스트림 데이터 처리를 위한 시계열 데이터베이스 병렬 접근 기반 읽기 지연 개선 기법

황용하*, 노순현*
*서울대학교 전기 · 정보공학부
e-mail : {yhhwang, shnoh}@redwood.snu.ac.kr

Improving Read Latency for Stream Data Processing via Parallel Access of Time Series Database

Yong-Ha Hwang*, Soon-Hyun Noh*
*Dept. of Electronic Computer Engineering, Seoul National University

요 약

시계열 데이터 처리를 위해 방대한 양의 데이터를 스토리지에서 빠르게 읽어와 처리하려는 움직임이 많아지고 있다. 이를 위해 스토리지의 read latency 를 개선하기 위한 여러 기법들이 제안되었지만, 이 기법들은 분산 노드의 스토리지 자원을 충분히 활용하지 못한다는 한계가 있다. 따라서 우리는 시계열 데이터를 실시간으로 처리하기 위해 스토리지에 병렬적으로 접근하여 read latency 를 개선하는 기법을 제안한다. 제안된 기법은 분산 환경에서 스토리지에 병렬적으로 접근하여, 각 노드에서 부분적으로 데이터를 읽어와 전체 데이터를 읽어오는 지연시간을 줄인다. 우리는 제안된 기법을 여러 노드로 구성된 분산 환경에서 구현하였다. 제안된 기법을 적용한 결과, 전체 데이터를 읽어오는 read latency 가 기존 기법보다 28.04% 줄어든 것을 확인하였다.

1. 서론

최근 자율주행 자동차와 사물 인터넷(IoT, Internet of Things) 기술이 발전하면서 센서에 대한 수요가 증가하고 있다[1]. 스마트폰을 비롯한 모바일 기기에서도 다양한 센서를 활용하면서 센서에 의해 생산된 방대한 양의 시계열 데이터를 저장하고 관리할 데이터베이스가 필요하게 되었고, 이러한 시계열 데이터를 효율적으로 관리하기 위한 데이터베이스인 InfluxDB, OpenTSDB, Graphite 등이 등장하였다.

시계열 데이터를 처리를 위해서는 시계열 데이터 프레임워크가 필요하다. 시계열 데이터 프레임워크는 그 구조가 계층화되어 있으며, 스토리지 계층과 처리 계층, 2 개의 계층으로 구성되어 있다[2]. 스토리지 계층은 시계열 데이터를 저장하여 읽고 쓸 수 있는 환경을 지원하고, 처리 계층은 스토리지 계층에 저장된 데이터를 사용하여 시계열 데이터를 처리하는 역할을 수행한다.

처리 계층에서 시계열 데이터 처리를 위해 스토리지 계층의 데이터를 읽어 오는 것은 프로그램의 성능과 밀접한 연관이 있다. 스토리지에서 데이터를 읽어오는 데 소요되는 지연 시간인 read latency 를 개선하는 것은 전체 프로그램의 성능을 좌우하는 중요한 문제이며, 이를 개선하기 위한 연구들이 많이 고안되었다. 이들 연구는 두 가지로 분류된다. 첫째, 스토리지 내부 성능을 개선하는 방식이다. 이는 스토리지의 자체의 구성을 변경하거나 스토리지 동작 방식의 변화를 통해 성능을 향상시켜 read latency 를 줄이는 방식이다. Payer 는 SSD 와 HDD 를 모두 사용하는 이기종 스토리지인 Combo Drive 를 제안하였다[3]. Combo Drive 는 파일 유형에 따라 저장 장소를 다르게 설정함으로서 스토리지의 성능을 개선한다. 하지만 적용 환경에 따라 성능의 차이가 발생한다는 문제가 있다. Wu 는 SSD 사용 환경에서 read latency 를 줄이기 위해 light-overhead P/E (Program/Erase) 방식을 제안하였다[4]. NAND Flash 에서 P/E 경합으로 인한 read latency 문제를 해결하기 위한 것이다. 하지만 적용 범위가 SSD 에 국한되는 문제가 있다. 둘째, 스토리지 외부에서 스토리지 활용 기법 변경을 통해 read latency 를 줄이는 방식이다. Lloyd 는 클라우드 환경에서 로컬 스토리지에 데이터를 복사해 놓고, 로컬 스토리지에서 캐싱을 통해 read lantecy 를 줄일 수 있는 방법을 사용하였다[5]. 하지만 로컬 스토리지 설치를 위한 비용이 발생하고, 데이터의 일관성을 보장하기 위한 오버헤드가 추가된다는 문제가 있다.

