분산시스템 트랜잭션 추적 기술 연구

정남준*, 김동욱*, 홍수빈*, 이동혁*
한국전력공사 전력연구원*

A Study on the Tracking Technology of Distribution System

Nam-Joon Jung*, Dong-Wook Kim*, Su-Bin Hong*, Dong-Hyuk Lee*
KEPCO Research Institute*

Abstract - 현재의 비즈니스 환경은 조직의 점진적인 거대화, 조직들 간에 비즈니스 프로세스 상호운용을 위하여, 보다 광범위하고 복잡한 비즈니스 서비스들에 대한 처리를 요구하고 있다. 복잡해진 비즈니스 처리를 위해서는 여러 서버나 프로세스가 분산된 형태의 시스템 운영이 불가피한데, 그러한 환경에서 발생하는 오류 및 성능 문제를 처리하기 위한 방안으로 분산 추적(Distributed Tracing)이 요구되고 있다, 본 논문에서는 인터넷 서비스의 복잡도가 높아지면서 발생하는 문제의 해결을 위해서 필요한 분산 추적 기술의 기술 동향 및 서비스 아키텍쳐를 분석하고, 한전의 복잡한 구조의 업무 시스템에 적용하기 위한 방안을 개념적인 측면에서 설계하고자 한다.

1. 서 론

분산 추적은 인터넷 서비스가 '홈페이지 중심(2 tier)'에서 '기존 시스템의 정보를 웹에서 제공(3 tier)'으로 발전하고, 이어서 '웹이 기존 시스템의 모든 기능을 흡수(Multi-tier)'로 발전하면서 인터넷 서비스가 다양한 컴포넌트의 조합으로 구성되는 구조(=SOA Architecture)를 가지게 되면서 부상하게 된 기술이다[1].

즉, n 계층으로 구성된 현재의 시스템들은 시스템의 복잡도가 증가하였으며, 복잡도가 증가한 시스템의 장애나 성능 저하의 문제는 해결하기 어렵게 된다. 문제의 해결을 위해서는 많은 컴포넌트와 서버의 상태를 모두 살펴봐야 하고, 개별 컴포넌트의 기능과 컴포넌트 간의 연계성에 대한 파악도 이루어져야 한다. 이런 이유로 n 계층 구조의 시스템은 전체 시스템에 대한 가시성이 부족하고, 문제의 원인을 찾는 데 시간도 오래 걸리며, 문제의 원인을 발견하지 못하는 경우도 자주 발생한다[2].

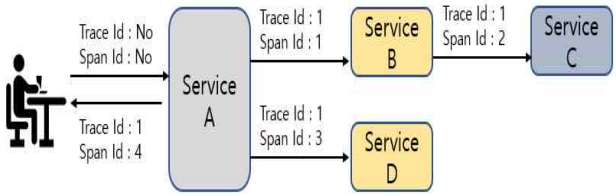
인터넷 서비스의 복잡도가 높아지면서 발생하는 문제의 해결을 위해서는 n 계층, 컴포넌트로 구성된 아키텍처를 효과적으로 추적할 수 있는 새로운 플랫폼이 필요하고, 이런 요구에 맞추어 개발된 개념이 분산 추적이다. 분산 추적은 기존 웹 환경에서도 효과를 발휘하지만, 특히 최근에 부상되는 마이크로서비스 아키텍처에 적합한 기술이다.

