메모리 로딩 시간을 고려한 GPU 병렬 알고리즘의 성능 개선 방안

배병걸[O], 이진우, 박일남, 임은진, 강승식
국민대학교 컴퓨터공학과
bazel1984@naver.com, sosuminjok@gmail.com, pin0156@naver.com,ejim@kookmin.ac.kr,
sskang@kookmin.ac.kr

Performance Enhancement of GPU Parallelism Algorithm including Memory Loading Time

Byunggul Bae[O], Jinwoo Lee, Il-Nam Park, Eun-Jin Im, Seung-Shik Kang
School of Computer Science, Kookmin University

요 약

GPU를 이용한 병렬 알고리즘은 어떤 메모리를 사용하는지에 따라 시스템의 전체적인 성능이 달라진다. 본 논문은 GPU 환경에서 실행되는 CUDA 프레임워크에서 병렬처리를 이용하여 문서 분류 시스템의 속도를 향상시키고자 할 때 메모리 로딩 시간이 전체적인 시스템의 성능에 미치는 영향을 연구하였다. 기존의 CPU 환경에서 구현했을 때와 비교하여 어느 정도의 성능 향상이 있었는지 실험하였으며 이전 연구에서 고려하지 않았던 메모리를 읽는데 걸리는 시간을 고려하여 현실적인 실행 시간을 비교하였다. 실험 결과에 의하면 CPU에서 구현했을 때의 연산 속도보다 GPU의 텍스쳐 메모리를 사용하여 구현하였을 때 문서 분류 성능이 향상되는 효과가 있음을 알 수 있었다.

주제어: GPU 병렬성, 메모리 로딩 시간, CUDA 프레임워크, 시스템 성능

1. 서론

현재 정보검색 분야에 있어서 가장 큰 화두는 소셜네트워크로 인하여 짧은 시간에 많은 수의 문서가 생성되는 대용량 문서처리 기술이다. 트위터와 페이스북을 비롯한 다양한 소셜 네트워크(Social Network)를 통하여 매우 많은 사람들이 실시간으로 자신들의 의견을 표출하는 문서를 생성하기 때문에 이러한 대용량 자료를 분석하기 위해 많은 문서들을 빠르게 처리 또는 분류할 수 있는 기술이 필요하다.[1] 이러한 목적으로 문서 분류(Text Classification) 또는 문서 클러스터링(Document Clustering) 기법이 필요하며 문서 분류 및 클러스터링 알고리즘은 문서간 유사도 계산이 주요 내용이므로 유사도 계산 성능이 전체적인 시스템의 성능에 미치는 영향이 가장 크다. 유사도 계산 기법은 통상적으로 벡터 공간 모델에서 사용되는 유사도 계산 기법으로 그 성능 측정이 가능하다.[2]

본 논문에서는 짧은 시간에 많은 문서들을 분류하는 문서 분류 시스템의 성능을 향상시키기 위해서 NVIDIA의 GPU 병렬처리 기술인 CUDA (Compute Unified Device Architecture) 프레임워크를 사용하여 알고리즘으로 구현했을 때 실행 속도를 향상하는 방안을 연구하였다.[3,4] 정보검색 기법으로 널리 사용되는 벡터 공간 모델은 문서에서 내용을 대표하는 자질(feature)들을 추출하고 문서를 자질 벡터(feature vector)로 변환하여 TF-IDF로 표현된 벡터간의 유사도를 코사인 계수로 계산하는 방법이다.[2] CUDA 프레임워크를 이용한 벡터 공간 모델에서의 코사인 유사도 계산 속도의 향상은 박일남 등에 의해 연구된 바 있다.[5] 이 연구에서는 벡터 공간에서 코사인 유사도 계산 속도가 CPU보다 CUDA 프레임워크를 이용한 GPU 병렬처리를 사용할 경우 속도가 향상되는 실험 결과를 보여주었다.

2. GPU 메모리 오버헤드 해결 방안

기존의 벡터 공간 모델에서 코사인 유사도 계산 기법을 GPU 프로그래밍으로 구현했을 때 그 연산속도가 매우 향상되는 것을 확인하였지만 이 연구에서는 단순히 연산 속도만을 비교하였기 때문에 메모리를 읽고 쓰는 속도를 고려하지 않았다.[6] 본 논문에서는 벡터 공간 모델을 이용하여 벡터 연산을 수행할 때 GPU 병렬처리 과정에서 빈번하게 발생하는 메모리 로드 시간을 고려하여 문서 분류 시스템의 실질적인 속도 향상을 분석하고자 하였다.

