




연구논문/작품 제안서

2018 년도 제 1 학기

논문/작품	○논문(<input checked="" type="checkbox"/>) ○작품(<input type="checkbox"/>) ※ 해당란에 체크
제목	Alluxio Data Prefetching
GitHub URL	https://github.com/dlehddms1115/alluxio_prefetch
팀원명단	이동은  (학번: 2015313106)

2018 년 3 월 22 일

지도교수 : 한 환 수 서명 

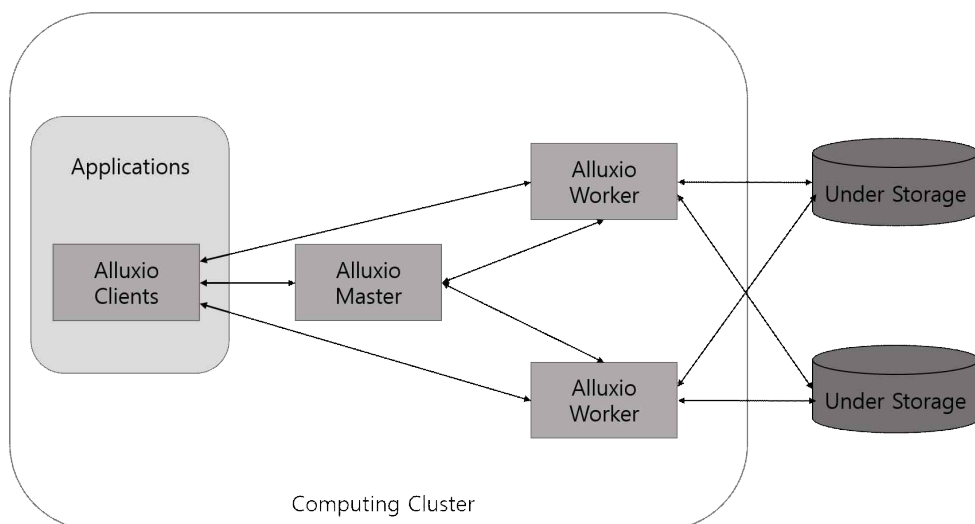
1. 과제의 필요성

1.1. Abstract

Alluxio는 Big Data를 위한 빠른 가상 스토리지 이다. Tachyon으로도 알려진 Alluxio는 메모리 중심적 가상 분산 스토리지 오픈 소스 시스템으로, 데이터 접근을 메모리 속도로 가능하도록 해주는 안정적인 스토리지이다. Alluxio는 파일 API와 커맨드라인 toolkit을 사용해 host의 RAM 공간에 접근할 수 있도록 한다. 분산 프레임워크인 Apache Spark는 Alluxio를 사용해서 데이터 접근 속도를 증가시킬 수 있다. 분산 프레임워크인 Apache Spark의 경우 많은 양의 데이터를 처리하는데 성능이 매우 중요하다. 때문에, Spark의 성능을 도와주는 Alluxio를 데이터 prefetching을 통해서 더 좋은 성능을 내도록 하는 작업이 매우 중요하다.

1.2. 서론

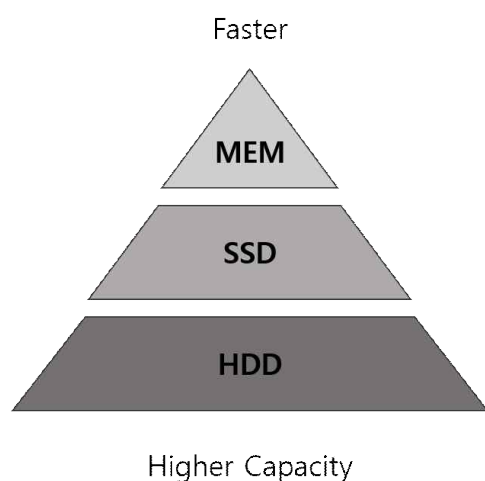
지금은 Big Data의 시대이다. 우리는 수많은 데이터를 생성하며, 이러한 데이터는 컴퓨팅 중심 플랫폼에서 급진적인 변화를 일으킨다. 계속 증가하는 데이터를 처리하기 위해 cluster computing이 널리 보급되고 있으며 map reduce 및 dyrad와 같은 데이터 집약적인 프레임워크가 제안된다. 그러나 흔히 사용되는 HADOOP map reduce는 복잡한 multi-stage의 처리와 interactive하고 ad-hoc한 쿼리들을 잘 처리하지 못하고 느리다는 단점이 있다. 이러한 파일 기반 데이터 통신은 과도한 I/O를 생성하여, 높은 성능의 병목의 원인이 된다. 이런 단점을 해결하기 위해 HDFS를 거치지 않고 RAM을 사용하는 SPARK라는 분산 프레임워크가 제안되었다. SPARK는 RAM을 read-only로 사용하여 fault에 대한 문제를 해결하고, HDFS를 거칠 때보다 빠른 속도를 자랑한다. 또한 data에 대해 수행되는 계산 순서를 나타낸