2. 본 론

2.1 관련 연구

2.1.1 분산 추적 서비스의 구성 및 이슈

분산 추적은 특정 작업이 여러 서비스와 연결되어 수행될 때 이에 대한 추적성을 제공한다. 클라이언트 또는 API 클라이언트가 서버로 호출한 하나의 요청을 Trace라고 하고, 서비스 컴포넌트 간의 호출을 Span이라 한다. 하나의 요청은 [그림 1]에서 확인할 수 있는 것과 같이 같은 Trace ID를 사용한다. 하나의 요청에 속하는 서비스 간의 호출은 각각 다른 Span ID를 사용한다. 분산 추적서비스를 통하여 전체 트랜잭션 시간은 Trace로 추적할 수 있고, 서비스별 시간은 Span으로 추적할 수 있다. 분산 추적을 구현하는 데 필요한 구성요소는 분산 추적을 적용할 '자체 애플리케이션 코드 및 서비스'와 분산 추적 기능을 제공하는 '공유 라이브러리 및 서비스' 그리고 분산 추적 라이브러리의 정보를 저장하고 관리하는 '외부 서비스'로 분리하여 생각할 수 있다. 그리고 이러한 구성요소를 구현하기 위해서는 '표준화된 분산 추적 API', '표준화된 추적 컨텍스트 정의 및 전파', '표준화된 추적 데이터 정의', 그리고'상호운영 가능한 추적 솔루션'이 필요하다[1].

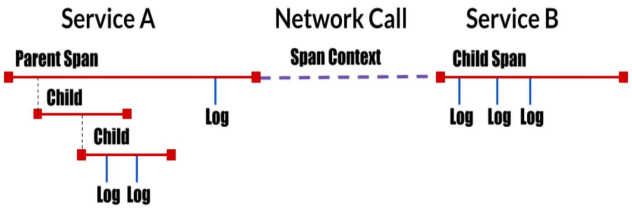


<그림 1> 분산 추적 서비스 구성

2.1.2 분산 추적 서비스의 표준화

분산 추적 서비스의 표준화를 위하여 Open Tracing project가 진행 중이다(최근 Open Telemetry가 Open Tracing Project SDK를 수용하도록 확장). Open Tracing은 분산 추적과 관련된 기술의 표준화를 위하여 1.서로 다른 조직에서 개발하여 실행하는 사용자 정의 애플리케이션 코드의 추적, 2.공유 라이브러리 및 공유 서비스의 추적 정보 교환, 3.런타임 호환성 제공, 4.각 구성요소가 어떤 추적 솔루션을 사용하여도 추적할 수 있도록 하는 것과 같은 4가지 과제를 목표로 표준화를 진행 중이다[2].

Open Tracing을 통한 분산 추적의 표준화는 기본 개념이나 데이터 모델을 Google의 Dapper를 기반으로 하고 있다. 표준화의 진행은 초기 단계(version 1.1)로 분산 추적의 기본적인 모델과 추구하는 개념은 다음과 같다. 분산 추적 모델에서 span context는 추적시스템이 다운 스트림 서비스에 전파하는 Trace 식별자, span 식별자 그리고 기타 데이터를 포함한다.



<그림 2> 분산 추적 모델

2.1.3 수동 방식의 분산 추적 구현 기술

<표 1> 분산 추적 구현 기술의 비교

구분	장 점	단 점
수동 방식	용도에 맞는 기능제공으로 API 단순 버그의 가능성이 낮음	사용하는 코드의 수정이 필요.
자동 방식	사용하는 코드의 수정 불필요	개발에 많은 노력 필요 추가적인 기능 구현이 어려움

분산 추적 기술의 표준화 동향과 별도로 현재 제공되는 분산 추적 플랫폼을 분석하면 개발자가 제공되는 API를 이용하여 프로그램을 수정/재작성하는 작업을 수행해야 하는 수동 방식과 기존 프로그램의 수정 없이 분산 추적 기능을 수행하는 자동 방식으로 나누어 볼 수 있다.

수동 방식은 마이크로소프트사의. Net 환경에서 제공하는 API를 활용하는 방식과 Zipkin과 같이 자체 컨테이너(Finagle)를 사용하고, 수정된 API를 이용하여 분산 추적을 수행하는 방식이 많이 사용되고 있다. 마이크로소프트의 분산 추적 기술은 ,Net 환경을 기반으로 하고 있다. .Net은 자바의 VM(Virtual Machine) 개념을 확대한 것으로 실행파일의 크기가 작아지는 이점이 있다.

