엔터프라이즈 레벨의 데이터 파이프라인 모니터링 시스템 개발

채소영, 박지수, 김혜미
에스투더블유 기업부설연구소

schae@s2w.inc, groot@s2w.inc, hmkim@s2w.inc

Development of Enterprise-Level Data Pipeline Monitoring System

So-Young Chae, Ji-Su Park, Hye-Mi Kim
Knowledge Engineering, S2W

요 약

데이터 처리 과정에서 데이터 손실 및 장애 상황을 감지하고 예방하기 위한 모니터링 시스템의 필요성이 증가하고 있다. 복잡한 데이터 파이프라인에서 각 단계를 실시간으로 관찰하고 문제 상황에 신속하게 대응하기 위해서는 종합적인 모니터링 시스템을 구축하는 것이 중요하다. 본 논문에서는 엔터프라이즈 레벨의 파이프라인 모니터링 시스템을 개발하여 데이터 파이프라인의 안정성을 향상하고 데이터의 신뢰성을 높이고자 하였다. 모니터링을 데이터, 애플리케이션, 운영, 그리고 외부 서비스 및 인프라 관점으로 분류 및 설계하고 각 관점에 따라 어떤 방식으로 활용되었는지 소개한다. 본 논문에서 개발한 모니터링 시스템을 통해 비즈니스 및 연구 분야의 데이터 처리 작업을 보다 효과적으로 관리하고, 문제 상황을 조기에 탐지하여 안정성을 향상시킬 수 있을 것으로 기대된다.

1. 서론

데이터는 비즈니스 및 연구 분야에서 핵심 자원으로 인식되고 있다. 데이터의 가치와 중요성은 지속적으로 증가하여, 이를 효율적으로 활용하기 위한 조직에서 데이터 처리를 위한 파이프라인(pipeline)[1]은 필수적인 요소이다. 데이터 수집부터 활용까지의 과정에는 복잡한 단계 및 도구가 포함되어 있으며, 다양한 변수가 존재하기도 한다. 이러한 복잡성과 외부적인 요인으로 인해 데이터 처리 과정에서 문제가 발생할 수 있으며, 이를 빠르게 감지하고 조치하는 것은 매우 중요하다.

데이터 파이프라인 모니터링 시스템은 관리자로 하여금 데이터 처리 과정의 각 단계를 실시간으로 관찰하고, 문제가 발생하면 빠르게 대응할 수 있도록 해준다. 이를 통해 데이터 손실, 처리 지연, 그리고 시스템 장애를 방지하고 데이터 품질과 신뢰성을 유지하는 데 도움을 줄 수 있다. 본 논문에서는 이러한 데이터 파이프라인 모니터링의 중요성을 강조하고, 특히 엔터프라이즈 레벨 모니터링의 필요성을 제시하고자 한다.

본 논문에서는 엔터프라이즈 레벨 데이터 파이프라인 모니터링의 필요성을 제시하고, 모니터링 유형을 4 가지로 나누어 설계 및 구현 방법을 상세히 설명한다. 또한, 모니터링 시스템을 활용했던 실제 사례를 소개하여 데이터 처리 프로세스의 효율성과 안정성을 높이는데 기여하는 실질적인 방법을 제시한다.

2. 엔터프라이즈 레벨 모니터링의 필요성

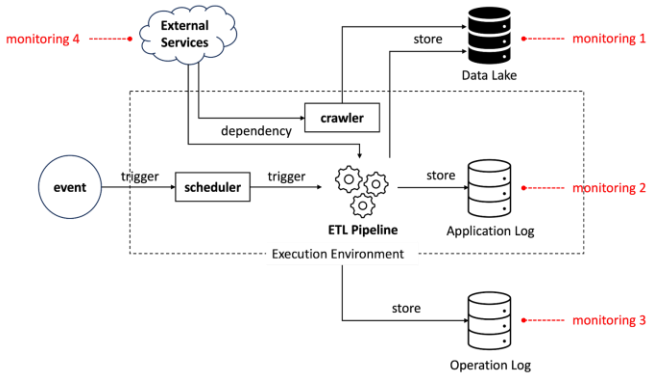
엔터프라이즈 환경에서는 다양한 모니터링 유형이 요구되며, 파이프라인 장애 상황을 인지하거나 정상적인 상황에서도 데이터 처리 단계별 로그를 분석하여 데이터의 전반적인 흐름을 실시간으로 파악할 수 있어야 한다. 예컨대 결과물에서 특정 데이터가 누락된 상황이 발생한 경우, 단순하게는 데이터가 저장되는 스토리지를 조회하는 방법을 떠올릴 수 있다. 외부 서비스 또는 툴을 사용하여 데이터를 수집하거나 처리하고 있다면, 네트워크 연결 장애 또는 서비스 호환성 등의 이슈가 발생했는지 확인할 수도 있다. 또한, 데이터 파이프라인의 각 단계에서 발생하는 이벤트 및 오류를 사전에 로깅하면 데이터가 어느 단계

