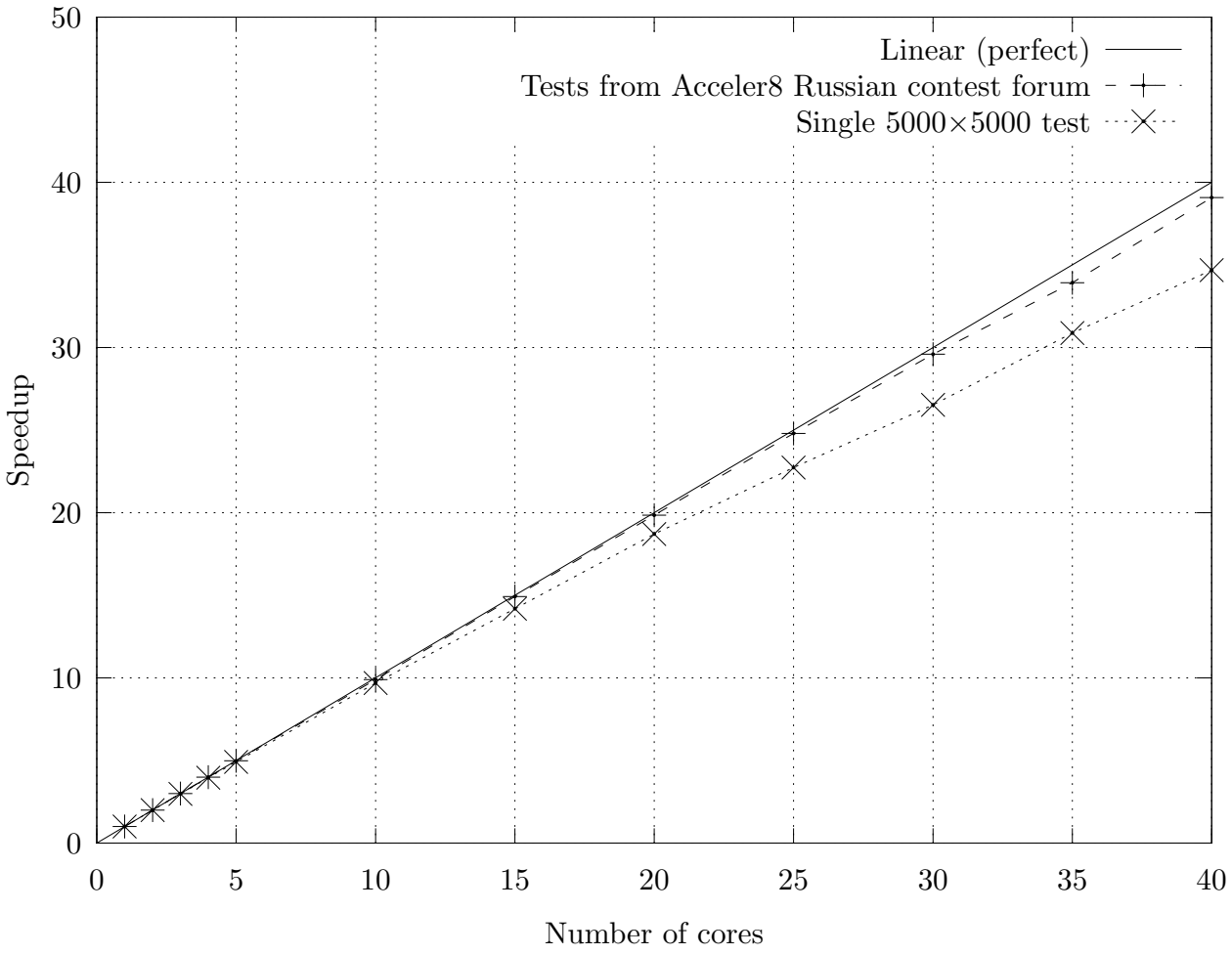

Figure 1: Measured speedup for different inputs on a 40-core machine