Carsten Hood
CSCE 221H-200

Lab Report
Performance Evaluation of STL Sort

Introduction

This report outlines the setup and results of the scalability tests of the Sort algorithm in the Standard Template Library. The Sort algorithm rearranges the elements in a range based on two input iterators into ascending order.

Theoretical Analysis

The complexity of STL Sort is linearithmic, O(n*log(n)), where n is the input size, the number of elements in the range. More than linear time is required, as elements must be rearranged, which depends on the initial disorder of the range.

Experimental Setup

The experiments described in this report were performed on a Mac computer running OS X 10.9.1 with a 2.7 GHz Intel Core i7 processor and 8GB RAM memory. C++ code was compiled using Apple LLVM 5.0.

The standard library ctime was used to time the operations. Executions were repeated and averaged to compensate for the clock accuracy as necessary; for small input sizes, many thousands of executions were averaged.

To test the running time of STL sort, a vector of doubles was used as input. The vector was initialized with a set capacity for each test, the first at 2 and the last at 67108864 elements. In each test the vector was repeatedly populated with random values and then sorted, with multiple executions being averaged. The time taken to populate the vector by the Generate algorithm is linear, O(n) and the following operation, STL Sort, takes O(n*log(n)) time. O(n) and O(n*log(n)) can be taken as O(n*log(n)), and so Generate can be mostly disregarded.

Experimental Results

Figure 1 shows the running time of STL Sort vs the input size n on a log-log plot. The algorithm appears to scale somewhat linearly and does not necessarily appear linearithmic as expected. This may be due to the influence of the linear Generate algorithm.

Figure 2 shows Time/Expected Time, or Time/n*log(n), vs. size on a log-x plot. This plot reveals the constants C and n0 from the equation: 0 <= f(n) <= c*g(n) for all n >= n0. From the plot, C is 5.005E-09 and n0 is 16. After this pair (n0, C), all points fall beneath the line y = C. The red and blue arrows mark C and n0.

Figure 3 shows the line represented by C*Expected Time, or 5.005E-09 * (n*log(n)), in red, over the running time of STL Accumulate, in blue. Where the lines intersect at n = 16 represents n0.

Summary

This report shows an experimental performance evaluation of the STL Sort algorithm. Its O(n*log(n)) complexity is demonstrated through experimental tests, with asymptotic constants and n0 shown.