에서 누락되었는지 파악할 수 있다. 엔터프라이즈 레벨의 모니터링 시스템이 구축된 환경에서는 위와 같은 과정을 자동화하여 모니터링에 필요한 시간과 자원을 절감하는 효과를 얻을 수 있다.

뿐만 아니라, 데이터 흐름에서 비정상적인 패턴을 감지하고 경고를 생성하는 기능을 추가할 수도 있다. 예를 들어, 데이터가 일정 기간동안 도착하지 않거나 예상되는 양보다 적게 도착하는 경우, 관련 담당자에게 신속하게 알림을 보내거나 누락된 데이터에 대한 자동화된 조치를 취할 수 있다.

이처럼 엔터프라이즈 레벨의 데이터 파이프라인 모니터링 시스템은 이상 상황을 더 신속하게 인지하고 대응할 수 있게 해주며, 데이터 파이프라인의 안정성과 신뢰성을 향상시키는 데 중요한 역할을 수행한다.

3. 설계 및 구현



(그림 1) 데이터 처리 프로세스에 따른 모니터링 유형 분류.

파이프라인 내부의 데이터 처리 프로세스는 다음과 같다. (그림 1)에서 보이는 것과 같이 먼저 수집기(crawler)가 소스 데이터를 수집하여 저장소(Data Lake)에 저장한다. 그 다음 특정 이벤트(event)에 의해 스케줄러(scheduler)가 동작하고, 스케줄링 정책에 따라 ETL 파이프라인이 동작하면서 순수 데이터 및 애플리케이션 수준의 로그 데이터를 각 저장소에 저장한다. 이때, 수집기와 ETL 파이프라인 모듈에서는 외부 서비스(External Services) 연동이 필요할 수 있다. 각 모듈은 특정한 운영 환경에서 실행되며, 운영 수준에서 생성되는 로그 데이터도 별도의 저장소에 저장된다.

위와 같은 과정을 통해 저장되는 데이터, 로그 및 외부 서비스와의 연결 상태 등을 모니터링하기 위한 유형 4 가지를 아래와 같이 제시한다. 참고로, 그림에 나타내지 않은 물리(physical) 인프라 수준에서는 프로메테우스(Prometheus)와 그라파나(Grafana)와 같은 도구를 사용한 별도의 모니터링 시스템으로 관리하기 때문에 본 논문에서는 다루지 않았다.

1) 데이터 관점의 모니터링

데이터 수집 또는 ETL 파이프라인 중간 과정에서는 다양한 형식의 데이터를 정형화하여 각 저장소에 저장한다. 데이터에는 고유 식별자, 저장 일시, 버전 등과 같은 메타(meta) 데이터와 실질적인 정보가 담긴 원본(raw) 데이터가 포함될 수 있다. 우리는 데이터의 품질을 보장하고 무결성을 확보하기 위해 양적 및 질적 모니터링을 사용하였다.

양적 모니터링의 경우, 각 모듈별로 저장된 데이터의 개수를 확인하여 데이터 누락 및 손실이 발생한 단계를 파악할 수 있다. 예를 들어, 데이터 A 를 파싱하는 모듈이 존재하지 않거나 모듈에 장애가 발생한 경우, 수집 데이터 저장소에서 데이터 A 의 개수는 수집량만큼 측정되겠지만 파싱 데이터 저장소에서는 개수가 0 이거나 정상 기준치보다 현저히 줄어들 수 있다. 또한, 데이터 개수 추이를 분석하면 데이터 양이 증가/감소하는 시기를 시계열로 파악할 수도 있다. 이때, 고유 식별자(unique identifier)를 통해 중복된 데이터를 제거하고 ETL 파이프라인에 사용된 변환 및 정제 로직을 적용하여 데이터의 편차(variation)를 최소화하는 기능을 추가하였다.

