MSAP.ai

2025.10

LLM과 APM, 세션 클러스터를 통합한 차세대 운영 인텔리전스 백서

이 백서는 이러한 문제를 해결하기 위해 특허 NP25073-KR 에 기반한 혁신적 통합 아키텍처를 제시합니다. 즉, 대규모 언어 모델 (LLM), 애플리케이션 성능 관리 (APM), 그리고 세션 클러스터 기술을 하나의 유기적 플랫폼으로 융합하여 운영 데이터의 단절을 해소하고 지능형 관측 및 제어를 가능하게 하는 방안을 심층적으로 다룹니다.

Contact Us

02-6953-5427

hello@msap.ai

www.msap.ai

© 2025 MSAP.ai. All rights reserved.

Contents

© 2025 MSAP.ai., All Rights Reserved.

© 2025 MSAP.ai., All Rights Reserved.

© 2025 MSAP.ai., All Rights Reserved.

© 2025 MSAP.ai., All Rights Reserved.

© 2025 MSAP.ai., All Rights Reserved.

1장. 서론

1.1 LLM · APM · 세션 클러스터 통합 아키텍처를 통한 차세대 운영 인텔리전스의 실현

현대 IT 운영 환경은 MSA와 쿠버네티스 중심의 클라우드 네이티브 전환으로 인해 데이터 사일로와 복잡한 운영 도구 분절이라는 근본적 한계에 직면하고 있습니다.

이 백서는 이러한 문제를 해결하기 위해 특허 NP25073-KR 에 기반한 혁신적 통합 아키텍처를 제시합니다. 즉, 대규모 언어 모델 (LLM), 애플리케이션 성능 관리 (APM), 그리고 세션 클러스터 기술을 하나의 유기적 플랫폼으로 융합하여 운영 데이터의 단절을 해소하고 지능형 관측 및 제어를 가능하게 하는 방안을 심층적으로 다룹니다.

백서에서는 다음과 같은 핵심 내용을 포함합니다.

- 필요성: APM이 URL 단위의 트랜잭션 성능만 보는 한계를 극복하고, 세션 클러스터가 사용자 전체 여정을 보존할 수 없었던 문제를 해결하기 위해 두 데이터 영역을 통합해야 하는 기술적 배경을 설명합니다.
- 효과: LLM이 세션ID · 사용자 식별인자 · 트랜잭션 로그를 연결하여 '동시접속자수', 'URL 단위 부하', 'CPU 사용률 상관관계' 등을 자연어로 질의하고 분석할 수 있는 지능형 운영 인텔리전스 환경을 구현합니다.
- 적용 사례: OPENMARU Cluster (세션 클러스터)와 OPENMARU APM 간 실제 연계 사례를 통해, 공공기관 및 대규모 엔터프라이즈 환경에서 LLM 기반 운영 자동화(VibeOps)가 어떻게 실현되는지 보여줍니다. 특히 장애 발생 시점의 동시 사용자 분석, URL별 병목 탐지, 세션 제어 명령 자동화 등 실질적인 운영 효과를 구체적으로 제시합니다.

이 백서는 단순한 기술 백서가 아닙니다. 운영 데이터와 AI를 결합하여 운영자가 자연어로 시스템을 이해하고 제어할 수 있는 새로운 운영 패러다임, 즉 '지능형 운영 인텔리전스'의 실현 방향을 제시하는 기술적 청사진입니다.

LLM의 언어 이해 능력과 APM · 세션 데이터의 정확성을 결합함으로써, 이 백서는 AI 없이는 운영을 논할 수 없는 시대의 표준 아키텍처를 정립하고자 합니다.

© 2025 MSAP.ai., All Rights Reserved.

1.1.1 LLM·APM·세션클러스터를 통합하는 필요성

APM, 세션 클러스터, LLM은 각각 강력한 기술이지만, 독립적으로 사용될 때 명확한 기술적 한계를 드러냅니다. 아키텍처 관점에서 이 세 가지 기술의 통합은 선택이 아닌 필수이며, 그 이유는 다음과 같은 세 가지 핵심 논점으로 증명됩니다.

1. 데이터 단절의 문제 APM은 개별 트랜잭션(URL 요청 등)의 성능을 마이크로초 단위까지 정밀하게 추적하지만, 정작 그 트랜잭션을 발생시킨 사용자의 전체 활동 흐름, 즉 '세션 컨텍스트'를 놓칩니다. 반대로 세션 서버는 현재 접속 중인 사용자의 세션 상태는 관리하지만, 성능상의 제약으로 인해 사용자의 과거 트랜잭션 이력을 저장하지 못하고 세션이 종료되면 데이터를 소멸시킵니다. 이처럼 두 시스템의 데이터가 단절되어 있어 "특정 사용자가 어떤 경로를 거쳐 장애 상황에 도달했는가?"라는 근본적인 질문에 답하기 어렵습니다.
2. 운영 복잡성 심화 MSA와 쿠버네티스 환경에서는 하나의 사용자 요청이 수많은 서비스(Pod)를 거쳐 처리됩니다. 장애가 발생하면 운영자는 문제의 원인을 찾기 위해 APM, 세션 관리 도구, 분산된 로그 시스템 등 여러 시스템을 오가며 데이터를 수동으로 조합하고 연관 관계를 추론해야 합니다. 이러한 파편화된 분석 과정은 시간 소모가 클 뿐만 아니라, 복잡도가 높은 문제에서는 정확한 원인 규명(Root Cause Analysis) 자체를 불가능하게 만들기도 합니다.
3. LLM의 잠재력과 한계 LLM은 자연어 처리 능력과 방대한 지식을 바탕으로 복잡한 질의에 대한 답변을 생성하는 데 탁월한 잠재력을 보입니다. 하지만 LLM은 스스로 데이터를 생성하거나 검증할 수 없습니다. 정확하고 신뢰할 수 있는 실시간 운영 데이터를 컨텍스트로 제공받지 못하면, 그럴듯하지만 사실이 아닌 답변을 생성하는 '환각(Hallucination)' 현상을 일으키거나 무의미한 분석 결과를 내놓을 뿐입니다. LLM을 진정한 운영 인텔리전스 도구로 활용하기 위해서는 정제되고 구조화된 데이터 공급이 전제되어야 합니다.

1.1.2 "동시접속자수"와 "URL(트랜잭션) 단위"를 LLM이 이해하도록 만드는 목표

본 백서가 추구하는 핵심 기술 목표는 '동시접속자수'와 같은 추상적인 운영 지표와 개별 'URL(트랜잭션)' 로그처럼 파편화된 데이터를 LLM이 분석하고 추론할 수 있는 구조화된 정보로 변환하는

© 2025 MSAP.ai., All Rights Reserved.

것입니다. 이는 단순히 데이터를 한곳에 모으는 것을 넘어, 각 데이터 조각을 의미 있는 타임라인으로 연결하는 과정을 포함합니다.

이 아키텍처의 핵심은 사용자 식별인자, 세션 ID, 트랜잭션 데이터를 하나의 타임라인으로 완벽하게 연결하는 데 있습니다. 사용자가 로그인하여 세션을 시작하고, 여러 페이지(URL)를 이동하며 트랜잭션을 발생시키고, 로그아웃하기까지의 모든 과정을 하나의 연속된 흐름으로 재구성합니다. 이렇게 구조화된 데이터는 LLM에게 "특정 사용자가 겪은 전체 여정"이라는 명확한 컨텍스트를 제공하며, 이를 통해 "CPU 사용량이 급증했을 때 어떤 사용자들이 어떤 작업을 하고 있었나?"와 같은 고차원적인 질문에 대한 정확한 답변 생성을 가능하게 합니다.

1.2 참조 시스템 개요

본 백서에서 논의될 차세대 운영 인텔리전스 아키텍처는 세 가지 핵심 시스템 컴포넌트의 유기적인 결합을 통해 완성됩니다. 각 컴포넌트는 독립적인 역할을 수행하면서도, 서로 데이터를 주고받으며 더 큰 시너지를 창출합니다. 이 섹션에서는 OPENMARU Cluster, OPENMARU APM, 그리고 이 둘을 LLM과 통합하는 특허 기반 아키텍처의 역할을 개괄적으로 설명하여 독자의 이해를 돕고자 합니다.

1.2.1 OPENMARU Cluster: IMDG 기반 세션 클러스터링·WAS 고가용 구성 개요

- 기술 정의: OPENMARU Cluster는 인메모리 데이터 그리드(IMDG, In-Memory Data Grid) 기술을 기반으로 하는 세션 클러스터링 솔루션입니다. 기존에 각 WAS 인스턴스의 메모리에 저장되던 사용자 세션 데이터를 외부의 분산된 메모리 영역으로 분리하여 중앙에서 저장하고 관리합니다.
- 핵심 기능: 이 아키텍처의 가장 큰 장점은 고가용성(High Availability) 확보입니다. 특정 WAS 인스턴스에 장애가 발생하더라도 세션 정보는 외부 데이터 그리드에 안전하게 보존되므로, 사용자는 재로그인 없이 다른 정상 WAS 인스턴스를 통해 중단 없이 서비스를 이용할 수 있습니다. 또한, 사용자 접속이 급증할 경우 간단히 데이터 그리드 노드를 추가하는 것만으로 손쉽게 수평 확장이 가능합니다.
- 차별점: 단순 Key-Value 저장소인 Redis가 최종 일관성(Eventually Consistent) 모델을

© 2025 MSAP.ai., All Rights Reserved.

따르는 것과 달리, 데이터 그리드 방식은 여러 노드에 걸쳐 데이터의 강한 일관성(Strong Consistency)을 보장합니다. 이는 금융 거래나 동시 요청이 빈번한 엔터프라이즈 환경에서 세션 데이터의 정합성을 지키는 핵심적인 차이점이며, 노드 추가 시 수동 개입이 필요한 Redis와 달리 자동 리밸런싱 기능은 대규모 환경의 운영 복잡성을 근본적으로 해결합니다.

1.2.2 OPENMARU APM: 실시간 애플리케이션 성능 수집과 동시 사용자·TPS·URL 응답시간 계측 구조

- 기본 기능: OPENMARU APM은 WAS에 설치된 에이전트를 통해 실시간으로 애플리케이션의 성능 데이터를 수집하고 모니터링하는 시스템입니다. URL별 응답시간, 초당 트랜잭션 수(TPS), 동시 사용자 수, DB 쿼리 성능 등 서비스 품질과 직결되는 핵심 지표를 계측합니다.
- 확장된 역할: 본 아키텍처에서 OPENMARU APM은 단순 성능 모니터링 도구를 넘어, 전체 데이터 통합의 허브 역할을 수행합니다. 이는 세션서버의 휘발성(ephemeral) 데이터를 APM의 영구 저장소(permanent storage)로 이전시켜, 모든 사용자 여정을 시간에 구애받지 않고 분석할 수 있는 '시스템 오브 레코드(System of Record)'를 구축하는 핵심적인 설계입니다. OPENMARU Cluster(세션서버)로부터 실시간 세션 ID와 해당 세션의 사용자 식별인자를 수집하고, 이를 APM 에이전트가 수집한 트랜잭션 데이터와 매핑하여 APM 트랜잭션 데이터베이스에 영구적으로 저장합니다.

1.2.3 특허 명세서에 정의된 LLM 연동형 AI API 서버·세션서버·APM서버의 통합 구조 개요

특허 명세서(NP25073-KR)에 기술된 통합 아키텍처는 앞서 설명한 두 시스템을 LLM과 연결하여 지능형 운영을 가능하게 합니다. 전체 시스템은 다음과 같은 4가지 핵심 컴포넌트의 상호작용을 통해 동작합니다.

- AI API 서버: 사용자의 자연어 질의나 명령을 수신하는 중앙 컨트롤 타워입니다. 입력된 자연어를 분석하여 요청의 성격에 따라 '현재' 상태 조회가 필요하면 **세션툴**을, '과거' 이력 분석이 필요하면 **APM툴**을 선택적으로 호출합니다.

© 2025 MSAP.ai., All Rights Reserved.

- 세션서버 (OPENMARU Cluster): '지금 무슨 일이 일어나고 있는가?'(What is happening right now?)라는 실시간 질의에 답하는 데이터 소스입니다. 현재 활성 상태인 세션의 정보(세션 ID, 사용자 식별인자)만을 제공합니다.
- APM서버 (OPENMARU APM): '과거에 무슨 일이 있었는가?'(What has happened?)라는 이력 분석 질의에 답하는 영구적인 데이터 소스입니다. 사용자의 모든 과거 트랜잭션과 세션 컨텍스트가 영구적으로 저장된 시스템의 유일한 진실 공급원(Single Source of Truth)입니다.
- LLM서버: AI API 서버로부터 정제된 데이터와 분석 요청을 전달받는 두뇌 역할을 합니다. 이 데이터를 기반으로 자연어 질의에 대한 심층적인 답변을 생성하거나, 특정 사용자를 로그아웃시키는 등의 제어 명령을 해석하여 실행 가능한 명령으로 변환합니다.

지금까지 서론에서는 현대 IT 환경의 문제의식에서 출발하여, 이를 해결하기 위한 기술 통합의 필요성과 아키텍처의 개요를 살펴보았습니다. 다음 장부터는 이 아키텍처를 구성하는 핵심 설계 원칙과 데이터 흐름을 심층적으로 증명해 나가겠습니다.

2장. 세션 클러스터링과 트랜잭션 식별의 기술적 전제

2.1. 웹 기반 트랜잭션에서의 세션 문제

현대적인 클라우드 네이티브 환경은 탄력적인 확장과 자동 복구를 핵심 가치로 삼고 있습니다. 그러나 이러한 동적인 아키텍처는 전통적인 웹 애플리케이션 서버(WAS)가 상태(State)를 관리하는 방식과 근본적으로 충돌합니다. WAS 인스턴스 내부에 사용자의 세션 정보를 저장하는 기존의 방식은 Pod가 수시로 생성되고 소멸하는 클라우드 환경에서 더 이상 유효하지 않으며, 이는 사용자의 서비스 경험과 시스템 전체의 안정성에 직접적인 위협이 됩니다. 따라서 상태 정보를 애플리케이션 외부에서 안정적으로 관리하는 새로운 아키텍처의 도입은 선택이 아닌 필수가 되었습니다.

2.1.1. 세션 메모리의 휘발성과 "현재 세션만 보이는" 한계

웹 애플리케이션 서버(WAS)의 세션 메모리는 그 구조상 두 가지 본질적인 한계를 가지고 있습니다. 이는 장애 분석과 사용자 경험 관리에 심각한 제약을 만듭니다.

© 2025 MSAP.ai., All Rights Reserved.

첫째는 휘발성(Volatility) 문제입니다. 세션 데이터는 물리적으로 서버의 휘발성 메모리(RAM)에 저장됩니다. 이는 세션이 정상적으로 종료되거나, 예측하지 못한 서버 장애로 인해 WAS 인스턴스가 중단될 경우 해당 메모리에 저장된 모든 세션 데이터가 영구적으로 소멸됨을 의미합니다. 사용자의 장바구니 정보나 로그인 상태가 순식간에 사라지는 현상은 바로 이 휘발성 때문에 발생합니다.

둘째는 현재성(Present-Only)의 한계입니다. WAS 세션 메모리는 성능상의 제한으로 인해 사용자의 과거 접속 기록 등을 저장하지 못하고, 오직 '현재 진행 중인' 세션의 상태 정보만을 유지하도록 설계되었습니다. 과거에 사용자가 어떤 경로로 서비스를 이용했는지, 어떤 요청을 보냈는지에 대한 이력 데이터는 세션 메모리에 저장되지 않습니다. 결과적으로, 세션 데이터만으로는 과거 사용자의 트랜잭션 이력을 추적하거나 특정 장애가 발생한 시점의 정확한 맥락을 파악하는 것이 원천적으로 불가능합니다.

2.1.2. WAS 장애 시 세션 유실과 재접속 폭주가 만드는 추가 부하 문제

단일 WAS 인스턴스의 장애는 단순히 해당 서버에 접속된 사용자들의 불편으로 끝나지 않고, 전체 시스템의 안정성을 위협하는 연쇄적인 악영향을 초래할 수 있습니다.

장애가 발생한 WAS 인스턴스의 메모리에 저장되어 있던 모든 사용자 세션 정보는 즉시 유실됩니다. 이로 인해 해당 서버에 접속해 있던 사용자들은 예고 없이 강제 로그아웃을 경험하게 되며, 서비스를 계속 이용하기 위해 거의 동시에 재접속을 시도하게 됩니다.

이러한 갑작스러운 재접속 트래픽 폭주는 정상적으로 운영 중이던 나머지 WAS 서버들에게 심각한 추가 부하를 가중시킵니다. 이는 마치 연쇄 추돌 사고와 같아서, 하나의 장애가 다른 서버들의 과부하를 유발하고 결국 전체 시스템이 마비되는 대규모 장애로 확산될 수 있는 잠재적 위험을 내포하고 있습니다.

2.1.3. 세션만으로는 "어떤 URL에서, 어떤 사용자가, 어떤 시점에" 문제가 났는지 알 수 없는 이유

서비스 운영 중 특정 기능에서 문제가 발생했을 때, 세션 데이터만으로는 장애의 근본 원인을 특정하기가 매우 어렵습니다. 그 이유는 세션 데이터의 목적과 구조적 한계 때문입니다.

세션 데이터는 본질적으로 사용자의 로그인 상태를 유지하는 데 초점이 맞춰져 있을 뿐, 사용

© 2025 MSAP.ai., All Rights Reserved.

자가 요청한 구체적인 트랜잭션의 상세 이력을 담고 있지 않습니다. 즉, 세션 ID는 존재하지만 그 세션 안에서 사용자가 어떤 URL을 호출했고, 어떤 파라미터를 전송했는지와 같은 행위 기록은 포함하지 않습니다.

결론적으로, 운영팀은 “어떤 URL에서, 어떤 사용자가, 어떤 시점에” 오류를 겪었는지 명확히 추적할 수 없게 됩니다. 이는 문제의 원인을 파악하고 해결하는 데 막대한 인력과 시간을 소요하게 만드는 대표적인 운영 비효율의 원인이 됩니다.

이처럼 세션이 가진 휘발성, 현재성, 그리고 정보의 불완전성이라는 본질적인 한계들은, 결국 상태(State)를 WAS 외부의 전문화된 시스템에서 관리하는 새로운 아키텍처의 등장을 필연적으로 만들었습니다.

2.2. OPENMARU Cluster의 IMDG 기반 구조

앞서 제기된 전통적인 세션 관리의 한계를 극복하기 위한 핵심 전략은 ’세션의 외부화(Session Externalization)’입니다. 이는 세션 데이터를 WAS의 생명주기로부터 완전히 분리하여 독립적인 외부 저장소에서 관리하는 개념입니다. OPENMARU Cluster는 바로 이 전략을 In-Memory Data Grid(IMDG) 기술을 기반으로 구현한 솔루션입니다. IMDG를 통해 세션 데이터를 여러 노드에 걸쳐 분산 및 복제함으로써, 특정 WAS 인스턴스의 장애와 무관하게 세션의 연속성을 보장하고, 트래픽 변화에 따라 유연하게 확장할 수 있는 고가용성 아키텍처를 완성합니다.

2.2.1. 세션을 WAS 밖(IMDG)으로 빼는 이유와 다중 WAS 인스턴스 간 공유 방식

세션을 WAS 외부의 IMDG(In-Memory Data Grid)에 저장하는 아키텍처는 다음과 같은 두 가지 핵심적인 이점을 제공합니다.

- 고가용성 확보: 세션 데이터를 WAS의 생명주기와 완벽하게 분리함으로써, 특정 WAS 인스턴스에 장애가 발생하더라도 세션 정보는 IMDG 클러스터 내에 안전하게 보존됩니다. 사용자의 요청은 다른 정상적인 WAS 인스턴스로 즉시 전달되어 중단 없이 서비스를 이어갈 수 있습니다. 이는 개별 서버의 장애가 전체 서비스 중단으로 이어지는 것을 방지하는 강력한 Failover 메커니즘을 제공합니다.

© 2025 MSAP.ai., All Rights Reserved.

- 탄력적 확장성: 예기치 못한 사용자 접속 폭주 상황에서 기존 방식은 WAS 증설만으로는 한계가 있습니다. OPENMARU Cluster는 WAS 인스턴스뿐만 아니라 IMDG 노드를 간단히 추가하는 것만으로 응답 시간 저하 없이 시스템 전체를 수평적으로 확장(Scale-out)할 수 있습니다. 추가된 노드는 클러스터에 자동으로 편입되어 데이터 분산 및 부하 분담에 즉시 참여하게 됩니다.

