GPU 기반 데이터 집약 시스템에 대한 비교연구

최병욱 ○ 구상언 서영균

경북대학교 컴퓨터학부

mxg130@gmail.com, tkddjs311@gmail.com, yksuh@knu.ac.kr

A Survey of GPU-based Data-intensive System

Byeonguk Choi ○ Sangun Gu Youngkyoon Suh

School of Computer Science & Engineering, Kyungpook National University

요 약

최근 몇 년 동안 빅데이터의 등장으로 기존 데이터 시스템에서 다루기 힘든 엄청난 양의 많은 데이터가 쏟아지고 있다. 이에 따라, 기존 시스템으로 데이터를 처리하기에 비용적·시간적 한계의 임계치에 이르게 되었다. 이러한 문제를 해소하기 위해, GPU를 이용한 데이터 집약 시스템이 등장하기 시작했다. 하지만 (i) *그러한 시스템들이 무엇이 있고*, (ii) *각 시스템들이 어떤 특성이 있는지 잘 알려져 있지 않으므로*, 사용자들은 어떤 시스템들이 자신의 어플리케이션에 유용하고 적합한지 알기 어려운 실정이다. 이러한 간격을 좁히기 위해, 본 연구는 최근에 등장한 GPU 기반 데이터 집약 시스템을 소개하고 시스템 간의 특장점을 비교 분석한다. 우리는 먼저 GPU를 이용한 데이터 처리의 이점이 무엇인지 살펴보고, 이를 이용해 어떻게 성능을 극대화 시킬 수 있는지 살펴본다. 이어, 이러한 시스템들을 비교 분석하여 어떤 쟁점들이 있는지 테이블로 제시하고, 끝으로 결론을 맺는다.

1. 서 론

GPU(Graphics Processing Unit)는 기본적으로 그래픽 처리를 위해 사용하는 장치이다. 그러나 2000년대, GPGPU(General-Purposed computation on Graphics Processing Unit) [1] 에 대한 연구가 진행되었고 이를 통해 GPU의 수 많은 코어를 이용하여 병렬처리를 수행함으로 그래픽처리 뿐만 아니라 일반적인 계산까지 처리하고자 하였다. 이는 현재 OpenCL, CUDA [2]와 같은 성과로 이어졌다.

GPU의 작동방식을 NVIDIA의 CUDA로 살펴본다. [3,4] GPU는 여러 개의 Streaming Multiprocessor(SM)을 포함하고 있다. 이런 SM은 또 다시 여러 개의 스칼라 프로세서로 구성된다. 이 스칼라 프로세서는 하나의 계산을 처리하는 하나의 처리 단위가 된다. 즉, 수없이 많은 프로세서가 GPU 내에 계층적으로 존재한다. CUDA에서 GPU를 이용하기 위해 최대 1024개의 스레드로 구성된 블록이라는 단위를 이용하는데, 이 블록들을 각 SM에 할당함으로써 계산을 병렬적으로 처리한다.

CPU의 사용에 더하여 이러한 GPU의 병렬성을 이용하면 기존의 데이터 집약시스템의 성능이 더욱 향상될 수 있다. 예를 들면, 수 많은 데이터를 GPU의 코어에 할당하여 병렬 처리를 수행함으로써, 적은 코어를 가진 CPU에 비해 데이터 처리에서 이점을 가진다. 이에 따라, 다양한 GPU 기반 데이터 집약 시스템들이 등장하였다.

다양한 GPU기반 데이터 집약 시스템이 등장하고 있지만, 그러한 시스템들이 무엇이 있고 어떤 특징을 가지는지 잘 알려지지 않아 적합한 시스템을 비교 및 선정하기가 어려운 실정이다. 따라서 어떤 시스템들이 있고 어떤 특징들을 갖는지 비교 분석하여 알아보고자 한다.

본 연구는 다음과 같이 구성된다. 2장은 다양한 데이터 집약 시스템들이 GPU를 어떻게 이용하였는지 살펴본다. 3장에서는 2장에서 소개한 여러 시스템들이 어떤 공통점과 차이점을 가지고 있는지 비교하여 본다. 마지막으로 4장에서는 위의 연구를 통한 결론을 맺는다.

2. 주요 GPU 기반 데이터 집약 시스템

본 장에서는 GPU 기반 데이터 집약 시스템들을 살펴본다. 특히 해당 시스템에서 GPU 를 이용한 구현사항들에 대해 집중하여 살펴보고자 한다.

