

CS492 Distributed Systems & Algorithms:

Mr.CL

20030767 최종욱, 20050145 김준기, 20060080 김민국

2009년 12월 24일

1 Introduction

우리는 지난 학기 동안 분산 시스템의 기본적인 구조와 Map-Reduce, 그리고 분산 시스템 상에서 이용되는 여러 알고리즘에 대해 공부해왔다. 특히나 3가지 프로젝트를 통해 Hadoop을 이용한 Map-Reduce 프레임워크 기반으로 실제 프로그램을 구현해 보면서 이에 관해 더욱 깊은 이해를 할 수 있었다. 이를 통해서 알게 된 점은 분산 시스템이 대용량의 데이터를 처리하는데 무척 용이하다는 점이다. 우리는 이러한 이점을 잘 활용하여 가능한 한 의미 있는 결과를 얻기 위해 matrix multiplication이라는 주제를 선택하였다.

Matrix multiplication은 과학 연산과 데이터 마이닝 등 여러 분야에서 많이 이용되는 기본적인 연산이며 매우 방대한 크기의 matrix를 이용하는 경우가 자주 발생한다. 이 때문에 이를 최적화시키기 위한 많은 시도가 이루어져 왔다. 우리는 분산 시스템을 통해서 좀 더 대용량의 matrix를 가지고 multiplication을 수행하려 한다.

그러나 방대한 크기의 matrix를 다룰 수 있게 하는 것도 중요하지만, 연산 수행에 필요한 시간을 단축키는 노력도 함께 이루어져야 한다. 따라서 더 빠르게 계산을 수행할 수 있도록 matrix multiplication 과정에서 GPU를 이용하였다. GPU는 그 성격상 실수와 matrix 연산에 특화되어 일반 CPU보다 훨씬 더 나은 성능을 보여준다.

이러한 아이디어를 바탕으로 하여, 우리는 GPU를 이용한 분산 시스템 상에서의 Hadoop을 이용한 matrix multiplication 알고리즘을 설계하

였다. 분산 시스템을 통해서 대용량의 데이터에 대한 확장성(scalability)을 얻고, GPU를 통해 연산의 속도 향상을 이룸으로써 이전보다 개선된 matrix multiplication을 수행하고자 한다.

2 Design Overview

2.1 Assumptions

우리가 목표로 하는 시스템의 효과적인 설계와 구현을 위해 약간의 가정이 필요하다.

- Matrix의 각 원소들은 single precision floating point format을 통해서 표현된다.
- 각 matrix는 dense matrix이다.

2.2 Block Approach

Dense matrix multiplication을 수행할 때 I/O 병목에 의해 계산 속도가 제약되는 이유는 $O(n^3)$ 의 곱셈 횟수와 $O(n^2)$ 의 element를 불러오는 횟수가 matrix가 커질수록 큰 차이가 나기 때문이다. 특히 이러한 I/O 병목 문제는 multi-processor나 네트워크 기반 클러스터로 규모를 확대할수록 더욱 심각해지는데, 이는 프로세서끼리 혹은 클러스터 상의 컴퓨터끼리의 통신 대역폭이 제한되어 있기 때문이다. 따라서 우리는 프로세서의 성능을 최대한으로 활용하기 위해 프로세서가 처리할 데이터가 가능한 한 지속적으로 공급될 수 있도록, 혹은 데이터를 한 번 불러왔을 때 가능한 한 많이 반복

적으로 연산에 활용하는 알고리즘과 코드를 작성해야 한다.

이 문제를 해결하기 위해 우리는 matrix를 block 단위로 쪼개어 적절히 이를 분산시켜 multiplication을 수행하는 방법을 선택하였다. 만약 각 block들이 적당히 큰 상황이라면, 우리는 각 노드 상에서 이러한 block들의 각 element들을 한번 불러와 적당한 횟수만큼 multiplication에 다시 이용할 수 있을 것이다. 이때 각 block들은 일반적으로 square matrix지만 아닌 경우도 있을 수 있으므로 알고리즘에 따라서 적절한 block decomposition을 구현하는 것이 중요하다.

2.3 NVidia CUDA

실험에 사용된 클러스터 상의 모든 노드에는 NVidia의 그래픽 카드가 장착되어 있다. NVidia에서는 CUDA라는 기술을 통해 개발자들이 C 언어로 직접 GPU에서 동작하는 프로그램을 만들 수 있도록 하고 있다. 이것을 기반으로 BLAS API¹를 구현한 CUBLAS를 이용하면 matrix multiplication을 GPU를 이용하여 수행할 수 있다.

2.4 Apache Hama Project

TODO: Hama 프로젝트를 선택한 이유, 하지만 도입에 실패한 이유 설명 (HBase의 성능 제약 등)

3 Implementation

TODO: 최종적으로 Hadoop 위에 바로 코딩해서 올린 구조 설명

3.1 Data Model

3.2 Map-Reduce Execution

4 Performance Analysis

TODO: 성능 측정 계획/결과 설명

5 Conclusion

TODO: 부딪힌 문제점들 설명 (Hama와 HBase의 설치 중 삽질한 것, CUDA 드라이버 관련 삽질한 것, JCublas에서 double/float 삽질한 것 등)

TODO: 앞으로 개선할 점과 계속해서 연구한다면 어떤 것들을 해보고 싶다 등등

¹Basic Linear Algebra Subprograms. Vector-Vector, Vector-Matrix, Matrix-Matrix 연산들을 단계별로 정의하고 있다.