문서 분류 시스템의 성능 향상에 큰 영향을 미치는 요인의 하나는 주기억장치에 저장되어 있는 데이터를 GPU의 메모리로 로드하는 것이며 이 방법에는 두 가지 방법이 있다. 첫 번째 방법은 주기억장치의 전체 데이터를 GPU로 로드하는 방법으로 전역 메모리를 사용하는 방법과 두 번째는 텍스쳐 메모리(texture memory)를 사용하는 방법이 있다. 텍스쳐 메모리는 GPU와 같이 주로 3차원 그래픽스 하드웨어에서 텍스쳐 데이터를 저장하는 기억공간이다. 이 메모리는 TRAM이라는 특수한 RAM으로 구현되어 읽기/쓰기 속도가 매우 빠르며, 그래픽스 하드웨어의 성능을 향상시키는데 사용된다.

3. 실험 및 평가

본 실험에서는 병렬처리를 이용하여 연산 속도를 빠르게 함으로써 실행 시간을 단축시키기 위하여 메인 메모리의 데이터를 GPU 메모리로 로드하는 시간 비용을 최소화하기 위해 다양한 메모리들을 대상으로 실험하였다. 본 연구에서는 전역 메모리, 그리고 텍스쳐 메모리를 사용하여 연산을 하고 그 연산에 필요한 시간을 비교하였다.

<표 1> 실험 환경

CPU	Intel Core i7 980x - 6 core (12 threads) - 3.33 GHz
GPU	NVIDIA GeForce GTX 580 - 512 CUDA Cores (32 Cores/SM x 16 SM) - 2.0 CUDA Compute Capability - 384bit Memory Interface Width - 192.4GB/sec Memory Bandwidth

<표 2> 실험 데이터

문서 개수 × 문서 개수	10×10, 10×100, 100×10, 100×100

실험은 표 1의 환경에서 수행하였으며 표 2와 같이 랜덤 함수에 의해 무작위로 제작된 대상 문서들을 사용하여 문서 분류를 위한 유사도 계산 실험을 수행하였다. 본 연구에서는 문서 유사도 측정 수치를 벡터 공간 모델에서 주로 사용하는 코사인 유사도와 유클리드 거리의 계산 시간을 비교하였다. CPU에서 실행했을 때 속도와 CUDA 프레임워크에서 GPU 병렬처리를 통해 실행했을 때 시간을 비교하였다. 여러 가지 유형의 메모리를 이용하여 GPU에서 실험한 결과는 표 3, 표 4와 같다.

이 실험 결과는 실제 CPU에서 실행한 시간과 GPU에서 실행한 시간을 비교한 것이다. 10x10은 10개의 입력 문서와 10개의 입력 문서를 서로 비교한 것을 의미한다. 10x10 문서 비교에 대해 CPU 연산 시간을 기준 시간 1로 하여 다양한 메모리들과 데이터들을 대상으로 실험하였다. 실험 결과, 코사인 유사도와 유클리드 거리를 척도로 사용한 경우 모두 데이터 개수가 적을 때는 CPU에서 연산속도가 빨랐으나 데이터 개수가 증가할수록 GPU 연산속도가 빨라진다.

<표 3> 코사인 유사도에 의한 성능 비교

	CPU	GPU-Global	GPU-Texture
10x10	1	4	4
10x100	16	22	14
100x10	16	14	15
100x100	156	52	32

<표 4> 유클리드 거리에 의한 성능 비교

	CPU	GPU-Global	GPU-Texture
10x10	1	4	4
10x100	9	24	15
100x10	9	14	15
100x100	88	52	33

4. 결론

문서 분류 및 클러스터링과 같이 각 문서에 대해 동일한 연산을 반복하는 응용 시스템에서 CPU 대신에 GPU를 이용했을 때 성능 향상 효과를 측정하기 위하여 메모리 오버헤드를 고려하여 전체적인 시스템의 성능 실험을 수행하였다. 실험 결과에 따르면 문서의 양이 적으면 GPU를 사용하는 것보다 GPU보다 CPU에서 연산 속도가 빠르지만 문서의 양이 많아질수록 병렬처리를 이용한 GPU가 더 성능이 좋아지는 것을 알 수 있다. 대용량 문서처리에 있어서 GPU의 전역 메모리와 텍스쳐 메모리를 사용할 경우 CPU보다 빠르며 그 중에서도 텍스쳐 메모리가 가장 빠른 연산 속도를 보여주었다.

참고문헌

[1] S. A. Dudani, "The Distance-weighted k-Nearest-Neighbor Rule," IEEE Transaction on Man and Cybernetics, 1976.

[2] G. Salton, A. Wong and C. S. Yang, "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol.18, no.11, pp.613-620, 1975.

[3] 정영훈, CUDA 병렬프로그래밍, 프리렉, 2011.

[4] Jason Sanders, CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley, 2010.

[5] 박일남, 배병걸, 임은진, 강승식, "GPU 병렬성을 이용한 문서 유사도 계산 성능 개선", 정보처리학회 논문지 B, vol.19-B, no.4, pp.243-248, 2012.