Zero Copy를 이용한 CSR 희소행렬 연산

윤상혁*, 전다윤*, 박능수*
*건국대학교 컴퓨터공학과
sgryoon97@konkuk.ac.kr, jeondayoon27@gmail.com, neungsoo@konkuk.ac.kr

CSR Sparse Matrix Vector Multiplication Using Zero Copy

SangHyeuk Yoon*, Dayun Jeon*, Neungsoo Park*
*Dept. of Computer Science, Konkuk University

요 약

APU(Accelerated Processing Unit)는 CPU와 GPU가 통합되어있는 프로세서이며 같은 메모리 공간을 사용한다. CPU와 GPU가 분리되어있는 기존 이종 컴퓨팅 환경에서는 GPU가 작업을 처리하기 위해 CPU에서 GPU로 메모리 복사가 이루어졌지만, APU는 같은 메모리 공간을 사용하므로 메모리 복사 없이 가상주소 할당으로 같은 물리 주소에 접근할 수 있으며 이를 Zero Copy라 한다. Zero Copy 성능을 테스트하기 위해 희소행렬 연산을 사용하였으며 기존 메모리 복사대비 크기가 큰 데이터는 약 4.67배, 크기가 작은 데이터는 약 6.27배 빨랐다.

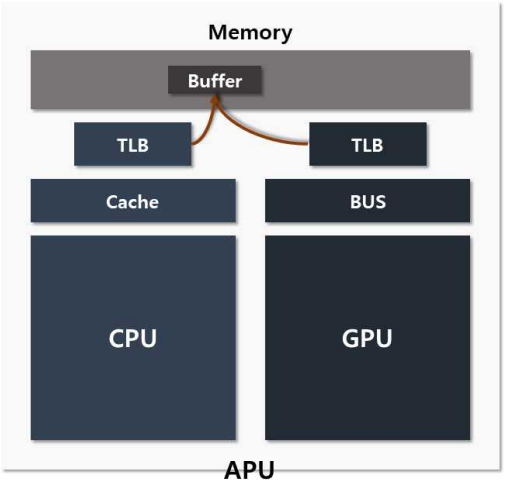
1. 서론

최근 프로세서의 성능이 높아지면서 CPU와 GPU 통합 프로세서인 APU(Accelerated Processing Unit)의 GPU 성능 또한 높아지고 있다. 과거 모바일 프로세서에서 주로 사용되던 APU는 현재 고성능 데스크톱을 위한 제품까지 확장되고 있다. 본 논문은 AMD의 최신 APU인 AMD Ryzen 7 4700g 프로세서에서 희소행렬 처리 성능을 측정하였다. 또한 APU 아키텍처에 최적화된 메모리 사용 방법을 소개한다.

2. APU 아키텍처

CPU와 GPU가 분리된 이종 컴퓨팅 환경과 달리 APU는 CPU와 GPU가 통합되어있는 프로세서이다. 따라서 APU는 CPU와 GPU가 같은 메모리 공간을 공유한다. APU에서 CPU와 GPU는 각각 가상주소 공간을 관리하며 서로 다른 TLB(Translation Look-aside Buffer)를 갖고 있다. 서로 다른 가상주소 관리 체계로 같은 물리 주소를 가리킬 수 있다[1]. 그러므로 이종 컴퓨팅 환경과 달리 CPU와 GPU 간 메모리 복사 없이 가상주소 할당으로 메모리 복사를 대체할 수 있으며 이를 Zero Copy라 한다. 하지만 CPU와 GPU가 통합되어있으므로 메모리 대역폭을 공유하여 속도 하락이 발생할 수 있으며 캐시 일관성 문제가 존재한다. 그림 1은 APU 아키텍처 다이어그램이다.

OpenCL에서 Zero Copy 기능을 사용하기 위해 clEnqueueMapBuffer api를 사용한다[2]. GPU에서 사용할 메모리 객체를 생성한 후 clEnqueueMapBuffer를 사용하면 CPU에서 접근 가능한 주소 공간을 반환한다. 그림 2는 OpenCL을 이용한 Map Buffer 의사코드이다.