하지만 기존 연구들은 단일 스토리지에서의 성능 향상 만을 고려할 뿐, 분산 환경으로 구성될 경우 read latency 개선을 위해 모든 분산 노드에서의 스토리지 자원을 충분히 활용하지 못한다는 문제가 있다. 이와 같은 기존 연구의 문제점을 해결하기 위하여, 본 논문에서는 로드 밸런서의 동작 방식을 수정하여 시계열 데이터베이스에 병렬적으로 접근해 데이터를

읽어오는 방법을 제안한다. 구체적으로, 읽어야 하는 전체 데이터를 분할하여 각각의 시계열 데이터베이스 노드에서 부분적으로 읽는 방식을 통해 분산 환경에서 read latency 를 개선하기위한 기법을 제안한다.

우리는 제안한 기법을 검증하기 위해 기존의 시계열 데이터베이스 접근 방식을 수정하였고 일련의 실험을 통해 read latency 가 기존 기법에 비해 17.12% 개선된 것을 확인하였다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2 장에서는 스트림 데이터 처리를 위한 분산 환경과 시계열 데이터를 관리하기 위한 시스템에 대해 설명한다. 3 장에서는 대상 시스템과 본 논문이 해결하고자 하는 문제를 정의한다. 4 장에서는 문제를 해결하기 위한 기법에 대해 자세히 설명하고, 5 장에서 제안하는 기법을 적용하기 위한 실험 환경에 대해 설명하고 적용 결과를 분석한다. 마지막으로 6 장에서 논문의 결론을 맺는다.

2. 배경 지식

본 장에서는 스트림 데이터 처리를 위한 분산 환경과 시계열 데이터를 관리하는 시스템에 대해 설명한다.

2.1 Spark Streaming

Spark 는 분산 처리와 머신 러닝 분야에서 다양하게 활용되고 있는 빅데이터 처리를 위한 병렬 분산 처리 플랫폼이다[6]. Spark Streaming 은 Spark 에서 동작하는 라이브러리 중 하나로서, 마이크로 배치 처리 기능을 이용하여 실시간 스트림 데이터 처리를 구현한 라이브러리이다. Spark Streaming 은 시계열로 연속해 생성되는 센서, 웹 시스템 데이터를 실시간으로 처리하기에 적합하며, 데이터의 실시간 통계 처리가 필요한 곳에 경향 분석을 위해 주로 사용된다.

2.2 HAProxy

HAProxy 는 HA (High Availability), 로드 밸런싱 기능을 제공하는 로드 밸런서로서 TCP 와 HTTP 기반 애플리케이션을 중계한다[7]. HAProxy 는 frontend 와 backend 계층으로 구성되어 있으며, frontend 는 클라이언트와 통신하고 데이터의 포맷을 검증한다. B ackend 는 서버와 통신하고 서버의 상태를 관찰하여 서버간 로드를 조절한다. HAProxy 는 사건 기반(event-driven) 방식으로 동작하며, 클라이언트로부터 데이터가 수신되면 설정된 로드 밸런스 방식에 따라 서버에 데이터를 분배한다.

2.3 InfluxDB

InfluxDB 는 InfluxData 에서 개발한 시계열 데이터베이스로서 대용량 시계열 데이터를 저장하기 위한 스토리지 관리 기법과 빠른 시계열 데이터 검색 기능을 제공한다[8]. InfluxDB 는 여러 노드로 구성할 수 있으며, 각 노드는 InfluxDB-Relay 를 통해 서로 연결된다. InfluxDB 에 시계열 데이터를 저장하면 InfluxDB-Relay 를 통해 모든 노드에 동일한 데이터가 저장되어 관리된다.

3. 대상 시스템과 문제 정의

본 장에서는 대상 시스템에 대해 설명하고, 시계열 데이터베이스 접근 과정에 대한 분석을 통해 논문에서 해결하고자 하는 문제를 정의한다.

