분산 시스템에서 점진적 갱신을 사용한 향상된 Prefetching

송경태, 김재형, 박상현
연세대학교 컴퓨터과학과
e-mail : skt8776@yonsei.ac.kr, { jaehyungkim, sanghyun } @ cs.yonsei.ac.kr

Enhanced Intra-block Prefetching using incremental update in Distributed System

Kyungtae Song, Jaehyung Kim, Sanghyun Park
Yonsei University, Computer Science Department

요 약

구글이 발표한 구글 파일시스템과 맵리듀스 논문 이후로 여러 오픈소스 분산시스템들이 나타남에 따라 상용시스템을 이용하지 않아도 대용량의 데이터를 처리 할 수 있는 서비스들이 등장하고 있다. 특히 분산 환경에서 병렬 처리를 제공하는 맵리듀스 프레임워크를 사용한 데이터 처리 방식은 가장 보편적인 개발 방법이다. 하지만 기존 상용 시스템에 비하여 처리 속도가 느린 단점이 있어 이를 보안하기 위해 학계 및 업계에서 많은 시도가 있어 왔다. 특히 데이터 지역성(data locality)를 향상시키기 위해 많은 시도들이 이루어져 왔고, 그 중 대표적인 것이 Prefetching 기법이다. 기존 Prefetching 기법에서는 Data의 Computation 시점과 정 반대의 시점에서 진행되므로, Prefetching이 완료되기 전까지 Prefetching의 이득을 얻을 수 없다는 단점이 있다. 그래서 본 논문에서는 Prefetching 방법을 개선하여 수행속도와 Prefetching속도에 따른 동적 prefetching 기법을 소개하고자 한다.

1. 서 론

가트너(Gartner)가 빅데이터 용어를 3V(Volume, Velocity, Variety)로 정의한 이후 빅데이터에 대한 관심이 뜨겁다. 특히 아파치(Apache) 재단[1]을 중심으로 다양한 오픈소스 분산 처리 시스템이 대중에게 공개되면서 많은 이들이 저렴한 비용으로 빅데이터를 처리할 수 있는 환경을 갖출 수 있게 되었다. 이로 인해 많은 연구자들 및 개발자들이 빅데이터 오픈소스 관련 연구, 개발을 하고 있다. 하둡(Hadoop)[2]은 대표적인 오픈소스 빅데이터 플랫폼으로서, 분산 처리시스템 MapReduce framework와 분산 파일 시스템 HDFS(Hadoop Distributed File System)로 구성된다. MapReduce는 수행해야 할 작업을 여러 개의 작은 단위로 쪼개어 서로 다른 서버에서 병렬적으로 연산을 할 수 있도록 해주는 프로그래밍 모델이다.

MapReduce의 작동방식은 다음과 같다. HDFS에 처리해야 할 데이터가 저장되어 있고, 그것을 나누어 놓은 것을 input split이라 부른다. Input split은 블록의 크기로 제한되고, 서버에서 수행되는 각 작업들 중 Map 작업을 수행하는 노드는 하나의 input split에 할당되어 연산을 수행하게 된다. Map 작업으로 생성된 결과물을 중간 결과물(Intermediate output)이라고 부르고, 중간 결과물들은 combine, shuffle의 작업을 거쳐 reduce 작업을 수행하는 노드로 전달된다.

데이터 Prefetching이란, 데이터 locality를 활용하기 위하여 연산에 쓰이고 있지 않은 데이터이더라도 추후에 쓰일 가능성이 있는 데이터를 미리 캐시 등으로 옮겨 놓는 것을 말한다. 데이터 locality를 활용하기 위한 다양한 시도들이 꾸준히 있었기 때문에 기존 연구들은 쉽게 접할 수 있다.. [2][3][4] 그 중 HPMR - Seo et al.[5]의 논문은 MapReduce Framework 상에서 Prefetching과 Preshuffling을 이용하여 MapReduce 연산 속도를 향상시킨 논문이다. 특히 논문에서 제시된 intra-block prefetching 방법은 Bi-directional prefetching으로써, prefetching으로 인한 부작용이 생기지 않는 방법이다. 하지만, 본 연구자는 부작용을 생각해서 만든 방법 때문에 prefetching으로 취할 수 있는 이득이 적어졌다는 것을 발견하게 되었다.

