데이터베이스에서의 GPU 활용 동향 연구

최영환, 여은지, 이형석, 임효상
연세대학교(원주캠퍼스) 컴퓨터정보통신공학부
e-mail : {big.choi, ejyeo, hyungseoklee, hyosang}@yonsei.ac.kr

A Study on the Database Research Trends of Exploiting GPU Capabilities

Young-Hwan Choi, Eunji Yeo, Hyungseok Lee, Hyo-sang Lim
Computer Telecommunications Engineering Division, Yonsei University

요 약

GPU(Graphic Processing Unit)의 높은 계산 능력과 병렬성은 그래픽 분야뿐만 아니라 다양한 분야에서 활용되고 있다. 본 논문에서는 데이터베이스 분야에서 GPU 가 활용되고 있는 연구들을 소개한다. 이를 통하여 데이터베이스 분야에서 GPU 활용의 중요성과 이러한 연구가 활발히 이루어져야 하는 필요성을 보인다. 또한 각각의 연구들이 GPU 의 병렬성을 어떻게 활용하고 있는지 분석하여 다른 데이터베이스 관련 연구들에서도 GPU 가 어떻게 활용될 수 있는지 중요한 단서들을 제공한다.

1. 서론

GPU (Graphic Processing Unit)는 그래픽 연산을 위한 하드웨어 장치이다. 최근에는 NVIDIA 사에서 제공하는 CUDA (Compute Unified Device Architecture)와 같이 GPU 를 이용하여 범용적인 연산을 수행할 수 있는 라이브러리들이 제공되고 있다. 범용 라이브러리의 발전으로 인하여 그래픽 분야가 아닌 다양한 분야에서도 GPU 의 높은 계산 능력과 병렬성을 이용할 수 있게 되었다. 특히 데이터베이스 분야에서도 GPU 를 활용하여 데이터베이스 연산을 빠르게 처리하고자 하는 많은 연구들이 진행되고 있다.

데이터베이스 분야의 GPU 활용 연구들의 공통적인 접근법은 대량의 데이터를 분할하고 분할된 영역에 대한 동일한 연산을 동시에 처리하여 수행 속도를 높이는 것이다. 이러한 방식은 크게 다음과 같은 두 가지 측면에서 유효하다: 1) 대량의 데이터를 처리하는 응용의 경우, 데이터를 나누어서 처리하여 병렬성을 높일 수 있다. 2) 계산집약적인 연산이 사용되는 응용의 경우, 연산 자체를 나누어서 처리하여 병렬성을 높일 수 있다. 본 논문에서는 이러한 GPU 의 병렬성을 이용한 접근법이 데이터베이스의 각 응용에서 어떻게 활용되었는지를 설명한다. 이를 통하여 데이터베이스 분야에서 GPU 활용의 중요성과 필요성을 보인다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 데이터베이스에서 자주 사용되는 일반적인 연산들을 GPU 기반으로 처리하기 위한 활용을 살펴본다. 제 3 장에서는 GPU 의 그래픽 정보 처리 특성을 살려서 공간 데이터베이스에서 활용되는 연구를 살펴본다. 제 4 장에서는 데이터베이스 보안에서 GPU 활용을 살펴보고 제 5 장에서는 그 외의 다양한 데이터베이스 응용에서 활용되는 GPU 연구에 대하여 살펴본다. 마지막으로 제 6 장에서는 결론을 맺는다.

2. 데이터베이스 기본 연산 처리에서의 GPU 활용

Naga Govindaraju 등[1]에서는 predicates, boolean combination, aggregation 연산등과 같이 비교 연산이 가장 큰 비용을 차지하는 데이터베이스 연산들을 GPU 를 이용하여 빠르게 계산하는 방법을 제안하였다. 이 방법은 대량의 데이터를 여러 개의 부분으로 분할하여 각각 GPU 의 스레드에 할당하고, 스레드 별로 독립적으로 비교 연산을 수행하여 전체 데이터에 대한 비교 연산을 병렬화 함으로써 수행 성능을 높였다. 이 방법은 GPU 의 특징인 SIMD(Single Instruction Multiple Data) 구조를 사용하여, 동일한 비교 연산을 서로 다른 데이터 부분에 대해서 동시에 수행하게 함으로써 병렬처리의 이득을 얻었다.