질적 모니터링은 저장된 데이터의 품질과 신뢰성을 확인하는 과정이다. 이 과정에서는 가장 최근에 저장된 데이터의 저장 일시를 지속적으로 모니터링함으로써 데이터의 최신성 및 정확성을 검증했다. 예컨대, 타임스탬프(timestamp) 값을 기반으로 데이터가 정해진 스케줄에 따라 올바르게 저장되고 있는지 관찰하고, 일정 기간동안 데이터가 업데이트되지 않았다면 담당자에게 문제 상황에 대해 알림을 전송하도록 했다.

2) 애플리케이션 로그 관점의 모니터링

데이터는 요구사항 및 목적에 따라 변환되어 다양한 애플리케이션에 녹아들 수 있다. 애플리케이션 구현을 위해서는 ETL 파이프라인이 동작하는데, 이때 발생하는 오류나 디버그 레벨의 로그를 모니터링하면 문제 상황을 보다 투명하게 파악할 수 있다.

우선, 파이프라인에서 공통적으로 사용하는 라이브러리에 로깅 인터페이스를 추가하여 모니터링을 필요로 하는 곳에서 언제든 가져다가 구현할 수 있도록 만들었다. 그리고 Fluentd[2]를 사용하여 파이프라인에서 출력하는 로그를 수집 및 파싱하고, Elasticsearch 에 저장하기 적합한 형식으로 가공했다. 애플리케이션 로그에는 로그 레벨, 이벤트 세부 정보, 저장 일시 등이 포함된다.

이처럼 모듈별로 저장된 로그는 위험 정도에 따라

관리자에게 알림을 전송하거나, Elasticsearch 쿼리 또는 Kibana 대시보드를 통해 디버깅이 필요한 로그 데이터를 살펴보고 이상 징후를 식별하는 데 사용할 수 있다. 애플리케이션 로그를 활용하면, 휴리스틱(heuristic) 방법에 의존하여 파이프라인 오류 상황을 추측하는 것에서 한 단계 나아가 논리 기반(logic-driven)의 원인 분석이 가능하다.

3) 운영 로그 관점의 모니터링

운영 로그 모니터링은 보다 근본적인 관점에서 파이프라인 헬스 체크(health check)를 가능하게 한다. 우리는 애플리케이션을 보다 안정적으로 운영하기 위해 쿠버네티스(Kubernetes)[3] 환경을 사용한다. 쿠버네티스 이벤트 로그는 클러스터에서 발생하는 모든 이벤트와 관련된 정보를 포함하며, 이벤트 로그 중에서는 중요도가 높거나 모니터링 대상이 되는 이벤트를 필터링하는 로직이 필요하다. 예를 들어, 특정 네임스페이스의 이벤트 또는 특정 레이블을 가진 파드 등을 모니터링 대상으로 선정할 수 있다.

이제, 이벤트 로그 중 파드의 생성 및 종료에 따른 상태값을 주기적으로 확인하여 비정상적인 패턴을 감지하는 기능을 추가한다. 또한, YAML 파일에 설정된 스케줄링 주기를 파싱하여 지정된 시간에 실행되지 않은 프로세스가 존재하는지 여부를 파악한다. 쿠버네티스 이벤트 로그를 분석하면 클러스터의 상태 및 이벤트를 실시간으로 추적하거나, 운영상의 장애 및 이상 징후를 조기에 감지하여 대응할 수 있다.

4) 외부 서비스 및 인프라 관점의 모니터링

일반적으로 데이터 수집기나 파이프라인 모듈은 독립적인 프로세스로 동작하지 않으며, 외부 서비스(API)를 활용하거나 내부의 다양한 인프라를 사용하는 등 서로 다른 서비스와 연결되어 있다.

데이터 파이프라인이 외부 서비스와 상호 작용하는 경우, 외부 서비스의 가용성과 응답 시간을 모니터링한다. 서비스가 다운되거나 응답 시간이 길어지는 경우 이를 신속하게 감지하여 조치를 취한다. 또한, 외부 서비스의 업데이트나 변경사항을 모니터링하고, 파이프라인이 이러한 변경에 대응할 수 있도록 관리한다. 데이터 파이프라인이 실행되는 인프라(서버, 네트워크, 스토리지)의 상태와 성능 또한 점검 대상이다. 리소스 부족, 네트워크 병목, 스토리지 용량 등과 같은 인프라 관련 문제를 감지하고 해결해야 한다.

