探討 PIM 技術如何影響線性回歸的效能

On Exploring How PIM Technique Affects the Performance of Linear Regression

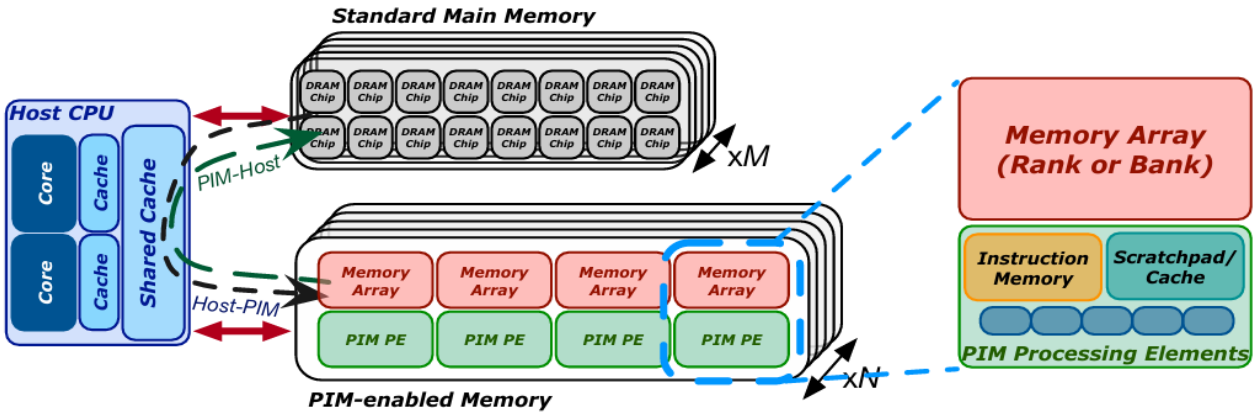
指導教授：何建忠
專題成員：劉彥誠
開發工具：C、Python、
UPMEM DPU toolchain
測試環境：Linux Ubuntu 20.04

一、簡介：

機器學習被廣泛應用於各種領域，此類型的演算法在訓練過程中需要存取大量的訓練資料，通過不斷迭代更新參數來提升模型的表現。然而，在當前以處理器為中心的系統（如 CPU、GPU）中，由於記憶體有延遲及頻寬的限制，在記憶體和處理器之間進行大量數據移動，將導致大量的執行周期和能量消耗，可能成為效能的瓶頸。

PIM（Processing-in-Memory）是一種異於 Von Neumann 架構的方法，旨在通過將運算的能力與記憶體晶片整合，以減少數據移動帶來的延遲。為了探討 PIM 技術對於上述問題的影響，本專題使用 UPMEM 的通用 PIM 架構，並挑選 Linear Regression 為實驗對象，因為其在進行梯度下降時，需要存取所有的訓練資料，而且也具有低運算強度和低 temporal locality 的特性，屬於 memory-bound workload。在訓練的過程中，會重複執行以下兩個步驟直到收斂：
1. 將所需的資料傳到 DPU 上計算梯度、2. 將計算結果傳回 CPU 上更新權重。

以下為系統架構圖：



二、測試結果：

目前現實中的 PIM 系統存在固有的缺點，即由於在記憶體附近或內部建構處理元件的難度和成本，其硬體比傳統處理器（如 CPU、GPU）受到更多限制。因此，通用 PIM 架構支援的指令集相當有限，而且難以執行複雜的運算。

為了評估 PIM 對效能的影響，實驗將從以下幾個面向進行：

1. 在 DPU 上使用不同資料型別 / 對乘法進行優化，共分成四種版本：

   （1）32-bit floating-point（LIN-FP32）
   （2）32-bit fixed-point（LIN-INT32）
   （3）fixed-point with hybrid precision（LIN-HYB）
   （4）fixed-point with hybrid precision and built-in functions（LIN-BUI）

2. 對於四種版本，在一個 DPU 上使用不同數量的 tasklet（PIM thread）執行

3. 對於四種版本，進行 weak scaling 和 strong scaling 實驗（使用 synthetic dataset）

以下是 LIN-FP32 使用一個 DPU 和一個 tasklet 的執行結果：

```
Allocated 1 DPU(s)

iter_time = 500, learning_rate = 0.0001, m = 2048, n = 16
Predefined weight: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
Successfully generate input data.

training at host, float...

Trained weight at host:  1.61,  2.56,  3.62,  4.59,  5.62,  6.54,  7.53,  8.61,
                         9.51, 10.66, 11.56, 12.61, 13.50, 14.56, 15.58, 16.60

Load input data to DPUs
Run program on DPU(s)...
DPU iter 0...
DPU iter 100...
DPU iter 200...
DPU iter 300...
DPU iter 400...

Trained weight at DPU:  1.61,  2.56,  3.62,  4.59,  5.62,  6.54,  7.53,  8.61,
                        9.51, 10.66, 11.56, 12.61, 13.50, 14.56, 15.58, 16.60

MAE on DPUs = 39.3054, avg Y = 3569.9282, error rate = 1.10%

init CPU-DPU Time (ms): 0.064000      syn CPU-DPU Time (ms): 2.191000
DPU kernel Time (ms): 800581.933000   DPU-CPU Time (ms): 4.809000
CPU reduction Time (ms): 0.581000

Outputs are equal.
```