Beingsheng He 등[2]에서는 단순한 비교 연산에서 더 나아가서 조인 연산을 GPU 를 이용하여 효율적으로 처리하는 방법을 제안하였다. 이 연구에서는 블록 중첩 조인(Block Nested Join) 혹은 해시 조인(Hash Join)을 수행하는 과정에서 GPU 의 병렬성을 활용하여 각각의 블록 혹은 해시 버킷(bucket)을 하나의 스레드에 할당하여 독립적으로 조인을 수행함으로써 조인 처리 성능을 향상시켰다. 또한 병렬성을 극대화 하기 위하

· 이 논문은 2014 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 일반연구자지원사업 지원을 받아 수행된 것임(2012R1A1A1042875).

여, 각 스레드 간의 I/O 충돌(conflict)을 최소화 하는 방향으로 블록과 해시 버킷을 나누는 방법을 제시하였다.

Sebastian Bress 등[3]에서는 앞에서 설명한 다양한 데이터베이스 연산들이 복합적으로 나타나는 복잡한 형태의 질의를 GPU 를 이용하여 어떻게 효율적으로 처리할 지에 초점을 맞추었다. 이 연구에서는 GPU 를 이용하기 위해서는 디바이스로의 데이터 전송등과 같은 부가적인 작업이 필요하여 추가적인 로드가 걸린다는 것에 착안하여, 질의처리 과정 중 병렬성으로 인한 이득을 얻기 어렵고 단순한 작업들은 CPU 가 처리하고 병렬성으로 얻을 수 있는 이득이 큰 작업은 GPU 에서 처리하도록 질의 수행 계획(query processing plan)을 생성함으로써 CPU 와 GPU 를 함께 이용하여 질의를 처리하는 방법을 제안하였다.

Peter Bakkum 등[4]에서는 데이터베이스 연산들을 GPU 에서 수행할 때 사용자들에게 생소한 CUDA 언어가 아닌 기존의 SQL (Structured Query Language)을 이용할 수 있도록 하는 프로그래밍 인터페이스를 개발하였다. CUDA 는 일반적인 범용 프로그램을 작성하는 기능을 제공하지만 SQL 을 직접 지원하지는 않는다. 이 연구에서는 CUDA 인터페이스 위에 SQL 인터페이스를 추가하고, 가장 많이 사용되는 SELECT 질의에 대해서 메모리 매핑과 스레드 구현 등의 복잡한 작업을 대신 수행하는 기능을 제공한다.

3. 공간데이터 처리에서의 GPU 활용

GPU 는 그래픽 연산을 처리한다는 그 본연의 역할에 맞게 공간 데이터베이스에서의 질의를 빠르게 수행하는데 활용되어 왔다. Chengyu Sun 등[5]에서는 공간 질의 처리 중 수행 비용이 큰 정제(refinement) 과정을 GPU 를 이용하여 병렬적으로 수행하는 방법을 제시하였다. 일반적으로 공간 질의 처리는 1) 질의 결과 후보를 찾는 과정과 2) 결과 후보들이 실제 질의 결과가 맞는지를 검증하는 정제 과정으로 나뉜다. 이 때, 질의 결과 후보를 찾는 과정은 공간 색인 등을 사용하여 빠르게 수행할 수 있으나, 정제 과정은 후보들이 질의 조건을 만족하는지 하나하나 비교해야 하기 때문에 많은 수행 시간을 필요로 한다. 이 연구에서는 결과 후보들을 분할하여 각각을 독립적인 스레드들에서 정제과정을 수행함으로써 병렬화하여 성능을 높이는 방법을 제시하였다. Bogdan Simion 등[6]에서는 정제 과정뿐만 아니라 공간 질의 처리의 전 과정으로 까지 GPU 의 활용을 확장하였다. 이 방법은 질의 결과 후보를 찾는 과정에서도 공간 객체들을 분할하여 여러 개의 스레드에서 동시에 수행하도록 함으로써 병렬성을 높였다.