이에 본 연구는 prefetching의 부작용이 발생하지 않으면서 prefetching으로 얻을 수 있는 이득을 최대화 하는 방법에 대하여 다뤄보고자 한다. 수식을 통해 prefetching을 시작할 수 있는 적당한 지점을 계산하고, 그 계산된 지점부터 bi-directional prefetching이 아닌 one directional prefetching을 이용하여 최대한 많은 양의

※ 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2015R1A2A1A05001845).

데이터를 prefetch하고 더 빠른 수행속도를 보이는 향상된 prefetching 방법을 제시하고자 한다.

이 논문의 구조는 다음과 같다. 2장에서는 HDFS와 기존 논문 HPMR에서의 prefetching 방식에 대하여 살펴보고, 3장에서는 향상된 prefetching 방식에 대하여 소개한다. 마지막 4장에서는 논문의 결론과 향후 연구 방향을 제시한다.

2. 배경

2.1 HDFS

HDFS는 구글 파일 시스템을 본 따서 만든 빅데이터 처리 분산파일시스템으로서 Master/Slave 구조를 갖고 있다. HDFS 클러스터는 하나의 네임노드와 여러 개의 데이터 노드로 구성된다. 네임노드는 데이터 노드의 상태 관리 및 복구, 데이터 배치 등 클러스터를 관리하는 Master 서버의 역할을 한다. 데이터 노드는 Slave 서버 역할을 하며 각 서버의 저장장치를 다루는 기능을 한다. 데이터에 대한 HDFS에서 데이터를 저장할 때는 블록 단위로 나누어 저장하며, 하나의 블록은 128MB의 기본값을 갖는다. 또한 내고장성을 위하여 각 블록을 기본값(3개)만큼 복사하여 서로 다른 서버에 배치한다.

2.2 HPMR – bi-directional prefetching

HPMR - Seo et al.[3]은 MapReduce cluster 상에서 data locality를 향상 시키기 위하여 prefetching방법을 제시한 논문이다. Prefetching 방법으로 블록 내(intra-block) prefetching과 블록 간(inter block) prefetching이 제시되어 있었는데, 본 연구에서는 블록 내(intra-block) prefetching 만을 고려한다. HPMR[3] 논문에서 제시한 prefetching방법은 (그림 1)과 같이 bi-directional prefetching으로써, 한쪽에서는 복잡한 연산이 수행되고 있을 때 다른 한쪽에서는 추후에 필요한 데이터를 불러오는 방식이다. Intra-block prefetching에서 고려한 요소는 processing bar를 지속적으로 관찰하여 computation과 prefetching이 동기화 되도록 하는 것과 bi-directional 방식을 이용하여 불필요한 read와 네트워크 사용이 발생하지 않도록 하는 것이다.

3. 아이디어

3.1 향상된 prefetching 방법

기존 방식(그림 1)은 one directional prefetching을 진행하였을 때 Prefetching속도보다 연산 속도가 빨라지게 되면 Prefetching의 이점을 얻지 못하고, 오히려 네트워크 bandwidth만 소모하게 되므로 Bi-directional 하게 Prefetching을 진행하였다.

하지만 기존의 Bi-directional prefetching은 Prefetching이 끝난 이후에 Prefetching으로 얻은 이득을 활용하게 된다. 즉, Prefetching 진행 중에는 Prefetching된 데이터에 대하여 이득을 얻을 수 없다는 뜻이다. 반면 one direction으로 진행을 하되, prefetching 시점을 적당한 지점부터 수행하면 prefetching과 prefetched computation이 동시에 이루어지면서 항상 prefetching 하고 있는 부분이 연산이 진행되고 있는 부분 보다 앞서도록 할 수 있을 것이다. 그래서 본 연구에서는 Bi-direction이 아닌 one direction이더라도 부작용은 없고 bi-direction 보다 장점이 있는 방안을 제시한다.

