Spark Streaming 기반 시스템에서 실시간 고장 복구를 지원하기 위한 고장 복구 시간 추정과 체크포인팅 기법

김정호[1)O], 양범준[2)], 이은성[2)], 홍성수[1), 2)]
[1)]서울대학교 융합과학기술대학원 융합과학부
[2)]서울대학교 전기정보공학부
{jhkim, bjyang, eslee, sshong}@redwood.snu.ac.kr

Fault Recovery Time Estimation and Checkpointing Technique for Real-Time Fault Recovery in Spark Streaming System

Jungho Kim[1)O], Beomjoon Yang[2)], Eunseong Lee[2)], Seongsoo Hong[1),2)]
[1)] Department of Transdisciplinary Studies, GSCST, SNU
[2)]Department of Electrical and Computer Engineering, SNU

요 약

세계 유수의 OEM들이 집중적으로 연구 중인 자율주행 자동차 실현을 가능케 하는 기술들 중 하나는 클라우드 연결 자동차이다. 이를 위해 실시간성을 보장하는 클라우드가 필수적이며 어려운 실시간 요구사항들을 충족해야 한다. 이를 지원하기 위해서는 데이터 분석 프레임워크가 필수적이며 현존하는 기술들 중에 가장 많은 요구사항들을 충족하는 것은 Spark Streaming이다. 하지만 이 프레임워크는 언급된 요구사항들 중 하나인 실시간 고장 복구를 충족하지 못하고 있다. Spark Streaming의 고장 복구 기법은 우선 런타임에 상태의 변형 히스토리를 기록해둔다. 고장 발생 시 그것에 기반하여 손실된 상태를 재연산하여 복구한다. 하지만 히스토리의 길이에 비례하여 재연산 시간이 증가되기 때문에 바운드된 복구 시간이 보장되지 않는다. 본 논문에서는 이러한 문제를 해결하기 위한 고장 복구 시간 추정과 체크포인팅 기법을 제안한다. 이 기법은 런타임에 변형 히스토리에 기반한 고장 복구 시간을 추정하고 시간 제약 충족 여부에 따라 해당 상태를 고장 감내 스토리지에 체크포인팅한다. 이를 Spark Streaming 1.4.1에 적용하고 실험을 통해 고장 복구 시간이 시간 제약 이내로 바운드 됨을 확인하였다.

1. 서 론

세계 유수의 자동차 OEM들이 집중적으로 연구 중인 자율주행 자동차 실현을 가능케 하는 기술들 중 하나는 클라우드 연결 자동차이다[1](cloud connected vehicle). 이를 위한 실시간 클라우드 데이터센터는 차량에게 자율주행에 관련된 실시간 서비스를 제공하기 위해 다양하고 어려운 요구사항들을 충족해야만 한다. 이 요구사항들로는 실시간 스트림 데이터 처리, 실시간 상태 기반(stateful) 작업, 실시간 고장 복구가 있다.

이러한 요구사항들을 충족하기 위해서는 데이터 분석 프레임워크가 필수적이며 현존하는 기술들은 배치와 스트림 프로세싱으로 구분된다. 전자에 해당하는 대표적인 것으로는 Hadoop이 있으며, 후자에는 Spark Streaming, Storm이 있다[2][3][4]. 표 1은 언급된 요구사항들에 대한 현존 프레임워크들의 충족 여부를 나타낸다. Hadoop과 Storm은 언급된 요구사항들을 충족하지 못하는 반면 Spark Streaming은 부분적으로 충족하며 이에 관련하여 개선될 필요가 있다. 이러한 요구사항들 중에서 본 논문은 Spark Streaming 기반 시스템에서 실시간 고장 복구를 충족하기 위한 기법을 다룬다.

표 1. 현존 프레임워크의 적합성 평가 요약

요구사항	Hadoop	Spark Streaming	Storm
실시간 스트림 데이터 처리	불충족	부분충족	불충족
실시간 상태 기반 작업	불충족	부분충족	불충족
실시간 고장 복구	불충족	부분충족	불충족

이러한 데이터 분석 프레임워크에서 고장 복구를 위해 여분의 자원을 확보하기 위한 많은 기법들이 제안되어 왔다[5]. 이 기법들은 HW 레벨 또는 SW 레벨로 구분 된다. HW 레벨 기법들은 같은 작업을 수행하는 여분의 컴퓨팅 서버를 추가로 설치하는데, 이는 높은 설치 및 유지보수 비용을 서비스 제공자에게 부담시킨다. SW 레벨 기법들은 상태의 복제본 백업과 상태의 변형 히스토리 백업 기법으로 구분된다. 전자는 상태가 갱신 될 때마다 서버들에 분산된 여러 개의

