UPMEM PIM 기반 원주율의 몬테카를로 추정 구현

곽재혁[1], 오광진[1]
[1]한국과학기술정보연구원
jhkwak@kisti.re.kr, koh@kisti.re.kr

Implementation of Monte Carlo estimation of Pi based on UPMEM PIM

Jae-Hyuck Kwak[1], Kwang Jin Oh[1]
[1]Korea Institute of Science and Technology Information

요 약

폰노이만 구조를 따르는 기존의 컴퓨터 시스템은 프로세서와 메모리의 역할이 구분되어 있으며 프로세서는 메모리에 저장된 명령어와 데이터를 불러와 실행한다. 이 과정에서 메모리와 프로세서 간에 발생하는 데이터 이동은 메모리 집약적인 응용을 처리하는데 있어서 심각한 오버헤드를 야기할수 있다. PIM(Processing-In-Memory)은 데이터 이동 병목을 해결하기 위해서 메모리에 프로세서의 능력을 통합하는 기술로서 최근의 메모리 기술의 발전으로 주목받고 있다.

본 논문에서는 UPMEM사의 상용 PIM 제품을 기반으로 몬테카를로 방법을 이용한 원주율 추정을 구현하고 성능 확장성을 분석하였다.

1. 서론

폰노이만 구조를 따르는 기존의 컴퓨터 시스템은 프로세서와 메모리의 역할이 구분되어 있으며 프로세서는 메모리에 저장된 명령어와 데이터를 불러와 실행하게 된다. 이와 같은 환경에서 메모리와 프로세서 간에 발생하는 데이터 이동은 낮은 데이터 재사용성을 가지는 신경망, 데이터베이스, 그래프 처리와 같은 메모리 집약적인 응용을 처리하는데 심각한 오버헤드를 야기할 수 있다.

PIM(Processing-In-Memory)는 데이터 이동 병목을 해결하기 위해서 메모리에 프로세서의 능력을 통합하는 기술로서 과거의 연구는 시뮬레이션이나 단순화된 하드웨어 평가를 기반으로 많이 수행되었으나 최근의 메모리 기술의 발전으로 삼성전자의 HBM-PIM, AxDIMM, SK하이닉스의 AiM, 알리바바의 HB-PNM 등 PIM에 대한 새로운 연구 결과가 소개되고 있다.

본 논문에서는 UPMEM사의 상용 PIM 제품을 사용하여 몬테카를로 방법을 이용한 원주율 추정을 구현하였다. UPMEM PIM은 메모리에 DPU라고 불리는 RISC기반의 범용 프로세서를 내장하고 있으며 UPMEM SDK를 이용하여 범용적인 프로그래밍이 가능하다. 그러나 하드웨어 제약으로 인해 DPU 아키텍처를 따라 프로그래밍되어야 하며 본 논문에서는 원주율 추정 구현을 통해서 이를 살펴보고 성능 확장성에 대해 분석하였다.

2. UPMEM PIM 및 DPU 아키텍처

그림 1은 UPMEM PIM[1][2] 및 DPU 아키텍처를 도식화한 것이다.

