CS 6501 - HW5 - PIM Evaluation

Huy Nguyen - mpc5ya

Q1: GEMV Performance Evaluation on PIM and CPU

I evaluated the performance of the General Matrix-Vector Multiplication (GEMV) benchmark on a 4096 × 4096 matrix and a 4096 × 1 vector using multiple DRAM configurations with PIMeval-PIMbench.

Specifically, I tested various bank-level High Bandwidth Memory (HBM) configurations: 1, 4, 8, 16, and 32 ranks, and also the Aquabolt. The performance was evaluated in terms of total execution time (host + PIM) and total energy consumption. For reference, we also ran the baseline GEMV benchmark on CPU to compare execution time.

Experimental Setup

Each configuration was tested using the command:

```
./gemv.out -r 4096 -d 4096 -c ../../../configs/hbm/PIMeval_Bank_RankX.cfg -v true
```

```
Running GEMV for matrix row: 4096 column: 4096 and vector of size: 4096
```

where `RankX` refers to the number of HBM ranks used (1, 4, 8, 16, 32).

Results

Config	PIM Time (ms)	Data Copy (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.549	2.442	2.991	1.387
Bank 4 (Aquabolt)	0.778	0.611	1.389	0.686
Bank 4 (Device Bank)	0.426	0.611	1.037	3.665
Bank 8	0.406	0.305	0.711	7.291
Bank 16	0.395	0.153	0.548	14.544
Bank 32	0.390	0.076	0.467	29.049
CPU Baseline	**Average Time over 10 runs: 3.11 ms**			

Table 1: Performance and energy results of GEMV across different HBM PIM configurations.

Discussion

As shown in the table, increasing the number of ranks consistently reduced the total runtime due to increased parallelism in the PIM architecture. For instance, the total execution time dropped from 2.99 ms (Rank 1) to 0.47 ms (Rank 32), demonstrating over 6× speedup. Both the PIM Time and the Data Copy Time did decrease with the larger numbers of banks.

However, this performance gain comes at the cost of increased energy consumption, which rises sharply with higher ranks (e.g., from 1.387 mJ at Rank 1 to 29.049 mJ at Rank 32).

When compared to the CPU baseline average runtime of 3.11 ms, all PIM configurations showed performance improvements, especially at 8+ ranks. The results confirm the advantage of leveraging bank-level parallelism in HBM for compute-intensive workloads like GEMV.

I believe there exists an optimal PIM configuration for the specific input size of 4096×4096. Overall, PIM demonstrated significant speedup compared to the CPU baseline and exhibited more consistent performance across multiple runs.

The `PIMeval_Bank_Rank4.cfg` file in the repository uses the Aquabolt simulation target, which differs from the other configurations based on `PIM_DEVICE_BANK_LEVEL`. To ensure a consistent comparison, I created a new configuration file using the standard bank-level device model. Under this model, we observe a clear trend: as the number of ranks increases, both PIM time and host time decrease, while energy consumption rises. In contrast, the Aquabolt configuration for 4 ranks results in slightly slower total execution time compared to its bank-level counterpart but achieves significantly better energy efficiency—even outperforming the Bank 1 configuration in terms of energy consumption.

Q2: RMS and Layer Normalization Benchmarking

Implementation Notes

During the RMSNorm and LayerNorm benchmarking, I encountered an overflow issue when using 32-bit unsigned integers with an input size of 16384. To resolve this, I switched to 64-bit unsigned integers (uint64_t) to ensure correctness and prevent arithmetic wraparound during accumulation.

Additionally, a value of 1 was added to the root mean square and square root computations to stabilize the normalization output and match expected results.

1. RMS Norm with input size 128

Config	**PIM Time** (ms)	**Data Copy** (ms)	**Host Time** (ms)	**Total Time** (ms)	**Total Energy** (mJ)
Bank 1	0.000284	0.000037	0.000582	0.000903	0.000762
Bank 4	0.000284	0.000009	0.000557	0.000850	0.002755
Bank 8	0.000284	0.000005	0.000194	0.000483	0.005413
Bank 16	0.000284	0.000002	0.000575	0.000861	0.010728
Bank 32	0.000284	0.000001	0.000251	0.000536	0.021359
CPU (avg)	**0.000157 ms**				

Table 2: RMS normalization performance for input length 128 across different bank-level HBM PIM configurations and CPU baseline.

2. RMS Norm with input size 4096

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000385	0.001192	0.000708	0.002285	0.002557
Bank 4	0.000304	0.000298	0.000678	0.001280	0.004239
Bank 8	0.000296	0.000149	0.000268	0.000713	0.006898
Bank 16	0.000293	0.000075	0.000712	0.001080	0.012217
Bank 32	0.000291	0.000037	0.000732	0.001060	0.022854
CPU (avg)	**0.018053 ms**				

Table 3: RMS normalization performance for input length 4096 across different bank-level HBM PIM configurations and CPU baseline.

