CSE 504: Assignment 3
Memory Allocation Optimization based on Access Frequency
Due date: Monday, December 4

In this assignment, you will analyze the access frequency of allocated data objects and then implement a pass to optimize their allocation based on the analysis results. This assignment is intended to give you insight into how compiler optimization is performed in LLVM/Clang.

To complete this assignment, you will perform the work described below. The report and the files are to be made available to the TA for evaluation by midnight on the due date. Do **not** delete interim versions of your files or anything else that documents how you have performed your assignment tasks.

Here is what you should do:

For this work you will use your account on Seawulf and the version of LLVM/Clang that you have already installed. This assignment depends on the first two assignments.

The things you need to accomplish are listed as follows.

1) For a given program, identify all data objects allocated using the OpenMP allocate directives, which you have implemented in Assignment 1. If your assignment 1 does not function correctly, the alternative is to identify all data objects allocated using malloc().

2) Analyze memory read and write frequency for those identified data objects, which you have partially implemented in Assignment 2. Extra credit will be given if cross function memory access frequency analysis is implemented, i.e., you take into consideration loads/stores to a certain data object in other functions where it is used.

3) Leverage the analysis results to optimize data allocation for a system with the following types of memory:

1. Fast memory: read latency is 1; write latency is 1; capacity is 1GB; use malloc_fast() for data allocation.

2. Normal memory: read latency is 2; write latency is 2; capacity is 4GB; use malloc() for data allocation.

3. Non-volatile memory: read latency is 2; write latency is 10; capacity is 16GB; use malloc_nvm() for data allocation. Non-volatile memory does not lose data when powered off.

Design your scheme to use different allocation APIs for data objects with different characteristics to optimize performance or other goals you may have. Your scheme should at least consider their read/write access frequency and size. These 3 steps should all be done in a single LLVM pass. You should also write different programs to test your optimization pass.

As always, do **not wait** until the due date to do your work: otherwise the machine may become very full around the due date.

You will need to submit your files. Please also leave suitably named files in an appropriately named directory in your course account in case we need to inspect them. Your documentation should provide any explanations needed. Please only submit the pass file and test programs that you wrote. You should not modify any existing LLVM/Clang file in this assignment. Please send the file to the TA by midnight on the due date.

Please also submit a report for your work. Your report should describe the implementation of your memory analysis pass and the test results. It should be no more than 4 letter-size pages in length. Email your report in pdf or word format to the TA by midnight on the due date.

Grading:

Grades from A through F will be assigned. Grades will be based upon the correctness of your work, completeness of your responses, and the overall quality of your work. Please do your own work: do NOT copy code or text from other class participants or from any other source.