다중 WAS 인스턴스의 세션 공유 방식은 아키텍처의 확장성과 운영 효율성을 결정하는 핵심 요소입니다. IMDG는 모든 노드가 동등한 역할을 수행하는 P2P(Peer-to-Peer) 아키텍처를 기반으로 데이터를 동적으로 분산 및 복제합니다. 이는 Redis 등에서 사용하는 클라이언트-서버 및 Primary/Replica 샤딩 모델과 근본적인 차이를 보이며, 대규모 엔터프라이즈 환경에서 다음과 같은 명확한 아키텍처 우위를 제공합니다.

1. 진정한 선형적 확장성: P2P 모델에서는 WAS 노드를 추가하는 것이 곧 데이터 그리드의 용량과 처리 능력 증가로 직결될 수 있습니다. 반면, 고정된 해시 슬롯(Hash Slot)을 사용하는 Redis 클러스터는 노드 추가 시 슬롯을 재분배(Resharding)하는 복잡한 과정이 필요하여, WAS 확장과 데이터 저장소 확장이 별개의 운영 작업으로 분리됩니다.
2. 운영 단순성: IMDG 클러스터에 새로운 노드가 참여하면 데이터는 자동으로 재조정(Rebalance)됩니다. 이는 수동 개입이나 복잡한 도구가 필요한 Redis의 리샤딩에 비해, 동적으로 서버 증설이 빈번한 클라우드 환경에서 운영 부담을 획기적으로 줄여줍니다.
3. 네트워크 병목 감소: P2P 구조는 데이터 지역성(Data Locality)을 높일 수 있으며, 모든 세션 I/O가 특정 Primary 노드와의 클라이언트-서버 통신에 의존하지 않으므로 네트워크 병목 현상을 완화할 수 있습니다.

결론적으로 IMDG의 P2P 아키텍처는 대규모의 동적인 환경에서 요구되는 자동화된 확장성, 운영 유연성, 그리고 높은 안정성을 제공하여 엔터프라이즈급 세션 클러스터링에 가장 적합한 구조입니다.

© 2025 MSAP.ai., All Rights Reserved.

2.2.2. Tomcat/JBoss/이기종 WAS 간 세션 공유 패턴과 공공기관 조달 활용 구조

OPENMARU Cluster의 가장 큰 장점 중 하나는 특정 벤더나 기술 스택에 종속되지 않는 높은 기술적 유연성입니다. 이를 통해 복잡하고 다양한 기업 및 공공기관의 IT 환경에 손쉽게 적용될 수 있습니다.

OPENMARU Cluster가 지원하는 주요 환경은 다음과 같습니다.

- 주요 오픈소스 WAS 지원: `Tomcat`, `JBoss`와 같이 널리 사용되는 오픈소스 WAS 환경에서 안정적인 세션 클러스터링을 구성할 수 있습니다.
- 이기종 WAS 간 세션 공유: 서로 다른 종류의 WAS(예: Tomcat과 JBoss) 인스턴스들이 혼재된 환경에서도 하나의 논리적인 클러스터로 묶어 세션 정보를 원활하게 공유할 수 있습니다.
- 서로 다른 웹 애플리케이션 간 세션 공유: 단일 도메인 내에서 운영되는 여러 웹 애플리케이션이 동일한 사용자 세션을 공유해야 하는 경우(SSO, Single Sign-On)에도 효과적으로 활용될 수 있습니다.

또한, OPENMARU Cluster는 제품의 신뢰성과 안정성을 바탕으로 조달청 디지털서비스몰에 정식으로 등록되어 있습니다. 이를 통해 다수의 공공기관에서 복잡한 계약 절차 없이 신속하게 솔루션을 도입하여 대국민 서비스의 안정성을 확보하는 데 활용하고 있습니다.

2.2.3. 세션ID-사용자 식별인자 매핑을 위한 필수 필드 구성

단순히 세션의 상태를 유지하는 것을 넘어, 장애 분석 및 사용자 행위 추적을 위해서는 익명의 세션을 실제 사용자와 연결하는 데이터 구조가 반드시 필요합니다.

이를 위해 OPENMARU Cluster가 관리하는 세션 데이터 내에는 다음 두 가지 핵심 필드가 포함되고 서로 명확하게 매핑(mapping)되어야 합니다.

- 세션 ID (Session ID): 사용자가 웹사이트에 최초 접속 시 생성되는, 해당 통신 세션을 식별하기 위한 고유 키입니다.

© 2025 MSAP.ai., All Rights Reserved.

- 사용자 식별인자 (User Identifier): 로그인 ID, 고객 번호 등 시스템 내에서 실제 최종 사용자를 특정할 수 있는 고유한 값입니다.

이 두 필드의 매핑은 2.1.3절에서 제기된, 세션만으로는 '누가, 무엇을, 언제' 실행했는지 알 수 없었던 문제를 해결하는 첫 번째이자 가장 중요한 단계입니다. 이 매핑 정보는 사용자가 로그인하는 시점에 웹서버 등에서 추출되어 **세션 ID**와 함께 세션서버(IMDG)로 전송됩니다. 이로써 우리는 익명으로 보였던 수많은 세션 ID와 실제 사용자를 연결하는 첫 번째 단서를 확보하게 됩니다. 이 **세션 ID - 사용자 식별인자** 매핑 구조는 이후 APM 및 LLM과의 연동을 통해 특정 사용자의 트랜잭션 흐름을 심층적으로 분석하는 데 가장 핵심적인 전제 조건이 됩니다.

지금까지 살펴본 바와 같이, IMDG 기반의 세션 외부화는 WAS 장애에 대한 안정성과 확장성 문제를 해결했습니다. 하지만 "어떤 사용자가 어떤 기능을 사용하다가 문제가 발생했는가?"라는 질문에 답하기 위해서는 트랜잭션 단위의 더욱 상세한 데이터가 필요하며, 이는 APM(Application Performance Management)을 통한 추가적인 데이터 수집을 통해서만 완성될 수 있습니다.

2.3. URL(트랜잭션) 레벨까지 내려가기 위한 추가 수집 포인트

LLM(거대 언어 모델)이 사용자의 복잡한 행동 패턴을 이해하고 "지난주에 로그인 오류가 가장 많았던 기능이 무엇인가?"와 같은 자연어 질의에 정확히 응답하기 위해서는 데이터의 깊이가 달라져야 합니다. 안정적으로 유지되는 세션 정보만으로는 부족하며, 사용자의 모든 개별 요청, 즉 '트랜잭션' 단위의 구체적인 컨텍스트 데이터가 결합되어야 합니다. APM을 통해 수집되는 트랜잭션 데이터는 LLM이 사용자의 의도와 시스템의 반응을 연결하여 분석할 수 있게 만드는 필수적인 정보입니다.

2.3.1. 웹서버/WAS 단의 URL·HTTP Method·쿼리 파라미터·호출 시간

사용자의 구체적인 행위, 즉 하나의 '트랜잭션'을 명확하게 정의하기 위해 APM 에이전트는 웹서버 또는 WAS 레벨에서 다음과 같은 핵심 데이터 항목들을 수집해야 합니다.

- URL: 사용자가 시스템에 요청한 서비스의 정확한 경로로, "무엇을" 요청했는지 나타냅니다. (예: `/orders/checkout`)

© 2025 MSAP.ai., All Rights Reserved.

- HTTP Method: 요청의 종류(GET, POST, PUT, DELETE 등)를 나타내며, 사용자가 조회, 생성, 수정, 삭제 중 어떤 행위를 의도했는지 알려줍니다.
- 쿼리 파라미터 (Query Parameters): URL에 포함된 부가적인 요청 값으로, 트랜잭션을 더욱 상세하게 정의합니다. (예: `?productId=123&quantity=2`)
- 호출 시간 (Timestamp): 해당 트랜잭션이 발생한 정확한 시점으로, 모든 이벤트의 시간 순서를 재구성하는 기준이 됩니다.

이 정보들이 결합될 때 비로소 우리는 "어떤 사용자가, 어떤 시점에, 어떤 데이터를 가지고, 어떤 기능을 요청했는지"를 명확하게 식별할 수 있습니다.

2.3.2. 세션ID-사용자 식별인자 매핑 정보의 세션서버 등록

데이터의 흐름을 명확히 이해하는 것이 중요합니다. 앞서 **2.2.3**절에서 설명했듯이, 사용자가 웹사이트에 로그인하면 웹서버(또는 WAS)는 해당 요청을 처리하면서 **사용자 식별인자**를 식별하고 이를 현재 요청의 **세션ID**와 함께 세션서버(IMDG)로 전송하여 저장합니다.

이 과정은 단순히 세션 정보를 유지하는 것을 넘어, 이후 APM이 데이터를 종합할 때 효율성을 극대화하는 중요한 사전 작업입니다. APM 서버가 특정 트랜잭션과 관련된 세션 정보를 조회해야 할 때, 세션서버에서 **세션ID** 하나만으로 **사용자 식별인자**까지 한 번에 가져올 수 있도록 데이터를 미리 구조화하는 핵심적인 단계입니다.

2.3.3. 세션데이터를 APM 트랜잭션 DB에 "영구화"하는 구조의 필요성

본 아키텍처를 완성하는 마지막 단계는 분산된 두 종류의 데이터를 하나로 통합하고 영구적으로 보관하는 과정입니다. 이는 APM 서버가 중심적인 역할을 수행합니다.

1. 먼저, APM 서버는 각 WAS에 설치된 에이전트로부터 개별 트랜잭션 정보(URL, 호출 시간, 응답 시간 등)를 실시간으로 수집합니다.
2. 동시에, APM 서버는 세션서버(IMDG)에 질의하여 해당 트랜잭션이 발생한 시점의 세션 정보(세션ID, 사용자 식별인자)를 가져옵니다.
3. 마지막으로, APM 서버는 수집된 두 데이터를 **세션ID**를 공통 키(Matching Key)로 사용하여 결합한 후, 이 통합된 데이터를 APM 트랜잭션 데이터베이스(DB)에 영구적으로 저장합니다.

© 2025 MSAP.ai., All Rights Reserved.

이러한 '영구화' 구조는 세션 메모리가 가졌던 '휘발성'과 '현재성'이라는 근본적인 한계를 완벽하게 극복합니다. 이제 우리는 특정 시점에 사라지는 휘발성 데이터가 아닌, 서비스 시작부터 현재까지의 모든 사용자 트랜잭션 이력을 분석할 수 있는 견고한 데이터 기반을 마련하게 됩니다.

결론적으로, IMDG를 통해 세션 관리의 안정성을 확보하고, APM을 통해 수집된 깊이 있는 트랜잭션 데이터를 **세션ID**로 연결하여 영구적으로 결합함으로써, 비로소 LLM을 통해 사용자의 모든 행위를 자연어로 분석하고 나아가 제어할 수 있는 모든 기술적 전제가 완성되었습니다.

3장. OPENMARU APM과 세션/URL 연계 수집 설계

3.1 OPENMARU APM의 기본 수집 모델

정확하고 신뢰도 높은 애플리케이션 성능 데이터 수집은 시스템 상태를 진단하고 미래의 장애를 예측하는 관측성(Observability)의 출발점입니다. 그러나 단순히 시스템 지표를 수집하는 것을 넘어, 개별 사용자의 경험과 직접 연결될 때 그 가치는 극대화됩니다. 이 섹션에서는 OPENMARU APM의 기본 데이터 수집 모델을 분석하고, 이 모델이 어떻게 사용자의 세션 정보와 결합하여 고차원적인 분석을 수행하기 위한 필수적인 기반이 되는지 설명합니다.

3.1.1 에이전트 기반 호출 트레이스와 자바 기반 웹서비스 계측 구조

OPENMARU APM은 에이전트(Agent) 기반 호출 트레이스 메커니즘을 통해 애플리케이션의 성능을 심층적으로 분석합니다. 이 에이전트는 모니터링 대상인 WAS(Web Application Server)에 연결되어, 자바 기반 웹 서비스에서 발생하는 모든 개별 트랜잭션의 흐름을 계측합니다.

이는 일반적으로 런타임에 바이트코드 조작(Bytecode Manipulation) 기술을 통해 구현됩니다. 이 방식을 통해 애플리케이션 소스 코드를 직접 수정하지 않고도 메소드 호출을 가로채고 성능 데이터를 수집할 수 있습니다. 사용자 요청이 시스템에 들어오는 순간부터 응답이 나갈 때까지, 에이전트는 각 메소드 호출, 데이터베이스 쿼리, 외부 API 호출 등 트랜잭션의 모든 단계를 추적합니다. 이를 통해, 단편적인 시스템 지표(CPU, 메모리)만으로는 파악할 수 없는 애플리케이션 내부의 병목 지점과 지연 원인을 명확하게 식별할 수 있습니다. 이 정밀한 계측 구조는 이후 사용자 세션 정보와 결합하여 "어떤 사용자가 어떤 기능에서 지연을 겪었는가"를 분석하는 핵심적인 데이

© 2025 MSAP.ai., All Rights Reserved.

터 소스가 됩니다.

3.1.2 수집 지표

APM은 시스템의 상태를 종합적으로 진단하고 잠재적 문제를 예측하기 위해 다음과 같은 핵심 성능 지표(KPI)를 수집합니다. 각 지표는 고유한 분석적 가치를 제공합니다.

- 동시 사용자 수 (Concurrent Users)
  - 분석적 가치: 특정 시점에 활성 트랜잭션을 처리하고 있는 사용자 수를 나타냅니다. 이 지표는 시스템이 현재 감당하고 있는 실질적인 부하를 측정하는 기준으로, 급격한 증가는 잠재적인 성능 저하의 전조가 될 수 있습니다.
- 초당 트랜잭션 (TPS, Transactions Per Second)
  - 분석적 가치: 시스템의 처리 용량을 나타내는 가장 직접적인 지표입니다. TPS의 변화 추이를 통해 서비스의 활성도와 성능 변화를 파악하고, 용량 증설 계획의 근거로 활용할 수 있습니다.
- 평균 응답시간 (Average Response Time)
  - 분석적 가치: 사용자 경험의 질을 나타내는 핵심 지표입니다. 응답시간이 길어질수록 사용자 만족도는 저하되므로, 이 지표를 지속적으로 모니터링하여 서비스 수준 협약(SLA)을 준수하고 있는지 확인할 수 있습니다.
- CPU 사용률
  - 분석적 가치: 애플리케이션 로직의 비효율성이나 과도한 연산 부하를 진단하는 데 사용됩니다. 특정 트랜잭션이 발생할 때 CPU 사용률이 급증한다면, 해당 기능의 코드 최적화가 필요함을 시사합니다.
- DB 쿼리 응답시간
  - 분석적 가치: 전체 응답시간 중 데이터베이스가 차지하는 비중을 분석하여, 비효율적인 SQL 쿼리나 인덱스 문제를 식별하는 데 결정적인 역할을 합니다.
- 커넥션 수 (Connection Count)

© 2025 MSAP.ai., All Rights Reserved.

- 분석적 가치: 데이터베이스나 외부 시스템과의 커넥션 풀 상태를 모니터링합니다. 커넥션 고갈은 심각한 서비스 장애로 이어질 수 있으므로, 임계치 관리가 필수적입니다.

3.1.3 장애 구간 축소와 시간축 재생(Replay)을 위한 데이터 보관 전략

세션서버의 가장 큰 제약은 세션 데이터를 휘발성 메모리(volatile session memory)에 저장한다는 점입니다. 세션이 종료되면 관련 데이터는 즉시 소멸되므로 과거 트랜잭션 이력을 추적할 수 없습니다. 본 아키텍처는 이 문제를 해결하기 위해 APM을 통해 수집된 모든 트랜잭션 및 성능 지표 데이터를 APM 트랜잭션 데이터베이스에 영구적으로 보관하는 전략을 채택합니다. 이는 단순한 데이터 저장을 넘어, 장애 분석과 근본 원인 분석(RCA)을 위한 전략적 자산이 됩니다.

- 장애 구간 축소: 특정 장애가 발생했을 때, 운영자는 해당 시간대의 데이터를 집중적으로 분석하여 문제의 범위를 효과적으로 좁힐 수 있습니다. 예를 들어 “오전 10시 5분부터 10분간 응답시간이 급증했다”는 사실이 확인되면, 해당 시간대의 트랜잭션, CPU, DB 쿼리 데이터만을 집중 분석하여 원인을 빠르게 규명할 수 있습니다.
- 시간축 재생 (Replay): 영구 보관된 데이터를 통해 과거 특정 시점의 시스템 상황을 마치 비디오를 재생하듯 재구성할 수 있습니다. 이를 통해 장애 발생 당시의 부하 패턴, 비정상적인 트랜잭션 흐름 등을 면밀히 검토하여 일회성으로 나타났다 사라지는 문제를 해결하는 데 중요한 단서를 제공합니다.

이 모델은 시스템 상태에 대한 포괄적인 뷰를 제공하지만, 이 익명의 성능 지표들이 그것을 생성한 특정 사용자 여정과 연관될 때 비로소 진정한 잠재력이 발현됩니다.

3.2 세션서버-APM서버 교차 수집

전통적인 관측성 아키텍처의 근본적인 결함은 시스템 성능 지표와 실제 사용자 경험 데이터 사이의 단절에 있습니다. 우리는 APM과 세션서버 사이에 직접적인 데이터 통로를 설계함으로써 이 간극을 메우고, 고립된 데이터 포인트들을 사용자-시스템 상호작용에 대한 통합된 서사로 변환합니다.

이 아키텍처의 핵심은 세션 관리를 위해 전용 인메모리 데이터 그리드(IMDG) 기반 세션서버를 채택한 것입니다. 일반적인 대안인 Redis를 세션 저장소로 사용하는 경우, 직렬화 오버헤드, 비동기 복제로 인한 장애 극복(Failover) 시 데이터 유실 가능성, 동시 요청에서의 ‘마지

© 2025 MSAP.ai., All Rights Reserved.

막 쓰기 승리(Last-Write-Wins)' 문제 등 운영상 치명적인 리스크에 노출될 수 있습니다. 반면 IMDG 기반 아키텍처는 세션 데이터의 강한 일관성(Strong Consistency)과 자동 데이터 재조정(Rebalancing) 기능을 제공하여, 엔터프라이즈 환경에서 요구되는 데이터 무결성과 확장성을 보장하는 전략적 선택입니다.

3.2.1 세션서버에서 세션ID·사용자 식별인자를 추출해 APM으로 올리는 플로우

특허 명세서(NP25073-KR)에 기술된 아키텍처는 두 시스템 간의 데이터 연계를 위한 명확한 플로우를 정의합니다.

1. 사용자 접속 및 세션 생성: 사용자가 웹사이트에 접속하면, 세션서버는 해당 사용자의 트랜잭션을 위한 **세션ID**를 생성하여 IMDG 기반 세션메모리에 저장합니다.
2. 사용자 식별인자 전송: 동시에 웹서버는 로그인 정보 등을 통해 획득한 **사용자 식별인자**(예: 사용자 ID)를 세션서버로 전송하고, 세션서버는 이를 **세션ID**와 매칭하여 세션메모리에 함께 보관합니다.
3. APM서버의 정보 수집: APM서버는 두 가지 경로로 정보를 수집합니다.
    - WAS 에이전트를 통해 개별 **트랜잭션** 정보를 수집합니다.
    - 세션서버의 세션메모리에 직접 접근하여, 현재 활성화된 **세션ID**와 그에 매칭된 **사용자 식별인자** 정보를 수집합니다.

이 데이터 흐름을 통해, 휘발성 메모리에만 존재하던 사용자의 세션 정보가 영구적으로 저장될 APM 트랜잭션 데이터와 결합될 준비를 마치게 됩니다.

3.2.2 APM 트랜잭션 DB에서 "세션ID=트랜잭션"으로 매칭하는 키 설계

APM 트랜잭션 데이터베이스 내에서 **세션ID**는 단편적인 데이터를 의미 있는 사용자 경험 이력으로 변환하는 핵심적인 공통 키(Common Key) 역할을 합니다.

세션서버로부터 수집된 **세션ID**와 **사용자 식별인자** 정보는 APM 트랜잭션 DB에 저장될 때, 동일한 **세션ID**를 가진 트랜잭션 정보와 매칭되어 하나의 레코드로 통합됩니다. 즉, **사용자 식별**

© 2025 MSAP.ai., All Rights Reserved.

인자, 세션ID, 트랜잭션 상세 정보(URL, 응답시간 등)가 모두 연결된 상태로 영구 저장됩니다.

이러한 키 설계는 "어떤 URL의 응답시간이 3초 걸렸다"는 단편적인 성능 데이터를 "사용자 'John'이 장바구니 페이지(**/cart**)에 접근했을 때 3초의 지연을 겪었다"는 구체적이고 실행 가능한(actionable) 정보로 변환시키는 결정적인 역할을 합니다.

3.2.3 실시간 세션과 과거 트랜잭션의 조회 전략

데이터 조회 요청의 성격에 따라 질의 대상을 효율적으로 분리하는 전략은 시스템 전체의 성능을 최적화합니다.