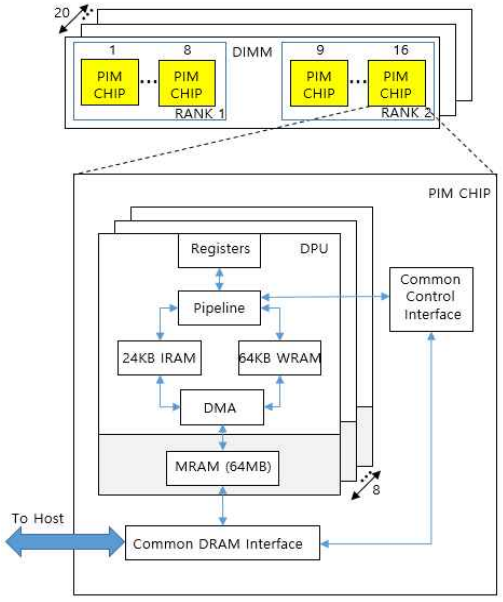


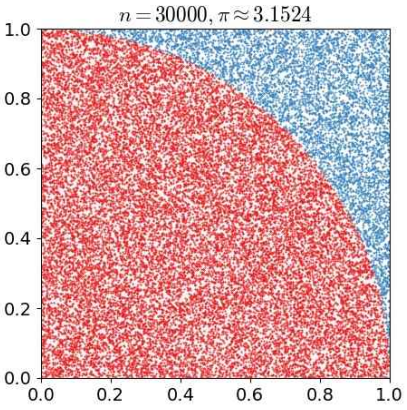
그림 1 UPMEM PIM 및 DPU 아키텍처

UPMEM PIM 모듈은 랭크당 8개의 PIM 칩으로 구성되어 총 16개의 PIM 칩으로 구성되며 PIM 칩은 8개의 DPU를 가진다. DPU는 32비트 RISC 기반의 범용 명령어 집합을 가지며 인스트럭션을 위한 IRAM, 스크래치 패드나 캐시로 사용되는 WRAM이 있으며 파이프라인은 IRAM으로부터 인스트럭션을 읽고 WRAM을 load나 store를 통한 데이터 저장 공간으로 사용한다. DMA는 MRAM과 WRAM 혹은 IRAM간의 데이터 이동에 사용된다. DPU는 14단계 인터리브 파이프라인 프로세서로 24개의 하드웨어 쓰레드를 지원하며 각 DPU는 자신의 64MB 메모리 뱅크에만 직접 접근 가능하다.

3. 몬테카를로 방법 및 원주율 추정

몬테카를로 방법[3]은 수학적인 결과를 얻기 위해서 반복된 난수 샘플링에 의존하는 계산 알고리즘이다. 몬테카를로 방법은 가능한 입력의 도메인을 정의하고 도메인에 대한 확률 분포에서 임의로 입력을 생성한 후 입력에 대한 결정론적인 계산을 수행하고 결과를 집계하는 패턴을 가지는데 대표적인 예시로는 원주율 추정이 있다.

그림 2에서 보는 것처럼 몬테카를로 방법을 이용한 원주율 추정은 (0,0)을 중심으로 변의 크기로 2r을 가지는 정사각형 안에 난수 (x,y)를 생성하며 동일한 중심으로 반경 r을 가지고 정사각형에 내접하는 원 내부에 있는 포인트와 총 포인트의 비율을 계산하면 원주율을 추정할 수 있다.



$$\frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

$$\frac{\pi}{4} = \frac{\text{num of points generated inside the circle}}{\text{total num of points generated inside the square}}$$

$$\pi = 4 * \frac{\text{num of points generated inside the circle}}{\text{total num of points generated inside the square}}$$

그림 2 몬테카를로 원주율 추정

4. UPMEM PIM 기반 몬테카를로 원주율 추정 구현

UPMEM SDK[4] 프로그래밍은 호스트 파트와 DPU 파트로 나뉜다. 호스트 파트는 다음과 같이 구현된다. (1) R보다 작은 난수 x, y를 생성하고 각각 입력데이터 배열에 저장한다. UPMEM PIM에서 실수형 데이터 처리는 소프트웨어적으로 에뮬레이션되어 매우 느리므로 정수형 데이터를 사용한다. (2) 각 입력데이터 배열은 그림 3과 같이 DPU의 개수에 맞게 DPU로 분배되는데 전송 효율을 높이기 위해서 비동기 전송 모드(DPU_XFER_ASYNC)를 사용하였다. (3) DPU 실행을 요청하고 완료되기를 기다린다. (4) 각 DPU로부터 계산된 결과를 전송받아서 DPU갯수만큼의 원소 개수를 가지는 출력데이터 배열에 저장하고 원주율을 추정한다.