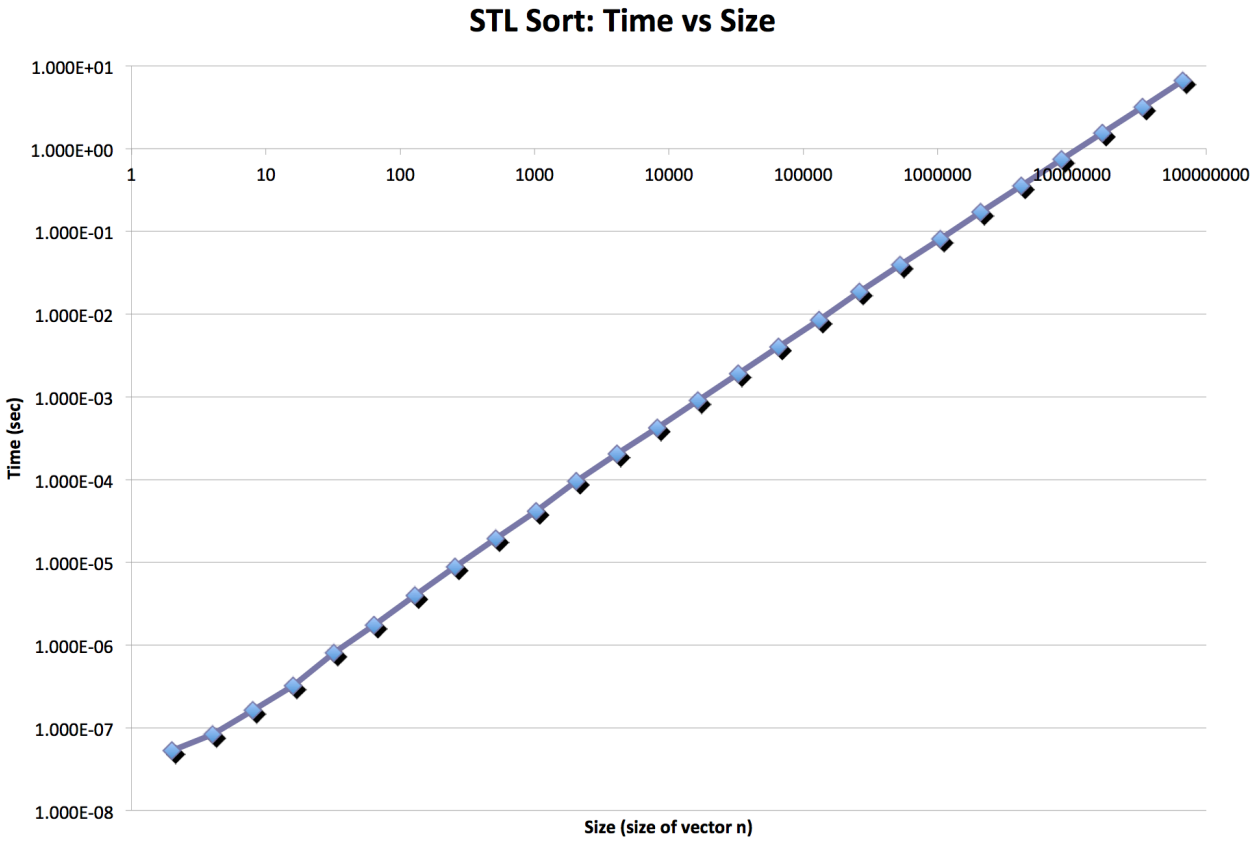

Figure 1: Execution time vs input size for STL Sort on a log-log plot.