[그림 1] Alluxio 구조

그래프인 DAG를 이용해 lazy-execution으로 효율적인 계산이 가능하게 한다. Lineage(계보)로 대강의 execution plan이 다 정해진 뒤 계산을 시행하므로 자원이 배치되는 상황 등을 미리 고려해서 효율적인 계산이 가능하다는 장점이 있다.

Apache Spark는 Alluxio를 사용하여 데이터 접근 속도를 증가시켜 전체적인 성능을 증가시킬 수 있다. Alluxio는 스토리지 시스템과 computation framework와 어플리케이션 사이에 위치하여 메모리 중심적 디자인을 구성한다. Alluxio는 computational framework와 분산 스토리지가 분리되었을 때 가장 큰 효율을 보인다. Alluxio는 데이터 access의 공통 인터페이스를 제공하는 동시에 빅데이터 어플리케이션의 속도를 최대로 증가시키는 역할을 한다. Alluxio가 under 스토리지 시스템들을 어플리케이션이 모르도록 하기 때문에 모든 under 스토리지가 모든 어플리케이션과 framework를 커버할 수 있다.



[그림 2] Alluxio Tiered Storage

Alluxio는 tiered 스토리지를 제공하여 메모리뿐만이 아닌 다른 스토리지 타입도 커버할 수 있도록 한다. 현재 Alluxio는 [그림 2]와 같은 Mem, SSD, HDD의 3가지 유형의 스토리지를 제공한다. Alluxio의 높은 tier일수록 데이터를 읽어오는 속도가 빠르나, 용량이 적다. 많은 양의 데이터를 처리하는 데에 데이터를 읽어오는 속도는 전체 성능에 큰 영향을 끼치기 때문에 Alluxio내에서 사용할 데이터를 미리 높은 tier로 prefetch하는 것은 전체 어플리케이션의 성능 향상에 큰 도움이 될 것이다. Data를 읽는 시간이 오래 걸리는 SSD나 HDD에 저장된 데이터를 어플리케이션과 비동기적으로 MEM으로 읽어오도록 코드를 구성하여 성능의 증가를 확인할 것이다.

2. 선행연구 및 기술현황

2.1. C. Chen, T. Hsia, Y. Huang, and S. Kuo, "Scheduling-Aware Data Prefetching for Data Processing Services in Cloud", 2017 IEEE International Conference on Advanced Information Networking and Applications, 2017

Scheduling-Aware Data Prefetching (SADP) 기법을 제안해 데이터 prefetching과 데이터 eviction을 포함하여 대용량 클라우드 데이터 센터의 데이터 prefetching

문제점을 해결하였다.

2.2. O. Yildiz, A. Zhuo, and S. Ibrahim, "Eley: On the Effectiveness of Burst Buffers for Big Data Processing in HPC Systems", 2017 IEEE International Conference on Cluster Computing, 2017

HPC(High Performance Computing) 어플리케이션의 효율을 보장하면서 빅데이터 어플리케이션의 효율을 높이기 위해서 Eley라는 burst buffer solution을 제시하였다. Eley는 두 어플리케이션에서 input 데이터를 읽어오는 시간을 줄이기 위해 데이터를 prefetch하여 computing node의 근처에 저장하도록 하였다.

2.3. Y. Chen, H. Zhu, H. Jin, and X.-H. Sun, "Algorithm-level Feedback-controlled Adaptive data prefetcher: Accelerating data access for high-performance processors", Parallel Comput., vol. 38, no. 10-11, pp. 553-551, Oct. 2012.

Algorithm-level Feedback-controlled Adaptive (AFA) data prefetcher를 제안하여 기존의 prefetching이 어플리케이션의 접근 패턴이 변할 때 효율적이지 못한 점을 해결하였다.

2.4. Y. Chen, S. Byna, and X. H. Sun, "Data access history cache and associated data prefetching mechanisms", in Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007. SC '07, 2007, pp. 1-12

새로운 cache 구조인 Data Access History Cache(DAHC)를 제안하고 관련된 prefetching 기술을 연구했다. DAHC는 data prefetching의 이점을 얻는 현실적인 접근을 제시하고 관련된 prefetching 기술은 기존의 접근보다 더 효율적임을 증명했다.