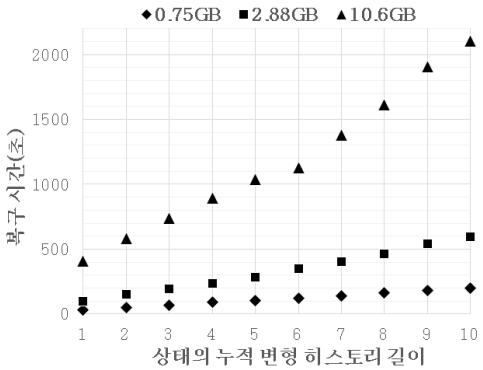

그림 1. 누적 변형 히스토리 길이에 따른 복구 시간

복제본들을 동시에 갱신하고 동기화한다. 하지만 이는 정상 동작 시에 클라우드 내 네트워크 과부화와 동기화 오버헤드를 크게 야기한다. 후자는 작은 크기의 변형 히스토리만을 저장하고 관리하여 고장 시에만 그 변형 히스토리에 기반하여 상태를 재연산하고 생성하기 때문에 앞서 언급된 오버헤드들을 효과적으로 줄인다. Spark Streaming의 고장 복구 기법은 이 분류에 속한다. 하지만 누적 변형 히스토리의 수가 증가될수록 비례하여 복구 시간이 증가되기 때문에 복구 작업이 바운드된 시간(bounded time) 내에 처리되지 못한다. 그림 1은 Spark Streaming 기반 시스템에서 0.75, 2.88, 10.6GB 크기의 상태들에 대한 복구 시간이 누적된 변형 히스토리 길이에 비례하여 증가됨을 나타낸다.

본 논문에서는 이 문제를 해결하기 위해 고장 복구 시간 추정과 체크포인팅 기법을 제안한다. 이 기법은 고장 복구 시에 수행되는 네 가지 작업들에 대하여 소요 시간을 런타임에 추정한다. 그 추정된 시간이 시간 제약에 대한 충족 여부에 따라 해당 상태들을 고장 감내 스토리지(fault-tolerant storage)에 저장한다.

본 연구진은 제안된 기법을 Apache Spark Streaming 1.41이 탑재된 서버에 구현했다. 대상 시스템에 제안된 기법 적용 전후로 고장 복구 시간을 측정한 결과 시간 제약 내에 완료 됨을 확인하였다.

2. Spark Streaming 시스템의 동작과 고장 복구 기법

Spark Streaming 시스템은 마이크로 배치(micro batch) 프로세싱 형태로 입력 데이터를 처리하며 초 단위 프로세싱(second-scale processing)을 제공하기 위해 상태 변형 작업을 메모리 상에서 수행한다. 이를 위해 상태를 RDD (resilient distributed dataset)라는 인메모리(in-memory) 데이터 구조로 관리한다. 변형 히스토리는 RDD에 수행된 변형(transformation)들의 시퀀스(sequence)이며 리니지(lineage)라고 불린다. 그림 2은 Spark Streaming 기반 시스템을 나타낸다.

이 시스템은 정상 동작에 관련된 세 가지 컴포넌트들과 비정상 동작에 관련된 두 가지 컴포넌트들로 구성된다. 전자에 해당하는 것들로는 입력 데이터 수신자, 중간 상태 연산자, 최종 결과값 연산자가 있다. 입력 데이터 수신자는 입력 스트림 데이터를 받아서 마이크로 배치 형태의 RDD로 생성하여 중간 상태 연산자에게 전달한다. 이 수행자는 전달 받은 RDD와 이전 중간 결과값에 해당하는 RDD를 입력으로 받아서 RDD 변형을 수행하여 새 RDD를 생성한다. 생성된 RDD는 중간 결과값으로 메모리 상에 저장되며 최종 결과값 연산자에게 전달된다. 최종 결과값 연산자는 전달된 RDD를 입력으로 받아서 액션(action) 작업을 수행하여 최종 결과값을 생성한다.

후자에 해당하는 것들로는 상태 백업 수행자와 고장 복구 수행자가 있다. 앞서 1장에서 언급됐다시피 이들은 리니지라는 변형 히스토리에 기반하여 동작된다. 상태 백업 수행자는 상태의 초기 상태에 해당하는 RDD와 리니지를 고장 감내 스토리지에 주기적으로 저장한다. 이들은 입력 데이터 수신자와 중간 상태 연산자로부터 전달 받는다. 고장 복구 수행자는 우선 고장 발생 여부를 감지한다. 고장이 발생한 경우에 강제 종료된 수행 환경을 복구하고 백업된 초기 상태에 해당하는 RDD를 고장 감내 스토리지에서 메모리로 읽는다. 마지막으로 그 RDD와 리니지를 기반으로 재연산하여 고장 시의 RDD를 복구한다.