- 실시간 세션 조회: "현재 어떤 사용자가 어느 페이지에 머물고 있는가?"와 같은 실시간 상태 조회 요청은 세션서버의 세션메모리를 직접 질의합니다. 세션메모리는 현재 진행 중인 세션 데이터만을 휘발성으로 관리하므로 매우 빠른 응답이 가능합니다.
- 과거 트랜잭션 이력 조회: "어제 오후 3시에 장애를 유발한 사용자의 모든 요청 내역을 분석해달라"와 같은 과거 이력 조회는 영구 저장된 APM 트랜잭션 데이터베이스를 질의합니다. 이는 대용량의 과거 데이터를 분석하는 데 최적화되어 있습니다.

이러한 분리는 과거 데이터에 대한 리소스 집약적인 분석 쿼리가 운영 중인 저지연 세션서버의 성능을 저하시키는 것을 방지하여 실시간 사용자 경험을 보존하는 핵심적인 아키텍처 원칙입니다.

모든 트랜잭션에 사용자 ID를 성공적으로 결합함으로써, 우리는 시스템 부하를 보다 미묘하게 이해하기 위한 기반을 마련했으며, 이를 통해 종종 모호하게 사용되는 '동시접속자수'라는 지표를 외과적 정밀함으로 해부할 수 있게 되었습니다.

3.3 동시접속자수 관점 정리

'동시접속자수'는 시스템 용량 산정과 성능 분석의 핵심 지표이지만, 이 용어는 종종 혼용되어 오해를 불러일으킵니다. 측정 방식과 관점에 따라 그 의미가 크게 달라질 수 있기 때문입니다. 따라서 시스템 부하를 정확하게 이해하고 예측하기 위해서는 '동시접속자수'의 개념을 명확히 정의하고 구분하는 것이 매우 중요합니다.

© 2025 MSAP.ai., All Rights Reserved.

3.3.1 APM의 동시 사용자 수와 세션서버의 활성 세션 수의 차이

시스템 부하를 해석할 때 가장 명확히 구분해야 할 두 지표는 APM이 측정하는 '동시 사용자 수'와 세션서버가 관리하는 '활성 세션 수'입니다.

활성 세션 수 (Active Session Count)

- 측정 주체: 세션서버
- 의미: 웹사이트에 로그인하여 세션을 유지하고 있는 모든 사용자의 수. 사용자가 단순히 창을 열어두고 아무런 활동을 하지 않아도 활성 세션으로 집계됩니다.
- 분석적 가치: 서비스의 잠재적 사용자 풀(Pool)의 크기를 나타내며, 라이선스나 메모리 용량 산정의 기준이 될 수 있습니다.

동시 사용자 수 (Concurrent User Count)

- 측정 주체: APM
- 의미: 특정 시점에 실제로 서버에 요청을 보내 트랜잭션을 처리 중인 사용자의 수. 즉, 시스템에 실질적인 부하를 유발하고 있는 사용자를 의미합니다.
- 분석적 가치: 시스템의 CPU, 스레드 등 실시간 리소스 사용량과 직접적인 상관관계를 가지며, 성능 병목 현상을 분석하는 데 결정적인 지표입니다.

예를 들어, 1,000명의 사용자가 로그인 상태(활성 세션 수 = 1,000)이더라도, 특정 순간에 50명만이 상품 조회 버튼을 클릭했다면 동시 사용자 수는 50이 됩니다. 이 두 지표를 혼동하는 것은 용량 계획에서 치명적인 오산으로 이어지며, 시스템이 감당해야 할 실제 트랜잭션 부하가 아닌 로그인 수에 기반하여 값비싼 인프라를 과잉 공급하는 결과를 초래하는 경우가 많습니다.

3.3.2 URL별 동시 접속량과 웹서버 리소스(CPU, 스레드풀) 상관분석

시스템 전체의 동시 사용자 수를 파악하는 것을 넘어, 부하를 유발하는 원인을 특정 기능 단위로 식별하는 것이 중요합니다. 이를 위해 특정 URL별 동시 접속 트래픽과 웹서버의 핵심 리소스(CPU 사용률, 스레드풀 점유율 등) 사용량 간의 상관관계를 분석하는 방법론이 사용됩니다.

예를 들어, APM 데이터를 분석한 결과 `/api/v1/search` URL에 대한 동시 요청이 50을 넘어서는 순간, 웹서버의 CPU 사용률이 90%까지 치솟고 스레드풀이 고갈되는 패턴을 발견할 수

© 2025 MSAP.ai., All Rights Reserved.

있습니다. 이는 시스템 전체의 문제가 아니라, 특정 '검색' 기능이 자원을 과도하게 소모하는 병목 지점임을 명확히 알려줍니다. 이 분석을 통해 개발팀은 전체 시스템을 재설계하는 대신, 해당 URL의 로직과 관련 쿼리를 집중적으로 최적화하여 문제를 해결할 수 있습니다.

3.3.3 LLM이 이해하기 쉬운 형태로의 메트릭 변환

분석된 모든 지표와 상관관계 데이터의 최종 목적지는 인간 운영자뿐만 아니라, 자연어 기반으로 상호작용할 거대 언어 모델(LLM)이 될 수 있습니다. LLM이 데이터를 효과적으로 이해하고 처리하기 위해서는, 수집된 원시 데이터를 구조화된 포맷으로 변환하는 과정이 필수적입니다.

이 과정에서 핵심은 데이터를 JSON 형식으로 구조화하는 것입니다. 예를 들어, 특정 시간대의 URL별 성능 지표는 다음과 같이 표현될 수 있습니다.

```
{ "timestamp": "2024-10-21T10:05:00Z", "url": "/api/v1/search", "concurrent_users": 52, "avg_response_time_ms": 3500, "cpu_utilization_percent": 91.5, "thread_pool_active": 198, "thread_pool_max": 200 }
```

더 나아가, 이러한 데이터를 클라우드 네이티브 컴퓨팅 재단(CNCF)의 표준인 OpenTelemetry 스키마와 연계하면 데이터의 상호운용성이 극대화됩니다. 이는 특정 벤더의 솔루션에 종속되지 않고, 다양한 관측성 도구와 LLM이 데이터를 일관된 방식으로 해석하고 활용할 수 있는 기반을 마련하는 현대적인 접근 방식입니다.

결론적으로, 3장에서 제시된 설계는 개별적인 APM 성능 지표 수집에서 시작하여, 세션 정보를 결합해 사용자 경험의 맥락을 부여하고, 최종적으로는 LLM이 분석 가능한 유의미한 데이터로 정제되는 완전한 파이프라인을 구축합니다. 이 체계적인 데이터 연계와 변환 과정은 단순히 과거의 문제를 해결하는 것을 넘어, "CPU 사용량이 급증한 원인이 된 URL은 무엇이고, 어떤 사용자들이 영향을 받았는가?"와 같은 복합적인 질문에 자연어로 답할 수 있는 차세대 운영 인텔리전스를 실현하는 핵심적인 기반이 될 것입니다. 궁극적으로 이 아키텍처는 관측성을 사후 대응적인 분석 도구에서 사전 예방적인 대화형 인텔리전스 파트너로 변모시키며, 복잡한 운영 질문에 즉각적이고 맥락을 이해하는 답변을 자연어로 제공하는 새로운 차원의 VibeOps를 가능하게 합니다.

© 2025 MSAP.ai., All Rights Reserved.

4장. LLM 연계 아키텍처 (특허 NP25073-KR)

기존의 애플리케이션 성능 관리(APM) 및 세션 관리 시스템은 방대한 데이터를 수집하지만, 정작 운영자가 문제의 본질을 파악하고 신속하게 조치하기 위해서는 복잡한 대시보드를 탐색하고 정형화된 쿼리를 학습해야 하는 높은 진입 장벽이 존재했습니다. 이러한 한계는 장애 대응 시간을 지연시키고, 잠재적 문제점을 사전에 분석하는 데 큰 제약으로 작용해 왔습니다. 본 장에서는 자연어 인터페이스, 즉 대규모 언어 모델(LLM)을 APM 및 세션 관리 시스템에 통합하는 아키텍처를 분석합니다. 이 접근법은 단순히 사용자 편의성을 높이는 것을 넘어, IT 운영의 패러다임을 바꾸는 혁신적인 전략입니다. IT 의사결정자는 이 아키텍처를 통해 복잡한 시스템 상태를 직관적인 대화로 파악하고, 나아가 시스템 제어까지 수행함으로써 운영 효율성과 문제 해결 능력을 극대화하는 실질적인 가치를 얻을 수 있습니다. 이어질 내용에서는 이 아키텍처를 구성하는 핵심 요소, 데이터 흐름, 그리고 안정적인 운영을 보장하는 프롬프트 설계 원칙을 심도 있게 살펴보겠습니다.

4.1. 핵심 구성요소 분석

LLM 연계 아키텍처의 진정한 가치를 이해하기 위해서는 각 기술 컴포넌트의 명확한 역할과 유기적인 상호작용을 파악하는 것이 중요합니다. 이 시스템은 AI API 서버, LLM 서버, 세션서버, APM서버라는 네 가지 핵심 요소로 구성됩니다. 각 요소는 독립적으로 고유의 기능을 수행하면서도, 사용자의 자연어 질의 하나를 처리하기 위해 마치 잘 짜인 오케스트라처럼 협력합니다. 이러한 유기적 결합을 통해 복잡한 데이터 조회는 물론, 시스템 제어까지 자연어로 수행하는 혁신적인 사용자 경험을 제공합니다.

4.1.1. AI API 서버: 지능형 요청 라우터

AI API 서버는 이 아키텍처의 중앙 관제탑 역할을 수행합니다. 이는 단순한 API 게이트웨이를 넘어, 사용자의 자연어 입력을 분석하고 그 의도를 파악하여 가장 적절한 내부 시스템으로 요청을 분배하는 지능형 오케스트레이터(Orchestrator)입니다.

사용자가 “현재 접속 중인 사용자 세션을 보여줘”와 같은 질의를 입력하면, AI API 서버는 ’세션’이라는 키워드를 분석하여 ’세션툴(Session tool)’을 호출합니다. 반면, “어제 오후 3시부터 4시 사이의 주문 처리 트랜잭션 이력을 요약해줘”와 같은 질의는 ’과거 이력’과 ’트랜잭션’이라는 맥

© 2025 MSAP.ai., All Rights Reserved.

락을 파악하여 'APM툴(APM tool)'을 호출하도록 결정합니다. 이처럼 AI API 서버는 사용자의 추상적인 자연어 질의를 시스템이 이해하고 실행할 수 있는 구체적인 데이터 요청(세션서버 호출 또는 APM서버 호출)으로 변환하는 핵심적인 첫 단계를 책임집니다. 이러한 지능적 라우팅은 실시간 요청을 위해 거대한 APM 영구 저장소를 불필요하게 조회하거나, 반대로 과거 데이터 분석을 위해 휘발성 세션서버에 부하를 주는 일을 방지하여 시스템 자원을 효율적으로 사용하는 핵심 전략입니다.

4.1.2. LLM 서버: 데이터 기반 응답 생성 엔진

LLM 서버는 아키텍처의 '두뇌'이자 최종 응답을 생성하는 핵심 엔진입니다. AI API 서버가 사용자의 질의를 해석하고 필요한 데이터를 요청하면, 세션서버 또는 APM서버로부터 해당 데이터가 LLM 서버로 전달됩니다. LLM 서버는 이 구조화된 데이터를 입력된 최초의 자연어 질의(프롬프트)와 함께 처리하여, 최종 사용자가 쉽게 이해할 수 있는 자연어 형태의 분석 보고서나 답변을 생성합니다.

이 서버의 역할은 단순히 데이터를 문장으로 바꾸는 것에 그치지 않습니다. 예를 들어, APM서버로부터 받은 응답 시간, CPU 사용률, 에러 발생 횟수 등의 원시 데이터를 종합하여 "해당 시간대에 특정 URL에서 응답 시간이 급증했으며, 이는 CPU 사용률 증가와 동반되었습니다. 주된 원인으로 추정되는 부분은…"과 같이 통찰력이 담긴 분석 결과를 제공합니다. 즉, 데이터를 인간의 언어로 번역하고, 그 안에 숨겨진 의미를 해석하여 가치 있는 정보로 가공하는 역할을 수행합니다.

4.1.3. 세션서버 및 APM서버: 실시간과 과거 데이터의 원천

이 아키텍처의 데이터는 목적에 따라 명확히 분리된 두 개의 원천, 즉 세션서버와 APM서버에서 제공됩니다. 두 서버의 역할 차이는 실시간성과 영속성이라는 핵심 키워드로 요약할 수 있습니다.

구분	세션서버 (Session Server)	APM서버 (APM Server)
주요 데이터	현재 진행 중인 (Real-time) 세션 데이터	과거 트랜잭션 이력 (Historical) 데이터
관리 대상	세션 ID, 사용자 식별인자 등 활성 세션 정보	URL 호출 이력, 응답 시간, CPU 사용률, 에러 정보 등

© 2025 MSAP.ai., All Rights Reserved.

구분	세션서버 (Session Server)	APM서버 (APM Server)
저장 방식	휘발성 메모리 (In-Memory Data Grid)	영구 저장소 (APM 트랜잭션 데이터베이스)
데이터 생명주기	세션이 종료되면 데이터 소멸	영구적으로 보관 및 관리
핵심 역할	실시간 사용자 상태 추적 및 관리	과거 성능 데이터 분석 및 장애 원인 규명

세션서버는 '지금 이 순간'의 시스템 상태 파악에, APM서버는 과거 데이터 기반의 추세 분석 및 근본 원인 규명에 최적화되어 있습니다. 특히 세션서버의 데이터 저장 방식으로 인메모리 데이터 그리드(IMDG)를 채택한 것은 중요한 아키텍처적 결정입니다. IMDG는 P2P(Peer-to-Peer) 아키텍처를 통해 모든 노드가 동등하게 데이터를 분산 및 복제하므로, 노드 추가만으로 성능과 용량이 선형적으로 확장되고(Linear Scalability) 장애 발생 시에도 데이터 유실 없이 강력한 일관성(Strong Consistency)을 보장합니다. 이는 마스터-슬레이브 구조를 가진 일반적인 클라이언트-서버 방식 캐시(예: Redis)가 장애 전환(Failover) 시 데이터 유실 위험이 있거나 마스터 노드가 병목이 될 수 있는 한계를 극복하는 핵심적인 차별점입니다. 이처럼 견고한 실시간 데이터 소스와 안정적인 과거 데이터 소스의 분리는 이 아키텍처가 엔터프라이즈급 신뢰성을 확보하는 기반이 됩니다.

4.2. 세션과 LLM 을 이용한 주요 적용 시나리오

LLM 연계 아키텍처의 진정한 효율성과 유연성은 사용자의 질의 목적에 따라 시스템 내부의 데이터 흐름이 동적으로 변화하는 방식에 있습니다. 모든 요청이 동일한 경로를 따르는 것이 아니라, '실시간 조회', '과거 분석', '시스템 제어' 등 각기 다른 시나리오에 최적화된 경로를 통해 처리됩니다. 이러한 목적별 데이터 플로우를 이해하는 것은 시스템이 어떻게 지능적으로 요청을 처리하고 자원을 효율적으로 사용하는지를 파악하는 데 핵심입니다.

© 2025 MSAP.ai., All Rights Reserved.

4.2.1. “현재 진행 중인 세션” 조회 경로

사용자가 실시간 시스템 현황, 예를 들어 “현재 활성 세션은 총 몇 개이며, 특정 사용자가 접속 중인지 확인해줘”와 같은 질의를 할 때 데이터는 다음과 같은 경로로 흐릅니다.

1. 자연어 질의: 사용자가 AI API 서버에 “현재 진행 중인 세션” 관련 질의를 입력합니다.
2. 요청 라우팅: AI API 서버는 질의의 핵심이 ’현재’와 ’세션’에 있음을 파악하고, 세션툴을 호출합니다.
3. 데이터 조회: 세션툴은 세션서버에 접속하여 현재 활성화된 세션 데이터를 조회합니다. 이 데이터는 세션서버의 휘발성 메모리에 저장되어 있어 매우 빠른 응답이 가능합니다.
4. 응답 생성: 조회된 실시간 세션 데이터는 AI API 서버를 거쳐 LLM 서버로 전송됩니다. LLM은 이 데이터를 기반으로 사용자 질의에 맞는 자연어 응답을 생성하여 반환합니다.

이 경로는 최신 상태의 데이터를 신속하게 확인하는 데 최적화되어 있으며, 영구 저장소에 불필요한 부하를 주지 않는 효율적인 구조입니다.

4.2.2. “과거 트랜잭션 이력” 조회 경로

사용자가 특정 기간의 성능 분석이나 장애 원인 분석을 위해 “지난 주 화요일에 발생한 주문 실패 트랜잭션 목록을 보여줘”와 같이 과거 데이터를 조회할 때의 흐름은 다릅니다.

1. 자연어 질의: 사용자가 AI API 서버에 “과거 트랜잭션 이력” 관련 질의를 입력합니다.
2. 요청 라우팅: AI API 서버는 ‘과거’, ‘트랜잭션’, ‘이력’ 등의 키워드를 분석하여 APM툴을 호출합니다.
3. 데이터 조회: APM툴은 APM 서버의 영구 트랜잭션 데이터베이스(DB)에서 사용자가 요청한 조건(기간, 특정 트랜잭션 등)에 맞는 이력 데이터를 조회합니다.
4. 응답 생성: 조회된 과거 데이터가 AI API 서버를 거쳐 LLM 서버로 전송되고, LLM이 이를 분석하고 요약하여 최종 자연어 보고서를 생성합니다.

이 경로는 영구적으로 저장된 방대한 데이터를 심층적으로 분석하고, 시간의 흐름에 따른 패턴이나 특정 시점의 문제 원인을 파악하는 데 사용됩니다.

© 2025 MSAP.ai., All Rights Reserved.

4.2.3. 자연어 명령에 따른 제어 경로

이 아키텍처는 단순 조회를 넘어, 자연어 명령을 통해 시스템을 직접 제어하는 강력한 기능을 제공합니다. 이 경우, 명령의 성격에 따라 제어 경로가 명확히 분리됩니다.

- 세션 제어: "ID 'user123'의 세션을 강제로 종료해줘"와 같은 명령은 AI API 서버에 의해'세션툴'로 라우팅됩니다. 세션툴은 세션서버에 직접 명령을 전달하여 해당 세션 데이터를 삭제하거나 변경함으로써 즉각적인 제어를 수행합니다.
- 트래픽 제어: "/admin 페이지로의 접근을 차단해줘"와 같은 명령은 AI API 서버에서 'APM툴'로 전달됩니다. APM툴은 이 명령을 APM서버를 경유하여 실제 애플리케이션이 구동되는 WAS(Web Application Server)의 에이전트로 전송하고, 에이전트가 해당 URL에 대한 접근 제어 규칙을 실시간으로 적용합니다.

이처럼 조회와 제어, 그리고 제어의 대상에 따라 데이터 플로우를 명확하게 분리함으로써 시스템의 안정성과 예측 가능성을 크게 높입니다. 이처럼 정교한 제어와 분석을 안정적으로 수행하기 위해서는, LLM이 주어진 역할 내에서 정확하게 작동하도록 만드는 잘 설계된 프롬프트가 필수적입니다.

4.3. LLM 프롬프트 설계 원칙

LLM의 강력한 언어 능력을 엔터프라이즈 시스템에 안정적으로 통합하기 위해서는 잘 설계된 프롬프트가 무엇보다 중요합니다. 프롬프트 설계는 단순히 LLM에게 질문을 던지는 행위가 아니라, LLM이 시스템의 명확한 가이드라인 안에서 일관되고 예측 가능하게 작동하도록 제어하는 핵심적인 엔지니어링 과정입니다. 부정확하거나 모호한 응답을 방지하고, 특히 시스템 제어 명령의 오작동 리스크를 최소화하기 위해 다음과 같은 설계 원칙이 반드시 필요합니다.

4.3.1. 시스템 프롬프트의 명확성: 필수 정보 명시

LLM이 사용자의 모호한 질의로부터 정확한 데이터를 조회하도록 강제하기 위해, 시스템 프롬프트(LLM에게 역할을 부여하고 제약 조건을 설정하는 사전 지침)에는 다음과 같은 핵심 파라미터가 반드시 포함되거나, 질의에서 추출되도록 유도해야 합니다.

© 2025 MSAP.ai., All Rights Reserved.

- 세션 ID (Session ID) 및 사용자 식별인자 (User Identifier): 분석 대상을 특정 개인이나 세션으로 명확히 한정하여, 불필요한 데이터 조회를 막고 분석의 정확도를 높입니다.
- URL: 분석 범위를 특정 기능이나 페이지의 트랜잭션으로 좁혀 문제의 원인을 구체적으로 파악하도록 돕습니다.
- 시간 범위 (Time Range): "어제 오전 9시부터 10시까지"와 같이 조회할 데이터의 시간적 범위를 명확히 지정하여, LLM이 방대한 데이터 속에서 헤매지 않고 필요한 정보에만 집중하도록 만듭니다.