Jia Pan 등[7]에서는 고차원 데이터에 대한 *k* 최근접 질의(*k*-NN, *k*-Nearest Neighbor Query)를 GPU 를 이용하여 빠르게 수행하는 방법을 제안하였다. 이 연구의 특징은 *k*-NN 질의를 처리하는 과정에서뿐만 아니라, 색인 구조 생성에도 GPU 의 병렬성을 활용했다는 것이다. 이 논문은 먼저 *k*-NN 질의 처리를 위해서 많이 사용되는 색인 구조인 LSH (Locality Sensitive Hash)를 생성하는데 있어서 고차원 데이터들을 분할하여 스레드에 할당하여 해시 값을 계산함으로써 속도를 높였다. LSH 는 위치적으로 가까운 데이터들끼리 유사한 해시 값을 가지도록 함으로써 인접한 데이터들을 클러스터링하여 빠르게 찾을 수 있도록 하는 방법이다. 다음으로, *k*-NN 질의 처리에 있어서 해시 테이블 검색을 통해서 얻은 결과 후보들로부터 정확한 순위(ranking)을 계산하는데 있어서 각각의 후보들을 스레드에 할당하여 병렬적으로 계산하도록 하였다.

4. 데이터베이스 보안에서의 GPU 활용

GPU 를 기반으로 한 데이터베이스 암호화와 복호화의 효율적인 처리에 대한 연구들이 진행되고 있다. Engine-CUDA[8]에서는 대칭키 암호화 알고리즘인 AES(Advanced Encryption Standard)를 CUDA 를 이용하여 수행할 수 있도록 구현하였다. 대칭키 암호화 알고리즘은 암호화 하고자 하는 데이터를 블록으로 나누어서 동일한 키를 적용하여 암호화 하는 방식이기 때문에 상대적으로 병렬화가 쉽다는 특징을 가진다. Heeseung Jo 등[9]에서는 Engine-CUDA 를 MySQL 에 적용하는 연구를 하였고, 이때 데이터의 크기에 따라서 CPU 에서의 암호화와 GPU 에서의 암호화 중 더 비용이 작은 쪽으로 수행하도록 하였다.

Keon Jang 등[10]에서는 SSL(Secure Sockets Layer)의 성능을 가속하기 위해서 GPU 를 이용하였다. SSL 은 인터넷상에서 데이터를 안전하게 주고 받기 위한 표준으로, 먼저 데이터를 암호화할 키를 주고 받기 위해서는 비교적 수행 비용이 큰 공개키 알고리즘인 RSA 를 사용하고, 이렇게 받은 키를 사용하여 실제 데이터는 비교적 수행비용이 작은 대칭키 알고리즘인 AES 를 사용하여 암호화하는 방식을 사용함으로써, 대칭키 암호화 알고리즘과 공개키 암호화 알고리즘의 장점을 동시에 취하였다. 이 연구는 다수의 클라이언트들로부터 전송되는 여러 개의 메시지가 수신되는 상황을 가정하여 서버에는 이 다수의 메시지를 해독하는데 들어가는 비용을 줄이는 방법을 제시하였다. 이 방법의 핵심은 메시지들을 GPU 의 스레드 개수만큼 축적한 후 한번에 암호화나 복호화를 수행한다는 것이다.

Owen Harrison 등[11]에서는 RSA 암호화 알고리즘을 GPU 를 이용하여 빠르게 수행하는 방법을 제시하였다. RSA 알고리즘은 암호화 및 복호화에 계산 복잡도가 큰 연산을 사용함으로써 안전성을 높인다. 최근 RSA 는 1024-bit 크기의 키를 사용하고 있는데, 이는 암호화의 안전성을 높이는 것과 동시에 암호화와 복호화에 필요한 계산비용도 함께 높아지는 결과를 얻었다. 이 연구에서는 이러한 문제를 해결하기 위하여 RSA 의 연산을 GPU 를 이용하여 병렬적으로 처리하도록 하였다. 이 연구에서는 먼저 한 개의 스레드가 한 개의 데이터에 대한 연산을 수행하고, 이러한 스레드를 동시에 여러 개 수행하는 병렬처리 방법을 사용하였다. 다음으로, 1024-bit 크기의 하나의 정수에 대한 연산을 여러 개의 스레드에 나누어서 처리하는 방법을 제시하였다.

