ECE 459: Programming for Performance
Assignment 4

Yang Yongren

April 4, 2018

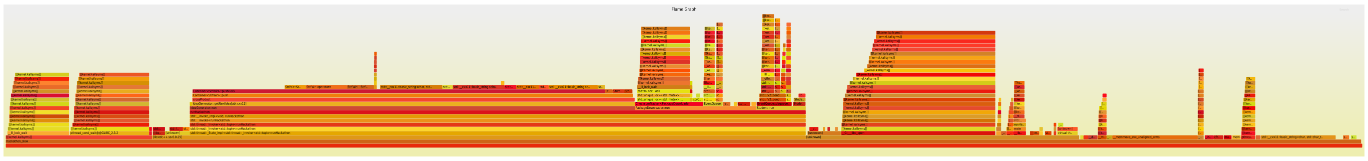

Figure 1: before optimization

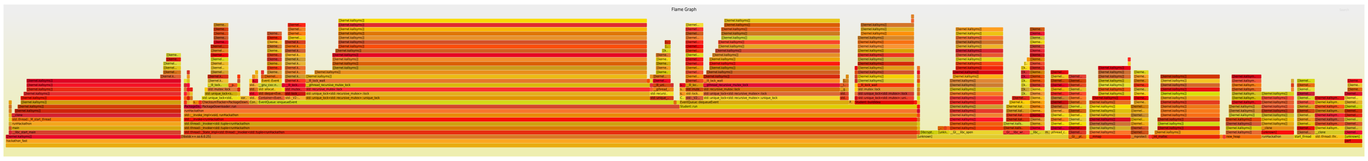

Figure 2: after optimization

	Time (s)
Not Optimized	1.906
Optimized	0.0266

Table 1: Hackathon execution benchmark

Speedup = 1.906/0.0266 = 72X.

Modification 1: Container Class:

Observation: Before optimization, it can be seen from the flame graph that Container::push/push_back takes up almost all the time of the idea generator. The original Container code has a very time-consuming push method, which allocate new memory and copy the old one to it every single time.

Fix: To make the program faster, I simply change the data structure of Container from heap memory allocation to std::deque, the program can easily push to and pop from dynamically sized queue, which saves a lot of time. After this fix, we can barely see the idea generator from the flame graph anymore, this is because it only uses very little amount of time to push. The whole program has a 2X speedup after this implementation.

Modification 2: Package/Ideas Read Input:

Observation: The flame graph shows that the readFileLine function takes a lot of time inside the PackageDownloader. By observing the code, it can be seen that the readFileLine function read from the begnning of file and probe to the target line every time, which is very time-consuming. The getNextIdea function inside the IdeaGenerator has the similar issue, every time it trying to get an idea, it will read the whole product file and the customer file again, and get the single idea, which is unnecessary.

Fix: To fix the readFileLine from the PackageDownloader, I store the whole file inside a private Container class member, the read process is done in the constructor, so the program does not have to read file every single time, it only needs the line number index to access the Container class member. The fix of getNextIdea from the IdeaGenerator is similar, the program read files and does the cross product to get the ideas inside the constructor, and then put all the ideas into a private Container class member, so the program can save time every time it trying to get an idea, it only needs the idea index to access the Container. After the fix, time used by PackageDownloader is shortened, since the readFileLine disappeared from the flame graph. The getNextIdea from the IdeaGenerator is also shorter on the flame graph.

Modification 3: Checksum Manipulation Using uint8_t:

Observation: In the flame graph, function xorChecksum seems takes lots of time in both PackageDownloader class and Student Class. By looking at the code, the whole checksum manipulation is done by using strings. The program takes strings from file, convert them to hex, then convert the hex to uint8_t, xor them and finally, convert them back to string. The process is very complicated and not necessary. The process wastes a lot of time since string in C++ is very slow and all the three Class are using it heavily.

Fix: I simplify the code by directly convert the string to uint8_t after the input is read, and only convert them back to string until the console needs to print. So the whole ChecksumTracker.h and utils functions are being modified. To use the updateGlobalChecksum function, a uint8_t need to be passed in, the xorChecksum function will xor it with the shared global checksum which allocated as uint8_t with size of SHA256_DIGEST_LENGTH and then update it. The sha256 function also returns a uint8_t pointer, so the whole process does not need string to byte or byte to string conversion only except printing the checksum at the end. By removing unnecessary strings, xorChecksum disappeared from the flame graph, and all three Class::run() does not have any time-consuming process that takes lots of time on the flame graph, only mutex lock to prevent race condition which is necessary. After finish all the three major fix, the program reaches more than 50X speedup.