이러한 필수 파라미터 명시는 단순히 데이터를 필터링하는 것을 넘어, LLM을 검증 가능한 특정 데이터 컨텍스트에 '그라운딩(Grounding)'시키는 핵심 전략입니다. 이는 LLM이 사실과 다른 정보를 생성하는 '환각(Hallucination)' 현상을 근본적으로 억제하고, 엔터프라이즈 환경에서 요구되는 사실 기반의 신뢰도 높은 응답을 보장하는 첫 번째 안전장치입니다.

4.3.2. 구조화된 출력 포맷: 예측 가능한 응답

LLM의 응답은 자연스럽지만 때로는 비정형적일 수 있어, 후속적인 기계 처리나 시각화에 어려움을 줄 수 있습니다. 이를 해결하기 위해 프롬프트는 LLM이 항상 일관된 구조의 결과물을 반환하도록 강제해야 합니다. 예를 들어, 다음과 같은 명확한 키-값 구조로 응답을 유도할 수 있습니다.

`"누가(사용자ID) 언제(타임스탬프) 무엇을(URL) 얼마만큼(응답시간·CPU·TPS)"`

이러한 원칙은 LLM을 단순한 대화형 도구에서 예측 가능하고 기계적으로 파싱(Parsing)할 수 있는 '자연어 API(Natural Language API)'로 변모시킵니다. 시스템은 이 구조화된 결과를 손쉽게 처리하여 자동으로 리포트를 생성하거나 대시보드에 시각화할 수 있어, 데이터 활용 가치를 극대화하고 다른 시스템과의 통합을 용이하게 합니다.

4.3.3. 제어 명령의 안전성 검증: 운영 리스크 최소화

자연어 명령이 실제 운영 시스템의 상태를 변경할 수 있다는 점은 매우 강력한 기능인 동시에 가장 큰 리스크입니다. 따라서 사용자의 의도와 다른 치명적인 오작동을 방지하기 위해 다층적인 안전장치가 필수적입니다. 자연어 명령을 운영 시스템에 안전하게 적용하기 위해서는 특허 아키텍처를 기반으로 다음과 같은 강력한 안전장치를 구현하는 것이 필수적입니다.

© 2025 MSAP.ai., All Rights Reserved.

- 허용 명령 화이트리스트: 시스템은 사전에 정의된 안전한 명령어 집합(예: '조회', '강제종료', '차단')만을 인식하고 실행합니다. 만약 사용자가 "서버를 재시작해"와 같이 허용 목록에 없는 위험한 명령을 입력하면, 시스템은 이를 실행하지 않고 거부하거나 경고 메시지를 반환하여 잠재적 사고를 원천적으로 차단합니다.
- URL 패턴 검증: 트래픽 제어와 같은 명령이 적용될 대상 URL이 의도치 않게 시스템 전체에 영향을 미치는 것을 방지합니다. 예를 들어, 와 같이 지나치게 광범위한 패턴의 URL 차단 명령이 입력되면, 시스템은 이를 위험한 명령으로 간주하고 실행 전에 관리자의 확인을 요구하거나 더 구체적인 패턴을 입력하도록 유도합니다.

이처럼 체계적인 아키텍처와 정교한 프롬프트 설계 원칙의 결합은 LLM 기술을 단순한 대화형 AI를 넘어, 복잡한 엔터프라이즈 환경을 안정적으로 분석하고 제어할 수 있는 신뢰할 수 있는 운영 도구로 격상시킵니다.

5장. LLM을 활용한 지능형 APM: 동시 접속자 및 트랜잭션 패턴 분석 자동화

5.1 동시접속자수 파악 로직

서론: 활성 사용자 식별의 전략적 중요성

시스템의 부하를 이해하는 첫걸음은 동시 접속자 수를 파악하는 것입니다. 그러나 단순히 시스템에 연결된 세션의 수를 세는 것만으로는 충분하지 않습니다. 서비스 안정성을 확보하고 미래의 용량을 정확하게 계획하기 위해서는, 특정 시간대에 실제로 활발하게 활동하는 '활성 사용자'를 식별하는 것이 필수적입니다. 본 섹션에서는 `OPENMARU APM`이 수집하는 애플리케이션 성능 데이터와 `OPENMARU Cluster`가 실시간으로 관리하는 세션 데이터를 결합하여, LLM(거대 언어 모델)이 동시 접속 현황을 다각적으로 분석하고 그 의미를 해석하는 새로운 접근법을 제시하고자 합니다.

© 2025 MSAP.ai., All Rights Reserved.

5.1.1 시간창(Sliding window)별 활성 세션 카운트

정확한 활성 사용자 수를 파악하기 위해 '슬라이딩 윈도우(Sliding Window)' 개념을 도입합니다. 이는 "지금 이 순간"이 아닌, "최근 5분"과 같이 지정된 시간 구간 동안 활동한 고유 세션의 수를 계산하는 방식입니다. 이를 통해 일시적인 접속 폭증과 실제 지속적인 부하를 구분할 수 있습니다.

이러한 실시간 집계의 신뢰성은 **OPENMARU Cluster**의 아키텍처적 선택에 기인합니다. In-Memory Data Grid(IMDG) 기반 아키텍처는 데이터의 강한 일관성(Strong Consistency)과 P2P 구조를 통한 선형적 확장성을 보장합니다. 이는 일반적인 Redis 기반 세션 스토어의 최종 일관성(Eventual Consistency) 모델과 근본적인 차이를 만듭니다. Redis의 경우, 장애 조치(Failover) 과정에서 발생할 수 있는 데이터 유실이나 비동기 복제로 인한 데이터 불일치는 LLM이 부정확하거나 오래된 세션 수를 분석하게 만들어 지능형 APM의 신뢰도를 저해할 수 있습니다. 반면, IMDG는 LLM 분석에 필요한 데이터의 정확성과 실시간성을 아키텍처 수준에서 보장합니다.

5.1.2 APM의 "장애 시간 동시 사용자수"와의 비교·보정 로직

운영 환경에서는 두 가지 종류의 '동시 사용자' 데이터가 존재합니다.

1. APM의 '장애 발생 시점 동시 사용자 수': **OPENMARU APM**이 특정 트랜잭션에서 임계값 초과나 에러가 발생한 순간에 기록하는 성능 메트릭입니다. 이는 문제의 심각성을 판단하는 중요한 지표입니다.
2. 세션 서버의 '실시간 활성 세션 수': **OPENMARU Cluster**가 슬라이딩 윈도우 방식으로 집계하는 현재 활동 중인 전체 세션 수입니다.

LLM은 이 두 데이터를 비교하고 보정하여 장애 상황에 대한 훨씬 깊이 있는 컨텍스트를 제공합니다. 예를 들어, 운영자는 다음과 같은 분석 결과를 얻을 수 있습니다.

LLM 분석 보고 예시: "APM 상 장애 발생 시점의 동시 사용자는 500명으로 기록되었지만, 실제 활성 세션은 1,200개였습니다. APM 메트릭과 세션 데이터의 상관관계 분석 결과, 이 차이는 특정 이벤트 페이지(**/event/promo**) URL에 사용자가 집중적으로 몰리면서 발생한 것으로 확인됩니다. 해당 URL의 부하가 전체 시스템에 영향을 미쳤을 가능성이 높습니다."

이처럼 LLM은 두 지표의 차이를 단순한 숫자가 아닌, '어디에 사용자가 집중되었는가'라는 구체적인 현상으로 해석하여 문제의 근본 원인에 더 가깝게 다가갈 수 있도록 돕습니다.

© 2025 MSAP.ai., All Rights Reserved.

5.1.3 LLM이 “이 구간이 피크였으니 원인을 보여줘”라는 질의를 이해하도록 하는 예시 질의

자연어 질의는 지능형 APM의 핵심 기능입니다. 운영자는 복잡한 쿼리나 대시보드 조작 없이, 대화하듯 시스템에 질문을 던질 수 있습니다.

질의 예시:

> “오늘 오전 9시에서 10시 사이 사용자 접속이 가장 많았던 피크 시간대를 찾고, 해당 시간대에 발생한 문제의 원인을 분석해줘.”

이 질의가 입력되면, 시스템은 다음과 같은 단계로 처리합니다.

1. 질의 이해 및 데이터 소스 식별: AI API 서버가 자연어 질의를 받아, ’피크 시간대’와 ’원인 분석’이라는 핵심 의도를 파악합니다. 이를 위해 APM 트랜잭션 데이터베이스와 세션 서버 데이터가 필요함을 인지합니다.
2. 피크 구간 특정: AI API 서버는 데이터 융합 프로세스를 지휘합니다. APM툴을 통해 APM 트랜잭션 데이터베이스를 자율적으로 쿼리하여 오전 9시부터 10시 사이의 동시 사용자 수와 활성 세션 수 추이를 분 단위로 분석하고, 가장 높은 수치를 기록한 특정 시간 구간(예: 9시 32분 ~ 9시 37분)을 피크로 정의합니다.
3. 연관 데이터 수집 및 컨텍스트 생성: 시스템은 특정된 피크 구간 동안 발생한 다른 이상 지표들을 자동으로 수집합니다. 예를 들어, 특정 URL의 응답 시간 급증, 에러 발생 트랜잭션 목록, DB 커넥션 풀 고갈, 특정 WAS 노드의 CPU 사용량 급증과 같은 데이터를 연계합니다.
4. LLM 기반 원인 분석 보고: 수집된 모든 연관 데이터를 풍부한 컨텍스트로 구성하여 LLM에 전달합니다. LLM은 이 정보를 바탕으로 “피크 시간대는 9시 32분경이었으며, 주된 원인은 `/api/search` URL에서 발생한 데이터베이스 슬로우 쿼리로 인해 전반적인 응답 시간이 지연되었기 때문입니다.” 와 같은 인과관계를 설명합니다.

전체 사용자 수라는 거시적 지표를 통해 시스템의 전반적인 부하 상태를 파악했다면, 이제는 서비스 품질에 직접적인 영향을 미치는 개별 URL 단위의 미시적 분석으로 초점을 전환해야 합니다. 특정 기능의 성능 저하가 전체 사용자 경험을 어떻게 훼손하는지, 그리고 LLM이 이를 어떻게 정밀하게 진단하고 제어하는지 다음 섹션에서 자세히 살펴보겠습니다.

© 2025 MSAP.ai., All Rights Reserved.

5.2 URL 단위 분석

서론: 마이크로서비스 환경에서의 URL 분석의 중요성

현대의 마이크로서비스 아키텍처(MSA) 환경에서 시스템은 수많은 독립적인 기능(API 엔드포인트, 즉 URL)의 집합으로 구성됩니다. 이 환경에서는 전체 시스템의 평균 응답 시간은 정상이지만, 결제나 검색과 같은 특정 핵심 기능(URL) 하나의 성능 저하가 전체 서비스 경험을 치명적으로 저하시킬 수 있습니다. 따라서 URL 단위의 성능을 정밀하게 분석하는 것이 무엇보다 중요합니다. 본 섹션에서는 LLM이 어떻게 세션, URL, 사용자 정보를 입체적으로 연결하여 문제를 정확히 진단하고, 나아가 능동적인 제어까지 제안하는지 설명합니다.

5.2.1 세션ID→URL→사용자 식별자 3단 연계 테이블 구조

문제의 근본 원인을 파악하기 위해서는 단편적인 데이터를 넘어, 각 데이터를 유기적으로 연결해야 합니다. 본 시스템의 핵심 아키텍처는 다음의 세 가지 핵심 식별자를 연계하여 영구적으로 저장하는 데이터 통합 흐름에 있습니다.

1. 사용자 식별자 (User Identifier): 세션 생성 시 웹 서버에서 확보되어 **OPENMARU Cluster** 세션 서버로 전달되는 실제 사용자 정보(예: 로그인 ID, 고객 번호)입니다.
2. 세션 ID (Session ID): **OPENMARU Cluster**가 관리하는 실시간 연결 식별자입니다. 사용자 식별자와 매핑되어 관리됩니다.
3. URL (Transaction): **OPENMARU APM** 에이전트가 WAS 단에서 계측하는 개별 HTTP 요청(트랜잭션) 정보입니다.

특히 명세서에 기술된 바와 같이, **APM 서버**는 이러한 데이터들을 통합하는 중앙 허브 역할을 수행합니다. 즉, APM 에이전트로부터 트랜잭션 정보를 수집하는 동시에, **OPENMARU Cluster**로부터 세션 ID와 사용자 식별자 정보를 수집하여 이 세 가지 데이터를 상호 연관시켜 **APM 트랜잭션 데이터베이스**에 영구적으로 저장합니다. 이 영구적인 APM 트랜잭션 데이터베이스는 특허 명세서에서 설명하듯, 실시간으로만 존재하는 세션 서버의 휘발성 메모리와 명확히 대비됩니다.

이처럼 '어떤 사용자(사용자 식별자)가', '언제(타임스탬프)', '어떤 기능(URL)을 사용했을 때', '어떤 경험(응답 시간, 에러 여부)을 했는지'에 대한 기록이 세션 ID를 매개로 모두 연결되어 저

© 2025 MSAP.ai., All Rights Reserved.

장됩니다. 이처럼 구조화된 영구적 연결고리는 LLM이 분석을 수행할 때 발생할 수 있는 '환각(Hallucination)' 현상을 방지하는 사실적 기반, 즉 '그라운드 트루스(Ground Truth)' 역할을 합니다. 이 구조는 특정 사용자의 과거 행적을 시간의 흐름에 따라 완벽하게 추적하고 분석할 수 있는 기반이 됩니다.

5.2.2 URL별 응답시간·에러율·접근 IP·브라우저 패턴을 한 번에 뽑는 자연어 프롬프트 세트

3단 연계 데이터 구조는 운영자가 매우 강력하고 구체적인 질문을 자연어로 던질 수 있게 합니다.

프롬프트 예시:

> "어제 /api/v1/payment URL의 시간대별 평균 응답시간과 에러율을 보여주고, 가장 많은 에러를 발생시킨 IP 주소 상위 5개와 브라우저 유형을 알려줘."

LLM은 이 프롬프트를 처리하기 위해 5.2.1에서 설명한 3단 연계 데이터를 다음과 같이 활용합니다.

- 데이터 조회: AI API 서버는 APM툴을 통해 APM 트랜잭션 데이터베이스에서 `/api/v1/payment` URL을 가진 모든 트랜잭션을 조회합니다.
- 지표 종합: 조회된 트랜잭션 데이터를 기반으로 시간대별 평균 응답 시간과 에러율을 계산합니다. 동시에 에러가 발생한 트랜잭션에 한정하여, 연계된 세션 정보로부터 접근 IP 주소와 브라우저 유형(User-Agent)을 추출하고 빈도를 집계합니다.
- 구조화된 결과 생성: LLM은 추출 및 분석된 모든 정보를 종합하여, 요청된 모든 지표(응답 시간, 에러율, IP, 브라우저)를 포함하는 구조화된 Markdown 테이블 형식의 보고서를 생성합니다. 이를 통해 운영자는 단 한 번의 질의로 다각적인 분석을 완료할 수 있습니다.

5.2.3 특정 URL에 대한 트래픽 제어(차단·리다이렉트·세션 강제 종료) 제안까지의 LLM 체인

본 기능은 APM의 패러다임을 수동적인 관찰 도구에서 능동적인 지능형 방어 및 제어 시스템으로 전환하는 핵심적인 진화입니다. 이는 단순 모니터링을 넘어 AI 주도형 자동 복구(AI-driven

© 2025 MSAP.ai., All Rights Reserved.

Remediation)로 나아가는, 현대 IT 운영의 필연적인 단계입니다. 이 과정은 다음과 같은 단계로 이루어집니다.

1. 분석 및 제안 (Analyze & Propose): LLM이 APM 데이터를 실시간으로 분석하던 중, 특정 URL(`/api/v1/download`)에 비정상적인 트래픽 패턴(예: 특정 IP 대역에서의 초당 수백 회 요청)을 감지합니다. 이는 분산 서비스 거부(DDoS) 공격의 징후일 수 있습니다. LLM은 즉시 다음과 같은 해결책을 자연어로 제안합니다.
2. 명령 변환 (Translate & Formalize): 운영자가 '동의' 의사를 밝히면, AI API 서버는 이 자연어 제안을 WAS(Web Application Server) 에이전트가 이해할 수 있는 공식적인 트래픽 제어 명령으로 변환합니다. 이 명령은 보통 JSON 형식의 차단 규칙으로 정의됩니다.
3. 실행 (Execute): 변환된 명령은 APM 서버를 통해 실시간으로 해당 웹사이트의 모든 WAS 에이전트로 전송됩니다. 에이전트는 이 규칙을 즉시 메모리에 로드하여, 해당 조건에 맞는 모든 요청을 차단하기 시작합니다. 이와 동일한 방식으로 특정 사용자의 세션을 강제로 종료하거나, 특정 페이지 요청을 점검 페이지로 리다이렉트하는 조치도 실시간으로 실행될 수 있습니다.

개별 URL에 대한 심층 분석과 실시간 제어는 운영 효율성을 극대화합니다. 그러나 여기서 한 걸음 더 나아가, 시스템이 스스로 이상 징후를 감지하고 운영자의 개입 없이 근본 원인 후보까지 제시하는 완전 자동화 단계로 나아가는 것이 지능형 운영의 최종 목표입니다. 다음 섹션에서는 이러한 자동화된 분석이 어떻게 구현되는지 살펴보겠습니다.

5.3 이상 징후와 루트코즈(Root Cause) 후보 자동화

서론: 단순 경고(Alert)를 넘어 설명 가능한 인사이트(Explainable Insight)로

전통적인 모니터링 시스템은 사전에 정의된 임계값을 넘으면 경고(Alert)를 발생시키는 데 그칩니다. "응답 시간 2초 초과"와 같은 경고는 현상을 알려줄 뿐, '왜' 그런 현상이 발생했는지에 대한 답을 주지 못합니다. 이로 인해 운영자는 여러 대시보드를 오가며 수동으로 원인을 추적해야 하는 부담을 안게 됩니다. 본 섹션의 목표는 APM이 탐지한 원시적인 이벤트를 LLM이 어떻게 인간이 이해할 수 있는 '자동 원인 분석 보고서'로 변환하여, 운영자의 문제 해결 시간을 획기적으로 단축시키는지를 보여주는 것입니다.

© 2025 MSAP.ai., All Rights Reserved.

5.3.1 APM이 던지는 이벤트를 LLM이 설명형(왜 이 URL이 느려졌는가)으로 바꿔주는 패턴

LLM은 단편적인 이벤트를 원인·맥락·조치로 연결된 “설명 가능한 인사이트”로 변환합니다.

APM이 던지는 이벤트를 LLM이 설명형(왜 이 URL이 느려졌는가)으로 바꿔주는 패턴은 오픈마루 플랫폼 제품 중 지능형 관리 기능인 CogentAI의 핵심 요소입니다.

LLM의 가장 강력한 능력 중 하나는 단편적인 데이터들을 연결하여 하나의 완성된 서사(Narrative)로 재구성하는 것입니다. 이는 IT 운영 데이터를 단순 수치가 아닌 인간의 언어로 해석하고, 인과관계를 추론하여 문제 해결 시간을 획기적으로 단축시키는 VibeOps 모델의 핵심입니다.

다음은 APM 이벤트가 LLM을 통해 운영자가 이해하기 쉬운 설명형 리포트로 어떻게 변환되는지를 보여주는 ‘Before’와 ’After’ 예시입니다.

- Before (APM 원시 이벤트):

  트랜잭션: `/api/checkout` (POST), 응답 시간: 27.58ms (HTTP-500 상태). 10:48:49.967 시점 WAS 인스턴스 D에서 DB 쿼리 응답 시간 3,792ms. 동시 접속자 수 5,200명 (09:00-10:00 피크 구간). DB 쿼리: `SELECT NTT ID...` (MySQL PrepareStatement.execute()).

  - 이 정보만으로는 운영자는 무엇을 해야 할지 막막합니다.

- After (LLM 자동 분석 리포트):

  “오늘 09:45 경, 동시접속자수 피크(5,200명)가 발생하면서, 사용자 A00123 외 100여 명의 /api/checkout 트랜잭션이 영향을 받았습니다. 주요 원인은 데이터베이스 연결 시간 초과(3.792초 지연)로 인한 WAS CPU 부하 증가였으며, 이는 피크 부하를 처리하기 위한 DB 커넥션 풀 부족 또는 느린 SELECT NTT ID… 쿼리 때문인 것으로 보입니다. 이 시점 직전에 배포된 새로운 버전의 애플리케이션에 대한 롤백 또는 데이터베이스 쿼리 최적화를 권장합니다.”.