Prefetching 시점을 좀 더 구체화 해보면, 연산이 수행되는 전체 시간과 prefetch가 진행되는 전체 시간이 같아지는 지점에서 prefetch를 시작하게 된다. Prefetching을 시작할 지점을 구하는 식은 다음과 같다.

$$\frac{S}{V_c} + \frac{1-S}{V_{pc}} = \frac{1-S}{V_p} \quad \text{- 식(1)}$$

V_c = prefetch 되지 않은 상태에서의 연산 속도
V_{pc} = prefetch된 상태에서의 연산속도
V_p = prefetch 하는 속도
S = prefetch가 시작되는 지점 (전체 크기를 1로 설정하였다.)

($V_{pc} > V_p$를 가정하였다.)

기존의 bi-directional 방법은 총 prefetch하는 양이 V_c와 V_p에 의해서 결정된다. 이를 수식화 해보면,

$$\frac{S'}{V_c} = \frac{1-S'}{V_p} \quad \text{- 식(2)}$$

이다.

식(1)의 해는

$$S = \frac{V_c * V_{pc} - V_c * V_p}{V_c * V_{pc} + V_{pc} * V_p - V_p * V_c}$$

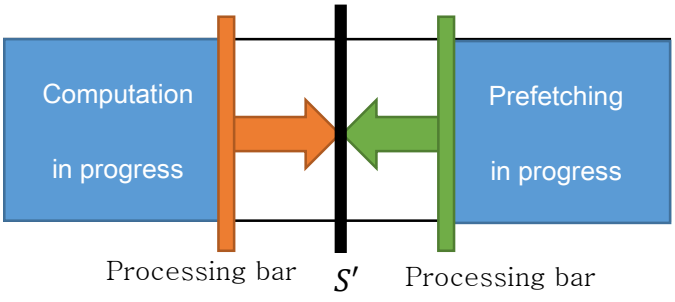
이고,

식(2)의 해는

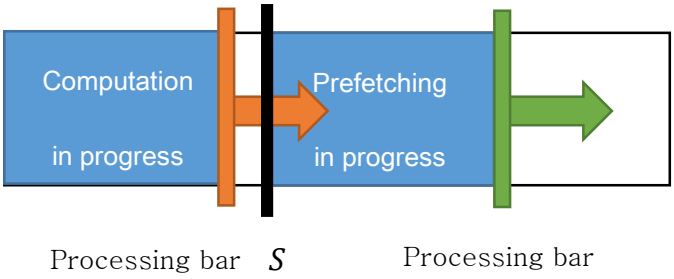
$$S' = \frac{V_c}{V_c + V_p}$$

이다.

여기서 $V_{pc} > V_p$ 이므로 S와 S'의 분모, 분자를 각각 ${V_p}^2, V_p$로 나누고, $\frac{V_c}{V_p}$와 $\frac{V_{pc}}{V_p}$를 각각 x,y (y < 1)로 치환하면 $S = \frac{xy-x}{xy+y-x}$, $S' = \frac{x}{x+1}$ 이 되어 $S < S'$임을 알 수 있다. 1-S, 1-S'이 각각 Prefetching 되는 부분 이므로 Prefetching 되는 부분이 기존 bi-directional 방법보다 많다는 것을 알 수 있다. 또한, 기존 방식보다 빠르게 prefetch된 데이터를 접할 수 있기 때문에 수행 시간 또한 빨라짐을 알 수 있다.