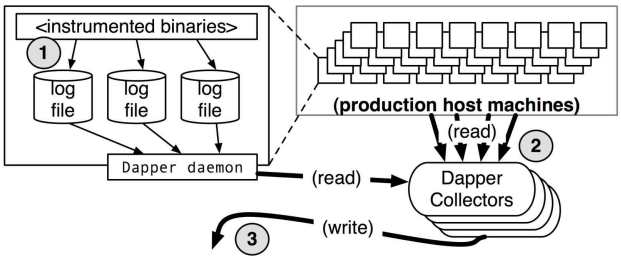
.Net을 이용한 분산 추적은 OpenTelemetry와 Application Insights의 두 가지 방법을 제공한다. OpenTelemetry와 Application Insights가 특정 업체에 종속적인 기술은 아니지만, 현 시점에서 .Net 환경에서 많이 사용한다[4].

```
.NET CLI

> dotnet run
Activity.Id:          00-7759221f2c5599489d455b84fa0f90f4-6081a9b8041cd840-01
Activity.ParentId:    00-7759221f2c5599489d455b84fa0f90f4-9a52f72c08a9d447-01
Activity.DisplayName: StepOne
Activity.Kind:        Internal
Activity.StartTime:   2021-03-18T10:46:46.8649754Z
Activity.Duration:    00:00:00.5069226
Resource associated with Activity:
    service.name: MySample
    service.instance.id: 909a4624-3b2e-40e4-a86b-4a2c8003219e

Activity.Id:          00-7759221f2c5599489d455b84fa0f90f4-d2b283db91cf774c-01
Activity.ParentId:    00-7759221f2c5599489d455b84fa0f90f4-9a52f72c08a9d447-01
Activity.DisplayName: StepTwo
Activity.Kind:        Internal
Activity.StartTime:   2021-03-18T10:46:47.3838737Z
Activity.Duration:    00:00:01.0142278
Resource associated with Activity:
    service.name: MySample
    service.instance.id: 909a4624-3b2e-40e4-a86b-4a2c8003219e

Activity.Id:          00-7759221f2c5599489d455b84fa0f90f4-9a52f72c08a9d447-01
Activity.DisplayName: SomeWork
Activity.Kind:        Internal
Activity.StartTime:   2021-03-18T10:46:46.8634510Z
Activity.Duration:    00:00:01.5402045
Resource associated with Activity:
    service.name: MySample
    service.instance.id: 909a4624-3b2e-40e4-a86b-4a2c8003219e

Example work done
```

〈그림 3〉 OpenTelemetry 기반 분산 추적 결과 가시화

2.1.4 자동 방식의 분산 추적 기술

자동 방식의 분산 추적 구현 기술은 Google의 Dapper 기술을 그 예로 들 수 있다. Google Dapper는 분산 트랜잭션의 어려움을 해결하기 위하여 메시지를 전송할 때, 애플리케이션 수준에서 메시지 간의 관계를 설명하는 태그를 추가하는 방법을 사용한다[4]. Google Dapper의 개념을 기반으로 개발된 분산 추적 제품은 트위터에서 사용하는 Zipkin, Jager, Pinpoint 등이 있다.



trace id	span 12	span 23	span 34	span 45	span 56	...
123456	*nil*	*nil*	**<data>**	**<data>**	*nil*	...
246802	**<data>**	*nil*	*nil*	*nil*	**<data>**	...
357913	*nil*	**<data>**	*nil*	*nil*	*nil*	...
...	...	...	...	...	...	...

(Central Bigtable repository for trace data)