© 2025 MSAP.ai., All Rights Reserved.

- LLM은 APM 트랜잭션 데이터베이스에 영구적으로 저장된 트랜잭션 정보(URL, 응답 시간)와 세션 서버에서 수집된 세션 ID 및 사용자 식별인자 정보를 수집하여 이를 연계 분석합니다. LLM은 이 데이터를 기반으로 근본 원인(Root Cause) 진단을 수행합니다.

5.3.2 LLM Observability(OpenLLMetry, Elastic APM의 LLM 트레이스)와의 정합성 확보

LLM 기반의 자동화 시스템을 도입할 때 가장 중요한 과제는 '신뢰성'입니다. LLM이 생성한 분석 결과를 무조건 신뢰할 수는 없으며, 특히 사실이 아닌 정보를 그럴듯하게 만들어내는 '환각(Hallucination)' 현상은 운영에 치명적일 수 있습니다. 따라서 LLM의 판단 근거를 추적하고 검증할 수 있는 'LLM 관찰 가능성(Observability)' 확보가 필수적입니다.

OpenLLMetry와 같은 도구를 활용하면 LLM의 내부 동작을 추적할 수 있습니다.

- LLM이 분석 보고서를 만들기 위해 어떤 APM 데이터를 요청했는가?
- 데이터를 바탕으로 어떤 프롬프트를 구성했는가?
- 최종적으로 왜 그런 결론(예: 'DB 풀 스캔이 원인이다')을 내렸는가?

이러한 전 과정을 추적함으로써, 시스템은 LLM이 내놓은 결과가 실제 데이터에 기반한 것인지, 아니면 논리적 비약이나 환각에 의한 것인지를 검증할 수 있습니다. 이는 LLM 분석 결과의 정합성을 확보하고, 운영자가 최종 판단을 내리는 데 중요한 근거를 제공합니다.

5.3.3 OpenTelemetry 스팬과 세션/URL 스팬을 한 질의에서 보게 하는 콘텍스트 결합

현대적인 분산 시스템 환경에서는 하나의 사용자 요청이 여러 마이크로서비스를 거쳐 처리됩니다. 이러한 분산 호출 흐름을 추적하는 표준 기술이 바로 OpenTelemetry이며, 각 단계를 '스팬(Span)'이라는 단위로 기록합니다.

본 시스템은 OpenTelemetry의 '스팬' 데이터와 독자적으로 수집한 '세션/URL' 데이터를 결합하여 전례 없는 수준의 통합 분석을 가능하게 합니다.

© 2025 MSAP.ai., All Rights Reserved.

시나리오 예시: 한 사용자가 모바일 앱에서 '주문하기' 버튼을 누릅니다. 이 요청은 다음과 같은 서비스들을 순차적으로 호출합니다. **주문 서비스 → 결제 서비스 → 재고 서비스**

- OpenTelemetry 트레이스: 이 세 서비스 간의 호출 흐름, 각 서비스에서의 소요 시간, 실패 지점 등을 추적합니다.
- 세션/URL 데이터: 이 요청을 시작한 최초의 '사용자 ID'와 '세션 ID', 그리고 모바일 앱의 종류와 버전 정보를 기록합니다.

이 두 데이터 소스를 결합하면, 운영자는 다음과 같은 고차원적인 통합 질의를 할 수 있습니다.

"사용자 'user-123'의 주문 요청이 '재고 서비스'에서 지연된 전체 과정을 보여줘."

시스템은 이 질의에 대해 사용자 세션 정보부터 시작하여, 주문, 결제, 재고 서비스로 이어지는 전체 분산 트레이스(스팬)를 하나의 타임라인으로 시각화하여 보여줍니다. 이를 통해 문제의 범위가 특정 사용자인지, 특정 서비스의 문제인지, 아니면 두 서비스 간의 네트워크 문제인지를 명확하게 진단할 수 있습니다.

결론 및 요약

본 장에서는 LLM을 활용한 지능형 APM이 어떻게 IT 운영을 혁신하는지 단계별로 살펴보았습니다. 동시 접속자 수라는 거시적 분석에서 시작하여, 개별 URL 단위의 정밀한 분석 및 실시간 제어를 거쳐, 최종적으로는 이상 징후에 대한 근본 원인을 자동으로 분석하고 제안하는 단계에 이르기까지의 과정을 다루었습니다.

OPENMARU APM이 수집하는 깊이 있는 성능 데이터와 **OPENMARU Cluster**가 관리하는 실시간 세션 데이터, 그리고 이들을 엮어주는 LLM의 추론 능력이 결합될 때, IT 운영은 더 이상 장애 발생 후 대응하는 데이터 중심의 '대응(Reaction)' 패러다임에서 벗어날 수 있습니다. 대신, 문제를 사전에 예측하고 근본 원인을 신속하게 해결하며, 나아가 시스템이 스스로 최적화하는 AI 기반의 '예측 및 자동화(Prediction & Automation)' 패러다임으로 전환됩니다. 이는 복잡한 클라우드 네이티브 환경에서 안정적인 서비스를 제공하기 위한 필수적인 진화 방향이며, 진정한 의미의 AIOps, VibeOps 및 선제적 SRE(Site Reliability Engineering)를 구현하는 핵심 기술 기반이 될 것입니다.

© 2025 MSAP.ai., All Rights Reserved.

6장. 자연어 질의/명령 시나리오: LLM을 통한 IT 운영의 혁신

6.1. 조회 시나리오: 대화형 인터페이스로 시스템의 맥락을 파악하다

전통적인 IT 운영 환경에서 시스템의 상태를 파악하는 주된 수단은 대시보드 기반의 모니터링이었습니다. 이는 정형화된 지표를 시각적으로 제공하는 데는 효과적이었으나, 운영자가 특정 문제의 근본 원인을 파악하거나 여러 지표 간의 상관관계를 분석하기 위해서는 복잡한 쿼리 언어를 학습하거나 여러 화면을 번갈아 가며 수동으로 데이터를 조합해야 하는 어려움이 있었습니다.

LLM(대규모 언어 모델) 기반의 자연어 조회 기능은 이러한 패러다임을 근본적으로 혁신합니다. 이제 운영자는 복잡한 쿼리 작성 능력 없이도, 마치 전문가와 대화하듯 시스템에 질문을 던지는 것만으로 즉각적인 통찰력을 얻을 수 있습니다. 이 새로운 접근 방식은 시스템의 깊은 맥락을 즉시 파악하게 하여 장애 대응 시간을 단축시키고, 나아가 IT 의사결정자가 기술적 장벽 없이 데이터에 직접 접근하여 비즈니스와 기술의 연결고리를 직관적으로 이해할 수 있게 만듭니다.

본 장에서 설명하는 모든 시나리오는 오픈마루 애플리케이션 플랫폼 제품들의 핵심 구성요소인 세션 클러스터링(Session Server)과 APM(APM Server)이 AI API 서버와 유기적으로 통합되어 어떻게 단순 모니터링을 넘어 지능형 운영을 실현하는지 보여줍니다. 실시간 활성 세션 데이터와 영구 저장된 트랜잭션 데이터를 결합하여 시스템의 맥락을 완벽하게 재구성하는 아키텍처에 기반합니다.

6.1.1. “오늘 09:00~10:00에 접속한 사용자 중 응답 3초 넘은 세션만 보여줘”

시나리오 분석: 지연 원인과 영향 사용자를 즉시 특정하다

이 질의는 단순한 요청을 넘어, 시스템 운영의 패러다임을 바꾸는 강력한 예시입니다. 과거에는 여러 필터를 수동으로 조합해야 했던 작업을 단 하나의 문장으로 완료하여, 서비스 품질 저하의 원인과 영향을 받는 사용자를 즉시 식별할 수 있습니다.

1. 자연어 해석 과정 사용자가 이 질문을 입력하면, AI API 서버를 통해 LLM은 문장을 구조적으로 분석합니다. “오늘 09:00~10:00”, “응답 3초 넘은”, “세션”과 같은 핵심 어구들을 인

© 2025 MSAP.ai., All Rights Reserved.

식하여 다음과 같은 구조화된 쿼리 파라미터로 변환합니다.

- **시간 범위**: 오늘 09:00:00 ~ 10:00:00
- **응답 시간 임계값**: 〉3000ms
- **조회 대상**: 세션 정보

2. 데이터 흐름 설명 변환된 쿼리는 과거 트랜잭션 데이터에 대한 조회이므로, AI API 서버는 이 요청을 APM툴로 라우팅합니다. **NP25073-KR_명세서초안**에 명시된 아키텍처에 따라, APM툴은 APM 서버를 호출합니다. APM 서버는 영구적으로 저장된 트랜잭션 로그가 담긴 'APM 트랜젝션 데이터베이스'에서 지정된 시간 범위와 응답 시간 조건을 만족하는 모든 트랜잭션 및 관련 세션 정보를 추출하여 LLM 서버로 전달합니다. 세션 서버의 세션 메모리는 현재 활성 세션만 관리하므로, 과거 데이터 조회는 반드시 APM 서버의 영구 데이터베이스를 통합니다.
3. 결과 및 가치 평가 LLM은 조회된 데이터를 바탕으로 "해당 시간대에 응답 지연을 겪은 사용자 목록과 세션 ID"를 명확하게 정리하여 보여줍니다. 이 기능의 비즈니스 가치는 명확합니다.

    - 선제적 고객 지원: 특정 시간대에 서비스 품질 저하를 경험한 사용자를 즉시 식별하여, 불만이 접수되기 전에 먼저 연락하여 문제를 해결하는 선제적 대응이 가능해집니다.
    - 신속한 문제 해결: 장애 발생 시, 영향을 받은 정확한 사용자 그룹을 특정함으로써 문제의 범위와 심각도를 빠르게 파악하고 해결의 우선순위를 정할 수 있습니다.

6.1.2. "특정 사용자ID가 오늘 어떤 URL을 몇 초씩 호출했는지 표로 만들어줘"

시나리오 분석: 개별 사용자 경험을 현미경처럼 들여다보다

고객 지원팀이나 개발팀이 특정 사용자의 문제(예: "결제가 안 돼요", "페이지가 안 떠요")를 재현하고 분석하는 것은 매우 어려운 작업입니다. 이 시나리오는 특정 사용자의 디지털 여정 전체를 재구성하여 문제 해결 시간을 획기적으로 단축시킵니다.

1. 사용자 여정 재구성 이 질의의 핵심은 '사용자ID'입니다. LLM은 이 키워드를 식별하고, AI API 서버는 APM툴을 통해 APM 서버의 트랜잭션 데이터베이스에 특정 사용자 식별자와

© 2025 MSAP.ai., All Rights Reserved.

연관된 모든 기록을 요청합니다. 시스템은 해당 사용자가 오늘 하루 동안 호출한 모든 URL, 각 URL의 응답 시간, 호출 시각 등 디지털 발자취를 시간 순서대로 재구성합니다.

2. 데이터 상관관계 분석 이 기능은 **NP25073-KR_명세서초안**에 명시된 핵심 아키텍처 덕분에 가능합니다. APM 서버는 단순히 트랜잭션만 기록하는 것이 아니라, 세션 서버로부터 수집한 '세션ID별 사용자 식별인자' 정보와 '사용자의 트랜젝션' 정보를 세션ID를 매칭 키로 하여 영구적으로 연결해 저장합니다. 따라서 'user-123'이라는 사용자 ID가 어떤 세션들을 거쳐 어떤 URL들을 호출했는지 완벽하게 추적할 수 있습니다.
3. 결과 제시 형식 LLM은 추출된 데이터를 단순히 나열하는 대신, 운영자가 가장 이해하기 쉬운 Markdown 표 형식으로 가공하여 결과를 생성합니다. 이를 통해 가독성을 극대화하고, 문제 분석에 필요한 핵심 정보를 한눈에 파악할 수 있습니다.

호출 URL	응답 시간(초)	호출 시각
/login/process	0.52	09:15:31
/main/dashboard	1.21	09:15:33
/products/view?id=123	4.15	09:16:02
/cart/add	0.88	09:16:10

이처럼 명확한 결과물은 특정 사용자의 장애를 재현하거나 원인을 분석하는 데 소요되는 시간을 분 단위로 단축시키는 강력한 효과를 가집니다.

6.1.3. "CPU 80% 넘은 시점과 동시접속자수 피크를 나란히 보여줘"

시나리오 분석: 이종 데이터를 결합하여 인과관계를 추론하다

숙련된 운영자는 시스템 리소스와 서비스 지표를 연관 지어 분석함으로써 문제의 근본 원인을 추론합니다. 이 고차원적 조회 기능은 서로 다른 두 종류의 데이터를 결합하여, 운영자에게 강력한 인과관계 추론의 단서를 제공합니다.

1. 이종 데이터 결합 이 질의는 두 개의 완전히 다른 데이터 소스를 결합해야 합니다.
    - 시스템 리소스 메트릭: 'CPU 80% 넘은 시점'의 데이터는 APM 서버에 수집된 인프라 성능 지표입니다.

© 2025 MSAP.ai., All Rights Reserved.

- 서비스 지표: '동시 접속자 수 피크' 데이터는 APM 서버의 트랜잭션 기록을 분석하거나, 실시간 활성 세션을 관리하는 세션 서버의 데이터를 집계하여 얻을 수 있습니다. AI API 서버는 이 두 가지 데이터를 각 소스에서 요청하고, LLM이 시간 축을 기준으로 두 데이터를 정렬하고 결합합니다. 이러한 이종 데이터의 결합은 MSAP COP가 APM, Session Clustering, Observability를 단일 플랫폼으로 제공하기에 가능한 고차원적 분석입니다. 분리된 도구로는 수동으로 상관관계를 추정해야 하지만, 통합 플랫폼 내에서는 자연어 질의 하나만으로 즉각적인 인과관계 추론이 가능해집니다.

2. 인과관계 추론 지원 결합된 데이터를 통해 운영자는 다음과 같은 핵심적인 질문에 대한 답을 얻을 수 있습니다.
    - "동시 접속자 수가 급증하여 CPU 부하가 높아졌는가?" (두 지표의 피크 시간이 일치하는 경우)
    - "CPU 급증이 동시 접속자 수와는 무관한, 특정 배치 작업이나 비효율적인 쿼리 때문인가?" (CPU 피크 시점에 동시 접속자 수 변화가 없는 경우) 이처럼 이 기능은 단순한 데이터 조회를 넘어, 운영자가 데이터 기반의 가설을 세우고 검증하는 과정을 지원합니다.
3. 시각적 결과 LLM은 두 데이터를 나란히 비교할 수 있는 형태, 예를 들어 시간대별 요약 테이블이나 텍스트 기반 차트 형식으로 응답을 생성할 수 있습니다.
4. 시간대별 CPU 사용률 및 동시 접속자 수 비교 | 시간 | CPU 사용률 (%) | 동시 접속자 수 | | :— | :— | :— | | 14:00 | 45% | 1,200 | | 14:05 | 85% | 5,500 (Peak) | | 14:10 | 60% | 3,100 |

이러한 직관적인 결과는 IT 의사결정자가 시스템 용량 증설, 성능 최적화, 혹은 마케팅 이벤트에 따른 부하 예측 등 전략적 판단을 내리는 데 강력한 근거를 제공합니다.

조회 기능을 통해 시스템에 대한 깊이 있는 통찰력을 얻는 방법을 살펴보았습니다. 이제 자연어를 사용하여 시스템을 직접 제어하는, 한 단계 더 나아간 시나리오들을 탐구하겠습니다.

© 2025 MSAP.ai., All Rights Reserved.

6.2. 제어 시나리오: 자연어 명령으로 즉각적인 조치를 실행하다

자연어 기반의 제어 기능은 IT 운영의 반응성을 극대화하는 전략적 도구입니다. 기존에는 운영자가 복잡한 관리 콘솔을 여러 단계에 걸쳐 조작하거나, 특정 스크립트를 찾아 실행해야만 가능했던 조치들을 이제는 단 한 문장의 명령으로 즉시 실행할 수 있습니다. 이는 운영자의 '의도'를 시스템이 직접 이해하고 실행하는 방식으로, 장애 대응 시간을 분 단위에서 초 단위로 단축시키고 치명적인 인적 오류(Human Error) 발생 가능성을 획기적으로 줄여줍니다.

본 장에서 설명하는 모든 제어 흐름은 **NP25073-KR_명세서초안**의 도 5 및 청구항 5에 명시된 아키텍처를 기반으로 합니다. AI API 서버가 자연어 명령의 의도를 파악하여, 실시간 세션 제어는 세션툴로, 트래픽 및 정책 제어는 APM툴로 라우팅하여 실행하는 구조입니다.

6.2.1. "이 사용자 세션 강제 로그아웃" → 세션툴을 통한 세션데이터 삭제/변경

시나리오 분석: 보안 위협에 대한 즉각적인 실시간 대응

어뷰징 사용자나 보안 위협이 의심되는 계정을 발견했을 때, 가장 중요한 것은 위협을 즉시 격리하는 것입니다. 이 시나리오는 운영자가 자연어 명령 한마디로 특정 사용자의 활성 세션을 즉시 무효화하여 시스템을 보호하는 강력한 보안 제어 기능입니다.

1. 명령 라우팅 AI API 서버는 "강제 로그아웃"이라는 명령의 의도가 '현재 활성 상태인 세션을 즉시 제어'하는 것임을 파악합니다. 이는 과거 데이터를 다루는 APM툴의 역할이 아니므로, AI API 서버는 이 명령을 세션툴로 정확하게 라우팅합니다. 세션툴은 실시간 세션 데이터를 직접 관리하는 세션 서버에 접속하여 명령을 전달합니다.
2. 실행 메커니즘 **NP25073-KR_명세서초안**에 따르면, 세션 서버는 OPENMARU Cluster의 핵심 기술인 고성능 IMDG(In-Memory Data Grid)에 세션 데이터를 저장합니다. 세션툴은 세션 서버의 IMDG에 직접 접속하여, 명령받은 특정 사용자 ID 또는 세션 ID에 해당하는 세션 데이터를 즉시 '삭제 또는 변경'합니다. 세션 데이터가 삭제되면 해당 사용자의 세션은 즉시 무효화되며, 다음 요청 시 시스템은 해당 사용자를 로그인 페이지로 리디렉션하게 됩니다.

© 2025 MSAP.ai., All Rights Reserved.

3. 여기서 IMDG를 선택한 것은 아키텍처적으로 중요한 결정입니다. 일반적인 Redis와 같은 Primary/Replica 샤딩 모델과 달리, IMDG의 Peer-to-Peer(P2P) 데이터 그리드 모델은 엔터프라이즈급 제어 기능에 필수적인 선형적 확장성과 강력한 고가용성을 제공합니다. '강제 로그아웃'과 같은 민감한 제어 기능은 노드 장애 시에도 세션 데이터의 유실이 없어야 하며, IMDG의 강력한 일관성(Strong Consistency)과 자동 데이터 재조정(Rebalancing) 기능은 Redis의 비동기 복제 방식에서 발생할 수 있는 데이터 유실 위험을 원천적으로 차단하여 운영의 안정성을 보장합니다.
4. 활용 사례 및 효과 이 기능은 보안 사고 대응(Incident Response)에서 매우 강력한 도구로 활용됩니다.
    - 어뷰징 사용자 차단: 비정상적인 트래픽을 유발하는 사용자를 발견 즉시 강제 로그아웃시켜 서비스 부하를 줄입니다.
    - 계정 탈취 대응: 특정 계정이 탈취된 것으로 의심될 때, 즉시 세션을 끊어 추가적인 피해를 막습니다.
    - 내부 정보 유출 방지: 퇴사자나 권한이 변경된 사용자의 비정상적인 접근이 감지될 때 즉각적으로 세션을 종료시킬 수 있습니다.

6.2.2. "/admin/* URL에 모바일 브라우저에서 접근 못 하게 해" → APM 경유 WAS 에이전트에 트래픽 제어 명령 전송

시나리오 분석: 배포 없이 실시간으로 트래픽 정책을 변경하다

운영 중인 서비스의 특정 기능에 긴급한 문제가 발생했거나, 특정 사용자 그룹에게만 접근을 제한해야 할 때, 서버를 재시작하거나 새로운 코드를 배포하는 것은 큰 부담입니다. 이 시나리오는 운영 중인 시스템의 트래픽 흐름을 동적으로 제어하는 유연하고 강력한 기능을 보여줍니다.

1. 명령 라우팅 및 실행 이 명령은 특정 URL 패턴(`/admin/*`)과 조건(모바일 브라우저)에 따른 트래픽 제어를 의미합니다. **NP25073-KR_명세서초안**의 아키텍처에 따라, AI API 서버는 이 명령을 APM툴로 전달합니다. APM툴은 이 명령을 APM 서버로 보내고, APM 서버는 최종적으로 실제 웹 애플리케이션이 동작하는 WAS(Web Application Server)에 설치된 에이전트에게 트래픽 제어 명령을 전송합니다. WAS 에이전트는 이 명령을 수신하여 메

© 2025 MSAP.ai., All Rights Reserved.