(그림 1) 기존의 bi-directional prefetching 방식



(그림 2) 개선된 prefetching 방식

3.2 Incremental Update

3.1에서의 식(1)을 풀려면 V_c, V_{pc}, V_p의 값을 알아야 한다. 이를 알기 위하여 본 연구에서 제시하는 방법은 Incremental Update이다. 방법은 다음과 같다.

1) 각 노드에서 처음 처리하는 블록에 대해서는 기존의 bi-directional 방식을 이용하여 prefetching을 처리한다.

2) 첫 블록의 prefetching이 끝나면 해당 블록을 처리하는데 걸린 시간을 이용하여 V_c, V_{pc}, V_p 의 값들을 각각 구한다.

3) 구해진 값을 이용하여 one directional 방식으로 prefetching을 수행한다. 이후 새로 얻어진 V_c, V_{pc}, V_p의 값들을 기존 값들에 더하여 평균치를 구한다.

4) 3)을 반복한다.

첫 블록을 처리할 때는 상수 값들을 알 수 없기 때문에 기존 prefetching을 사용한다. 그러므로 첫 블록에는 본 연구의 내용이 적용되지 않는다. 만일 이후의 블록들에서 computation processing bar와 prefetching processing bar가 만나는 순간이 오면 prefetching을 중단한다. 하지만 작업이 진행될수록 상수 값들이 점진적으로 갱신되어 보다 정확한 prefetching 지점을 구할 수 있게 된다. 하나의 MapReduce 작업은 128MB의 블록을 수천, 수만 개를 처리한다. 그러므로 점진적인 갱신을 통하여 결과로 얻어진 상수 값들을 계속 반영하면 효과적으로 유의미한 상수 값들을 얻을 수 있다.

4. 결론 및 향후 연구방향

Seo et al.[3]을 참고하여 HDFS, MapReduce 상에서 Data prefetching을 조금 더 효율적으로 할 수 있는 방법을 제시하였다.

기존의 bi-directional 방식이 아닌, one-directional 방식을 채택하여 prefetching된 데이터를 이용할 수 있는 시점을 빠르게 앞당겼고, 간단한 수학적 증명을 통하여 One-directional 방법이 bi-directional 방법과 비교하여 효용성이 있다는 것을 증명하였다. 또한, prefetching 시작 시점을 정하기 위하여 Incremental Update를 이용한 prefetching 방법을 고안하였다. Incremental Update를 통하여 점차 안정적인 prefetching 시작 지점을 찾을 수 있다.

HPMR[3] 논문에 따르면 prefetching 비율이 60%를 넘어가면 prefetching overhead로 인해 prefetching 비율이 높아짐에 따라 60%이상 구간에서 얻는 이득이 초반 0~60% 보다 낮다. 앞으로는 개선된 prefetching 방법을 구현함과 동시에 prefetching overhead를 줄일 수 있는 방안을 고려하여 HDFS, MapReduce 상에서 Data locality를 이용하여 보다 발전되고 효율적인 데이터 처리 방식을 구현함을 목표로 한다.

참고문헌

[1] The Apache Software Foundation https://www.apache.org

[2] Apache Hadoop https://hadoop.apache.org/

[3] C. L. Abad, Y. Lu and R. H. Campbell, "DARE: Adaptive Data Replication for Efficient Cluster Scheduling", Proc. of the International Conference on Cluster Computing, pp. 159-168, 2011.

[4] M. Zaharia, et al., "Delay Scheduling: Simple Techniques for Achieving Locality and Fairness in Cluster Scheduling", Proc. of the EuroSys'10, pp. 265-278, 2010.

[5] C.He, Y.Lu and D. Swanson, "Matchmaking: A New MapReduce Scheduling Technique", Proc. of the 3rd International Conference on Cloud Computing Technology and Science (CloudCom), pp. 40-47, 2011.

[6] Seo, S., Jang, I., Woo, K., Kim, I., Kim, J.S., Maeng, S.: HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment. In: IEEE International Conference on Cluster Computing and Workshops, CLUSTER 2009, pp. 1–8. IEEE (2009)