3. 고장 복구 시간 추정과 체크포인팅 기법

이 장에서는 실시간 고장 복구 요구사항을 충족하기 위해 제안된 기법을 설명한다. 이 기법은 Spark Streaming 기반 시스템에서 RDD를 복구하는데 소요되는 시간을 런타임에 추정하고 이를 바탕으로 시간 제약 충족 여부에 따라 현재 RDD를 고장 감내 스토리지에 체크포인팅한다. 그림 2는 제안된 기법의 동작 개관을 나타낸다. 제안된 기법은 고장 복구 시간 추정자와 상태 백업 결정자로 구성된다. 고장 복구 시간 추정자는 종단간(end-to-end) 복구 시간을 결정하는 순차적인 네 가지 작업들에 대한 소요 시간을 주기적으로 추정한다. 상태 백업 결정자는 추정된 결과값을 전달 받고 시간 제약 충족 여부를 판단한다. 충족되지 않는 경우에 한 해 상태 백업 수행자에게 RDD 백업을 요청한다.

고장 복구 수행자가 총 복구 시간을 추정하는 방식은 그 작업들이 정상 동작 시에 이미 수행됐던 작업인지 아닌지에 따라 다르게 동작한다. 전자에

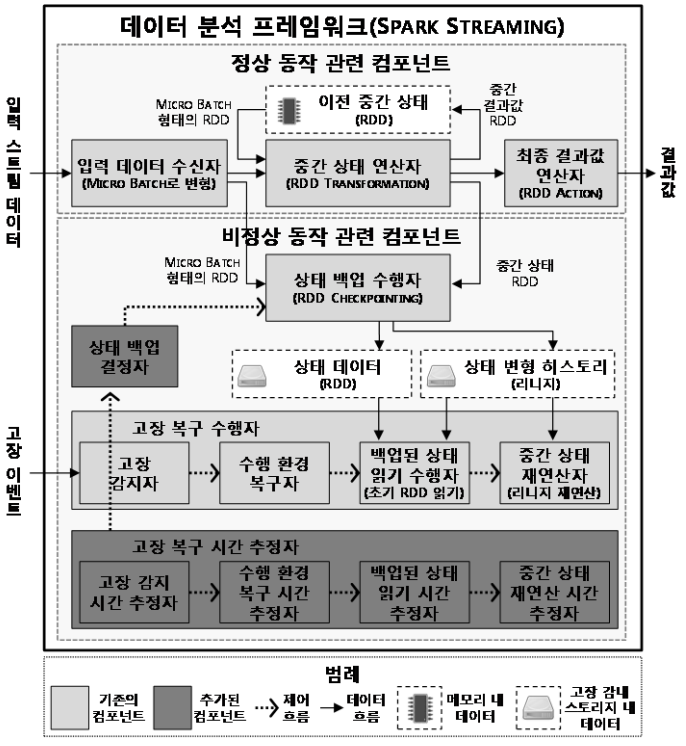

그림 2. Spark Streaming 기반 시스템과 제안된 기법의 동작 개관

해당하는 작업들의 소요 시간은 이미 수행됐던 작업들 중 일부의 소요 시간을 토대로 추정된다. 이에 해당하는 작업들로는 수행 환경 복구와 중간 상태 재연산이 있다. 후자의 경우에는 대상 시스템의 관련 내부 메커니즘에 의해 추정된다. 관련된 작업들로는 고장 감지와 백업된 상태 접근이 있다.

우선 수행 환경 복구 시간은 최악의 경우(worst case) Spark Streaming의 수행 단위인 잡이 재실행되는 경우의 소요시간이다. 따라서 그 시간은 최초 잡 실행 시 소요된 시간을 토대로 추정된다. 중간 상태 재연산 시간은 정상 동작 시에 RDD의 변형에 소요된 시간으로 추정된다. 이를 위해 Spark Streaming 잡이 주기적으로 마이크로 배치 프로세싱을 할 때마다 RDD의 변형 작업에 소요된 시간을 기록해둔다. 고장 감지 작업은 마스터 노드가 슬레이브 노드들이 주기적으로 전달하는 하트비트(heartbeat)의 유무를 토대로 수행된다. 따라서 최악의 경우 소요 시간은 하트비트의 주기와 같다. 그 주기는 `heartbeatInterval`라는 변수를 통해 조정 가능하며 기본값은 10초이다. 백업된 상태 접근 시간은 고장 감내 스토리지로부터 복구에 필요한 RDD들을 읽어 들이는 시간이며, 이는 그 스토리지의 읽기 성능에 의존적이다. 이 성능은 정적으로 확보해 둔 RDD의 크기에 따른 초당 읽기 시간을 토대로 추정된다.