5. 그 외 다양한 데이터베이스 응용에서의 GPU 활용

앞에서 소개한 데이터베이스 응용뿐만 아니라 최근에는 다양한 데이터베이스 응용으로 GPU 의 활용 영역이 확장되고 있다. Martin Krulis 등[12]에서는 이미지 유사 검색(image similarity search)에 GPU 를 활용하는 방법을 제시하였다. 일반적으로 이미지 유사 검색에서는 각각의 이미지에서 특성 백터를 추출하고, 추출된 특성 백터들 간의 거리(유사도)를 계산하는 방식으로 이루어진다. 이 때, 이미지의 개수가 많아지면 유사도 거리 계산의 횟수도 기하급수적으로 증가하기 때문에 효율적으로 처리하는데 어려움이 있었다. 이 연구에서는 대량의 이미지를 일정 개수씩 나누어서 스레드에 할당하여 유사도를 동시에 계산하는 병렬처리뿐만 아니라, 특성 백터 자체를 분할하여 하나의 유사도 계산을 여러 개의 스레드에서 동시에 병렬적으로 처리하는 방법도 함께 제시하였다는 특징을 가진다. 이는 일반적으로 1) 특성 백터의 크기가 크고 2) 유사도 계산이 매트릭스 연산으로 이루어져서 연산 자체를 병렬화하기 쉽다는 특징을 이용한 것이다. 특히 이 연구에서는 고비용의 병렬 CPU 머신과 저비용의 GPU 에서의 이미지 유사 검색 성능 비교 실험을 통해서, GPU 기반의 검색이 비용대비 월등히 우수한 성능을 보임을 입증하였다.

Naga Govindaraju 등[13]에서는 수십억 개 이상의 대량 데이터를 대상으로 정렬(sorting) 연산을 GPU 를 이용하여 빠르게 처리하는 방법을 제시하였다. 이 연구의 특징은 앞서 설명한 연구들에서 주로 채용했던 GPU 의 병렬 연산 기능뿐만 아니라, GPU 가 CPU 에 비해서 10 배이상 빠른 메모리 통신 속도를 가진다는 장점을 활용하여 I/O 를 최적화 하였다는 것이다. 대량의 데이터에 대한 정렬을 수행할 때는 일반적으로 외부 정렬(external sort) 방식을 사용하는데, 이때 데이터를 반복적으로 읽어와야 해서 I/O 비용이 정렬 연산 비용의 대부분을 차지하게 된다. 이 연구는 이러한 반복적인 I/O 를 GPU 의 글로벌 메모리에서 수행함으로써 정렬 연산의 속도를 높였다. 또한, I/O 의 효율성을 높이기 위해서 CUDA 에서 사용하는 범용적인 데이터구조가 아니라 GPU 에서 그래픽 처리에 사용되는 텍스처(texture) 표현 방식에 따른 데이터구조를 사용하였다는 특징을 가진다.

Naga Govindaraju 등[14]에서는 히스토그램 근사치(histogram approximation) 계산 등의 메모리 접근 비용이 큰 수치 통계(numerical statistics) 계산을 빠르게 수행하기 위하여 GPU 를 활용하였다. 이 연구는 히스토그램을 구성하는데 가장 큰 비용을 차지하는 정렬 연산을 일반적인 비교 연산이 아니라 GPU 에서 색상을 조합하기 위한 블렌딩(blending) 연산을 사용하여 빠르게 수행한다. 블렌딩 연산은 원래의 색상과 조합할 색상 코드 값 비교는 작업을 GPU 의 하드웨어 모듈을 사용하여 가속하는데, 이를 여러 개의 코어들에서 병렬적 처리함으로써 일반적인 비교 연산을 사용하는 것에 비해서 더 빠르게 정렬 작업을 수행할 수 있도록 한다.