모리에 정책을 로드하고, 이후 /admin/* 경로로 들어오는 요청의 User-Agent 헤더를 검사하여 모바일 브라우저에서의 접근을 즉시 차단합니다.

2. 정책의 동적 적용 이 방식의 가장 큰 장점은 '동적 제어'에 있습니다. 별도의 서버 재시작이나 배포 파이프라인을 거치지 않고, 운영자의 명령만으로 실시간으로 운영 중인 시스템의 접근 제어 정책을 변경할 수 있습니다. 이는 장애 대응이나 긴급 점검 시 서비스 중단 시간을 최소화하는 데 결정적인 역할을 합니다.
3. 활용 사례
    - 긴급 기능 점검: 결제 모듈 등 특정 기능에 버그가 발견되었을 때, 해당 URL로의 접근을 즉시 차단하여 피해 확산을 막습니다.
    - 베타 기능 제한: 특정 IP 대역이나 내부 사용자 그룹에게만 새로운 기능에 대한 접근을 허용하는 점진적 롤아웃(Canary Release)에 활용할 수 있습니다.
    - 디도스(DDoS) 공격 방어: 특정 패턴의 공격 트래픽이 감지될 때, 해당 URL이나 IP 대역을 신속하게 차단하여 시스템을 보호합니다.

6.2.3. "이 IP 대역은 1시간 차단해" → 자연어 명령을 룰 포맷으로 변환하는 LLM 템플릿

시나리오 분석: 운영자의 의도를 정확한 시스템 규칙으로 변환하다

LLM의 핵심 역량 중 하나는 비정형 데이터(자연어)를 정형 데이터(코드, JSON 등)로 변환하는 능력입니다. 이 시나리오는 LLM의 변환 능력을 활용하여, 운영자가 복잡한 문법을 몰라도 자연어로 말하듯 보안 정책을 시스템에 적용하는 과정을 보여줍니다.

1. 의도에서 규칙으로의 변환 운영자가 "IP 대역 xxx.xxx.xxx.0/24는 1시간 차단해" 라고 명령하면, LLM은 이 문장에 담긴 핵심 의도를 추출합니다.
    - **대상**: IP 대역 (xxx.xxx.xxx.0/24)
    - **조치**: 차단 (block)
    - **기간**: 1시간 (3600초) LLM은 이 정보를 바탕으로 방화벽이나 WAS 에이전트가 즉시 해석할 수 있는 구조화된 룰 포맷(예: JSON)으로 변환합니다.

© 2025 MSAP.ai., All Rights Reserved.

2. 템플릿 기반의 안정성 여기서 중요한 점은 LLM이 이 규칙을 마음대로 생성하는 것이 아니라는 점입니다. 시스템은 사전에 정의된 'LLM 템플릿'을 가지고 있으며, LLM은 자연어에서 추출한 파라미터를 이 템플릿에 채워 넣는 방식으로 작동합니다. 이 접근 방식은 LLM이 잘못된 문법의 규칙을 생성하거나 의도하지 않은 명령을 만들어내는 '환각(Hallucination)' 현상을 원천적으로 방지합니다. 이는 운영의 안정성을 보장하는 핵심적인 안전장치입니다.
3. 운영 효율성 증대 이 기능 덕분에 운영자는 더 이상 복잡한 방화벽 룰 문법이나 API 명세를 외울 필요가 없습니다. 공격 IP를 차단하거나 특정 국가에서의 접근을 막는 등의 보안 정책을 자연어로 즉시 적용할 수 있게 되어, 보안 위협 대응 속도와 업무 효율성이 극대화됩니다.

지금까지 살펴본 조회 및 제어 시나리오는 운영자와 시스템 간의 상호작용 방식을 근본적으로 변화시킵니다. 다음 장에서는 이러한 변화가 운영자 경험(UX) 측면에서 어떤 가치를 제공하는지 구체적으로 정리하겠습니다.

6.3. 운영자 경험(UX) 정리: 지능형 운영을 위한 인터페이스 설계

LLM 기반 운영 시스템의 성공은 단순히 기술적 성능이 뛰어난 것만으로는 보장되지 않습니다. 최종 사용자인 운영자가 시스템을 얼마나 쉽고, 직관적이며, 안전하게 사용할 수 있는지에 따라 그 가치가 결정됩니다. 아무리 강력한 기능이라도 사용하기 어렵거나, 작은 실수 하나가 대형 장애로 이어질 수 있다면 현장에서 외면받을 수밖에 없습니다.

따라서 본 섹션에서는 효과적인 LLM 연동 시스템을 구축하기 위한 핵심적인 운영자 경험(UX) 설계 원칙(Design Principles) 세 가지를 분석합니다. 이 원칙들은 단순한 편의성을 넘어, 시스템의 가치를 높이고 운영 리스크를 통제하는 아키텍처의 필수 요소입니다.

6.3.1. UX 원칙: 실행 가능한 통찰력(Actionable Insight) 제공–LLM 응답을 대시보드/링크/표로 반환하는 경우

LLM의 응답은 단순한 텍스트의 나열을 넘어, 운영자가 다음 행동을 결정할 수 있는 '실행 가능한 통찰력'을 제공해야 합니다. 정보의 가독성을 높이고, 더 깊은 분석으로 자연스럽게 연결될 수 있도록 다양한 형식의 응답을 활용하는 것이 중요합니다.

- Markdown 표 형식으로 정보 구조화 "특정 사용자ID가 오늘 어떤 URL을 몇 초씩 호출했

© 2025 MSAP.ai., All Rights Reserved.

는지 표로 줘"와 같은 요청에 대해, 데이터를 구조화된 Markdown 표로 제공하면 가독성이 비약적으로 향상됩니다. 각 열은 명확한 헤더를 가지며, 시간 순서대로 정렬되어 있어 사용자의 행동 패턴을 한눈에 파악할 수 있습니다.

- 기존 시스템과의 유기적인 연동 (대시보드 링크) "CPU 80% 넘은 시점"에 대한 분석 요청 시, LLM은 요약된 원인 분석 결과와 함께 다음과 같은 추가 정보를 제공해야 합니다.
- 여기서 제공되는 링크는 단순히 APM 대시보드의 메인 페이지가 아니라, 시간 필터가 정확히 '14:05'로 적용된 상세 메트릭 페이지의 URL이어야 합니다. 이는 LLM의 빠른 요약 능력과 기존 모니터링 시스템이 제공하는 심층 분석 능력을 결합하는 최적의 UX 설계입니다. 운영자는 LLM을 통해 문제의 개요를 파악한 뒤, 클릭 한 번으로 상세 데이터 분석을 이어갈 수 있습니다.

6.3.2. UX 원칙: 안전성(Safety) 최우선 확보–조회와 제어를 같은 프롬프트에서 허용하지 않는 가이드

시스템의 안정성을 보장하기 위해, 시스템의 상태를 변경하지 않는 '읽기(조회)' 작업과 상태를 변경하는 '쓰기(제어)' 작업은 반드시 명확하게 분리되어야 합니다. 이는 아키텍처 설계에서 CQRS(Command Query Responsibility Segregation, 명령 조회 책임 분리) 원칙을 LLM 기반 제어 플레인에 적용하는 것과 같습니다. 이 두 가지를 한 번의 명령으로 처리하도록 허용하는 것은 편리해 보일 수 있으나, 예측 불가능한 대규모 장애를 유발할 수 있는 매우 위험한 설계입니다.

- 복합 명령의 잠재적 위험성 만약 시스템이 다음과 같은 복합 명령을 허용한다고 가정해 봅시다.
- 운영자의 의도는 소수의 문제 세션을 정리하는 것이었을 수 있습니다. 하지만 만약 그 순간 데이터베이스 지연 등의 일시적인 문제로 수천 개의 세션 응답이 3초를 넘었다면, 이 명령 하나로 수천 명의 사용자가 예고 없이 로그아웃되는 대규모 서비스 장애가 발생할 수 있습니다.
- 필수적인 설계 가이드라인 안전한 시스템을 위해서는 AI API 서버 또는 LLM 프롬프트 설계 단계에서부터 조회 명령과 제어 명령을 명확히 구분해야 합니다. 또한, 모든 제어 명령을 실행하기 전에는 반드시 사용자에게 재확인 절차를 거치도록 설계하는 것이 필수적인 안전장치입니다.

© 2025 MSAP.ai., All Rights Reserved.

- 이러한 명시적인 확인 단계는 운영자의 실수를 방지하고, 명령의 영향 범위를 명확히 인지시킨 후 조치를 실행하도록 하여 시스템 안정성을 크게 높입니다.

6.3.3. UX 원칙: 효율성(Efficiency) 극대화–감시 대상 URL·사용자 그룹을 프롬프트로 등록해두는 방법–

운영 업무에는 반복적으로 조회하거나 감시해야 하는 대상 그룹이 존재합니다. 매번 동일한 URL 패턴이나 사용자 목록을 길게 입력하는 것은 비효율적입니다. 시스템이 자주 사용하는 대상을 '사용자 정의 단축키'나 '매크로(별칭)'로 등록할 수 있는 기능을 제공하면 운영 효율성을 극대화할 수 있습니다.

- 사용자 정의 별칭(Alias) 기능 운영자가 자연어 프롬프트를 통해 자주 사용하는 조회 대상을 자신만의 키워드로 등록할 수 있도록 지원합니다.
- 간결하고 직관적인 질의 이렇게 별칭을 등록해두면, 이후의 질의는 훨씬 간결하고 직관적으로 변화합니다.
- 이 기능은 복잡한 URL 패턴이나 긴 사용자 목록을 매번 입력해야 하는 번거로움을 제거하여 시스템 사용의 편의성을 크게 향상시킵니다. 이는 운영자가 더 중요한 분석과 판단에 집중할 수 있도록 돕는 실용적인 UX 설계입니다.

7장. 구현·보안·배포 고려사항

LLM(거대 언어 모델)을 APM 및 세션 관리 시스템에 통합하는 것은 단순히 기술적 연결을 넘어, 실제 운영 환경의 안정성과 성능을 보장하기 위한 깊이 있는 아키텍처 설계를 요구합니다. 특히, 현대적인 쿠버네티스/클라우드 네이티브 환경에서의 배포는 기존의 시스템 구축 방식과는 근본적으로 다른 접근을 필요로 합니다. 자동 확장(Auto-scaling)과 자동 복구(Auto-healing)와 같은 클라우드 네이티브의 핵심 원칙들은 상태를 저장하는(Stateful) 컴포넌트의 안정적 운영을 위해 특화된 설계가 반드시 선행되어야 함을 의미합니다. 기존의 정적인 서버 환경을 가정하던 배포 방식은 POD가 수시로 생성되고 소멸하는 동적인 쿠버네티스 환경에서는 더 이상 유효하지 않으며, 이는 시스템 전체의 아키텍처에 중대한 영향을 미칩니다.

© 2025 MSAP.ai., All Rights Reserved.

7.1 쿠버네티스/클라우드 네이티브 상의 배포

7.1.1 세션서버(IMDG)와 APM 서버의 상태 저장(Stateful) 구성

1. Stateful 서비스의 필요성 분석 클라우드 네이티브 아키텍처의 핵심은 애플리케이션(WAS)을 무상태(Stateless)로 유지하여 쿠버네티스의 이점을 극대화하는 것입니다. WAS POD가 상태를 내부에 저장하면, HPA(Horizontal Pod Autoscaler)에 의한 축소나 자동 복구 과정에서 POD가 사라질 때 사용자의 세션 데이터가 함께 유실되는 치명적인 문제가 발생합니다.
2. 이 문제를 해결하기 위해, 세션 데이터와 APM 데이터를 WAS 외부의 독립적인 저장소로 분리해야 합니다. **쿠버네티스 세션 외부화: IMDG 활용 전략** 문서에서 설명하듯, 세션 상태를 외부화함으로써 WAS POD는 언제든지 교체되거나 확장/축소될 수 있는 완전한 무상태 컴포넌트로 남을 수 있습니다. 결과적으로, 세션서버와 APM 서버는 이 외부화된 데이터를 안정적으로 보관해야 하므로 반드시 상태 저장(Stateful) 서비스로 구성되어야 합니다. 이는 시스템의 안정성과 데이터 무결성을 보장하는 필수적인 설계 결정입니다.
3. 세션서버를 위한 IMDG(In-Memory Data Grid) 선택 평가 세션 외부 저장소로는 Redis와 IMDG가 주로 고려되지만, 대규모 엔터프라이즈 환경에서는 아키텍처의 구조적 차이가 중요한 선택 기준이 됩니다. OPENMARU Cluster가 채택한 IMDG 방식과 Redis 방식은 다음과 같은 차이점을 가집니다.

비교 항목	OPENMARU Cluster (IMDG)	Redis
아키텍처	P2P(Peer-to-Peer) 데이터 그리드 구조. 모든 노드가 동등하며 데이터가 자동으로 분산 및 복제됩니다.	클라이언트-서버 및 Primary/Replica 구조. 데이터는 Primary 노드에 저장되고 Replica로 복제됩니다.

© 2025 MSAP.ai., All Rights Reserved.

비교 항목	OPENMARU Cluster (IMDG)	Redis
확장성	뛰어난 수평적 확장성. 노드를 추가하면 데이터가 자동으로 재조정(Rebalance)되어 성능이 선형적으로 확장됩니다.	수평 확장은 가능하나 노드 추가 시 슬롯(Slot) 재분배 과정이 필요하며, 쓰기 성능은 Primary 노드에 집중됩니다.
고가용성	각 노드가 데이터 복제본을 유지하므로 특정 노드 장애 시에도 다른 노드가 즉시 서비스를 이어받아 중단 없는 서비스 제공이 가능합니다.	Sentinel 또는 Cluster를 통해 장애 조치를 지원하지만, Primary 장애 시 Replica가 승격되는 동안 일시적 지연이 발생할 수 있습니다.
데이터 일관성	강한 일관성(Strong Consistency)을 보장하는 옵션을 제공하여 세션 데이터의 무결성을 높일 수 있습니다.	기본적으로 최종 일관성 (Eventually Consistent) 모델을 따르며, 비동기 복제로 인해 장애 시 데이터 유실 가능성이 있습니다.
운영 복잡성	자동화된 클러스터링 및 노드 리밸런싱으로 대규모 환경에서 운영 관리가 상대적으로 용이합니다.	클러스터 구성 및 관리가 상대적으로 복잡하며, 노드 추가 시 수동 개입이나 추가 설정이 필요할 수 있습니다.

결론적으로, IMDG의 P2P 아키텍처는 대규모 클라우드 네이티브 환경에 구조적으로 더 우수합니다. 자동화된 데이터 재조정(Rebalancing)은 선형적 확장성을 보장하며, 강한 일관성(Strong Consistency) 지원은 미션 크리티컬한 세션 관리에 요구되는 데이터 무결성을 담보하기 때문입니다.

1. APM 서버의 Stateful 특성 상세 세션서버의 데이터가 현재 활성 세션에 대한 휘발성 정보를 다루는 반면, APM 서버는 장기적인 분석을 위한 영구 데이터를 저장해야 합니다. 특허명세서(NP25073-KR)에 따르면, APM 서버는 사용자 트랜잭션 정보, 세션 ID, 사용자 식별

© 2025 MSAP.ai., All Rights Reserved.

인자를 수집하여 영구적인 **APM 트랜젝션 데이터베이스**에 저장합니다.

2. 이 데이터베이스는 반드시 Stateful로 구성되어야 합니다. 그 이유는 LLM이 과거 트랜잭션 이력을 분석하여 패턴을 찾거나, 특정 시점의 장애 원인을 추론하는 등 시계열 분석을 수행하기 때문입니다. 만약 APM 데이터가 휘발성이라면, LLM은 과거 데이터를 기반으로 한 깊이 있는 분석이나 통찰을 제공할 수 없게 됩니다. 따라서 APM 서버의 데이터베이스는 시스템의 지능형 분석 기능을 위한 핵심적인 상태 저장소 역할을 수행합니다.

7.1.2 LLM 서버의 GPU/CPU 노드 선택과 네트워크 대역폭 고려

1. LLM 서버 하드웨어 요구사항 평가 LLM의 추론(Inference) 성능은 사용자 경험과 직결되는 핵심 요소입니다. `NVIDIA_AgentAI_Development_v3.pdf` 문서에서 강조하듯, LLM의 잠재력을 최대한 활용하기 위해서는 고성능의 가속 컴퓨팅 인프라가 필수적입니다. 자연어 질의에 대한 응답이 지연된다면 시스템의 효용성은 크게 떨어질 것입니다.
2. 주요 하드웨어 선택 기준 LLM 서버를 구성할 때는 다음의 기준을 반드시 고려해야 합니다.

    - GPU vs. CPU: 거대 언어 모델의 추론 연산은 병렬 처리에 최적화되어 있으므로, 고성능 GPU는 필수적입니다. NVIDIA의 Hopper 또는 Blackwell 아키텍처 기반 GPU는 낮은 지연 시간(Latency)과 높은 처리량(Throughput, tokens/s)을 보장하여 실시간에 가까운 응답을 가능하게 합니다. 반면, Llama Nemotron Nano와 같은 소규모 모델은 엣지(Edge) 환경이나 개발/테스트 목적에 한해 CPU 노드에서 실행할 수 있으나, 이는 상당한 성능 저하를 감수해야 합니다.
    - 추론 최적화 (Inference Optimization): 하드웨어뿐만 아니라 소프트웨어 최적화도 중요합니다. NVIDIA `TensorRT-LLM`과 같은 라이브러리는 NVIDIA GPU의 성능을 극대화하는 핵심 컴포넌트입니다. 예를 들어, Llama 7B 모델에 대해 H100 GPU에서 추론 성능을 최대 4.6배까지 향상시킬 수 있으며, 이는 곧바로 더 빠르고 응답성 좋은 사용자 경험으로 이어집니다.
    - 네트워크 대역폭 (Network Bandwidth): 대규모 모델을 여러 노드에 분산하여 학습시키거나 추론하는 '3D Parallelism'과 같은 모델에서는 노드 간 통신이 병목이 될 수 있습니다. 따라서 InfiniBand나 RoCE(RDMA over Converged Ethernet)와 같은 고대역폭, 저지연 네트워킹 기술을 적용하여 노드 간 데이터 전송 시간을 최소화해야

© 2025 MSAP.ai., All Rights Reserved.

합니다.

7.1.3 멀티 클러스터·DR 환경에서의 세션 데이터 복제

1. 재해 복구(DR) 시나리오의 세션 복제 과제 엔터프라이즈 시스템에서는 지리적으로 분산된 여러 쿠버네티스 클러스터 간에 재해 복구(DR) 체계를 구축하는 것이 일반적입니다. 이때 가장 중요한 과제 중 하나는 주 데이터센터(Active DC)에 장애가 발생했을 때 백업 데이터센터(Standby DC)로 트래픽을 전환하더라도 사용자의 세션 상태를 그대로 유지하여 서비스 연속성을 보장하는 것입니다.
2. 세션 복제를 위한 IMDG 아키텍처의 우수성 이러한 멀티 클러스터 환경에서 IMDG 아키텍처는 구조적인 강점을 가집니다. **WAS 세션 클러스터링: Redis 대 IMDG 심층 비교 및 데이터 그리드와 Redis 비교 분석** 문서에 따르면, Hazelcast나 Infinispan과 같은 IMDG 솔루션은 데이터센터 간 데이터 복제를 위한 Cross-Datacenter Replication (XDR) 또는 WAN Replication이라는 강력한 내장 기능을 제공합니다. 이 기능을 통해 여러 클러스터의 세션 데이터를 실시간으로 동기화하여 한쪽 클러스터에 장애가 발생해도 다른 쪽에서 즉시 세션을 이어받을 수 있습니다.
3. 대안 아키텍처와의 비교 반면, Redis 오픈소스(OSS) 버전은 네이티브 액티브-액티브(Active-Active) 복제 기능을 지원하지 않습니다. 이로 인해 DR 환경을 구축하려면 별도의 외부 솔루션을 도입하거나 복잡한 스크립트를 통해 수동으로 데이터를 동기화해야 하므로, 구성이 복잡해지고 장애 전환(Failover) 시 데이터 유실의 위험이 커집니다. 결론적으로, 자동 리밸런싱을 지원하는 IMDG의 P2P 아키텍처와 내장된 WAN 복제 기능은 안정적인 멀티 클러스터 및 DR 환경 구축에 구조적으로 더 적합합니다.

이처럼 LLM 통합 플랫폼을 성공적으로 배포하기 위해서는 쿠버네티스 환경에 대한 깊은 이해를 바탕으로 각 컴포넌트의 특성에 맞는 아키텍처를 신중하게 선택해야 하며, 이는 시스템의 안정성과 직결되는 보안 및 접근 제어 전략으로 자연스럽게 이어집니다.