〈그림 4〉 Dapper의 데이터 수집 파이프라인 개요

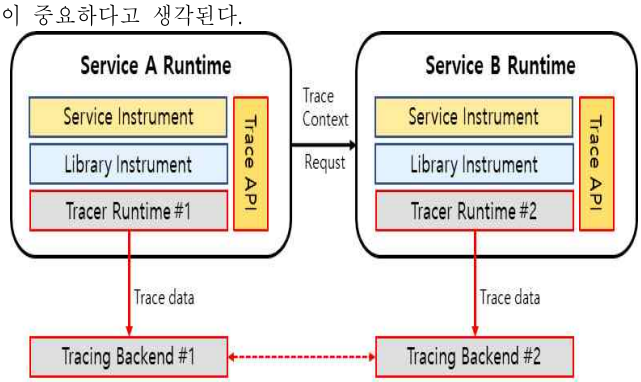
Pinpoint는 Google Dapper의 기본 개념을 바탕으로 개발된 제품으로 어느 정도 상용화의 수준에 근접한 제품이다. 현재, 국내 기업을 중심으로 사용 중에 있으며, 분산 추적의 표준화 방향인 Open Tracing의 개정에 맞추어 지속적으로 기능이 보완되고 있다. Pinpoint는 자동 방식의 분산 추적 기능을 제공한다. 즉, RPC 호출 코드를 가로채서 태그 정보를 자동으로 넣고, 식별하는 방식을 사용한다. 태그 데이터는 키의 집합으로 구성되며, 이 집합을 TraceId라고 정의한다. Pinpoint의 핵심 자료구조는 Span과 Trace, TraceId로 이루어져 있다. TraceId는 TransactionId(TxId)와 SpanId, ParentId로 이루어진 키의 집합이다. TransactionId는 메시지의 아이디이며, SpanId와 ParentId는 RPC의 부모 자식 관계를 나타낸다[1].

Jaeger는 분산 서비스간 트랜잭션을 추적하는 오픈소스 소프트웨어로서, 분산 트랜잭션을 모니터링하고 성능과 대기 시간을 최적화하며, 발생하는 문제의 해결을 위한 근본 원인분석(Root Cause Analysis, RCA)을 수행한다.

또한, Zipkin은 구글의 Dapper 구조를 기초로 개발된 여러 제품 중에서 가장 활성화되어 있는 오픈소스 제품으로서, Spring Cloud Sleuth를 통하여 트랜잭션 ID를 생성하여 분산 추적을 수행할 수 있다[5].

2.2 전력 업무 적용을 위한 고찰

본 연구에서 분산 추적 기술을 적용하고자 하는 분야는 일 평균 사용자가 15,000명이 많은 사용자가 사용하는 대규모 시스템에서 시스템 트랜잭션의 연계 관계를 분석하고 가시화하기 위한 업무이다. 기존 운영중인 업무시스템에 적용하기 위해서는 아래 <그림 5>와 같이 분산 추적을 위한 모듈(빨강색 박스)을 설치하여 트랜잭션 대표 ID, SpanID, pSpanID 등과 같은 데이터 생성과 수집이 요구된다. 실시간으로 운영되는 시스템의 안정적인 운영을 위한 관점에서는 본격적인 시스템 적용에 많은 검토가 필요한 사항이라는 것이다. 신규 개발되는 시스템 기능이나 모듈에는 필수적으로 분산 추적을 위한 아키텍처 반영을 하는 것이 중요하다고 생각된다.



〈그림 5〉 업무시스템 분산 추적 적용 아키텍처

3. 결 론

본 연구에서는 인터넷 서비스의 복잡도가 높아지면서 발생하는 문제의 해결을 위해서 필요한 분산 추적 기술의 기술 동향 및 서비스 아키텍처를 분석하고, 한전의 복잡한 구조의 업무 시스템에 적용하기 위한 방안을 개념적인 측면에서 검토하였다. 본 연구 결과를 반영할 수 있는 시스템 아키텍처 설계를 추진 예정이다.

[참 고 문 헌]

[1] Pinpoint, “https://d2.naver.com/helloworld/1194202”, 2015
[2] The OpenTracing project, “https://opentracing.io”
[3] .NET 분산 추적 개념, https://docs.microsoft.com/ko-kr/dotnet/core/diagnostics/distributed-tracing-concepts
[4] Benjamin H. Sigelman, “Dapper, a Large-Scale Distributed Systems Tracing Infrastructure, Google, 2010
[5] Jaeger를 활용한 분산 환경 서비스 로그 트레이싱 https://twofootdog.tistory.com/67