분산 로그 수집 시스템에서의 클라이언트 주도형 설정 주입을 위한 스레드 단위 제어 아키텍처

강찬욱, 정주연, 양승원

건국대학교

kan0202@konkuk.ac.kr, jjjjyy0704@konkuk.ac.kr, yks8967@konkuk.ac.kr

Thread-Level Control Architecture for Client-Driven Configuration Injection in Distributed Log Collection System

Kang Chan Wook, Jeong Ju Yeon, Yang Seung Won

Konkuk Univ.

요 약

본 논문은 기존 로그 수집 도구들의 복잡한 설정 구조와 운영 중 정책이 변경될 경우 전체 프로세스의 재시작과 같은 대규모 재구성을 필요로 함을 문제로 인식한다. 이러한 방식은 서비스의 연속성을 저하시키며, 시스템 운영의 예측 가능성에도 부정적인 영향을 미친다. 이에 본 연구에서는 통신을 통한 설정 변경을 지원하며, 설정 변경 시 프로세스 재시작, 환경 재구성 등의 절차를 최소화하면서도 로그 처리 정책을 세밀하게 제어할 수 있는 구조를 가진 클라이언트 주도형 아키텍처를 제안한다.

제안한 아키텍처의 핵심은 설정 주입을 지원하는 것뿐만 아니라 설정 변경의 영향을 전체 시스템이 아닌, 실행 단위 수준으로 제한하는 데 있다. 이를 위해 본 시스템은 ETag를 활용해 설정의 버전 및 변경 여부를 판단하며, ThreadNum을 이용해 변경이 필요한 Thread를 감지하고 해당 Thread에 한해서 선택적으로 중단 및 재구성을 수행한다. 그 결과, 다른 Thread의 동작을 유지한 채 부분적인 설정 변경이 가능해진다. 이를 통해 로그 수집 에이전트를 유연하게 제어 가능한 실행 단위의 집합으로 재정의할 수 있고, 결과적으로 로그 수집 시스템의 운영 효율성과 안정성을 크게 향상시킨다.

Ⅰ. 서 론

클라우드 환경과 마이크로서비스 아키텍처의 확산으로 인해, 분산 시스템에서 로그 수집 및 분석의 중요성은 지속적으로 증가하고 있다. 컨테이너 기반 환경에서는 서비스 인스턴스의 생성과 종료가 빈번하게 발생하며, 이에 따라 로그의 발생 위치와 형태, 그리고 로그 처리 정책 또한 운영 요구에 따라 자주 변경된다. 이러한 환경에서는 로그 수집 시스템이 단순한 로그 전달을 넘어, 로그 처리 동작을 유연하게 제어할 수 있는 구조를 요구받는다.

일반적으로 로그 수집 시스템에서 로그 처리 동작은 파싱, 필터링, 라우팅과 같은 처리 단계들의 조합으로 구성되며, 이러한 처리 흐름은 설정(configuration)을 통해 정의된다. Logstash, Promtail, Fluentd와 같은 대표적인 로그 수집 도구들은 입력 - 처리 - 출력 단계로 구성된 로그 처리 파이프라인을 설정 파일로 기술하는 방식을 채택하고 있다 [1][2][3]. 그러나 이러한 설정은 대체로 정적 파일 기반으로 관리되어, 실행 중 변경이 제한적이거나 서비스 재시작에 의존하는 경우가 많다.

이러한 정적 구성 모델은 클라우드 및 마이크로서비스 환경에서 운영 자동화와 유연성 측면의 한계를 드러낸다. 로그 처리 정책 변경 시 에이전트 재시작이나 설정 재배포가 요구되며, 분산 환경에서는 다수의 로그 수집 에이전트에 대한 설정 일관성 유지 또한 관리 부담으로 작용한다. 특히 로그 처리 정책을 주도적으로 변경해야 하는 환경에서는 실행 중인 로그 수집 시스템 설정을 안전하게 제어할 수 있는 통신 구조가 필요하다.

또한 기존 로그 수집 시스템에서는 설정 변경 단위와 로그 수집 실행 단위가 명확히 분리되어 있지 않아, 설정 변경이 시스템 전체 재시작과 같은 과도한 재실행으로 이어지는 경우가 많다. 이러한 구조에서는 설정 변경의 영향 범위를 세밀하게 제어하기 어렵고, 운영 안정성과 예측 가능성을 확보하는 데 한계가 존재한다.

본 논문은 이러한 문제를 해결하기 위해 클라이언트 주도형 설정 주입을 지원하는 통신 아키텍처를 제안한다. 제안하는 시스템은 웹 UI 대시보드를 설정 제어의 주체로 정의하고, 설정 주입 스레드와 로그 수집 및 전송 스레드를 분리된 실행 구조로 구성한다. 이를 통해 설정 변경의 영향 범위를 로그 수집 및 전송 스레드 단위로 제한하고, 변경이 필요한 실행 단위만을 선택적으로 재구성할 수 있도록 한다.