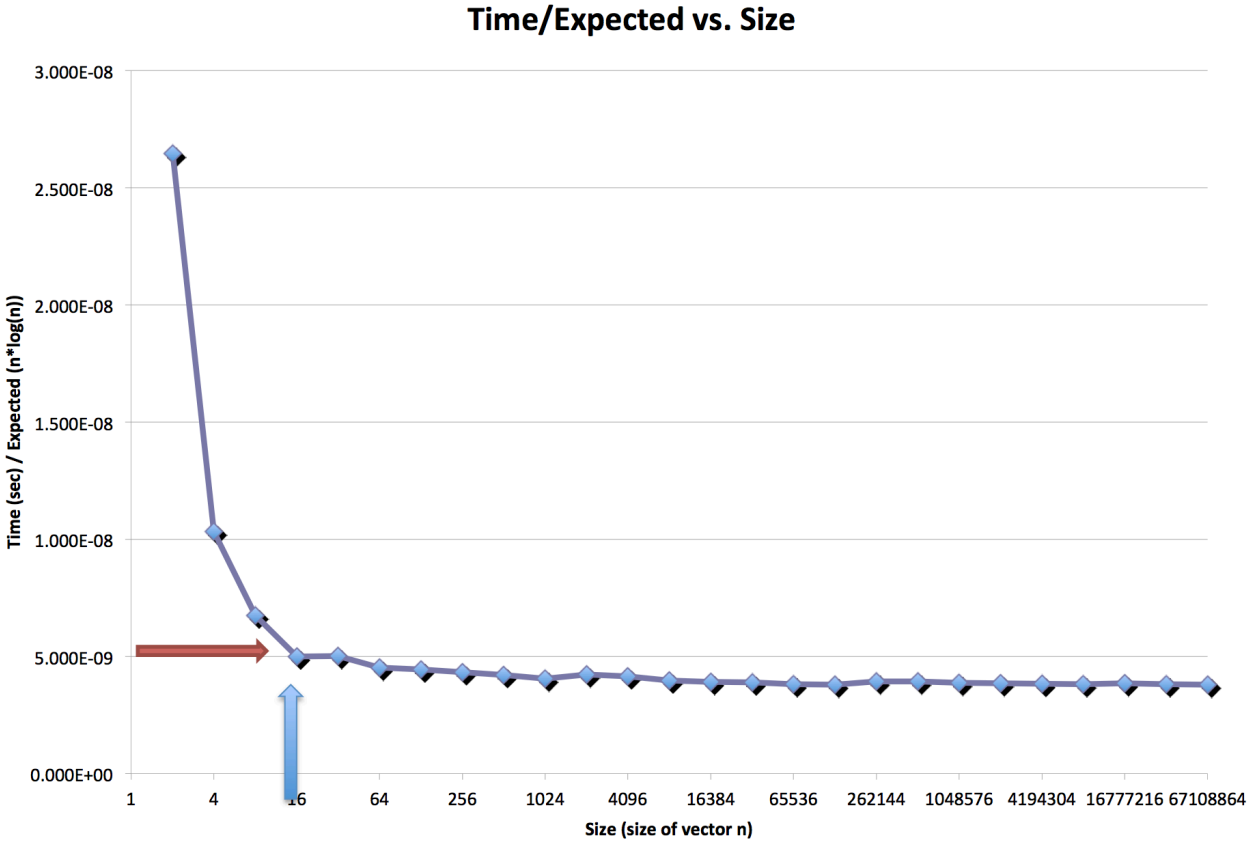

Figure 2: Time/Expected Time (n*log(n)) vs Size for STL Sort on a log-x plot.
The Big-O pair (n0, C) is marked by blue and red arrows.

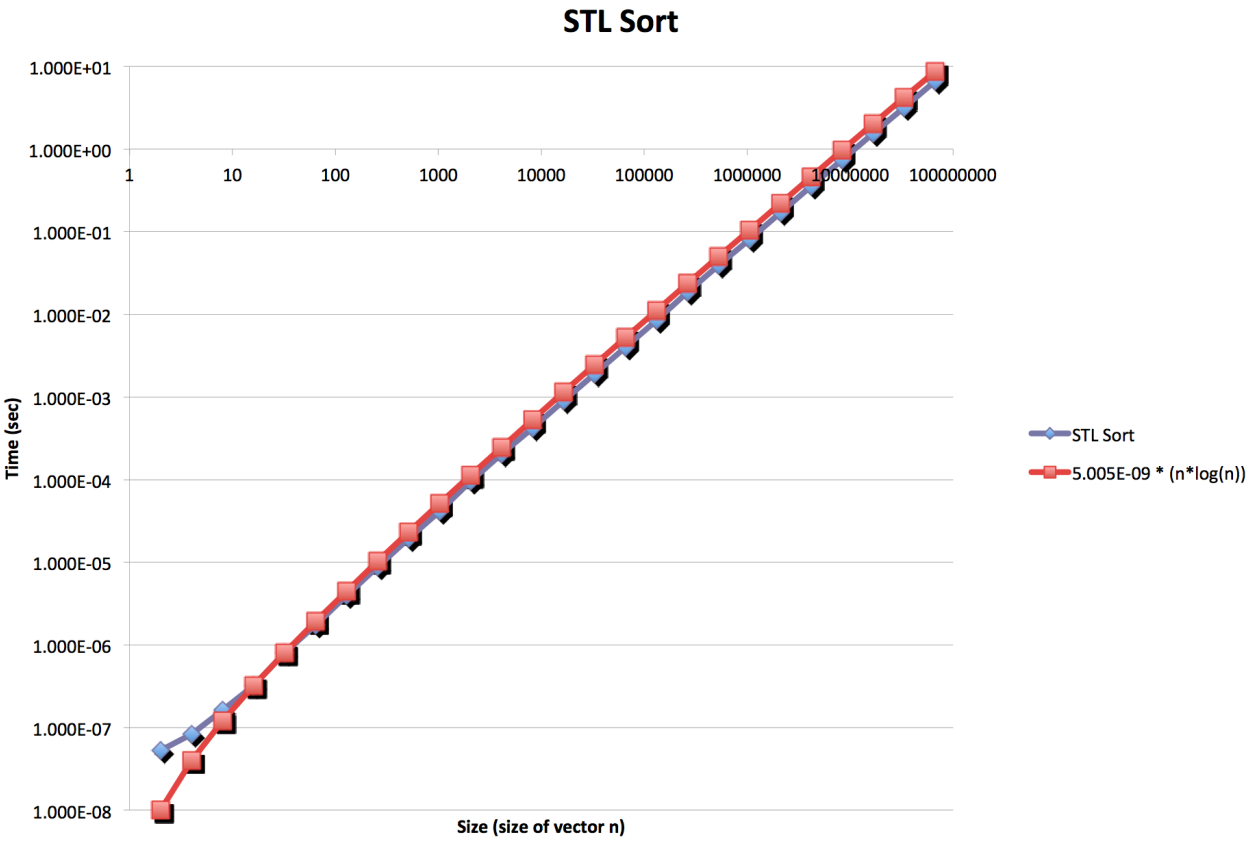

Figure 3: Execution time vs input size for STL Sort (in blue) on a log-log plot. Also shown in red is C*Expected Time, or 5.005E-09 * (n*log(n)).