7.2 보안·접근제어

LLM을 APM과 같은 핵심 운영 시스템과 통합할 때, 보안과 접근 제어는 단순한 부가 기능이 아닌 아키텍처의 핵심 요소로 설계되어야 합니다. 특히 자연어 명령을 통해 실시간 사용자 세션이나 운

© 2025 MSAP.ai., All Rights Reserved.

영 트래픽을 직접 제어할 수 있는 기능은 강력한 만큼 큰 위험을 내포하고 있습니다. 모호한 자연어 해석이나 LLM의 환각(Hallucination) 현상이 의도치 않은 서비스 장애로 이어지는 것을 방지하기 위해, 다층적인 안전장치를 마련하는 것이 필수적입니다.

7.2.1 자연어 명령이 곧바로 운영 트래픽을 건드리지 못하도록 하는 중간 승인 플로우

1. 직접 실행 경로의 위험성 특허 명세서(NP25073-KR, 도 5)에 묘사된 아키텍처는 사용자의 자연어 명령이 AI API 서버에서 분석된 후, 세션툴(Session Tool)이나 APM툴(APM Tool)을 통해 곧바로 대상 시스템에서 실행되는 자동화된 흐름을 보여줍니다. 이는 신속한 대응을 가능하게 하지만, 운영 환경의 안전성을 고려할 때 잠재적인 위험을 안고 있습니다.
2. 필수적인 보안 강화 제안: 중간 승인 워크플로우 특허가 자동화된 흐름의 가능성을 제시하더라도, 실제 엔터프라이즈 환경의 운영 안전성을 위한 모범 사례(Best Practice)는 반드시 중간 승인 단계를 포함하는 것입니다. 이를 통해 자동화의 효율성과 인간의 통제력을 결합할 수 있습니다.
3. 권장 승인 워크플로우 안전성을 극대화하기 위해 다음과 같은 “Human-in-the-loop” 워크플로우를 도입해야 합니다.
    1. LLM이 사용자의 자연어 명령을 분석하여 실행 가능한 제어 명령(예: 특정 세션 강제 종료, 특정 IP 대역 차단)을 생성합니다.
    2. 생성된 명령은 즉시 실행되지 않고 ‘승인 대기(Pending)’ 상태로 전환되며, 동시에 해당 권한을 가진 운영 담당자에게 알림(Notification)이 전송됩니다.
    3. 운영자는 LLM이 생성한 명령의 의도, 영향 범위, 잠재적 부작용을 검토합니다.
    4. 운영자의 명시적인 ’승인’이 있어야만 해당 명령이 APM 또는 세션툴로 전달되어 최종 실행됩니다.
4. 결론: 안전장치로서의 인간 개입 이러한 인간 개입 단계는 모호한 자연어 지시나 LLM의 오류로 인해 발생할 수 있는 대규모 서비스 장애를 예방하는 가장 중요한 안전장치입니다. 이는 시스템의 신뢰성을 확보하기 위한 필수적인 설계 원칙입니다.

© 2025 MSAP.ai., All Rights Reserved.

7.2.2 사용자 식별인자(고객ID, 주민번호 대체키 등) 마스킹/토큰화 전략

1. 사용자 식별인자의 데이터 흐름 특허 명세서(NP25073-KR)에 따르면, 시스템은 “사용자 식별인자”를 세션서버와 APM 서버에서 수집하여 분석 및 질의응답을 위해 최종적으로 LLM 서버로 전송합니다.
2. 내재된 보안 위험 분석 고객 ID나 주민등록번호 대체키와 같은 개인식별정보(PII)를 원본 그대로 LLM(사내 구축 모델 또는 외부 상용 API)에 전송하는 것은 심각한 데이터 프라이버시 침해 및 개인정보보호법과 같은 규정 준수 위반 리스크를 야기합니다.
3. 필수 보안 계층: 마스킹 및 토큰화 전략 이러한 리스크를 원천적으로 차단하기 위해, 다음과 같은 마스킹 및 토큰화 전략을 필수 보안 계층으로 적용해야 합니다.
    1. 토큰화 (Tokenization): 데이터가 LLM 서버로 전송되기 전, 고객 ID와 같은 원본 사용자 식별자는 암호학적으로 안전하고 역추적이 불가능한 토큰으로 대체됩니다.
    2. 격리된 매핑 서비스 (Isolated Mapping Service): 원본 식별자와 토큰 간의 매핑 정보는 아키텍처적으로 분리된 고도의 보안을 갖춘 ’토큰 관리 서비스’에만 저장됩니다. 이 서비스는 LLM이 접근할 수 없도록 철저히 격리되어야 합니다.
    3. 제어된 역토큰화 (Controlled Detokenization): 세션 종료와 같은 제어 명령을 실행해야 할 때, 권한이 부여된 APM 또는 세션툴만이 토큰 관리 서비스에 질의하여 토큰을 실제 식별자로 변환하고 대상 시스템에 명령을 수행합니다.

7.2.3 감사로그에 프롬프트·응답·실행된 트래픽 제어 명령을 모두 남기는 구조

1. 포괄적인 감사 로깅의 필요성 LLM 기반 제어 시스템에서는 모든 상호작용을 추적할 수 있어야 합니다. 이는 장애 발생 시 원인 분석, 보안 사고 포렌식, 그리고 규정 준수 감사를 위해 필수적입니다. “누가, 언제, 무엇을 요청했고, 시스템이 어떻게 반응했으며, 그 결과 어떤 조치가 취해졌는가”를 명확히 기록해야 합니다.
2. 필수 감사 로그 필드 정의 특허 명세서(NP25073-KR)에 기술된 시스템 아키텍처를 기반으로, 모든 LLM 상호작용에 대해 다음과 같은 정보를 포함하는 감사 로그를 생성하고 영구적으로 보관해야 합니다.

© 2025 MSAP.ai., All Rights Reserved.

필드 (Field)	설명 (Description)
Timestamp	이벤트가 발생한 정확한 시간 (ISO 8601 형식 권장)
Source User/System	프롬프트를 입력한 사용자 또는 시스템의 식별자
Original Prompt	사용자가 입력한 원본 자연어 질의 또는 명령의 전체 텍스트
LLM Response (Raw)	LLM이 생성한 응답의 가공되지 않은 전체 텍스트
Parsed Command	LLM 응답에서 추출된 실행 가능한 구체적인 명령 (예: `DELETE session_id`, `BLOCK ip_address`)
Target System	명령이 실행될 대상 시스템 (예: Session Server, APM/WAS Agent)
Execution Status	실행 상태 (예: Success, Failure, Pending Approval)
Execution Result	대상 시스템으로부터 반환된 결과 또는 오류 메시지

이처럼 강력한 보안 및 감사 체계를 구축하는 것은 시스템의 신뢰성을 확보하는 데 필수적이며, 이는 시스템의 품질과 성능을 검증하는 엄격한 테스트 단계로 이어져야 합니다.

7.3 품질·성능 시험

LLM, APM, 세션 관리 시스템이 유기적으로 결합된 이 복합 시스템을 실제 운영 환경에 배포하기 전, 안정성, 응답성, 정확성을 검증하기 위한 엄격한 품질 및 성능 테스트는 필수적입니다. 테스트의 목표는 현실적인 대규모 부하 조건 하에서 시스템이 예상대로 동작하고, 성능 병목이 없으며, 비즈니스 요구사항을 충족하는지 사전에 확인하는 것입니다.

© 2025 MSAP.ai., All Rights Reserved.

7.3.1 세션 10만·동접 5천 수준의 부하 시 LLM 질의 응답시간 측정

1. 테스트 시나리오 정의 실제 운영 환경과 유사한 부하를 시뮬레이션하기 위해, OPENMARU Cluster(IMDG)가 관리하는 활성 세션 100,000개와 APM이 처리하는 동시 사용자 5,000명 수준의 트랜잭션이 발생하는 시나리오를 구성합니다. 이 상태에서 운영자가 시스템에 자연어 질의를 입력하는 상황을 가정합니다.
2. 측정 방법론: End-to-End Latency 핵심 성능 지표는 'End-to-End Latency'로, 사용자가 자연어 질의를 제출한 시점부터 LLM이 생성한 완전한 응답을 수신하기까지 소요되는 총 시간을 측정합니다.
3. 질의 응답 지연 시간의 구성 요소 이 총 지연 시간은 다음과 같은 여러 단계의 합으로 구성됩니다.

    1. 데이터 검색 지연 (Data Retrieval Latency): AI API 서버가 사용자의 질의를 해석한 후, 필요한 컨텍스트 데이터를 확보하기 위해 APM 서버의 데이터베이스 및/또는 세션서버의 IMDG를 조회하는 데 걸리는 시간.
    2. LLM 추론 지연 (LLM Inference Latency): LLM 서버가 프롬프트와 검색된 컨텍스트 데이터를 입력받아 응답을 생성하는 데 걸리는 시간. 이 시간은 모델의 크기와 기반 GPU 하드웨어의 성능에 크게 좌우됩니다.
    3. 네트워크 지연 (Network Latency): AI API 서버, APM/세션서버, LLM 서버 등 각 컴포넌트 간에 데이터를 주고받는 데 소요되는 네트워크 전송 시간.

7.3.2 APM 에이전트가 추가한 오버헤드 계측

1. APM 에이전트 오버헤드의 개념 APM 에이전트는 모니터링 대상 WAS(Web Application Server)의 성능 데이터를 수집하기 위해 설치되지만, 이 과정에서 필연적으로 약간의 리소스(CPU, 메모리)를 추가로 소모하고 응답 시간에 미세한 지연을 추가합니다. 이를 '오버헤드(Overhead)'라고 하며, 이 수치가 허용 범위 내에 있는지 반드시 측정해야 합니다.
2. 오버헤드 측정을 위한 A/B 테스트 방법론

    - 기준선 설정 (A): 통제된 부하 테스트 환경에서, APM 에이전트를 설치하지 않은 상태의 WAS에 부하를 가하여 핵심 성능 지표(KPI)인 평균 응답 시간, 초당 트랜잭션 수

© 2025 MSAP.ai., All Rights Reserved.

(TPS), CPU 사용률, 메모리 사용량을 측정합니다. 이것이 성능의 기준선이 됩니다.

- 에이전트 설치 후 측정 (B): 동일한 WAS에 APM 에이전트를 설치한 후, 이전과 정확히 동일한 조건의 부하 테스트를 다시 수행하여 동일한 KPI들을 측정합니다.
- 오버헤드 계산: 각 KPI에 대해 (B) 측정값과 (A) 측정값의 차이를 백분율로 계산하여 오버헤드를 산출합니다. 예를 들어, `오버헤드(%) = ((B값 - A값) / A값) * 100`으로 계산할 수 있습니다.

7.3.3 LLM 모델 교체 시(사내 모델↔상용 API) 동작 검증 체크리스트

1. 검증 체크리스트의 필요성 시스템에 내장된 LLM을 다른 모델(예: 사내 구축 모델에서 상용 클라우드 API로 전환)로 교체하는 것은 단순한 설정 변경이 아닙니다. 이는 시스템의 기능, 성능, 비용, 보안 등 전반에 걸쳐 중대한 영향을 미칠 수 있으므로, 체계적이고 안전한 전환을 보장하기 위한 검증 체크리스트가 필수적입니다.
2. LLM 모델 교체 검증 체크리스트
3. 기능 검증 (Functional Verification)
   - ☐ Prompt Compatibility: 새로운 모델이 기존 시스템 프롬프트와 사용자 질의를 의도한 대로 정확하게 이해하는가?
   - ☐ Response Accuracy: APM/세션 데이터를 기반으로 사실에 입각한 정확한 답변을 제공하는가?
   - ☐ Command Generation: 구문적으로 올바르고 논리적으로 타당한 제어 명령을 생성하는가?
   - ☐ Format Adherence: Markdown 테이블, JSON 등 시스템이 요구하는 출력 포맷을 일관되게 준수하는가?
4. 비기능 검증 (Non-Functional Verification)
   - ☐ Latency: 새로운 모델의 평균 및 최대 응답 시간이 서비스 수준 협약(SLA) 기준을 만족하는가?
   - ☐ Throughput: 새로운 모델이 예상되는 최대 질의 부하를 감당할 수 있는가?
   - ☐ Cost Analysis: (상용 API의 경우) 새로운 모델의 운영 비용이 이전 모델 대비 어느 정도이며, 예산 범위 내에 있는가?

© 2025 MSAP.ai., All Rights Reserved.

5. 보안 및 규정 준수 검증 (Security & Compliance Verification)

   ☐ Data Privacy Policy: (클라우드 기반 API의 경우) 새로운 모델 제공사의 데이터 처리 정책이 회사의 보안 및 규정 준수 요구사항과 일치하는가?

   ☐ Input/Output Filtering: 유해하거나 부적절한 콘텐츠를 차단하는 안전 필터링 메커니즘이 기존 모델과 동등하거나 그 이상인가?

본 챕터에서 다룬 배포, 보안, 품질 테스트는 LLM 통합 플랫폼을 아이디어에서 실제 운영 가능한 엔터프라이즈 시스템으로 구현하기 위한 실질적인 고려사항들을 종합적으로 다루었습니다. 이러한 체계적인 접근을 통해 기술의 잠재력을 최대한 활용하면서도 안정성과 신뢰성을 확보할 수 있습니다.

8장. 적용 사례와 화면 예시

8.1 특허 명세서 기반 예시 화면 재구성

본 장에서는 특허(NP25073-KR)에 명시된 핵심 아이디어를 실제 운영 화면으로 시각화하여, LLM 기반 관제 시스템이 제공하는 직관적인 가치를 증명하는 것을 목표로 합니다. 이론적인 아키텍처가 어떻게 실용적인 사용자 경험으로 변환되는지 보여줌으로써, 복잡한 시스템 운영을 자연어 하나로 해결하는 혁신적인 패러다임을 구체적으로 제시합니다. 각 예시는 가상의 UI 화면을 상세히 묘사하여 독자가 실제 시스템을 사용하는 듯한 경험을 할 수 있도록 구성되었습니다.

8.1.1 특정 사용자의 기간별 트랜잭션 이력 조회 화면 (표·대시보드)

특정 사용자의 과거 행적을 추적하는 것은 장애 분석과 사용자 경험 최적화의 첫걸음입니다. 기존 시스템에서는 여러 로그 소스를 수동으로 조합하고 필터링해야 했던 이 복잡한 작업을, 제안된 시스템에서는 LLM을 통해 단일 자연어 질의로 해결할 수 있습니다. 이는 문제 해결 시간을 획기적으로 단축시키는 동시에, 운영자가 데이터가 아닌 문제의 본질에 집중할 수 있도록 만드는 전략적 가치를 지닙니다.

운영자는 LLM 기반 관제 시스템에 다음과 같이 질문할 수 있습니다.

"사용자 ID `kim-star`의 어제 오전 9시부터 10시까지의 모든 트랜잭션 이력을 표로 보여줘."

© 2025 MSAP.ai., All Rights Reserved.

이 질의에 응답하기 위해 시스템은 APM 서버의 영구 저장소, 즉 'APM 트랜잭션 데이터베이스'를 조회합니다. 이곳은 관측성 데이터를 위한 시스템의 영속성 계층(persistence layer) 역할을 합니다. 특허(NP25073-KR) 아키텍처에 따라, 사용자의 웹 세션이 종료된 후에도 트랜잭션 상세 정보, 세션 ID, 사용자 식별인자가 서로 매칭되어 영구적으로 저장되기 때문에 과거 이력 조회가 가능합니다. 이러한 아키텍처 분리는 의도적인 설계의 결과입니다. 세션 서버의 휘발성 인메모리 그리드(In-Memory Data Grid)는 실시간 세션에 대한 저지연(low-latency) 쿼리에 최적화된 반면, APM 데이터베이스는 과거 추이 분석 및 장애 사후 분석에 필요한 데이터의 영구적(durable) 보관을 위해 설계되었습니다.

LLM은 조회된 데이터를 바탕으로 다음과 같은 결과 화면을 생성하여 운영자에게 제시합니다.

- 사용자 **kim-star** 트랜잭션 이력 (2024-10-26 09:00 ~ 10:00)

타임스탬프	호출 URL	HTTP Method	응답시간 (ms)	상태 코드	세션 ID
2024-10-26 09:05:12	/main/dashboard	GET	150	200	sess-abc-123
2024-10-26 09:07:34	/api/v1/products/search	POST	850	200	sess-abc-123
2024-10-26 09:07:59	/product/detail/P0045	GET	320	200	sess-abc-123
2024-10-26 09:15:21	/cart/add	POST	1200	503	sess-abc-123

이 표는 단순히 텍스트 정보를 나열하는 것에서 끝나지 않습니다. 각 행(트랜잭션)은 클릭 가능한 하이퍼링크로 제공되어, 클릭 시 해당 트랜잭션의 모든 상세 호출 내역을 분석할 수 있는 OPENMARU APM의 상세 트레이스(Trace) 화면으로 즉시 연결됩니다. 이를 통해 운영자는 거시적인 이력 조회에서 미시적인 원인 분석까지 끊김 없이 탐색할 수 있습니다.

© 2025 MSAP.ai., All Rights Reserved.

이처럼 과거 데이터를 분석하는 것을 넘어, 실시간 시스템 상태와 사용자 활동을 결합하여 분석하는 예시를 다음 섹션에서 살펴보겠습니다.

8.1.2 일정 기간 사용자 수 vs 웹서버 CPU 사용률 비교 화면

시스템 부하와 사용자 활동 간의 상관관계를 파악하는 것은 용량 계획 및 성능 병목 분석의 핵심 과제입니다. 기존에는 CPU 사용률과 같은 시스템 메트릭과 동시 접속자 수와 같은 활동 로그가 별개의 대시보드에 존재하여 운영자가 두 데이터를 머릿속으로 조합하며 상관관계를 추론해야 했습니다. LLM 기반 시스템은 이 두 가지 이종 데이터를 맥락적으로 결합하여 단순 데이터 비교를 넘어 깊이 있는 인사이트를 도출합니다.

운영자는 다음과 같은 복합적인 질문을 할 수 있습니다.

"지난 1시간 동안의 전체 동시 접속자 수 추이와 웹서버 평균 CPU 사용률을 비교하는 그래프를 그려줘."

이 질의를 처리하기 위해 시스템은 두 개의 다른 데이터 소스를 동시에 활용합니다. 특허(NP25073-KR) 도면 4에서 묘사된 바와 같이, LLM은 APM 서버에서 수집한 '웹서버 CPU 사용률' 메트릭 데이터와, 세션서버 및 APM DB에서 집계한 '동시 접속자 수' 데이터를 결합하여 분석을 수행합니다.

LLM의 응답은 단순한 그래프 이미지가 아니라, 두 데이터 간의 상관관계를 분석한 텍스트 요약과 함께 제공되어 운영자의 빠른 의사결정을 돕습니다.

- 동시 접속자 수와 CPU 사용률 상관관계 분석 (2024-10-26 13:00 ~ 14:00)
  - 최대 동시 접속자 수: 13:35 경, 5,200명
  - 최고 CPU 사용률: 13:37 경, 85%
  - 분석: "동시 접속자 수가 5,000명을 넘어서는 시점부터 CPU 사용률이 급증하는 패턴을 보입니다. 특히 13:35 ~ 13:40 구간에서 상관관계가 가장 높게 나타났습니다. 해당 시간대의 부하 유발 URL에 대한 추가 분석이 필요합니다."

이와 같이 시스템의 거시적인 상태를 분석하는 것에서 더 나아가, LLM을 통해 특정 사용자의 실시간 활동을 미시적으로 추적하는 방법을 다음 예시에서 확인하겠습니다.

© 2025 MSAP.ai., All Rights Reserved.

8.1.3 자연어로 현재 세션 페이지 위치를 확인하는 예시

실시간 사용자 지원이나 이상 행위 탐지를 위해서는 특정 사용자의 현재 활동을 즉시 파악하는 것이 중요합니다. 예를 들어, 결제 페이지에서 오류를 겪는 고객을 지원하거나, 비정상적인 접근 패턴을 보이는 사용자를 추적할 때 이 기능은 결정적인 역할을 합니다. 이 기능은 과거 데이터가 아닌, 세션서버의 휘발성 메모리에 존재하는 실시간 데이터를 직접 조회하여 구현됩니다.

운영자는 특정 사용자의 현재 상태를 즉시 확인하기 위해 다음과 같이 질문할 수 있습니다.

"사용자 ID `lee-vip`는 지금 어느 페이지에 머물고 있어?"

이 질의는 과거 데이터가 아닌 '현재 진행 중인 세션'에 대한 정보이므로, 특허(NP25073-KR) 아키텍처에 따라 AI API 서버는 의도적으로 APM 데이터베이스를 우회하고, '세션툴(Session Tool)'을 통해 요청을 세션 서버로 직접 라우팅합니다. 이 전용 저지연 경로는 응답 시간이 핵심적인 실시간 질의를 위해 특별히 설계된 아키텍처입니다.