2.1 PG-STROM

PG-STROM [5]은 GPU 를 이용하여 SQL 워크로드를 처리하기 위한 PostgreSQL 의 확장기능이다. 따라서 PostgreSQL 의 워크로드 중 GPU 처리에 적합한 것을 PG-STROM 에서 처리한다. Scan, Join, Aggregation 과 같은 연산들이 이에 해당한다. 위와 같은 연산들을 처리하기 위해 Query Optimizer 가 GPU 실행을 선택하면 기존의 PostgreSQL 을 대신하여 PG-STROM 모듈을 호출하고 SQL 워크로드를 처리한다. GPU 에

의해 처리될 워크로드는 쿼리에 따라 다양하다. 따라서 PG-STROM 은 실행시간에 GPU 프로그램(CUDA)을 자동으로 구성하고 빌드하여 실행한다.

PG-STORM 이 PostgreSQL 위에 자연스럽게 구현 될 수 있도록 다음의 세가지 특징을 가진다.

첫째, 쿼리의 변화가 없다. SQL 구문을 이용해 이미 분석된 내부 데이터 구조를 참조하기 때문에 GPU 실행을 위한 특별한 SQL 명령어가 필요하지 않다. 둘째, PG-STROM 은 PostgreSQL 의 데이터 스키마와 저장소 형식을 이용한다. 따라서 데이터를 특별한 테이블로 옮기거나, 특별한 데이터 형식으로 변환시킬 필요가 없다. 셋째, PG-STROM 은 PostgreSQL 에 의해 지원되는 표준 인터페이스 위에 구현된다.

2.2 MapD

MapD[6,7] 는 빅데이터를 활용하기 위해 설계된 플랫폼이다. 크게 MapD Core 와 MapD Immerse 로 나뉘어진다.

MapD Core 는 GPU 를 위해 설계된 오픈소스 SQL 엔진이다. 따라서 GPU 의 가속을 통하여 수십억 개 행의 쿼리를 밀리세컨드로 처리한다. 표준 SQL 을 기반으로 동작하며 MapD Immerse 와 상호작용을 한다. SQL 쿼리를 Just-In-Time LLVM(Low-Level Virtual Machine) 기반 컴파일러로 GPU 및 CPU 에서 실행할 수 있는 코드로 변환하여 메모리 엑세스 및 캐시에서 효율적인 코드 작성이 가능하다. 또한 지능적으로 메인 메모리와 GPU 의 VRAM 에서 사용빈도가 높은 데이터에 대해 캐싱을 수행한다. 따라서 다양한 하드웨어 구성과 데이터 집합 크기에서 뛰어난 성능을 보인다.

MapD 는 분산된 클러스터에 대하여 선형적으로 스케일아웃(Scale-out)이 가능하다. 기존의 MapD 의 단일 노드 환경에서 사용되었던 확장성을 이용한다. 하나의 노드에서 MapD 는 쿼리가 실행되면 각 GPU 는 다른 GPU 에 독립적인 전체 데이터의 한 단위를 처리하고 CPU 에 의해 데이터가 집계된다. 따라서 단일 기계에서 여러 GPU 를 사용하는 경우를 축소된 분산 환경으로 생각할 수 있다. 그 위에 집계계층(Aggregation Layer)을 추가함으로써 분산된 환경에서의 스케일아웃이 가능하다. 이 환경은 중앙 집중형 인덱스를 유지하지 않기 때문에 노드의 수에 따른 선형적인 스케일아웃이 가능하다.

MapD 는 시스템 메모리와 함께 GPU 의 VRAM 을 투명하게 관리하여 성능을 극대화한다. 필요에 따라 시스템 메모리와 VRAM 간에 데이터를 이동시킨다.

MapD Immerse 는 MapD core 와의 커뮤니케이션을 통해 시각화를 수행한다. 대량의 데이터에 대해 분석한 결과를 다양한 시각적 화면으로 볼 수 있다. 시각화 과정에서 CPU 로의 복귀 없이 GPU 에서 쿼리 결과를 직접 렌더링한다. 그 과정에서 3 계층 캐쉬를 이용하여 렌더링을 단순화시키고 가속시킨다.

2.3 BlzeGraph

BlazeGraph[8-10]는 큰 그래프 데이터를 처리하기 위한 시스템으로 초기 등장 이후 GPU 를 이용한 버전(BlazeGraph GPU)을 발표하였다.