또한 ETag 기반 설정 버전 관리와 ThreadNum을 이용한 스레드 단위 제어 메커니즘을 도입하여, 로그 수집 에이전트 프로세스의 재시작 없이도 동적 설정 변경을 가능하게 한다. 본 논문은 이러한 설계를 통해 로그 수집 에이전트를 클라이언트와의 통신을 통해 동적으로 제어 가능한 실행 단위들의 집합으로 재정의하고, 분산 로그 수집 환경에서의 설계 및 구현 가능성을 검증한다.

Ⅱ. 시스템 개요

본 논문에서 제안하는 분산 로그 수집 시스템은 클라이언트, API 서버, 로그 수집 에이전트로 구성된다. 클라이언트는 웹 UI 대시보드 형태로 제공되며, 로그 수집 정책과 파이프라인 및 전송 설정을 정의하는 제어 주체로 동작한다. API 서버는 클라이언트와 로그 수집 에이전트 간의 중개 역할을 수행하며, 설정 관리와 인증을 담당한다. 로그 수집 에이전트는 실제 로그 파일을 수집하고 전송하는 실행 주체로서, 클라이언트로부터 주입된 설정을 기반으로 동작을 변경한다.

제안하는 시스템의 핵심 설계 원칙은 설정 제어 흐름과 로그 수집 실행 흐름을 명확히 분리하는 것이다. 이를 위해 로그 수집 에이전트 내부에서는 설정 주입을 담당하는 실행 단위와 로그 수집 및 전송을 담당하는 실행 단위를 스레드 수준에서 분리하여 구성한다. 이러한 구조는 설정 변경이

로그 수집 전체에 미치는 영향을 제어 가능한 범위로 제한하는 것을 목표로 한다.

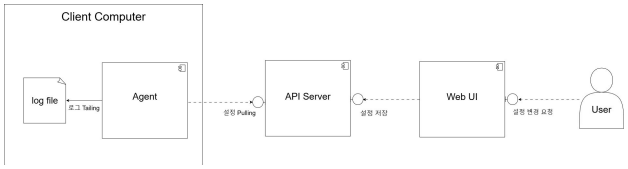

그림 1 목적 시스템의 전체 아키텍처 구성도

Ⅲ. 설정 주입 스레드 기반 제어 모델

로그 수집 에이전트는 설정 주입을 담당하는 전용 스레드를 통해 클라이언트로부터의 설정 변경을 수신한다. 이 스레드는 API 서버와의 HTTPS 통신을 통해 설정을 주기적으로 Pulling하며, 실행 중인 로그 수집 동작과 독립적으로 동작한다. 설정 주입 스레드는 로그 파일 접근이나 로그 전송과 같은 데이터 처리 작업을 수행하지 않으며, 오직 설정 변경 여부의 판단과 적용 대상 식별에만 관여한다.

설정 Pulling 방식은 클라이언트 주도의 설정 주입 모델을 간접적으로 구현한다. 클라이언트는 API 서버에 설정 변경을 등록하고, 로그 수집 에이전트는 주기적인 Pulling을 통해 해당 변경을 감지한다. 이를 통해 로그 수집 에이전트는 외부로부터 직접적인 Push 요청을 받지 않으면서도, 클라이언트가 설정 제어의 주체로 동작할 수 있다. 이러한 방식은 네트워크 접근 제어와 방화벽 환경에서도 안정적인 설정 전달을 가능하게 한다.

설정 주입 스레드와 로그 수집 및 전송 스레드 간의 상호작용은 다음과 같은 설정 업데이트 동작 시퀀스로 정리할 수 있다.

1. 클라이언트는 웹 UI 대시보드를 통해 로그 수집 및 전송 스레드의 설정을 변경한다.
2. 로그 수집 에이전트 내 설정 주입 스레드는 API 서버로부터 설정을 주기적으로 Pulling 하여 수신한다.
3. 수신된 설정의 ETag를 기존 설정의 ETag 와 비교하여 설정 변경 여부를 판단한다.
4. 설정 변경이 감지되면, 설정 주입 스레드는 변경된 설정에 포함된 ThreadNum 정보를 기반으로 영향을 받는 로그 수집 및 전송 스레드를 식별한다.
5. 식별된 로그 수집 및 전송 스레드만을 종료한 후, 새로운 설정을 반영한 로그 수집 및 전송 스레드를 생성하여 로그 수집을 재개한다. 이 과정에서 다른 스레드들은 중단 없이 기존 로그 수집을 지속한다.

