PIM programming exercise

In this assignment, you will learn how to write a program for PIM architectures exploiting the bank-level parallelism.

This assignment is worth 10 points.
The assignment submission is due on 11:59 pm ET on April 29, 2025.

Your tasks

For this assignment, you will use PIMeval-PIMbench (PIM simulator for bank-level and subarray-level processing in/near memory architectures developed at UVA) to

1. [4 points] Evaluate the performance of *gemv* benchmark (matrix dimension 4096x4096, vector dimension 4096x1) on different DRAM configurations.
    a. The GEMV (General Matrix Vector Multiplication) kernel is already implemented here. You need to evaluate the performance of this benchmark using different bank-level HBM PIM configurations. The list of configs you will evaluate on are:
        i. PIMeval_Bank_Rank1.cfg (and other configs in the same path with more ranks, i.e. PIMeval_Bank_Rank4.cfg, PIMeval_Bank_Rank8.cfg, etc.
    b. Report the total time and energy consumption for each configuration on PIM. Compare the total time with CPU baseline.

2. [6 points] Write and evaluate new benchmarks as described below:
    a. RMS normalization:

$$y_i = \frac{x_i}{\mathrm{RMS}(x)} * \gamma_i, \quad \text{where} \quad \mathrm{RMS}(x) = \sqrt{\epsilon + \frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

    For simplicity, you can assume γ_i, ϵ to be equal to 1.

    b. Layer normalization:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

    For simplicity, you can assume γ to be equal to 1, and β to be equal to 0. Here x, y are vectors.

Use vector length = 128, 4096, 8192, 16384 for both the benchmarks.

Since most of the common benchmarks that are amenable to PIM are already implemented in the PIMeval-PIMbench public github repository, your task is to write new benchmarks (RMS norm and layer norm). We have provided the code skeleton in the attached tar file (**assignment_PIM.tar.gz**), you need to fill in the code with appropriate PIM APIs (attached **API_Description.pdf**), build, run, and evaluate the code.

Submission: Submit only the PIM/<benchmark>.cpp file and a report comparing the execution time(diff. HBM config PIM vs CPU), energy (different HBM configuration for PIM)

Instructions:

Simulator setup:

1. git clone git@github.com:UVA-LavaLab/PIMeval-PIMbench.git
2. make -j USE_OPENMP=1

Example run:

3. cd PIMbench/gemv/PIM
4. ./gemv.out -c ../../../configs/hbm/PIMeval_Bank_Rank8.cfg -v true

Working with the Benchmarks:

You only need to copy the benchmark directories from the tar file to ***PIMeval-PIMbench/PIMbench/*** , go to ***<benchmark>/PIM/*** and work from there.
The code completion will need you to put some scalar operations on the host, follow the hints in the provided code and select appropriate APIs from the list of APIs. You can look vec-add/ , axpy/ benchmarks for reference,

Running the benchmarks:

From ***PIMeval-PIMbench/PIMbench/<benchmark>/PIM/*** path,

- `make -j USE_OPENMP=1` (to build your benchmark)
- `./<benchmark>.out <benchmark specific options> -v true -c ../../../configs/hbm/PIMeval_Bank_Rank*.cfg` (to run the benchmark with different bank configurations and other benchmark specific command line options.)

Evaluating the benchmarks:

Analyze the PIM output to report : PIM time, Host time, Total time and Total energy for different bank configurations (*PIMeval-PIMbench/configs/hbm/PIMeval_Bank_Rank*.cfg*). Use HBM bank-level PIM configuration for 1 ranks, 4 ranks, 8 ranks, 16, ranks, and 32 ranks. Compare the total time with CPU baseline time. CPU Baseline code is at *PIMbench/<benchmark>/baselines/CPU/*

NOTE: In case you get "stack smashing detected" error please try compiling with the following options:
-fsanitize=address -g -fno-omit-frame-pointer

The compile command would be:

```
g++ <benchmark>.cpp -std=c++17 -Wall
-I../../../libpimeval/include -I../../../util
-L../../../libpimeval/lib -lpimeval -O3
-Wno-unknown-pragmas -fsanitize=address -g
-fno-omit-frame-pointer  -fopenmp -o <benchmark>.ou
```

References

You can read more about the simulator and benchmarks here:
Architectural Modeling and Benchmarking for Digital DRAM PIM