Ali Panahi
Order Book Programming Problem
January 26, 2018

How to Build
```
$ cd PROJECT_PATH/bin
$ cmake .
$ cmake --build . --target all -- -j 4
```

How to run using file input:
$ ./pricer *target-size input-file*

How to run using console input:
$ ./pricer *target-size*

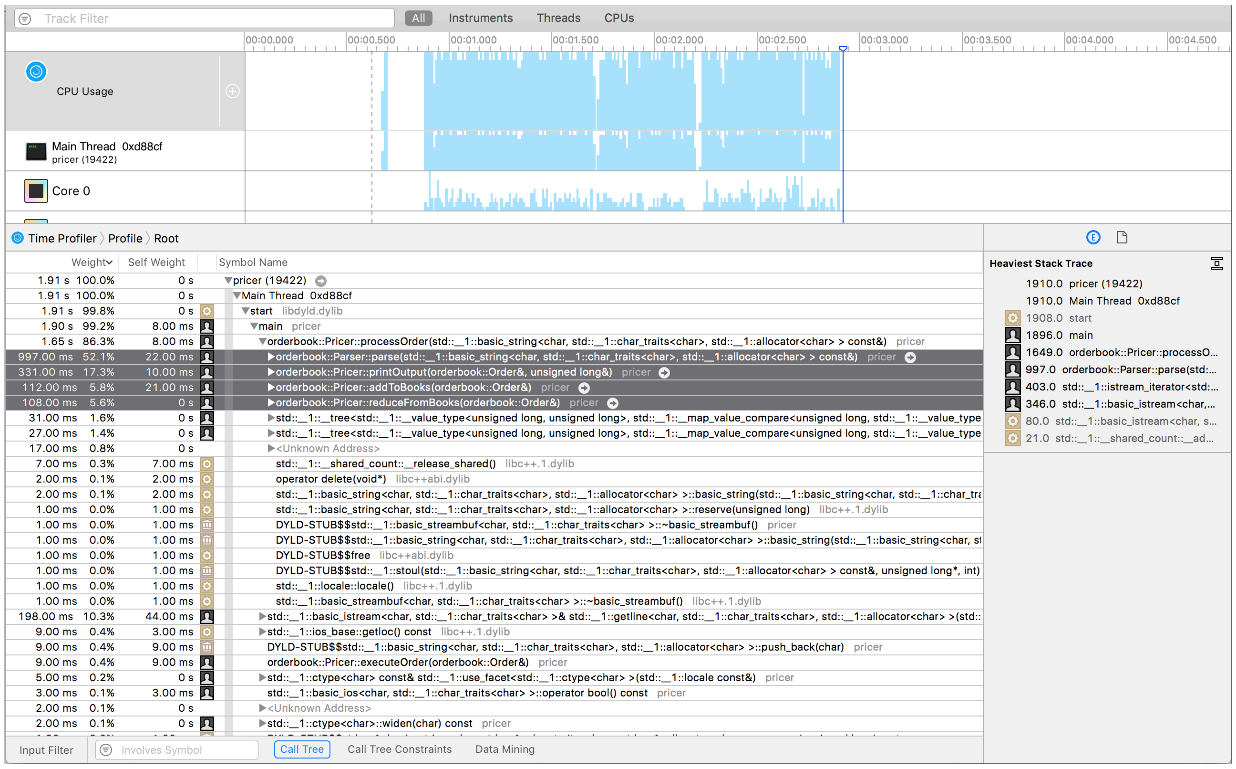
Description
The program coded in C++ and uses a *multimap* for storing the orders and an unordered_map for looking up the orders based on their order-id. The source code style lint-ed based on "Google C++ Style Guide". The program has been checked with *valgrind* for memory leakage. The result of the *Profiling* shows that on the *pricer.in* file and size-target 200, at peak just 5.1 megabyte used. Look at the end of this document for more info about the profiling.
The code could be optimized for more clarity using better design pattern but not done because of the time constraints.

What is the time complexity for processing an Add Order message?
A STL multimap been used for storing the orders sorted by their prices. Since multimap implemented using red-black tree, the insert have runtime complexity of O(log n).

