In the zip file you can see 2 folders

Original : Have the full code the benchmark and the test\_stress\_sort in the benchmark\_folder

Modified : Have the parallel version with std::sort and std::stable\_sort. This folder don't have the code of introsort, merge\_sort and test\_stress\_sort

All the code is in the src/countertree/util/sort. The code is running in the namespace boost::countertree. This code is a preliminary version, done for my counter tree library. If you consider useful, I will change the name space and document the code, the algorithms and the ideas obtained from the benchmarks.

The time measure is done with a thin layer of chrono functions using the steady\_clock.

In the benchmark/GCC/util\_sort/ are the benchmark programs. All the programs must be compiled with the openMP and pthread options. The benchmark\_sort\_03.cpp, benchmark\_sort\_04.cpp, benchmark\_sort\_05.cpp, must be linked with the TBB library.

benchmark\_sort\_03.cpp : sort 200 000 000 uint32\_elements, in different sets ( sorted, reverse sorted, generated with rand() function, with rand() / ( 100000000), with rand() % 10000, and equals ).

benchmark\_sort\_04.cpp : sort 100 000 000 uint64\_elements, in different sets ( sorted, reverse sorted, generated with rand() function, with rand() / ( 50000000), with rand() % 10000, and equals ).

benchmark\_sort\_05.cpp : sort elements of different sizes ( 8, 16, 32, 64, 128 and 256 bytes), in different sets as in the previous benchmarks. ( This benchmark need around 20 min for the execution)

benchmark\_sort\_indirect.cpp : benchmark comparing the parallel\_sort and parallel\_merge\_sort with the indirect\_parallel\_sort and the indirect\_parallel\_merge\_sort, with elements of different size

test\_stress\_sort.cpp : this program have a big vector of 100 000 000 elements uint64\_t. The program make parts of the same size for each HW core. On my computer 4 parts and 4 cores. The program execute the part sequentially checking the time, and after do the same but in parallel, each thread sort a part, and check the time. This is done with the std::stable sort and countertree::merge\_sort. The data input is the same for all.

In GCC, an example of the command line used for to compile the program is:

g++ -std=c++11 -Wall -fexceptions -fopenmp -fomit-frame-pointer -fexpensive-optimizations -O3 -I../../../../src -c benchmark\_sort\_03.cpp -o benchmark\_sort\_03.o

g++ -o bin/Release/benchmark\_sort\_03 benchmark\_sort\_03.o -pthread -fopenmp -s -lpthread -ltbb

Francisco Tapia

fjtapia@gmail.com