3. RMS Norm with input size 8192

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000500	0.002384	0.000702	0.003586	0.004555
Bank 4	0.000338	0.000596	0.000314	0.001248	0.006239
Bank 8	0.000311	0.000298	0.000379	0.000988	0.008485
Bank 16	0.000303	0.000149	0.000790	0.001242	0.013811
Bank 32	0.000300	0.000075	0.000227	0.000602	0.024462
CPU (avg)	**0.036464 ms**				

Table 4: RMS normalization performance for input length 8192 across different bank-level HBM PIM configurations and CPU baseline.

4. RMS Norm with input size 16384

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000730	0.004768	0.000677	0.006175	0.008550
Bank 4	0.000406	0.001192	0.000219	0.001817	0.010240
Bank 8	0.000352	0.000596	0.000268	0.001216	0.012493
Bank 16	0.000325	0.000298	0.000243	0.000866	0.016998
Bank 32	0.000317	0.000149	0.000776	0.001242	0.027680
CPU (avg)	**0.072827 ms**				

Table 5: RMS normalization performance for input length 16384 across different bank-level HBM PIM configurations and CPU baseline.

1. Layer Norm with input size 128

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000427	0.000037	0.000578	0.001042	0.001192
Bank 4	0.000427	0.000009	0.000745	0.001181	0.004183
Bank 8	0.000427	0.000005	0.000259	0.000691	0.008169
Bank 16	0.000427	0.000002	0.000306	0.000735	0.016143
Bank 32	0.000427	0.000001	0.000308	0.000736	0.032091
CPU (avg)	**0.001877 ms**				

Table 6: Layer normalization performance for input length 128 across different bank-level HBM PIM configurations and CPU baseline.

2. Layer Norm with input size 4096

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000585	0.001192	0.000842	0.002619	0.004620
Bank 4	0.000462	0.000298	0.000979	0.001739	0.007145
Bank 8	0.000449	0.000149	0.000975	0.001573	0.011135
Bank 16	0.000443	0.000075	0.000973	0.001491	0.019117
Bank 32	0.000440	0.000037	0.000974	0.001451	0.035080
CPU (avg)	**0.029313 ms**				

Table 7: Layer normalization performance for input length 4096 across different bank-level HBM PIM configurations and CPU baseline.

3. Layer Norm with input size 8192

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.000763	0.002384	0.000837	0.003984	0.008401
Bank 4	0.000517	0.000596	0.000802	0.001915	0.010931
Bank 8	0.000476	0.000298	0.000860	0.001634	0.014304
Bank 16	0.000463	0.000149	0.000460	0.001072	0.022300
Bank 32	0.000457	0.000075	0.000975	0.001507	0.038293
CPU (avg)	**0.057355 ms**				

Table 8: Layer normalization performance for input length 8192 across different bank-level HBM PIM configurations and CPU baseline.

4. Layer Norm with input size 16384

Config	PIM Time (ms)	Data Copy (ms)	Host Time (ms)	Total Time (ms)	Total Energy (mJ)
Bank 1	0.001119	0.004768	0.000848	0.006735	0.015962
Bank 4	0.000627	0.001192	0.001011	0.002830	0.018503
Bank 8	0.000545	0.000596	0.001112	0.002253	0.021891
Bank 16	0.000504	0.000298	0.000810	0.001612	0.028667
Bank 32	0.000491	0.000149	0.000451	0.001091	0.044718
CPU (avg)	**0.116427 ms**				

Table 9: Layer normalization performance for input length 16384 across different bank-level HBM PIM configurations and CPU baseline.

Discussion

Across all input sizes, we observe that increasing the number of banks in the PIM configuration consistently reduces both PIM execution time and data transfer time. This is expected, as higher bank counts allow for greater parallelism and better utilization of memory-level parallel execution.

For smaller inputs like 128, PIM execution time is relatively stable regardless of configuration, indicating the overhead of managing cores dominates and amortization is limited. However, energy consumption still increases with bank count due to higher idle or underutilized parallel cores.

As the input size increases (4096 and above), the performance gains from increased banks become more apparent. PIM time and total runtime drop significantly, especially moving from Bank 1 to Bank 8 or 16. However, energy consumption increases sharply beyond 16 banks, showing diminishing returns from scaling the memory banks too far.

RMS normalization is consistently faster than Layer normalization because it requires fewer operations—most notably avoiding subtraction and mean-shifting over multiple axes. Layer normalization results (not shown here) take approximately 2–3× more time and energy, reinforcing the added complexity of its operations.

CPU runtimes remain fairly constant per input size, but are significantly slower than even the lowest PIM configurations at large sizes. This demonstrates the strength of bank-level parallel PIM acceleration for vector-level normalization operations.

Conclusions

I really enjoy doing this HW as this looks similar to CUDA programming which I had some experience before. I would love to learn more about PIM this summer :D

Thanks Prof. Skadron and Khyati for all the help this semester of grading my late submissions! I will definitely be more active and on-time over the summer.