2.5. 연구목적

Spark가 Alluxio를 사용함에 있어서 더 좋은 성능을 낼 수 있는 방법으로 성능 최적화를 하기 위해 data prefetching 기법을 사용하는 것이다. Data prefetching 없이 Alluxio를 사용했을 때와의 성능 비교를 통해 Data prefetching이 Spark의 성능을 얼마나 증가시키는 지를 실험을 통해 알아낼 것이다.

2.6. 접근방법

Spark에서 application을 실행할 때 각 job에 필요한 data들을 Alluxio에 미리 prefetching을 해 놓음으로써, Spark의 성능을 최적화한다. Alluxio에서 data prefetching을 쉽게 구현할 수 있으면 좋겠지만, Alluxio가 data prefetching에 대한 염두 없이 제작된 시스템이기 때문에, Alluxio 코드 내에 data prefetching을 구현하는 데에 어려움이 예상된다는 한계가 있다.

3. 작품/논문 전체 진행계획 및 구성 (2페이지 내외)

3.1. 선행 연구 탐색 및 분석

- 주제의 구체화를 위해 선행 연구들을 조사하여 배경 지식과 최신 동향 분석.
- 기본 개념 습득 및 관련 논문 정리.

3.2. 실험 환경 설정

- 리눅스 기반의 실험.
- 실험에 사용할 Spark와 Alluxio의 버전 선정.
- Alluxio SSD에 사용할 파일 시스템 선정.
- 실험에 사용할 workload 선정.

3.3. 구현 및 실행

- Data prefetch가 가능하도록 Alluxio 코드 수정.
- 수정한 Alluxio 코드와 기존의 Alluxio 코드를 사용하여 같은 workload에 대한 성능 실험.

3.4. 결과 확인 및 분석

- 성능 실험 결과에 대해 정리 및 그래프 작성.
- 그래프 동향에 대한 비교 및 원인 분석.

3.5. 개선 방안 제시

- 실험 결과에 대한 개선 방안 탐구.
- 개선안 제시를 위해 추가 자료 조사 및 정리.

[표 1] 월별 졸업논문 진행 계획

	3	4	5	6	7	8	9	10	11
Github									
제안서, 서약서									
자료 조사 및 주제 구체화									
실험									
결과분석									
논문 중간보고서									
논문작성									
최종보고서									
논문 발표회									

4. 기대효과 및 개선방향

MEM에서 data를 읽어오는 속도와 SSD나 HDD에서 data를 읽어오는 속도는 현저하게 큰 차이가 난다. 때문에 자주 사용하는 data를 SSD나 HDD에서 MEM으로 prefetch를 하도록 수정한 Alluxio 코드를 통한 실험에서 기존의 Alluxio로 실험하였을 때의 성능 차이가 위에서 언급한 MEM에서 data를 읽어오는 것과 스토리지에서 data를 읽어오는 것과 같은 정도의 성능 차이가 나는 것을 기대한다.

Alluxio의 코드가 data prefetching을 고려하고 만들어진 코드가 아니기 때문에 Alluxio 코드를 수정하여 data prefetching을 구현하는 데에 많은 어려움이 예상되지만, 기존의 코드를 잘 분석하여 prefetching이 가능하도록 코드를 수정한다면 후에 자주 사용하는 data를 고르는 방식을 더 최적화하여 좋은 성능이 나올 수 있을 것으로 기대한다.

5. 참고문헌

- [1] C. Chen, T. Hsia, Y. Huang, and S. Kuo, "Scheduling-Aware Data Prefetching for Data Processing Services in Cloud", 2017 IEEE International Conference on Advanced Information Networking and Applications, 2017
- [2] O. Yildiz, A. Zhuo, and S. Ibrahim, "Eley: On the Effectiveness of Burst Buffers for Big Data Processing in HPC Systems", 2017 IEEE International Conference on Cluster Computing, 2017
- [3] Y. Chen, H. Zhu, H. Jin, and X.-H. Sun, "Algorithm-level Feedback-controlled Adaptive data prefetcher: Accelerating data access for high-performance processors", *Parallel Comput.*, vol. 38, no. 10-11, pp. 553-551, Oct. 2012.
- [4] Y. Chen, S. Byna, and X. H. Sun, "Data access history cache and associated data prefetching mechanisms", in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, 2007. SC '07, 2007, pp. 1-12
- [5] Alluxio – Open Source Memory Speed Virtual Distributed Storage
<https://www.alluxio.org/overview/architecture>