COLFAX RESEARCH | SOFTWARE EXPERTISE EVALUATION (SEE)

Multithreaded Filtering

Download lab file

File list:

- Makefile
- main.cc
- worker.cc*

(Only the changes in * files are used for grading)

Instructions:

In this lab, you will be working with an application that filters multiple 1D array datasets by the sum of the elements in the array. The input will be a 2D matrix (stored as 1D array) where each row represents a single dataset. The workload ouputs a sorted C++ vector containing the row indices of datasets that have sums greater than a given threshold.

```
void filter(const long n, // number of datasets
            const long m, // dataset size
            float *data,  // input data
            const float threshold, //filter cut-off
            std::vector &result_row_ind // output
            ) {
  for (long i = 0; i < n; i++) {
    float sum = 0.0f;
    for (long j = 0; j < m; j++) {
      sum += data[i*m + j];
    }
    if (sum > threshold)
      result_row_ind.push_back(i);
  }
  std::sort(result_row_ind.begin(),
            result_row_ind.end());
}
```

Add multi-threading to filter(), located inside "worker.cc".

Hints:

It may be tempting to parallelize this workload over *j* (across elements in a dataset). However this turn out to be rather inefficient for several reasons.

- Not enough parallelism in the loop for both multi-threading and vectorization.
- Too much overhead from entering/leaving parallel regions

We can avoid the above two performance problems, by parallelizing the *i* loop. Implement multi-threading over the *i* loop (across multiple datasets). Remember that appending to vector is NOT a thread safe operation.

Notes:

- We have provided a function for appending a vector to another vector for your convenience.
- You can use any parallelization framework for this task.

Running app:

The grading script uses the following command to run the application.

```
% KMP_HW_SUBSET=1t ./app $ALPHA
```

Grading:

1. Compile: The code compiles without error (0%)
2. Verification: The code generates the correct output (0%)
3. Performance: The application completes in under 0.5 seconds on the Xeon Phi 7210 system (100%)

Last transaction TID: 29131
We have graded your submission, and your grade is shown in the table

below

Solution: 选取文件 未选择文件

Resubmit

TID	Grade	Feedback	Submitted file	Submission time
29131	100 *,x	Compilation: PASSED Verification: PASSED Performance: PASSED	Submitted solution	2020/12/16 20:12:32

Legend:

* - your best grade for this lab so far

x - not yet submitted to the course platform. Grade transfer may take up to 5 minutes. To re-check the status, reresh this page

Coursera Learners:

- Coursera only remembers your highest score. If you re-attempt the lab and get a lower score, it will not be reflected in Coursera
- If your best grade does not get reflected in Coursera for more than 10 minutes, please let the system administrators know at labs@colfaxresearch.com