그래프 연산 처리에서 큰 문제점들이 존재하였다. 데이터의 비-지역성(Non-Locality)과 의존성, 메모리 버스에서 병목현상이 일어나는 문제가 존재하였다. 이런 문제에 대해 GPU 를 사용해 10 배 이상의 메모리 대역폭을 이용할 수 있고 로드밸런스(Load Balance)를 유지할 수 있다. 의존성 문제를 해결하기 위해 그래프에 대한 태스크를 연산과 데이터 의존성으로 분해하였다. GP 를 이용하기 위해 CUDA 를 이용하는데 정점(Vertex) 중심의 API 를 이용하여 단순한 확장가능 추상화(Simple Extensible Abstraction)가 가능하다.

대규모 그래프에서 데이터가 어떻게 구성되어 있는지 알 수 없다. 따라서 기존의 파티셔닝(Partitioning)을 적용하기 어렵다. 이 문제에 대한 단순해결책은 모든 그래프를 메모리에 올리는 것이다. 이를 위해 GPU 를 이용하고 SPARQL 쿼리가 소프트웨어에 의해 변환되어 GPU 에 의해 실행되어 가능하다. 현재 최대 256 개의 GPU 를 지원하고 약 1 억개에 이르는 Edge 를 충족시킬 수 있다.

2.4 SQREAM

SQREAM[11]은 10TB 이상의 거대한 빅 데이터를 분석하기 위해 전체를 메모리에 올리지 않는방식을 채택한 GPU 기반 데이터 집약 시스템이다. GPU 를 이용하여 처리량이 우수하고 많은 테이블과 데이터를 결합 시 수동인덱싱 또는 사전집계 없이 가능하다.

데이터를 저장 시 칼럼 스토리지(Columnar Storage) 방식을 이용한다. 또 항상 데이터를 압축하여 이용한다. 인덱싱을 대신해 들어오는 데이터를 압축하고 태그하는 태깅시스템을 사용한다. 인덱스를 이용하지 않는 방식을 통해, 데이터의 조작 시 복잡한 인덱스 구조를 수정할 필요가 없어 데이터 수에 비례하여 선형적으로 처리시간이 증가한다.

SQREAM 은 두 가지의 파티셔닝을 특징으로 한다. 첫번째로, 수직(Vertical) 파티셔닝은 컬럼의 부분집합을 선택적으로 접근하여 디스크 스캔과 메모리 입출력을 줄인다. 수평(Horizontal) 파티셔닝은 자동으로 저장소를 관리할 수 있는 청크로 나눈다. 위와 같은 접근방식을 통해 제한된 VRAM 을 최대한 활용한다.

2.5 Kinetica

Kinetica[12]는 In-memory Database System 이다. 데이터를 처리할 때 인덱싱하지 않고 분산된 데이터를 수집 및 추출하여 준비한다. 따라서 적은 인덱스로 데이터가 쓰여지고 바로 사용할 수 있다. 데이터를 수집하고 분석하며 통합한다. 이구조를 통해 데이터의 수집 및 처리를 동시에 진행하여 즉각적인 결과를

도출한다. 인덱싱이 적은 간단한 데이터 구조를 가지고있어 시스템은 데이터 크기에 비례하여 확장된다. 단순히 GPU 의 컴퓨팅 성능을 이용해 인덱스 의존도를 감소시켰기 때문이다. 즉, 하드웨어 비용을 더 작고 예측 가능하게 할 수 있다.

시각화를 지원한다. 메모리에 분산된 이미지에 대해 OpenGL 렌더링 파이프라인을 이용하여 자체 시각화 도구를 이용할 수 있다. 특히 시간 및 공간 분석에 최적화된 시각적 환경을 제공한다.

3. GPU기반 데이터 집약 시스템의 비교

본 장에서는 시스템을 비교 분석하고자 한다. 우선 전반적인 비교사항을 표1에 기재하였다. 공통적으로 CUDA를 이용하여 GPU를 사용하였다. 이 연구에 언급되지 않은 여러 GPU기반 데이터 집약 시스템이 존재한다. 이런 시스템들 대부분 CUDA를 이용하였다.