What is the time complexity for processing a Reduce Order message?
A STL unordered_map been used for storing the order-id to an iterator into the orders multimap. Since unordered_map implemented using hash map, the lookup have runtime complexity of O(1) and erasing from orders multimap using iterator have runtime complexity of O(1) thus in total it takes time complexity of Reduce Order is O(1).

If your implementation were put into production and found to be too slow, what ideas would you try out to improve its performance?
- CPU Profiling and Optimization and rewriting parts that are bottlenecks
- Memory optimization using tools like valgrind
- Using GPU computing
- Using data types that have less overheads
- Using better Hash functions
- Using cache where it's possible

CPU Profiling of the Pricer program

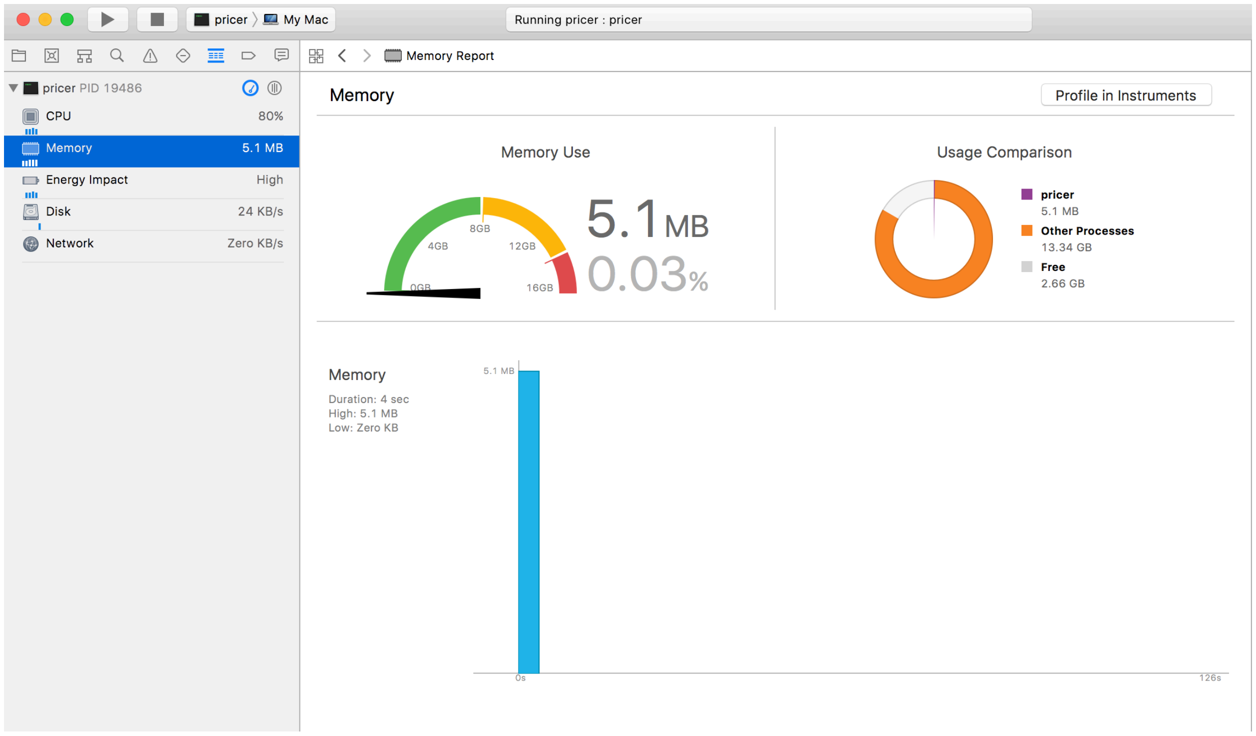


```
Time Profiler 〉 Profile 〉 Root

   Weight      Self Weight   Symbol Name
   1.91 s  100.0%     0 s    ▼pricer (19422)
   1.91 s  100.0%     0 s     ▼Main Thread  0xd88cf
   1.91 s   99.8%     0 s      ▼start  libdyld.dylib
   1.90 s   99.2%  8.00 ms      ▼main  pricer
   1.65 s   86.3%  8.00 ms       ▼orderbook::Pricer::processOrder(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&)  pricer
997.00 ms   52.1% 22.00 ms        ▶orderbook::Parser::parse(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&)  pricer
331.00 ms   17.3% 10.00 ms        ▶orderbook::Pricer::printOutput(orderbook::Order&, unsigned long&)  pricer
112.00 ms    5.8% 21.00 ms        ▶orderbook::Pricer::addToBooks(orderbook::Order&)  pricer
108.00 ms    5.6%     0 s         ▶orderbook::Pricer::reduceFromBooks(orderbook::Order&)  pricer
 31.00 ms    1.6%     0 s         ▶std::__1::__tree<std::__1::__value_type<unsigned long, unsigned long>, std::__1::__map_value_compare<unsigned long, std::__1::__value_type
 27.00 ms    1.4%     0 s         ▶std::__1::__tree<std::__1::__value_type<unsigned long, unsigned long>, std::__1::__map_value_compare<unsigned long, std::__1::__value_type
 17.00 ms    0.8%     0 s         ▶<Unknown Address>
  7.00 ms    0.3%  7.00 ms         std::__1::__shared_count::__release_shared()  libc++.1.dylib
  2.00 ms    0.1%  2.00 ms         operator delete(void*)  libc++abi.dylib
  2.00 ms    0.1%  2.00 ms         std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::basic_string(std::__1::basic_string<char, std::__1::char_tra