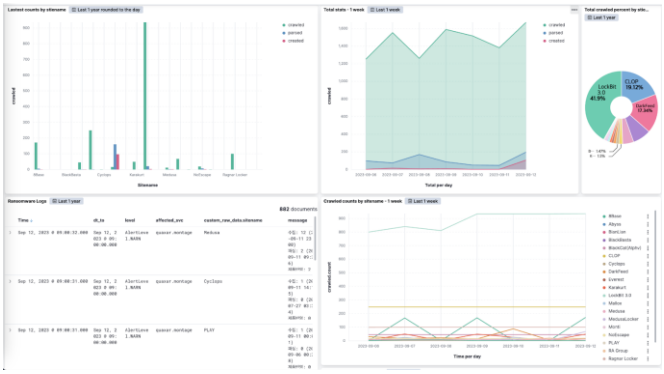
앞서 소개한 4 가지 유형의 데이터 파이프라인 모니터링을 시스템화하기 위해 다음과 같은 방법을 사용하였다. 먼저, 데이터 및 인프라 소스와 연결하고 필요한 정보를 획득하기 위한 커넥터(connector) 컴포넌트를 생성한다. 이는 데이터 및 인프라 접근에 필요한 계정 정보 및 연결 설정을 입력받아 해당 정보를 가져오는 역할을 수행한다. 각 유형에 따른 모니터링 항목은 의미 단위의 이벤트로 구성되며, 지정한 임계치(threshold)나 이전 데이터와의 비교 작업을 거쳐 장애 상황 여부 및 위험 수준을 평가받는다. 뿐만 아니라, 평가 결과에 따라 관련 담당자에게 리포팅 메시지나 경고 알림 등을 전달하는 알림(notifier) 컴포넌트도 구성하였다.

동일한 성격을 지닌 모니터링 대상은 각 애플리케이션으로 구현 및 실행되며, 데이터 및 서비스의 상태를 파악하는 작업을 수행한다. 시스템을 구성하는 각 모듈은 기능 단위로 추상화되어 있어 파라미터(parameter) 설정이나 새로운 애플리케이션 추가가 용이하도록 설계되었다. 이를 통해 데이터 파이프라인 모니터링 시스템을 효율적으로 유지하고 확장할 수 있도록 하였다.

4. 활용 예시

본 논문에서 개발한 모니터링 시스템을 활용해 모니터링 유형별 활용 가능성 및 범위에 대한 예시는 다음과 같다.

1) 데이터 관점의 모니터링



(그림 2) 랜섬웨어 데이터 모니터링 Kibana 대시보드.

우리는 사이버 공간에서 활동하는 랜섬웨어 공격 그룹에 대한 정보를 제공하는 서비스를 운영하고 있다. 서비스를 제공하기 위해 다양한 랜섬웨어 사이트로부터 데이터를 수집, 가공 및 분석하여 데이터베이스(database)에 저장하는 일련의 프로세스를 수행한다. 그러나 이 과정에서 데이터가 누락되거나 손실되는 일이 발생할 수 있으며 이는 서비스의 신뢰성을 저해할 수 있으므로, 데이터 무결성을 보장하기 위해 예상대로 데이터가 수집되고 서비스화되는지 모니터링하는 과정이 중요하게 고려되었다.

따라서 데이터 수집, 가공, 분석, 및 서비스화 과정 각각에서 데이터의 누락 여부를 철저히 모니터링하였다. 예를 들어, 각 단계에서는 랜섬웨어 사이트로부터의 데이터 양을 정기적으로 확인하여 데이터가 손실되지 않고 수집 및 처리되고 있는지 점검했다. 또한, 데이터의 최신성을 검증하기 위해 처리 시간을 지속적으로 관찰하고 이상 징후를 신속하게 탐지할 수 있도록 하였다.

모니터링 상황은 (그림 2)와 같이 Kibana 대시보드로 시각화하여 데이터 수집, 가공, 분석 및 서비스화 과정을 한눈에 파악할 수 있도록 했다. 이를 통해 데이터 파이프라인의 모니터링을 더 효과적으로 수행하고 랜섬웨어 공격에 대한 정보 제공 서비스의 신뢰성을 높일 수 있다.

2) 애플리케이션 로그 관점의 모니터링

데이터 파이프라인에는 패스워드가 필요한 압축 파일의 압축을 해제하거나, 식별자의 이름을 확인하는 작업과 같이 자동화하기 어려운 부분이 존재한다. 이 경우 파이프라인 처리 지연을 방지하기 위해 빠른 조치가 필요하다.