3.1 대상 시스템

본 논문의 대상 시스템 구성은 그림 1 과 같이 세 부분으로 구성되어 있다. 첫째, 시계열 데이터를 읽고 처리하기 위한 실시간 시계열 데이터 분석 프레임워크, 둘째, 클라이언트와 서버를 중계해주는 로드 밸런스 프레임워크, 마지막으로 시계열 데이터를 저장하고 관리하는 시계열 데이터베이스로 구성된다.

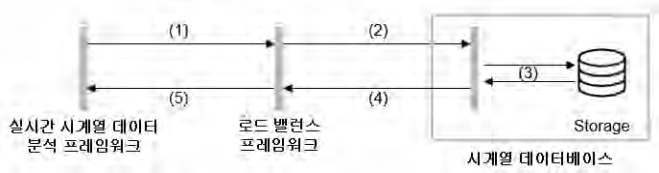
실시간 시계열 데이터 프레임워크에서는 실시간으로 시계열 데이터를 분석하고 처리하기 위해 Spark Streaming 을 사용한다. Spark Streaming 은 실시간 시계열 데이터 분석을 위한 Spark Driver 와 시계열 데이터베이스인 InfluxDB 를 연결 해 주기 위한 Spark-InfluxDB Connector 로 구성된다. Spark Driver 에서 시계열 데이터 분석을 위해 읽기 명령 (read operation)을 보내면 Spark-InfluxDB Connector 에서 InfluxDB 문법에 맞는 읽기 요청(read query)으로 변경하여 데이터를 송신한다.

로드 밸런스 프레임워크는 오픈 소스이며 HA 를 지원하는 HAProxy 를 사용한다. HAProxy 는 클라이언트로 실시간 시계열 데이터 분석 프레임워크를, 서버로 시계열 데이터베이스를 설정하여 각 서버 노드의 로드를 관리하고, 클라이언트의 요청을 서버로 전달한다.

시계열 데이터베이스는 대용량 시계열 데이터를 효율적으로 관리하기 위한 다양한 기능을 제공하는 InfluxDB 를 사용한다. InfluxDB 는 여러 개의 데이터베이스 노드로 구성되어 있으며, 각 노드들은 독립적인 PC 로 구성되어 네트워크를 통해 서로 연결되어 있다. 여러 개의 데이터베이스 노드들은 내결함성을 위해 동일한 데이터를 복제하여 저장한다.

(그림 1) 대상 시스템

3.2 시계열 데이터베이스 접근 과정 분석



(그림 2) Read Latency 분석

실시간 시계열 데이터 분석 프레임워크에서 시계열 데이터를 읽어 오기 위해 시계열 데이터베이스로 읽기 요청을 전송한 순간부터, 로드 밸런스 프레임워크를 거쳐 시계열 데이터베이스에서 해당 데이터를 읽어오는 과정은 그림 2 와 같다. 각각의 과정에서 소요되는 read latency 는 아래와 같이 5 단계로 분류할 수 있다.

(1) 실시간 시계열 데이터 분석 프레임워크에서 로드 밸런스 프레임워크로 읽기 요청이 전달되는 latency
(2) 로드 밸런스에서 로드 밸런싱을 통해 시계열 데이터베이스 노드에 읽기 요청이 전달되는 latency
(3) 스토리지에서 데이터를 읽어오는 latency
(4) 읽어온 데이터를 로드 밸런스 프레임워크로 전달하는 latency
(5) 읽어온 데이터를 실시간 시계열 데이터 분석 프레임워크로 전달하는 latency

3.3 문제 정의

분산 시스템 환경에서, 스토리지에 접근하여 데이터를 읽어 오는 read latency 를 개선하는 것은 전체 시스템의 성능을 좌우하는 중요한 문제이다. 따라서 시스템의 성능 향상을 위해서는 read latency 의 분석 및 개선 과정이 필요하다.