Ⅳ.로그 수집 및 전송 스레드 구조

로그 수집 에이전트는 다수의 로그 수집 및 전송 스레드를 통해 로그 파일을 수집한다. 각 로그 수집 및 전송 스레드는 특정 로그 파일 또는 로그 소스에 대응되며, 파일 변경 사항을 감지하여 로그를 읽고 이를 전송 모듈로 전달한다. 로그 수집 및 전송 스레드는 독립적인 실행 단위로 관리되며, 서로 간의 상태를 공유하지 않는다.

이러한 스레드 단위 구조는 로그 수집 동작을 세밀하게 제어할 수 있는 기반을 제공한다. 설정 변경이 발생하더라도, 모든 로그 수집 스레드를 동시에 중단할 필요 없이, 변경이 필요한 로그 수집 및 전송 스레드만을 선택적으로 재구성할 수 있다. 이를 통해 로그 수집 시스템은 실행 중에도 부분적인 설정 변경을 수용할 수 있다.

Ⅴ.ETag 및 ThreadNum 기반 설정 관리 메커니즘

설정 변경의 효율적인 판단과 적용 범위 제어를 위해, 본 시스템은 ETag와 ThreadNum을 결합한 설정 관리 메커니즘을 사용한다. ETag는 로그 수집 및 전송 스레드 설정의 버전을 식별하는 값으로 사용되며, 설정 주입 스레드는 Pulling 과정에서 수신한 ETag를 기존 값과 비교하여 설정 변경 여부를 판단한다.

ThreadNum은 각 로그 수집 및 전송 스레드를 식별하기 위한 식별자로, 설정 항목과 로그 수집 및 전송 스레드를 명시적으로 연결하는 역할을 한다. 설정 변경이 감지되면, 설정 주입 스레드는 변경된 설정에 포함된 ThreadNum 정보를 기반으로 재구성이 필요한 로그 수집 및 전송 스레드를 식별한다. 이후 해당 스레드만을 종료한 뒤, 새로운 설정을 반영한 스레드를 생성하여 로그 수집을 재개한다.

이 과정에서 로그 수집은 해당 스레드 범위에서만 짧은 시간 동안 제어된 방식으로 중단되며, 로그 수집 에이전트 프로세스 자체는 재시작되지 않는다. 이러한 설계는 설정 변경의 영향 범위를 명확히 정의하고, 불필요한 전체 재구성을 방지한다.

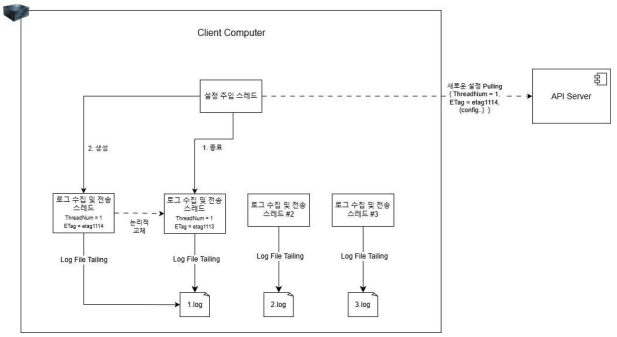

그림 2 로그 수집 및 전송 스레드 설정 변경 흐름도

Ⅵ. 결론

본 논문에서는 클라이언트 주도형 설정 주입을 지원하는 분산 로그 수집 아키텍처를 제안하였다. 제안한 시스템은 설정 주입 스레드와 로그 수집 및 전송 스레드를 분리된 실행 구조로 구성함으로써, 설정 변경의 영향 범위를 로그 테일러 스레드 단위로 제한할 수 있도록 설계되었다.

프로토타입 구현과 실험을 통해, 클라이언트에서 설정 변경이 발생하더라도 로그 수집 에이전트 프로세스를 재시작하지 않고, 변경이 필요한 로그 테일러 스레드만을 선택적으로 재구성할 수 있음을 확인하였다. 또한 설정 변경이 없는 경우에는 ETag 기반 설정 관리 메커니즘을 통해 불필요한 스레드 재구성이 발생하지 않음을 검증하였다.

이러한 결과는 제안한 아키텍처가 실행 중인 분산 로그 수집 환경에서 클라이언트 주도의 동적 설정 제어를 안정적으로 지원할 수 있음을 보여주며, 로그 수집 시스템을 동적으로 제어 가능한 실행 단위들의 집합으로 재정의할 수 있는 설계 가능성을 제시한다.

참 고 문 헌

[1] Fluentd Project, “Fluentd Configuration” Available: https://docs.fluentd.org/configuration, Accessed: Jan. 2026.

[2] Grafana Labs, “Promtail Documentation” Available: https://grafana.com/docs/loki/latest/send-data/promtail, Accessed: Jan. 2026.

[3] Elastic, “Logstash Configuration Files,” Available: https://www.elastic.co/docs/reference/logstash/config-setting-files, Accessed: Jan. 2026.