Assignment 3 Part 1: Report

Leslie Xie, Bojun Lin

February 7, 2020

Introduction

Our main goal is to find a suitable data adapter with high performance and code quality. To this end, we will be testing six different data adapters that were submitted in response to the RFP.

Description of analysis performed

The tests were performed on a machine with the following configuration:

Hardware

CPU - Intel(R) Core(TM) i7-8750H @ 2.20GHz - 6 cores, 12 processors

OS - Windows 10 Personal Edition

Laptop - Fully charged, on “best performance” mode

Software

Local

Python submissions - Python 3, MinGW shell

C++ submissions - MinGW/MSYS development environment, GCC

Using Team “chunkyboys” gen_large_sor.py script, we created a SOR file that would be tested against across all of the submissions. This file has 4 columns and 99999 rows. Tests started at different points to read from, as well as the different functions supported by the adapter. Two trials on the same build were done for each submission, and these trials’ results were averaged to yield the final result.

We will be measuring the average user+sys time and the RAM memory usage of the trials’ tests.

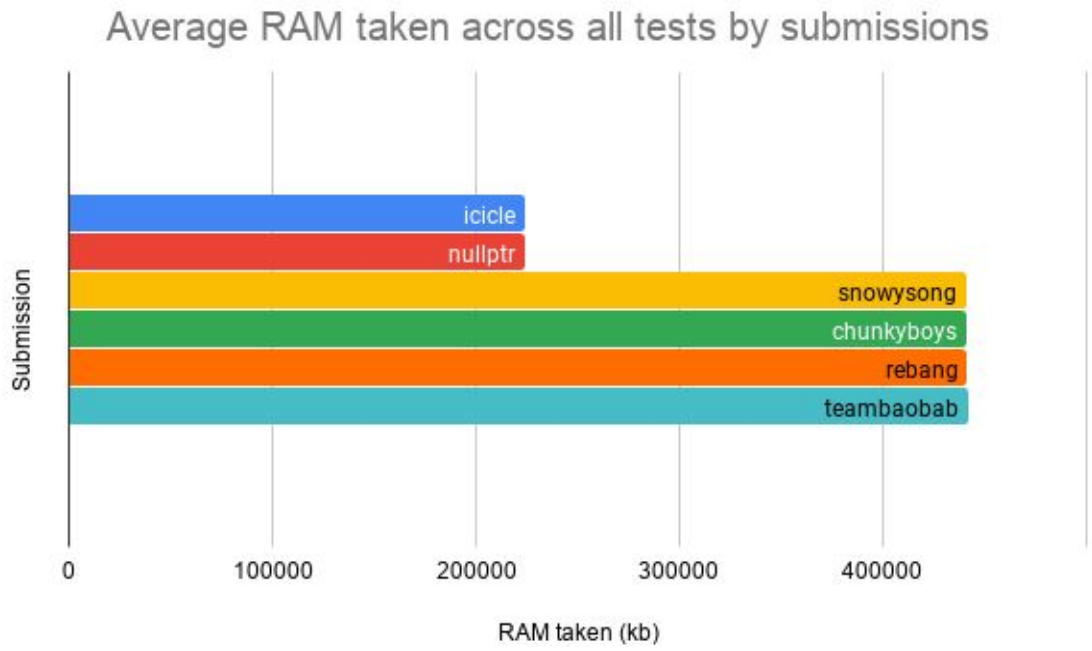
For each submission, the following scripts were used in place of the tests of the submission Makefiles:

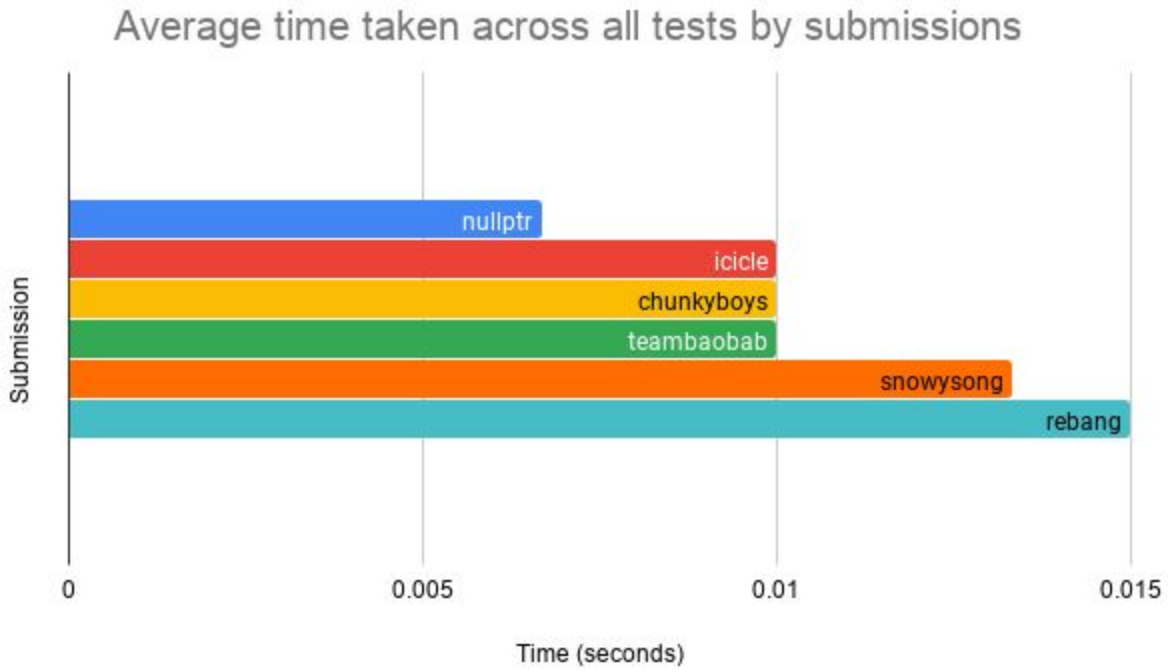
```
/usr/bin/time -v ./sorer -f "large.sor" -from 0 -len 10000 -print_col_type 3
/usr/bin/time -v ./sorer -f "large.sor" -from 5000 -len 10000 -is_missing_idx 2
0
/usr/bin/time -v ./sorer -f "large.sor" -from 10000 -len 10000 -print_col_idx 1
2
```

Comparison of product's relative performance

	Trial 1			Trial 2			Averaged			Total Averages
nullptr	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0	0	0	0	0	0	0	0	0	
sys time (seconds)	0	0.01	0.01	0.01	0.01	0	0.005	0.01	0.005	
total time (seconds)	0	0.01	0.01	0.01	0.01	0	0.005	0.01	0.005	0.006666666667
max resident set size (kb)	230912	223232	222208	223232	222208	222208	227072	222720	222208	224000
icicle	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0	0.01	0.01	0.01	0.01	0	0.005	0.01	0.005	
sys time (seconds)	0.01	0	0	0	0	0.01	0.005	0	0.005	
total time (seconds)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
max resident set size (kb)	231680	222464	222208	222464	222720	222208	227072	222592	222208	223957.3333
snowysong	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0	0	0	0.01	0.01	0	0.005	0.005	0	
sys time (seconds)	0.01	0.01	0.01	0	0	0.03	0.005	0.005	0.02	
total time (seconds)	0.01	0.01	0.01	0.01	0.01	0.03	0.01	0.01	0.02	0.01333333333
max resident set size (kb)	440320	440832	441088	441856	441088	441856	441088	440960	441472	441173.3333
chunkyboys	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0	0	0.01	0	0	0	0	0	0.005	
sys time (seconds)	0.01	0.01	0	0.01	0.01	0.01	0.01	0.01	0.005	
total time (seconds)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
max resident set size (kb)	441856	441856	440832	441344	440832	440832	441600	441344	440832	441258.6667
rebang	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0	0	0	0	0.01	0.01	0	0.005	0.005	
sys time (seconds)	0.01	0.03	0.01	0.01	0.01	0	0.01	0.02	0.005	
total time (seconds)	0.01	0.03	0.01	0.01	0.02	0.01	0.01	0.025	0.01	0.015
max resident set size (kb)	441856	440832	442112	441600	440832	441088	441728	440832	441600	441386.6667

teambaobab	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3	
user time (seconds)	0.01	0	0	0	0	0	0.005	0	0	
sys time (seconds)	0.01	0.01	0	0.01	0.01	0.01	0.01	0.01	0.005	
total time (seconds)	0.02	0.01	0	0.01	0.01	0.01	0.015	0.01	0.005	0.01
max resident set size (kb)	441344	441600	441344	442112	441856	441856	441728	441728	441600	441685.3333





Threats to validity
A common source of different results would be different software/hardware.

Recommendation to management
The submissions written in C++ used less RAM than the Python counterparts (~50.7% less from nullptr to teambaobab). Thus, the obvious choice is between the two submissions tested that were written in C++: nullptr and icicle.

nullptr however is roughly ~33% faster than icicle, at the very slight cost of more memory used (~.001%).

nullptr is consistently <.01 seconds in out-of-kernel time.

Comparing the code quality of nullptr and icicle, both are about equal in documentation (rather lackluster, missing design recipe components). Neither implementations use third-party libraries, only standard libraries to C++.

Both implementations also use wrapper classes for the four supported data types.

Therefore, we recommend going forward with **nullptr's** data adapter.