4. 실험 환경 및 검증 결과

본 연구진은 제안된 기법을 Spark Streaming에 구현하고 여섯 대의 컴퓨팅 서버들로 구성된 실험 환경에서 실험을 통해 실시간 고장 복구 요구사항 충족 여부를 검증했다. 구체적인 실험 환경은 표 2와 같다.

검증 시나리오는 자율주행 자동차를 지원하기 위한 클라우드에서 주로 수행될 주기적 대용량 지도 데이터 갱신 작업을 수행하는 것이다[1]. 그리고 수행 도중에 임의의 서버를 강제 종료하여 복구에 소요되는 시간을 측정하였다. 제안된 기법 적용 전후로 각각 40회씩 실험하였다. 실험에 사용된 지도 데이터는 대한민국 영토의 크기인 330km*330km에 대하여 9km/px 축척을 갖는다. 이 지도 데이터 크기는 OpenStreetMap이라는 공개 지도 데이터를 토대로 0.75GB으로 선정하였다.

그림 3은 실험 결과를 나타낸다. 회색 막대기는

표 2 실험 환경

하드웨어		
컴퓨팅 서버 5대	CPU	Intel E5400 2.7GHz dual-core
	메모리	DDR3 6GB
	스토리지	1TB (SSD * 1)
네트워크	토폴로지	Fully Connected Network
	대역폭	1Gb
소프트웨어		
데이터 분석 프레임워크		Spark Streaming 1.41
분산 파일 시스템		Hadoop 2.71
운영체제		Ubuntu 14.04.1 LTS

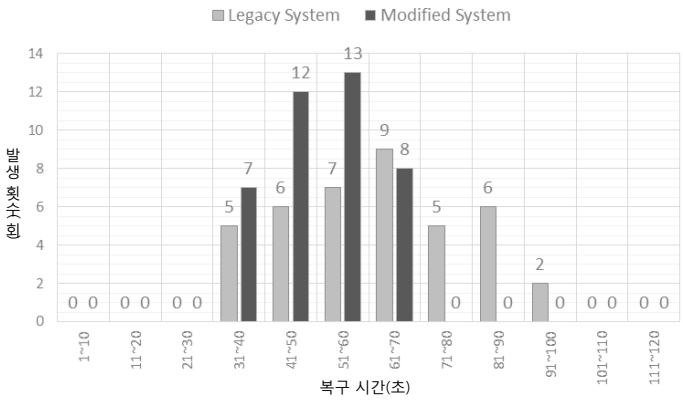

그림 3. 실험 결과

기존 Spark Streaming 시스템의 결과값이며 검은색 막대기는 제안된 기법이 적용된 시스템의 결과값이다. 제안된 기법이 적용된 시스템의 복구 시간이 시간 제약 70초 이내로 바운드 됨을 확인하였다. 그리고 제안된 기법에서 고장 복구 시간 추정에 의한 잡의 수행시간 지연 정도는 1% 이하로 무시할 정도의 수치였다.

5. 결론

본 논문에서는 자율주행 자동차를 지원하기 위한 실시간 클라우드에서 실시간 고장 복구 요구사항을 충족하기 위한 고장 복구 시간 추정과 체크포인팅 기법을 제안하였다. 본 연구팀은 Spark Streaming 1.4.1 기반 시스템에 제안된 기법을 적용하였고 실험을 통해 바운드된 시간 내에 복구 작업을 완료함을 확인하였다.

Acknowledgement

본 논문은 SK 하이닉스(주)의 위탁에 따른 '연구용역'의 결과입니다. (No.0421-20140084)

참 고 문 헌

[1] Burgstahler Daniel, et al., "RemoteHorizon.KOM: Dynamic Cloud-Based eHorizon", *Automotive meets Electronics (AmE 2016)*, 2016

[2] Dean Jeffrey and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *Communications of the ACM*, 2008

[3] Zaharia Matei, et al., "Discretized Streams: Fault-Tolerant Streaming Computation at Scale", *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2013

[4] Toshniwal Ankit, et al., "Storm@ twitter", *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2014

[5] Mehdi Nazari Cheraghlou, et al., "A Survey of Fault Tolerance Architecture in Cloud Computing", *Journal of Network and Computer Applications*, 2015