Changkyu Kim 등[15]에서는 GPU 를 이용하여 트리 검색을 빠르게 수행하는 방법을 제시하였다. 이 연구에서 메모리 접근 비용을 줄이기 위해 CPU 의 캐시(cache)와 메모리의 캐시라고 불리는 TLB(Translation Lookaside Buffer)를 이용하였다. 메모리에 저장되어 있는 데이터에서 트리 구조를 이용하여 탐색 작업을 수행할 때, 캐시 히트율을 높이기 위해서 두 가지 방법을 제시하고 있다. 먼저 1) 연관성이 높은 트리 노드(node)들을 캐시 최대 용량에 맞춰 그룹화하는 방법과 다음으로 2) 멀티 코어 시스템을 사용하여 캐시의 개수를 늘려 캐시 히트율을 높이는 방법을 제안한다. 첫 번째 방법은 CPU 가 트리 순회를 위해 하나의 그룹을 캐시에 적재하면 다음에 처리할 노드가 캐시에 포함될 확률이 높기 때문에 캐시 히트율을 높일 수 있다는 점을 고려한 방법이다. 두 번째 방법은 여러 개의 코어들을 이용하면 캐시의 개수가 늘어나고, 탐색에 필요한 비교작업을 동시에 많이 처리할 수 있기 때문에 메모리 접근 비용과 병렬처리를 함께 수행한 방법이다. 이 연구는 두 번째 방법을 멀티 코어 시스템보다 많은 코어를 사용할 수 있는 GPU 까지 확장하여 병렬성을 높였다.

6. 결론

GPU 는 높은 계산 능력과 병렬성으로 인해 범용적인 연산에서 성능을 향상시킬 수 있는 하드웨어로 주목 받고 있다. 본 논문에서는 데이터베이스 분야에서 GPU 를 이용하여 연산을 효율적으로 수행하는 연구 동향에 대해 살펴보았다. 해당 연구들을 통해 볼 수 있듯이 GPU 는 데이터베이스의 기본적인 연산부터 공간 데이터베이스의 질의 수행, 데이터베이스의 보안을 위한 암호화 수행, 그리고 그 외의 다양한 응용에 이르기까지 다양하게 이용되고 있다. 또한 본 논문에서는 데이터베이스 분야에서 GPU 활용의 중요성과 필요성을 보였다. 이렇게 다양한 연구에서와 같이 GPU 를 이용하면 비교적 저비용으로 큰 성능향상을 얻을 수 있다.

참고문헌

[1] Govindaraju, N.K., et al. Fast computation of database operations using graphics processors. in Proceedings of the 2004 ACM SIGMOD international conference on Management of data. 2004. ACM.

[2] He, B., et al. Relational joins on graphics processors. in Proceedings of the 2008 ACM SIGMOD international conference on Management of data. 2008. ACM.

[3] Breß, S., E. Schallehn, and I. Geist, Towards optimization of hybrid cpu/gpu query plans in database systems, in New Trends in Databases and Information Systems. 2013, Springer. p. 27-35.

[4] Bakkum, Peter, and Kevin Skadron. Accelerating SQL database operations on a GPU with CUDA. In Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. ACM, 2010.

[5] Sun, C., D. Agrawal, and A. El Abbadi. Hardware acceleration for spatial selections and joins. in Proceedings of the 2003 ACM SIGMOD international conference on Management of data. 2003. ACM.

[6] Simion, B., S. Ray, and A.D. Brown, Speeding up spatial database query execution using GPUs. Procedia Computer Science, 2012. 9: p. 1870-1879.
[7] Pan, J. and D. Manocha. Fast GPU-based locality sensitive hashing for k-nearest neighbor computation. in Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2011. ACM.
[8] Engine-CUDA: A cryptographic engine for CUDA supported devices,https://code.google.com/p/engine-cuda/
[9] Jo, H., et al., Data Encryption on GPU for High-Performance Database Systems. Procedia Computer Science, 2013. 19: p. 147-154.
[10] Jang, K., et al. SSLShader: cheap SSL acceleration with commodity processors. in Proceedings of the 8th USENIX conference on Networked systems design and implementation. 2011. USENIX Association.
[11] Harrison, O. and J. Waldron, Efficient acceleration of asymmetric cryptography on graphics hardware, in Progress in Cryptology–AFRICACRYPT 2009. 2009, Springer. p. 350-367.
[12] Kruliš, M., et al., Combining CPU and GPU architectures for fast similarity search. Distributed and Parallel Databases, 2012. 30(3-4): p. 179-207.
[13] Govindaraju, N., et al. GPUTeraSort: high performance graphics co-processor sorting for large database management. in Proceedings of the 2006 ACM SIGMOD international conference on Management of data. 2006. ACM.
[14] Govindaraju, N.K., N. Raghuvanshi, and D. Manocha. Fast and approximate stream mining of quantiles and frequencies using graphics processors. in Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005. ACM.
[15] Kim, C., et al. FAST: fast architecture sensitive tree search on modern CPUs and GPUs. in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. 2010. ACM.