그러나 데이터 파이프라인이 동작하는 중간에 관리자가 직접 개입하여 문제를 해결하는 것은 사실상 어려운 일이다. 이를 효율적으로 처리하기 위해 해당 파이프라인에서 발생되는 로그를 Elasticsearch 에 적재하면, 모니터링 시스템은 해당 로그를 분석하여 관리자에게 즉각적인 알림을 전달할 수 있다. 관리자는 알림을 받은 즉시 파이프라인에 필요한 작업을 수행하여 잘못된 데이터의 입력을 방지하거나 파이프라인 리소스 낭비와 같은 상황을 막을 수 있다.

3) 운영 로그 관점의 모니터링

우리는 데이터 파이프라인을 운영하는 쿠버네티스 클러스터에서 발생하는 모든 이벤트 로그를 수집하고 Elasticsearch 에 저장하고 있다. 수집된 이벤트 로그는 카테고리에 따라 분류되고 컨테이너 실행과 관련된 이벤트를 중심으로 관심 있는 로그만 선택적으로 추출한다.

모니터링 시스템은 해당 로그를 애플리케이션에 따라 나누고 컨테이너 실행 상태를 추적한다. 이를 통해, 쿠버네티스 클러스터에서 발생한 문제가 컨테이너, 파드, 리소스와 같은 환경 중 어느 부분에서 발생한 문제인지 확인하고 구체적인 원인을 빠르게 짚어볼 수 있다.

4) 외부 서비스 및 인프라 관점의 모니터링

애플리케이션 및 시스템 간에 데이터나 메시지를 비동기적으로 전달하고 관리하기 위해서 메시지 큐(queue)를 사용한다. 우리는 모니터링 시스템을 통해 매일 10 분마다 메시지 큐에 저장된 데이터의 양을 측정하여 임계치 이상의 데이터가 처리되고 있는지 확인하고 있다.

큐의 데이터가 소비(consume)되지 않고 임계치 이상의 데이터가 쌓여있다면 파이프라인이 정상적으로 동작하고 있는지 살펴보아야 한다. 데이터를 생산(produce)하는 파이프라인에 이슈가 생겼을 수도 있고, 큐의 데이터를 소비하는 파이프라인의 문제일 수도 있다. 애플리케이션 수준의 로그를 통해 탐지되지 않는 장애 상황은 얼마든지 존재할 수 있으므로 인프라 관점의 모니터링이 반드시 필요하다.

5. 결론

우리는 분산되어 있던 디버깅 포인트를 통합하고 모니터링 시스템에 내재화함으로써 데이터 및 처리 과정을 보다 효율적으로 관리하고 불필요한 자원 낭비를 방지할 수 있게 되었다. 로그를 디버깅하거나 인프라 상태를 점검하는 일은 이제 엔터프라이즈 레벨의 모니터링 시스템이 실시간으로 수행하고 있다. 장애 상황에서의 모니터링 알림이나 파이프라인에서 처리되는 데이터 상태를 확인하는 대시보드도 주기적으로 생성되어 관리자의 간섭을 최소화하고 가시성을 확보했다. 또한, 새로운 모니터링 요구사항을 수용하기 위해서는 기능별로 추상화된 모듈을 가지고 구현하기만 하면 되기 때문에 빠르고 유연하게 확장할 수 있다는 강점을 지닌다.

데이터 처리가 복잡한 환경일수록 운영을 최적화하고 데이터 무결성을 보장하기 위해 다차원적인 모니터링은 필수적이다. 본 논문에서 소개한 엔터프라이즈 레벨의 데이터 파이프라인 모니터링 시스템은 비즈니스 및 연구 분야의 데이터 처리 프로세스 효율성과 신뢰도를 향상시키는데 중요한 역할을 할 수 있을 것으로 기대된다.

참고문헌

[1] A. Raj, J. Bosch, H. H. Olsson and T. J. Wang, "Modelling Data Pipelines," 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Portoroz, Slovenia, 2020, pp. 13-20, doi: 10.1109/SEAA51224.2020.00014.

[2] Fluentd, fluentd v1.0 Documentation, 2020.9.29., https://docs.fluentd.org/

[3] kubernetes, Kubernetes API, 2023.7.12., https://kubernetes.io/docs/reference/kubernetes-api/