  1.00 ms    0.0%  1.00 ms         std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::reserve(unsigned long)  libc++.1.dylib
  1.00 ms    0.0%  1.00 ms         DYLD-STUB$$std::__1::basic_streambuf<char, std::__1::char_traits<char> >::~basic_streambuf()  pricer
  1.00 ms    0.0%  1.00 ms         DYLD-STUB$$std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::basic_string(std::__1::basic_string<char, st
  1.00 ms    0.0%  1.00 ms         DYLD-STUB$$free  libc++abi.dylib
  1.00 ms    0.0%  1.00 ms         DYLD-STUB$$std::__1::stoul(std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> > const&, unsigned long*, int)
  1.00 ms    0.0%  1.00 ms         std::__1::locale::locale()  libc++.1.dylib
  1.00 ms    0.0%  1.00 ms         std::__1::basic_streambuf<char, std::__1::char_traits<char> >::~basic_streambuf()  libc++.1.dylib
198.00 ms   10.3% 44.00 ms       ▶std::__1::basic_istream<char, std::__1::char_traits<char> >& std::__1::getline<char, std::__1::char_traits<char>, std::__1::allocator<char> >(std::
  9.00 ms    0.4%  3.00 ms       ▶std::__1::ios_base::getloc() const  libc++.1.dylib
  9.00 ms    0.4%  9.00 ms        DYLD-STUB$$std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::push_back(char)  pricer
  9.00 ms    0.4%  9.00 ms        orderbook::Pricer::executeOrder(orderbook::Order&)  pricer
  5.00 ms    0.2%     0 s        ▶std::__1::ctype<char> const& std::__1::use_facet<std::__1::ctype<char> >(std::__1::locale const&)  pricer
  3.00 ms    0.1%  3.00 ms        std::__1::basic_ios<char, std::__1::char_traits<char> >::operator bool() const  pricer
  2.00 ms    0.1%     0 s        ▶<Unknown Address>
  2.00 ms    0.1%     0 s        ▶std::__1::ctype<char>::widen(char) const  pricer

Heaviest Stack Trace
  1910.0  pricer (19422)
  1910.0  Main Thread  0xd88cf
  1908.0  start
  1896.0  main
  1649.0  orderbook::Pricer::processO...
   997.0  orderbook::Parser::parse(std:...
   403.0  std::__1::istream_iterator<std:...
   346.0  std::__1::basic_istream<char,...
    80.0  std::__1::basic_istream<char, s...
    21.0  std::__1::__shared_count::__ad...
```

Memory Profiling of the Pricer program



```
pricer PID 19486
CPU              80%
Memory         5.1 MB
Energy Impact    High
Disk          24 KB/s
Network     Zero KB/s

Memory

Memory Use
5.1 MB
0.03%

Usage Comparison
pricer
5.1 MB
Other Processes
13.34 GB
Free
2.66 GB

Memory
Duration: 4 sec
High: 5.1 MB
Low: Zero KB
5.1 MB
```