인덱싱은 대부분 사용하지 않는다. 기존의 시스템에 GPU를 더한 시스템은 기존의 인덱싱을 사용한다. GPU를 위해 설계된 시스템들은 인덱싱을 적게 사용한다. 인덱싱을 사용하면 빠른 접근이 가능하지만 구조를 유지하기 위한 오버헤드가 커진다. GPU의 병렬계산능력을 이용하면, 인덱싱 없이 빠른 데이터 처리가 가능하여 인덱싱을 사용할 필요가 없다. 또한 장비의 확장이나 데이터의 증가에 따라 선형적으로 처리시간이 변하여 예측이 가능한 이점이 존재한다.

파티셔닝과 압축은 시스템마다 사용에서의 차이가 있다. SQREAM과 Blazegraph는 다양한 방법으로 데이터를 처리하였고 MapD는 무공유 구조(Shared Nothing Architecture)를 이용한다.

각 시스템마다 내새우는 특장점이 존재한다. 우선 PG-STROM 은 PostgreSQL 과의 호환성을 특징으로 한다. MapD 는 빅데이터에 대한 편리한 시각화 기능을 가지고 있고 SQREAM 은 대용량(10TB 이상)의 처리를 목적으로 한다. Blzaegraph 는 그래프 데이터 처리에 특화되어 있고 Kinetica 는 전반적인 속도 향상 및 시각화, 타 도구와의 호환성 등 다양화를 강조하였다.

표 1 GPU 기반 데이터 집약 시스템들의 비교

System	MapD	PG-STROM	SQreamDB	Kinetica	Blazegraph
indexing	X	PostgreSQL	X	X	O(B+tree)
partitioning	A shard per GPU	X	O (Vertical, Horizonal)	X	O (key range shard)
compression	O	X	O (Always-on)	O	O
commercial or open source	partially open source	Open source	Commercial	Commercial	Open source
CPU or GPU	hybrid or CPU-only	hybrid			
in-memory	In-memory (RAM)	PostgreSQL	Non-in memory	In-memory (RAM)	Non-in memory
kill application	Visualizing big data	add to installed PostgreSQL	Big data(>1TB) Analysis	Speed up and compatibility	Handling graph data

4. 결론

본 연구는 증가하는 데이터를 처리하기 위한 한가지 방법인 GPU를 이용한 시스템에 대하여 소개하였다. 소개하지 못한 다양한 시스템들이 존재하지만 본 연구에서는 큰 특징을 가지고 있거나 최근에 등장한 시스템을 위주로 소개하고자 하였다.

전반적으로 GPU의 병렬성을 이용하여 가속시키고 인덱싱을 배제하였다. 다만 시스템의 목적이 다르기 때문에 메모리 사용과 활용도에 대하여 차이를 보였다.

다양한 GPU기반 데이터 집약 시스템들이 등장하지만 각 시스템마다 다른 기준의 성능평가를 시행하고 있어 직접적인 비교에 어려움이 따른다. 본 연구에서는 기재하지 못하였지만 다양한 성능분석을 위한 프레임워크인 BenchGAD (**Bench**marking framework for **G**PU **A**ccelerated **D**ata Intensive system) 를 제안한바 있다. 본 연구를 통해 BenchGAD를 비롯한 다양한 연구를 위한 중요한 토대가 될 것으로 기대된다.

Acknowledgement

"이 논문은 2018년도 정부 (과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받은 신진연구자사업에 의해 연구되었음" (No. NRF-2018R1C1B6006409).

참고문헌

[1] M. Seacker, V. Markl, "Big Data Analytics on Modern Hardware Architectures: A Technology Survey", *Business Intelligence Second European Summer School*, pp. 125-149, 2012

[2] NVIDIA: CUDA Zone, https://developer.nvidia.com /cuda-zone (2018년 4월 접속)

[3] NVIDIA, "NVIDIA CUDA C Programming Guide", 2012

[4] NVIDIA, "NVIDIAs Next Generation CUDA Compute Architecture: Fermi", 2009

[5] Heterodb, http://heterodb.com (2018년 4월 접속)

[6] MapD, "MapD & IBM for General-Purpose Computing on GPUs", 2017

[7] MapD, "MapD: The Extreme Analytics Platform", 2018

[8] Blzaegraph, "Introduction to Blazegraph Database"

[9] P. Howard, "Blazegraph GPU Bloor In Detail", 2015

[10] B. Thompson, J. Lewis, "High Performance with High Level Languages Abstraction without regret?", *GPU Technology Conference*, 2016

[11] SQREAM, "SQream DB Technical Whitepaper: A Database Designed for Exponentially Growing Data", 2017

[12] Kinetica, http://kinetica.com (2018년 4월 접속)