DPU 파트는 다음과 같이 구현된다. (1) DPU에 전달된 입력데이터 배열은 MRAM에 저장되는데 MRAM은 WRAM보다 접근 비용이 크기 때문에 mram_read를 사용하여 MRAM에 직접 접근하였다. WRAM에는 태스크릿 별로 캐시 (cache_x [NR_TASKLETS] [CACHE_SIZE], cache_y [NR_TASKLETS] [CACHE_SIZE])를 생성하여 MRAM과 WRAM간의 데이터 전송을 효율화하였다. MRAM과 WRAM의 소스 및 타겟 주소는 8바이트로 정렬되어야 함으로 DPU간 입력데이터 분배 및 캐시 크기는 주의가 필요하다. (2) DPU는 태스크릿으로 DPU에 전달된 입력데이터를 병렬처리한다. 각 입력데이터가 원 내부에 위치($x^2 + y^2 < R^2$)하는 경우 카운터 값을 증가시키는 계산을 반복 수행하며 최종 카운터 값이 호스트로 전달되는 결과 값이 된다.

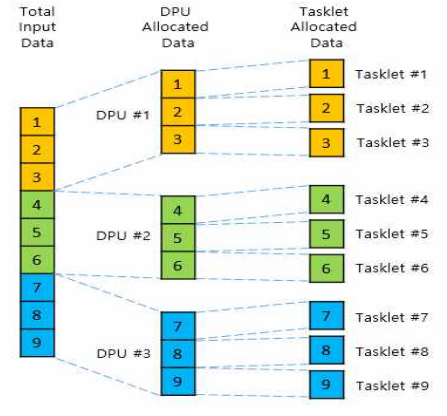


그림 3 UPMEM PIM의 입력데이터 분배

몬테카를로 원주율 추정의 강확장성(strong scalability)을 비교하기 위해서 총 입력데이터 개수를 고정하고 (1) 싱글 랭크(SR)에서 실행되는 DPU 갯수 1개-64개를 사용하는 경우와 (2) 멀티 랭크(MR)에서 실행되는 DPU갯수 64개-1024개를 사용

하는 경우를 나누어서 CPU-DPU 전송시간, DPU 실행시간, DPU-CPU 전송시간을 분석하였다.

그림 4, 그림 5를 보면 DPU갯수가 증가함에 따라 DPU 커널의 실행시간이 비례하여 감소하며 강확장성을 유지함을 알수 있다. 다만, DPU갯수가 증가함에 따라 DPU 커널의 실행시간보다 CPU-DPU 전송시간이 전체 실행 시간 중에서 더 큰 비중을 차지하고 있음을 알 수 있다. 따라서 CPU와 DPU간의 데이터 전송 최적화가 전체 성능 향상에 도움이 될 것으로 볼 수 있다.

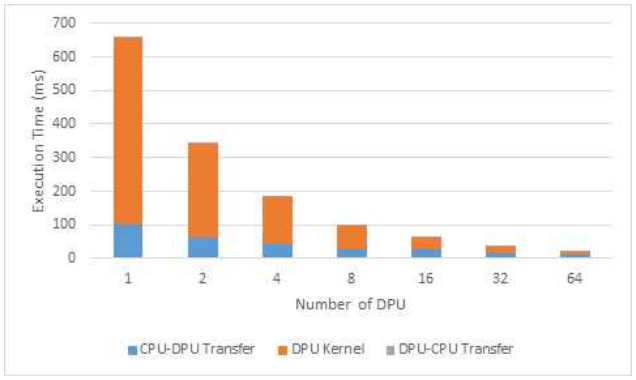


그림 4 강확장성 성능 분석 (SR, Total Points = 4,096,000)

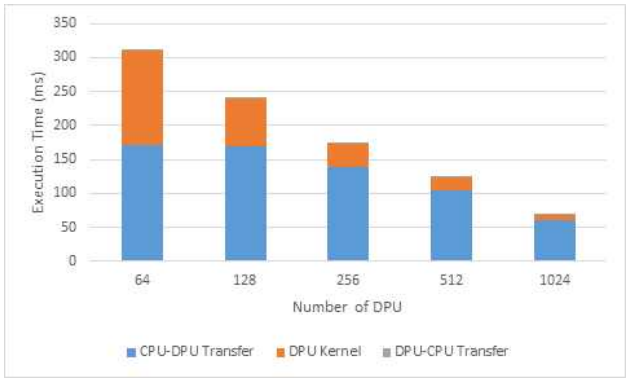


그림 5 강확장성 성능 분석 (MR, Total Points = 65,536,000)