(그림 1) APU 아키텍처

```
01  cl_mem gpu_mem
02
03  // Create buffer
04  gpu_mem = clCreateBuffer(...)
05
06  float* cpu_mem = clEnqueueMapBuffer(gpu_mem, ...)
07
08  // Write data at cpu_mem
09  cpu_mem[i] = ...
10
11  // Execute Kernel
12  clEnqueueKernel(...)
13
14  // Wait kernel to finish
15  clWait(...)
16
17  // Read data
18  print(cpu_mem[i])
19
20  // Repeat additional work
21
22  // Unmap buffer
23  clEnqueueUnmapMemObject(gpu_mem, cpu_mem, ...)
```

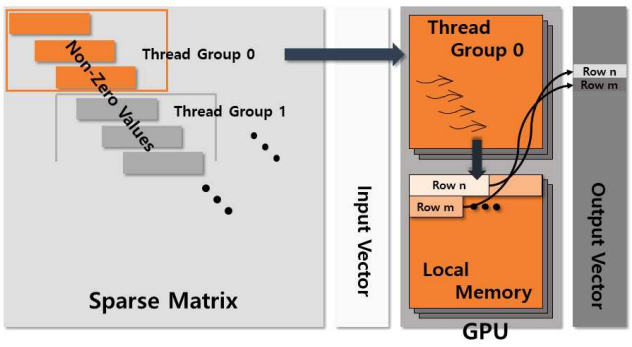
(그림 2) OpenCL Map Buffer 의사코드

3. CSR 알고리즘

희소행렬(Sparse Matrix)은 대부분의 값이 0인 행렬로 계산 분야에서 많이 사용되는 행렬이다. 대부분의 값이 0이므로 0이 아닌 값만 계산하여 희소행렬 연산 시간을 단축할 수 있다. 희소행렬 연산 시간을 단축하기 위해 사용되는 알고리즘 중 가장 대표적인 알고리즘은 CSR(Compressed Sparse Row) 알고리즘이다.

CSR 알고리즘은 희소행렬의 0이 아닌 값, 각 값의 열 정보, 각 행의 0이 아닌 값의 개수를 저장하는 3개의 배열로 희소행렬을 압축한다. 계산하려는 행의 0이 아닌 값의 개수를 확인하여 계산할 배열의 위치를 확인하고 해당 값들의 열 정보에 해당하는 위치의 벡터값을 모두 곱한 후 더하여 결과 벡터에 저장한다. 위 과정을 모든 열 마다 반복하여 계산을 완료한다.

GPGPU를 이용하여 희소행렬을 계산할 때 성능을 높이기 위해 반복적인 연산의 양을 최대한 늘려 각 스레드 그룹의 처리량을 증가시키는 것이 중요하다. 한 스레드 그룹은 희소행렬의 0이 아닌 값과 벡터를 곱한 중간 결과를 프로그래밍 가능한 GPU 캐시인 로컬메모리에 저장한다. 모든 스레드가 로컬메모리에 중간값을 저장하면 결과 벡터에 해당하는 위치에 로컬메모리값들을 더해 저장한다. 연산량을 최대한 높이기 위해 한 스레드 그룹이 사용 가능한 로컬메모리의 크기를 알아낸 후 각 스레드 그룹이 수용 가능한 행의 크기를 계산하여 각각 배정한다. 따라서 각 스레드 그룹이 계산하는 행의 길이는 다르다. 이를 CSR Stream 알고리즘이라 한다[3]. 그림 3은 GPGPU를 사용한 CSR 알고리즘을 설명한 그림이다.



