Sample Sort with TBB

Damir Ferizovic, David Vogelbacher

Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. In this paper we are going to compare our implementation of the parallel sample sort algorithm using the Intel TBB framework with the standard STL algorithm.

1 Algorithm

Sample sort is a parallel sorting algorithm that only distributes the input onto the the different threads once and then all threads sort their elements locally. The end result is then just the concatenation of the local results. Let p be the amount of available threads. In order to split the input, $p-1$ pivot elemens are selected and the elements of the input are grouped into p groups, depending on in between which pivot elements they lie (similiar to quicksort). Then, each thread sorts on of the groups. In order to get good pivot elements, oversampling is used. Initially, more than $S*p$ elements are selected randomly as possible pivot elements for a constant S. These elements are then sorted (sequentially, because the amount is still small). After that, every Sth element is selected as pivot. This oversampling helps finding good pivot elements with high probability, such that the resulting groups are of similar size.

2 Inplementation Details

2.1 Pivot selection

2.2 Grouping of elements

2.3 Local sorting

3 Experimental Results

Your hardware.

What do you benchmark.

Running time 1 and speedup plots 2 (for each generator, 64-bit integer and 32-bit floating point (not for non-comparative integer sorting algorithms).

Interpretation.

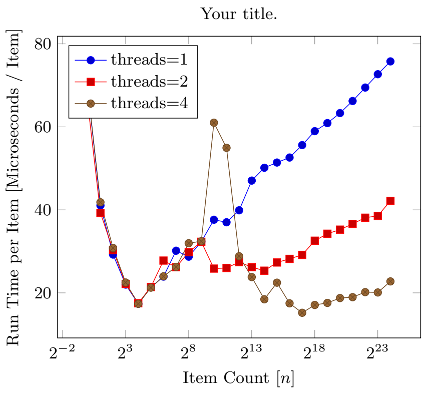

Fig. 1. Running times of `std::sort` with uniform input. Mean of 49 iterations.

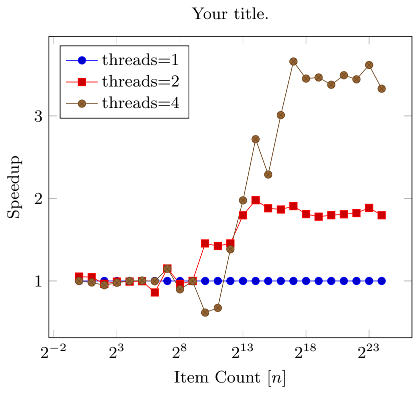

Fig. 2. Speedup of `std::sort` with uniform input. Mean of 49 iterations.

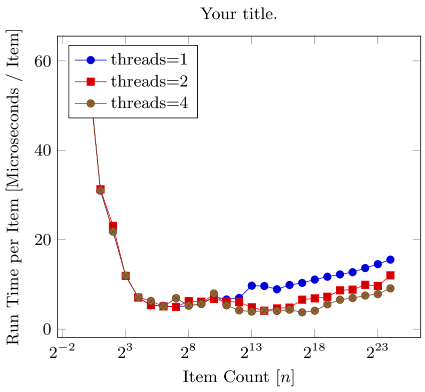

Fig. 3. Running times of `std::sort` with zero input. Mean of 49 iterations.

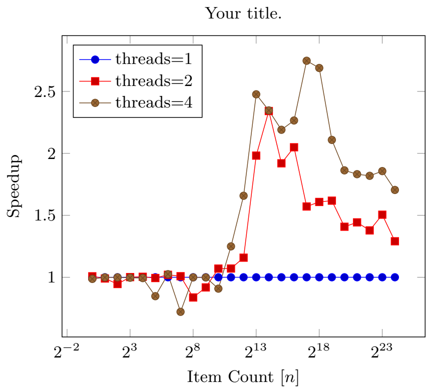

Fig. 4. Speedup of `std::sort` with zero input. Mean of 49 iterations.