그림 1 에서 시계열 데이터 분석을 위해 실시간 시계열 데이터 분석 프레임워크에서 읽기 요청을 보내면, 로드 밸랜스 프레임워크는 연결된 시계열 데이터베이스 노드들 중 로드가 적은 노드에 접근하여 요청된 데이터를 읽어온다. 이 때, 로드가 적은 하나의 노드에만 접근하여 데이터를 읽어오기 때문에 다른 노드들의 스토리지 자원에 여유가 있는 상황에서도 이를 활용하지 못하는 문제가 있다. 이러한 현상은 read latency 를 지연시켜 성능의 병목 지점이 된다.

Read latency 개선을 위해서는 그림 2 의 (1) ~ (5) 과정에서 소요되는 시간을 단축해야 한다. (1)과 (2)는 네트워크 전달 과정이므로 read latency 개선이 어렵다.

하지만 로드 밸런스 프레임워크 동작 방식을 데이터베이스 노드의 스토리지에 병렬적으로 접근하도록 수정함으로서, 스토리지에서 데이터를 읽어오는 (3)과, 읽어온 데이터를 네트워크를 통해 실시간 시계열 데이터 분석 프레임워크로 전달하는 과정인 (4), (5)의 read latency 를 개선할 수 있다. 우리는 제안하는 시계열 데이터베이스 병렬 접근 기법을 통해 read latency 를 개선하고자 한다.

4. 시계열 데이터베이스 병렬 접근 기법

제안된 기법은 로드 밸런서인 HAProxy 의 동작 방식을 수정한다. 이를 위해 기존 HAProxy 의 동작 방식을 파악하고 수정된 HAProxy 의 동작 메커니즘에 대해 설명한다.

기존 HAProxy 는 사건 기반 방식으로 동작하며 클라이언트로부터 읽기 요청이 도착하면 이를 처리하기 위한 별도의 태스크를 생성한다. 생성된 태스크는 로드 밸런스 과정에서 config file 을 분석하여 backend 서버로 등록된 시계열 데이터베이스인 InfluxDB 노드를 확인하고, 로드가 작은 노드를 선택하여 읽기 요청을 전달하는 방식으로 동작한다. 하지만 이는 3.3 장에서 언급한 바와 같이 분산 노드에 여유가 있는 스토리지 자원을 충분히 활용하지 못한다는 문제가 있다.

제안된 기법은 로드 밸런스 과정에서 여러 개의 태스크를 생성하여, backend 서버로 등록된 InfluxDB 노드에 병렬적으로 접근하도록 한다. 각 태스크는 실시간 시계열 데이터 분석 프레임워크에서 요청한 데이터를 InfluxDB 노드에 접근하여 읽어오도록 수정한다.

구체적으로, 실시간 시계열 데이터 분석 프레임워크에서 시계열 데이터 처리를 위해 InfluxDB 에 요청하는 전체 데이터를, HAProxy 에서 생성된 각 태스크에서 InfluxDB 노드에서 병렬적으로 접근하여 동시에 부분적으로 읽어와 전체 데이터를 읽어오는 read latency 를 개선한다.

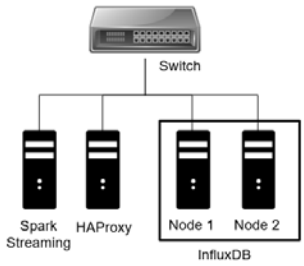
결과적으로 클라이언트인 실시간 시계열 데이터 분석 프레임워크는 전체 데이터를 서버인 InfluxDB 에 요청하고, HAProxy 는 이를 처리하기 위해 여러 개의 태스크를 생성하여 backend 서버에 등록된 InfluxDB 노드에서 병렬적으로 데이터를 읽어온다. 이를 통해 분산 시스템에서 여유 있는 스토리지 자원을 활용하여 read latency 를 개선할 수 있다.

5. 실험 및 검증

본 장에서는 제안된 기법을 적용하기 위한 실험 환경과 실험 결과에 대해 설명한다.

5.1 실험 환경

제안된 기법을 평가하기 위해 총 4 대의 PC 를 사용하였으며, 실험 환경 구성은 그림 3 과 같다. 네트워크의 의한 지연을 최소화하기 위해 하나의 스위치 아래 모든 PC 를 연결시켜 실험 환경을 구성하였다.



(그림 3) 실험 환경 구성

각 PC 의 성능은 표 1 과 같다.