몬테카를로 원주율 추정의 약확장성(weak scalability)을 비교하기 위해서 DPU에서 처리되는 입력데이터의 개수를 고정하고 (1) 싱글 랭크(SR)에서 실행되는 DPU갯수 1개-64개를 사용하는 경우와 (2) 멀티 랭크(MR)에서 실행되는 DPU갯수 64개-1024개를 사용하는 경우를 나누어서 CPU-DPU 전송시간, DPU 실행시간, DPU-CPU 전송시간을 분석하였다.

그림 6, 그림 7을 보면 DPU 개수가 증가하더라도 DPU 커널의 실행시간은 일정하게 유지되어 약확장성을 유지함을 알 수 있다. 그러나 DPU 개수가 증가함에 따라 CPU-DPU전송시간이 증가하여 전체 실행 시간 중에서 CPU-DPU 전송시간이 차지하는 비중이 점차 커지고 있음을 확인할 수 있다.

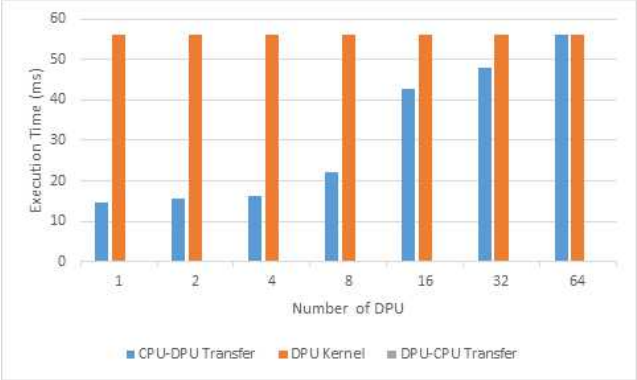


그림 6 약확장성 성능 분석 (SR, Total Points = 409,600 * nr_dpu)

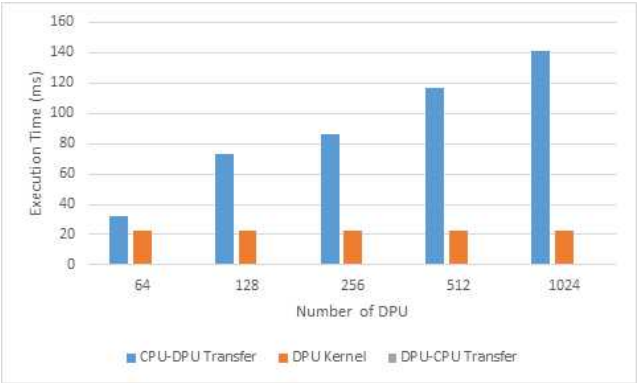


그림 7 약확장성 성능 분석 (MR, Total Points = 163,840 * nr_dpu)

5. 결론 및 향후 계획

본 논문에서는 UPMEM PIM을 기반으로 몬테카를로 방법을 이용한 원주율 추정을 구현하고 성능 확장성을 분석하였다. UPMEM PIM은 범용적인 프로그래밍이 가능하지만 하드웨어 제약으로 인해 응용의 특성에 맞게 소프트웨어 최적화가 요구되며 데이터 타입 및 분배, 전송과 관련한 최적화 연구가 필요할 것으로 생각된다.

※ 이 논문은 2023년도 한국과학기술정보연구원(KISTI)의 기본사업으로 수행된 연구입니다.(과제번호: K-23-L02-C06)

참고문헌

[1] Juan Gómez-Luna, et al., "Benchmarking a new paradigm: experimental analysis of a real processing-in-memory architecture.", https://doi.org/10.48550/arXiv.2105.03814, 2022.

[2] Juan Gómez-Luna, et al. "Evaluating Machine LearningWorkloads on Memory-Centric Computing Systems.", 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2023.

[3] Monte Carlo method, https://en.wikipedia.org/wiki/Monte_Carlo_method

[4] UPMEM SDK, https://sdk.upmem.com/2021.3.0/