LLM은 세션서버로부터 받은 실시간 데이터를 바탕으로 다음과 같이 간결하고 명확한 답변을 생성합니다.

"사용자 `lee-vip`는 현재 `/cart/checkout` 페이지(결제 진행 단계)에 3분 15초 동안 머물고 있습니다."

이 기능의 활용 범위는 개인 사용자에 국한되지 않습니다. 예를 들어, "현재 '관리자' 그룹에 속한 모든 사용자의 세션 위치를 알려줘"와 같이 특정 사용자 그룹 전체의 실시간 활동을 모니터링하는 데에도 활용될 수 있어, 보안 및 운영 관점에서 높은 응용 가능성을 가집니다.

지금까지 LLM을 활용한 다양한 조회 사례를 살펴보았으며, 다음 장에서는 단순 조회를 넘어 시스템을 직접 제어하는 운영 시나리오를 구체적인 워크플로와 함께 분석하겠습니다.

8.2 운영 시나리오별 워크플로

LLM의 진정한 가치는 단순히 데이터를 조회하고 분석하는 것을 넘어, 운영자의 의도를 깊이 이해하고 실제 시스템 제어까지 연결하는 'VibeOps'의 구현에 있습니다. 이는 운영자와 시스템의 관계를 '명령자-도구(master/tool)'의 관계에서 '협력적 파트너십'으로 근본적으로 재정의하는 것을 의미합니다. 기존의 스크립트 기반 자동화(AIOps)를 넘어, 문맥을 이해하고 의도를 파악하여 대응하는 '의도 중심 운영(intent-based operations)'으로의 진화입니다. 본 섹션에서는 장

© 2025 MSAP.ai., All Rights Reserved.

애 대응, 보안 통제, 성능 관리라는 세 가지 핵심 운영 시나리오를 통해, 자연어 명령이 어떻게 구체적인 시스템 조치로 이어지는지 그 단계별 워크플로를 제시합니다.

8.2.1 장애 시간대: “어느 URL이 동시접속자수를 튀게 만들었나”를 LLM으로 바로 묻는 절차

장애 발생 시 가장 중요한 것은 ‘골든 타임’ 내에 근본 원인(Root Cause)의 유력한 후보를 빠르게 식별하는 것입니다. 기존 방식에서는 운영자가 동시접속자수 대시보드, CPU 사용률 그래프, Top-N URL 목록 등 여러 화면을 오가며 수동으로 데이터를 비교하고 상관관계를 분석해야 했습니다. 이 워크플로는 그 모든 과정을 LLM에게 직접 질문함으로써, 운영의 패러다임을 ’탐색’에서 ’질의’로 전환하는 혁신을 보여줍니다.

1. 1단계 (상황 인지): 운영자는 APM 모니터링 대시보드를 통해 특정 시간대(예: 14:20~14:25)에 동시접속자 수가 평소 대비 비정상적으로 급증한 것을 인지합니다.
2. 2단계 (자연어 질의): 운영자는 별도의 데이터 조회 없이, LLM 인터페이스에 상황의 본질을 담아 다음과 같이 직접 질문합니다.
3. 3단계 (데이터 분석): 질의를 수신한 AI API 서버는 ’APM툴’을 호출하여 해당 시간대의 트랜잭션 데이터를 APM DB에서 추출합니다. LLM의 핵심 역량은 단순히 로그를 필터링하는 것을 넘어, 사용자 세션 수, 애플리케이션 트랜잭션 지표(TPS, 응답시간), 서버 리소스 지표(CPU)라는 세 가지 이종 데이터 스트림을 종합하여 인과 관계를 추론하는 데 있습니다. 이는 기존 방식으로는 여러 대시보드를 오가며 수동으로 분석해야만 했던 다차원적 상관관계 분석입니다.
4. 4단계 (결과 및 근거 제시): LLM은 분석 결과를 단순 텍스트가 아닌, 명확한 결론과 이를 뒷받침하는 근거 데이터를 함께 제시하여 운영자의 신뢰도를 높입니다.
    - 결론: “해당 시간대 동시접속자 수 급증은 `/event/promo/newYear` URL 호출이 폭주하면서 발생한 것으로 보입니다.”
    - 근거:

© 2025 MSAP.ai., All Rights Reserved.

순위	URL	해당 시간대 TPS	평균 응답시간 (ms)	비고
1	/event/promo/newYear	1,500	2,500	평시 대비 TPS 30배 증가
2	/api/v1/users/me	250	300	연관 호출로 동반 상승
3	/static/js/promo.js	200	150	정적 리소스

이처럼 신속하게 장애 원인을 분석하는 것을 넘어, 보안 위협에 대응하기 위해 특정 사용자의 접근을 즉시 통제하는 시나리오를 다음 절에서 살펴보겠습니다.

8.2.2 보안/통제: “이 사용자는 지금부터 특정 메뉴를 못 들어오게 해라” 실행 절차

어뷰징, 정보 유출 시도 등 보안 사고가 발생했을 때, 특정 사용자의 접근을 즉각적으로 차단하는 것은 서비스의 신뢰도와 데이터를 보호하기 위한 필수 조치입니다. 이 워크플로는 운영자의 자연어 명령이 어떻게 세션서버를 직접 제어하여 실시간으로 접근 통제 조치를 실행하는지 그 과정을 보여줍니다. 이는 분석과 실행 사이의 간극을 없애 즉각적인 대응을 가능하게 합니다.

1. 1단계 (상황 인지): 운영자는 모니터링 시스템을 통해 특정 사용자(abuse-user-01)가 비정상적인 패턴으로 /admin 메뉴에 반복적으로 접근하는 것을 발견합니다.
2. 2단계 (자연어 명령): 운영자는 상황의 심각성을 인지하고 LLM 인터페이스에 즉시 제어 명령을 내립니다.
3. 3단계 (명령 분석 및 라우팅): 특허(NP25073-KR) 아키텍처에 따라, AI API 서버는 이 명령이 단순 조회가 아닌 ’세션 제어’에 해당한다고 판단하고, 이 작업을 수행할 수 있는 ’세션 툴’을 호출합니다.
4. 4단계 (실행): 세션툴은 세션서버(OPENMARU Cluster)의 인메모리 데이터 그리드에 직접 접속하여 다음 두 가지 조치를 원자적으로 실행합니다.

© 2025 MSAP.ai., All Rights Reserved.

- 사용자 식별인자 `abuse-user-01`에 매핑된 세션 ID를 찾아 해당 세션 데이터를 삭제하여 즉시 강제 로그아웃시킵니다. 이 조치는 세션 그리드의 인메모리 상태를 직접 제어하는 것으로, 로그 기반 분석이나 배치(batch) 중심의 보안 도구로는 불가능한 수준의 실시간 통제력을 보여줍니다. 그 효과는 즉각적입니다.
- 동시에, 해당 사용자의 세션 저장소 프로필을 업데이트하여 `/admin` 접근을 차단하는 속성을 추가함으로써 재로그인 시에도 정책이 유지되도록 보장할 수 있습니다.

5. 5단계 (결과 보고): 조치가 성공적으로 완료되면 LLM은 운영자에게 명확한 실행 결과를 보고하여 상황이 종료되었음을 확인시켜 줍니다.

이러한 즉각적인 사후 대응뿐만 아니라, 잠재적인 성능 문제를 예방하기 위해 LLM이 선제적으로 정책을 제안하는 흐름을 다음으로 분석하겠습니다.

8.2.3 성능/용량: “이 URL은 상시로 2초 이내로 제한하라” 정책화를 LLM이 제안하는 흐름

안정적인 서비스 품질을 유지하기 위해서는 특정 기능의 성능 저하가 전체 시스템 장애로 확산되기 전에 선제적으로 대응하는 것이 중요합니다. 이 워크플로는 AIOps의 궁극적 목표, 즉 문제에 더 빨리 ‘대응’하는 것을 넘어 문제가 발생하기 전에 ‘예방’하는 단계로의 진화를 보여줍니다. 여기서 LLM은 시스템을 지속적으로 분석하여 안정성 개선 방안을 제안하는 ‘디지털 사이트 신뢰성 엔지니어(digital Site Reliability Engineer, SRE)’와 같은 역할을 수행합니다.

1. 1단계 (성능 데이터 분석): LLM은 정기적으로 OPENMARU APM의 트랜잭션 데이터를 분석합니다. 분석 중, `/report/monthly/generate` URL의 평균 응답시간이 지속적으로 3초를 초과하며 전체 시스템의 스레드를 과도하게 점유하는 경향을 발견합니다.
2. 2단계 (정책 제안): LLM은 이 문제를 단순한 경고로 알리는 대신, 구체적인 해결 방안을 담은 정책을 운영자에게 선제적으로 제안합니다.
3. 3단계 (운영자 승인): 운영자는 LLM이 제시한 문제 분석과 제안된 정책의 타당성을 검토한 후, ‘승인’ 버튼을 클릭하거나 “제안을 수락해”와 같은 자연어로 승인 의사를 전달합니다.
4. 4단계 (명령 변환 및 실행): 운영자의 승인이 확인되면, AI API 서버는 ’APM툴’을 호출합니다. 특허(NP25073-KR)에 명시된 바와 같이, APM툴은 “이 URL은 2초 이내로 제한하

© 2025 MSAP.ai., All Rights Reserved.

라"는 자연어 기반의 정책을 WAS 에이전트가 이해할 수 있는 기계적인 트래픽 제어 명령(예: JSON 형식의 룰셋)으로 변환합니다. 이 명령은 APM 서버를 통해 해당 URL을 처리하는 모든 WAS의 에이전트로 전송되어 실시간으로 적용됩니다.

5. 5단계 (결과 보고 및 모니터링): 정책이 성공적으로 적용되면, LLM은 운영자에게 최종 결과를 보고하고 후속 조치를 안내합니다.

지금까지 OPENMARU Cluster와 APM, LLM을 결합한 다양한 적용 사례를 살펴보았습니다. 마지막으로, 이 솔루션이 다른 표준 기술 및 경쟁 솔루션과 어떻게 연동되고 차별화되는지 비교 분석하겠습니다.

8.3 다른 관측성 스택과의 연동

현대적인 IT 환경에서 단일 솔루션이 모든 것을 해결하는 '만능 열쇠'가 되기는 어렵습니다. 대신, OpenTelemetry와 같은 개방형 표준과 생태계를 통해 다른 시스템과 유연하게 연동하는 능력이 솔루션의 가치를 결정합니다. 본 섹션에서는 제안된 LLM 기반 통합 관제 시스템이 개방형 표준과 어떻게 상호 작용하고, 다른 상용 솔루션과 비교하여 어떤 차별점을 가지는지, 그리고 국내 공공 시장에서의 특수성을 어떻게 활용하는지 분석합니다.

8.3.1 OpenTelemetry 기반 트레이스와의 상호 참조

OpenTelemetry는 특정 벤더에 대한 종속성을 줄이고 로그, 메트릭, 트레이스 같은 관측성 데이터를 표준화하는 클라우드 네이티브 환경의 핵심 기술입니다. OPENMARU APM이 제공하는 깊이 있는 트레이스 데이터와 OpenTelemetry의 표준 트레이스를 LLM이 어떻게 결합하여, 마이크로서비스 환경 전반에 걸친 엔드투엔드(End-to-End) 가시성을 확보하는지 살펴보겠습니다.

두 데이터의 상호 참조는 컨텍스트 전파(Context Propagation)를 통해 이루어집니다. OPENMARU APM 에이전트가 수집한 트랜잭션 정보에 OpenTelemetry 표준인 `trace_id`와 `span_id`를 속성으로 포함시키거나, 반대로 OpenTelemetry로 계측된 스팬(span)의 속성(attribute)에 OPENMARU가 관리하는 `session_id`와 `user_id`를 주입하는 방식으로 상호 연결 고리를 만듭니다.

이러한 연결을 통해 운영자는 LLM에게 다음과 같은 통합 질의를 할 수 있습니다.

© 2025 MSAP.ai., All Rights Reserved.

"OpenTelemetry trace ID `abcdef123456`와 관련된 OPENMARU APM의 세션 정보와 사용자 식별자를 알려줘."

이러한 연동을 통해 얻을 수 있는 가치는 명확합니다.

- 통합된 가시성: 사용자의 브라우저 클릭(OpenTelemetry 프론트엔드 계측)부터 백엔드 WAS의 DB 쿼리(APM 에이전트 계측)까지 끊김 없는 단일 트랜잭션 추적이 가능해집니다.
- 컨텍스트 보강 (Contextual Enrichment): 표준 OpenTelemetry 트레이스에 비즈니스적으로 중요한 세션 및 사용자 컨텍스트가 더해져, 추상적인 성능 데이터가 특정 사용자 경험에 대한 실행 가능한 인사이트로 전환됩니다.
- 투자 보호: 기업이 이미 OpenTelemetry 기반으로 구축한 관측성 파이프라인과 수집기를 그대로 활용하면서, OPENMARU 솔루션이 제공하는 세션 연계 분석 및 제어의 강점을 추가할 수 있습니다.

이처럼 개방형 표준과의 연동성을 확보하는 한편, 시중의 다른 LLM 기반 모니터링 기능과 비교했을 때 어떤 차별점을 갖는지 다음 절에서 비교해 보겠습니다.

8.3.2 Elastic/New Relic 등 LLM 모니터링 기능과의 비교 관점

최근 많은 관측성(Observability) 솔루션들이 LLM 관련 기능을 앞다투어 출시하고 있습니다. 하지만 이들 기능의 목적과 OPENMARU가 제안하는 모델 사이에는 근본적인 차이점이 존재합니다. 경쟁 솔루션 대부분은 'LLM을 사용하는 애플리케이션'을 모니터링하는 데 초점을 맞추는 반면, OPENMARU의 접근 방식은 'LLM을 활용하여' 기존 시스템을 운영하고 제어하는 데 중점을 둡니다.

두 접근 방식의 핵심적인 차이점은 다음과 같습니다.

관점	Elastic, New Relic 등의 LLM 모니터링	OPENMARU의 LLM 연계 (VibeOps)
주요 목적	LLM을 사용하는 애플리케이션을 모니터링 (예: 프롬프트, 응답, 비용 추적)	LLM을 활용하여 기존 웹 애플리케이션을 운영 및 제어

© 2025 MSAP.ai., All Rights Reserved.

관점	Elastic, New Relic 등의 LLM 모니터링	OPENMARU의 LLM 연계 (VibeOps)
분석 대상	LLM 애플리케이션의 성능 (LLM 자체의 응답시간, 토큰 사용량 등)	웹 애플리케이션의 성능 (동시 접속자수, URL 응답시간, CPU 사용률 등)
LLM의 역할	분석 및 관찰의 대상	분석 및 제어의 주체
사용자	LLM 기반 서비스를 개발/운영하는 개발자	기존 엔터프라이즈 시스템을 운영하는 IT 운영자

경쟁 솔루션들이 'LLM 옵저버빌리티'라는 특정 영역에 집중하는 동안, 본 솔루션은 특허(NP25073-KR) 기반 아키텍처를 통해 'LLM을 활용한 IT 스택 전체의 운영 자동화' 라는 훨씬 더 크고 핵심적인 엔터프라이즈의 과제를 해결합니다. 이는 단순 모니터링을 넘어 시스템과 대화하고 제어하는 VibeOps 패러다임의 구체적인 구현체이며, 시장에서 독보적인 비전을 제시합니다.

이러한 기술적 차별점과 더불어, 국내 공공 및 조달 시장 환경에서 OPENMARU Cluster가 갖는 독특한 강점을 마지막으로 살펴보겠습니다.

8.3.3 공공·조달 환경에서의 OPENMARU Cluster 적용 시 차이점

공공 및 금융 기관이 상용 소프트웨어를 도입할 때는 기술적 성능 외에도 안정적인 국내 기술 지원 체계, 투명하고 신속한 구매 절차, 그리고 관련 규정 준수 여부를 매우 중요하게 고려합니다. 이러한 비기능적 요구사항은 외산 솔루션이 쉽게 충족하기 어려운 장벽이 되기도 합니다.

가장 큰 차별점은 OPENMARU Cluster가 조달청 디지털서비스몰에 정식으로 등록된 제품이라는 사실입니다. 이는 공공기관에서 복잡한 수의계약이나 입찰 절차 없이, 표준화된 프로세스를 통해 신속하고 투명하게 제품을 도입할 수 있음을 의미하며, 이는 다음과 같은 핵심적인 장점으로 이어집니다.

- 신속하고 투명한 도입: 조달청 디지털서비스몰을 통해 표준화된 절차로 구매가 가능하여, 솔루션 도입에 소요되는 시간과 행정 비용을 크게 절감할 수 있습니다.
- 안정적인 국내 기술 지원: 외산 솔루션과 달리, 국내에 위치한 개발사가 직접 기술 지원과 유지보수를 제공합니다. 이는 문제 발생 시 언어 장벽 없이 신속하고 책임감 있는 대응을 보

© 2025 MSAP.ai., All Rights Reserved.

장하며, 공공 서비스의 연속성과 안정성 확보에 필수적인 요소입니다.

- 이기종 WAS 및 레거시 환경 지원: OPENMARU Cluster는 Tomcat, JBoss 등 다양한 상용 및 오픈소스 WAS를 지원하며, 여러 종류의 WAS가 혼재된 환경에서도 세션 클러스터링을 구현할 수 있습니다. 이는 다양한 시스템이 복잡하게 얽혀있는 공공기관의 IT 환경에 최적화된 유연성을 제공합니다.
- 검증된 안정성: 이미 다수의 공공기관 및 금융권에서 도입하여 운영 효율성과 안정성을 입증한 솔루션으로, 새로운 기술 도입에 따르는 리스크를 최소화할 수 있습니다.

결론적으로, OPENMARU Cluster는 뛰어난 기술력뿐만 아니라 국내 공공·조달 환경에 최적화된 구매 용이성과 신뢰도 높은 기술 지원 체계를 갖춤으로써, 공공 부문의 클라우드 네이티브 전환과 지능형 운영 도입을 위한 가장 확실하고 안정적인 선택지 중 하나입니다.

References & Links

1. OPENMARU Cluster – 제품 소개 페이지

   https://www.openmaru.io/product/openmaru-cluster/

2. OPENMARU APM – 제품 소개 페이지

   https://www.openmaru.io/product/openmaru-apm/

3. OPENMARU iAP – 인텔리전트 애플리케이션 플랫폼

   https://www.openmaru.io/product/openmaru-iap/

4. OPENMARU 공식 사이트 (제품군 통합 정보)

   https://www.openmaru.io/

5. MSAP.ai – AI 기반 MSA 플랫폼 소개

   https://www.msap.ai/msap-ai/

6. 특허 명세서 NP25073-KR – “LLM 연동형 AI API 서버·세션서버·APM서버 통합 구조” (국가 지식재산권정보서비스 KIPRIS 등록 정보)

© 2025 MSAP.ai., All Rights Reserved.

https://www.kipris.or.kr/

7. CNCF OpenTelemetry Project (관측성 데이터 표준 스키마)

   https://opentelemetry.io/

8. Kubernetes 공식 문서 – Pod, Service, Deployment 구조

   https://kubernetes.io/docs/concepts/

9. Redis 공식 문서 – 세션 저장소 및 일관성 모델(Eventual Consistency)

   https://redis.io/docs/latest/develop/data-types/

10. In-Memory Data Grid (IMDG) 기술 백서 – Hazelcast 공식 사이트

    https://hazelcast.com/resources/

11. OpenAI API Reference – LLM Prompt Engineering 기본 원칙

    https://platform.openai.com/docs/guides/prompt-engineering

12. Cloud Native Computing Foundation (CNCF) 공식 사이트

    https://www.cncf.io/

13. Spring Boot & APM Integration Guide (Spring 공식 문서)

    https://docs.spring.io/spring-boot/docs/current/reference/html/actuator.html

14. W3C HTTP Specification (HTTP Method, Query Parameter 정의)

    https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

15. DigitalServiceMall (조달청 디지털서비스몰) – OPENMARU Cluster 등록 정보

    https://www.digitalservice.kr/

© 2025 MSAP.ai., All Rights Reserved.

Contact Us

02-6953-5427

hello@msap.ai

www.msap.ai

MSAP.ai Blog

최신 기술 트렌드와
유용한 팁들을 가장 먼저
만나보세요.

MSAP.ai eBook

이제 나도 MSA 전문가
개념부터 실무까지

YouTube

클라우드 기반 기술과
인프라 전략을 다루는
전문 채널

MSAP.ai

엠에스에이피닷에이아이 | MSAP.ai

전화 : (02) 6953 - 5427
팩스 : (02) 469 - 7247
메일 : hello@msap.ai

© 2025 MSAP.ai. All rights reserved.