(그림 3) GPGPU에서 CSR 알고리즘

4. 실험

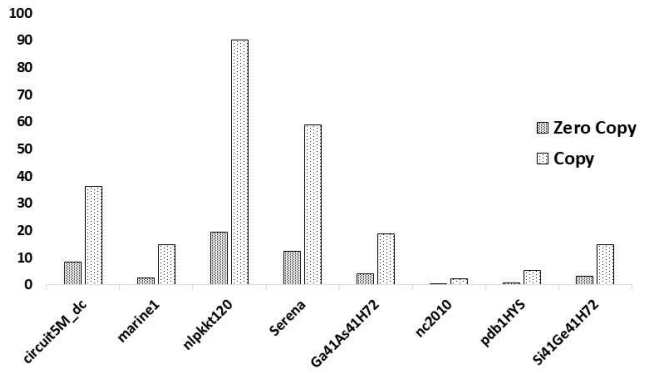
AMD의 APU인 Ryzen 7 4700g 프로세서를 사용하여 CSR 알고리즘 처리 속도를 측정하였다. APU에서 사용 가능한 Zero Copy 기술을 적용한 알고리즘과 기존 copy 방법을 사용한 알고리즘의 처리 속도를 비교하였다. 표 1은 실험 환경이다. 실험을 위한 희소행렬은 상용 행렬인 circuit5M_dc, marine1, nlpkkt120, Serena, Ga41As41H72, nc2010, pdb1HYS, Si41Ge41H72를 사용하였다[4]. 그림 4는 각 희소행렬 계산 시간을 나타낸 그래프이다. 시간 단위는 microseconds이며 Zero Copy를 적용한 커널의 처리시간과 기존 Copy를 적용한 커널의 처리시간을 나타낸다. Zero Copy를 적용한 처리시간은 메모리 매핑 시간을 포함하며 기존 Copy를 적용한 커널은 메모리 복사 시간을 포함한다.

<표 1> 실험 환경

APU	AMD Ryzen 7 4700g
MemoryUnit	16GB
OS	Ubuntu 20.04.1
OpenCL Version	2.0

실험 결과 커널의 처리 속도는 비슷하였지만 기존 Copy를 사용하였을 때 메모리 복사 시간이 처리 속도의 대부분을 차지하였다. 약 960Mb로 크기가 가

장 큰 nlpkkt120의 경우 Zero Copy를 적용하였을 때 기존 Copy 대비 약 4.67배 빨랐으며, 약 16Mb로 크기가 가장 작은 nc2010 희소행렬의 경우 Zero Copy를 적용하였을 때 약 6.27배 빨랐다. 연산 속도와 비교해 데이터 복사 시간이 클수록 Zero Copy를 적용하였을 때 효과가 커지는 것을 알 수 있다.



(그림 4) CSR 알고리즘 성능 비교

5. 결론

본 논문은 CPU와 GPU가 같은 메모리 공간을 공유하는 APU 아키텍처에 맞게 메모리 복사를 줄이는 방법과 그 효과에 대해 분석하였다. APU에서 CPU와 GPU는 각각의 가상 주소 공간 체계를 갖고 있어 가상 주소 변환으로 같은 물리 공간에 접근할 수 있다. 이 기법을 Zero Copy라 하며 이를 사용하여 메모리 복사 시간을 줄일 수 있다. 그 결과 기존 Copy 사용 대비 Zero Copy를 적용하였을 때 크기가 가장 큰 희소행렬의 처리 속도의 경우 약 4.67배 빨라졌으며 크기가 작은 희소행렬의 경우 약 6.27배 빨라졌다. 연산 시간 대비 데이터 복사 시간이 더 긴 경우 속도 향상이 더욱 크며 CPU와 GPU 사이 데이터 복사가 빈번하게 일어날 때 이 효과는 더욱 커질 수 있다.

감사의 글

이 논문은 대한민국 정부(과학기술정보통신부)의 재원으로 한국연구재단 슈퍼컴퓨터개발선도사업의 지원을 받아 수행된 연구임 (과제번호 : 2020M3H6A1084853)

참고문헌

[1] Pierre Boudier, Graham Sellers “Memory System on FUSION APU, Benefit of Zero Copies” AMD Fusion Developer Summit, June. 2011

[2] AMD ROCm OpenCL Optimization Guide “https://rocmdocs.amd.com/en/latest/Programming_Guides/Opencl-optimization.html#optimization-opencl”

[3] J. L. Greathouse and M. Daga, “Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format”, SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, pp. 769-780, Nov. 2014

[4] SuiteSparse Matrix Collection “https://sparse.tamu.edu/”