<표 1> 실험 환경

	노드	Spark Streaming, HAProxy	InfluxDB
하드웨어	CPU	Intel i7-4770	Intel i7-3770
	Memory	32GB	6GB
	Storage	SSD 500GB	HDD 1TB
소프트웨어	OS	Linux 16.04.1	Linux 16.04.1
	program	Spark Streaming 2.2	InfluxDB 1.4
		HAProxy 1.6.3	

실험을 위한 워크로드는 NOAA (National Oceanic and Atmospheric Administration) Center 에서 제공하는 시계열 데이터를 사용하였다. 이 워크로드는 해수의 온도와 높이 정보를 제공하며, 6 초 간격으로 측정한 15,258 개의 데이터가 존재한다.

실험은 2 개의 InfluxDB 노드에서 전체 해수 정보를 기존 기법으로 읽어오는 데 걸리는 read latency 와 제안된 기법으로 읽어올 때의 read latency 를 각각 5 회 측정하여 평균을 비교하였다. Read latency 는 그림 2 에서 설명한 바와 같이 각 단계별로 분류하여 측정하였다.

5.2 실험 결과

<표 2> Read latency 측정 결과

Read latency	기법 미적용 [ms]	기법 적용 [ms]
(1)	1.213	1.7
(2)	4.06	3.27
(3)	594.683	460.652
(4)	7.054	6.54
(5)	9.134	9.053
Total	616.144	481.215

표 2 는 실험 결과를 나타낸다. 제안된 기법을 적용한 경우 두 개의 InfluxDB 노드에서 데이터를 읽어오는 데 소요된 read latency 를 표시하였다.

실험 결과, 기존 기법은 전체 해수 정보를 읽어오는데 총 616.144ms 의 시간이 소요되었다. 반면 제안된 기법을 사용한 결과, (3), (4), (5) 단계에서 read latency 가 감소하였으며, 최종적으로 전체 해수 정보를 읽어오는 데 481.215ms 가 소요되어 기존 기법에 비해 28.04%의 read latency 개선을 보였다.

6. 결론

본 논문에서는 분산 컴퓨팅 환경에서 read latency 개선을 위한 시계열 데이터베이스 병렬 접근 기법을 제안하였다.

기존 기법은 분산 환경에서 하나의 시계열 데이터베이스 노드에만 접근하여 전체 데이터를 읽어와, 다른 노드의 스토리지 자원을 활용하지 못해 read latency 가 지연되는 문제가 있었다. 제안된 기법은 연결된 시계열 데이터베이스에 병렬적으로 접근하여 전체 데이터를 부분적으로 읽어오도록 수정함으로서 read latency 를 개선하였다. 본 연구진은 시계열 데이터 분석 환경에서 제안된 기법을 적용 후 기존 기법보다 read latency 가 28.04% 개선된 것을 확인하였다.

Acknowledgement

본 연구는 중소벤처기업부의 기술혁신개발사업의 일환으로 수행하였음. [S2392603 , IoT 시계열 빅-데이터 실시간 저장 분석 및 시각화 플랫폼 기술 개발]

참고문헌

[1] https://news.kotra.or.kr/user/globalBbs/kotranews/4/globalBbsDataView.do?setIdx=243&dataIdx=35413
[2] https://aws.amazon.com/ko/streaming-data/
[3] Payer, Hannes, et al. "Combo drive: Optimizing cost and performance in a heterogeneous storage device." *First Workshop on Integrating Solid-state Memory into the Storage Hierarchy*. Vol. 1. No. 1. 2009.
[4] Wu, Guanying, and Xubin He. "Reducing SSD read latency via NAND flash program and erase suspension." *FAST*. Vol. 12. 2012.
[5] Lloyd, Wyatt, et al. "Stronger Semantics for Low-Latency Geo-Replicated Storage." *NSDI*. Vol. 13. 2013.
[6] Zaharia, Matei, et al. "Spark: Cluster computing with working sets." *HotCloud* 10.10-10 (2010): 95.
[7] www.haproxy.org/download/1.6/doc/intro.txt
[8] https://docs.influxdata.com/influxdb/v1.0/concepts/storage_engine/