MSAP.ai

msap.ai

APM

Application Performance Management

APM 기반 AIOps 장애 분석 데모

PROJECT CODE	DATE
APM-AIOPS-V04	2026.08

CONTENTS

Agenda

전 4개 섹션 · 본문 13장

KEY FOCUS 알람에서 멈추지 않고 원인과 조치까지 — APM · LLM · MCP 연계 시연

SECTION 01

장애 시나리오 데모

외부 서비스 지연 · 외부 API 에러 · DB 지연 — SHOW CASE 01–03

외부 서비스 지연에 따른 장애

AIOPS Boost　WAS 장애 발생

APM 기반 AIOPS 서비스

운영자 질문

"지금 장애 원인 분석해줘"

자동 분석 지표

TPS · 응답시간 · 오류율 · Apdex 를 즉시 산출

상세 분석

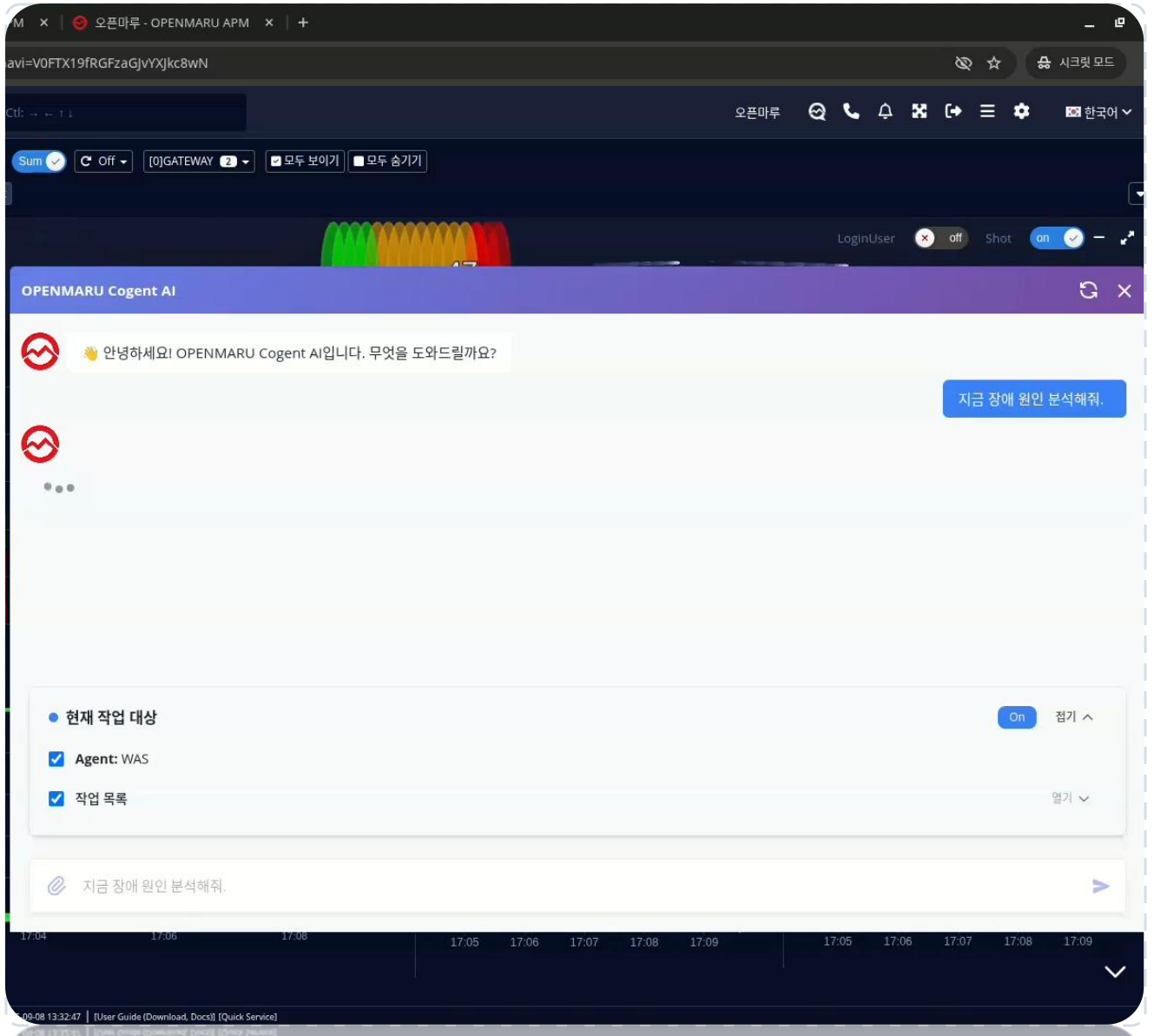
특정 트랜잭션의 처리 시간과 Trace 정보 제공
외부 API 지연까지 탐지해 병목 지점 자동 식별

통합 분석 구조

MSA 기반 MCP 구조로 모니터링 데이터 · 로그 ·
외부 연동 시스템 정보를 통합 분석

즉각 조치 제안

외부 API 점검 · WAS 타임아웃 설정 검토
원인–대응 과정을 빠르게 수행



질문 네 번으로 끝나는 WAS 트러블슈팅

APM 에 축적된 성능 데이터와 로그를 LLM · MCP(Model Context Protocol)로 연결 — 자연어 질문에 원인 분석과 실행 방안 제시

1 장애 탐지 및 요약

"지금 발생한 장애 원인을 분석해줘."

AI 자연어 처리

질문을 장애 분석 요청으로 변환 후 TPS · 응답시간 · Apdex 를 MCP 로 호출

AI 답변

"특정 구간에서 응답시간이 50ms → 2800ms 로 급증, Apdex 하락. 외부 API 호출 지연."

2 상세 성능 분석

"응답시간이 왜 이렇게 길어졌는지 알려줘."

AI 복합 분석

정상 구간과 장애 구간의 성능 패턴 비교, 오류율 · TPS · Trace 동시 분석

AI 답변

"/slowTest1 트랜잭션의 외부 HTTP 호출(20초)이 병목. 내부 WAS 는 정상 처리."

3 병목 지점 식별

"어느 트랜잭션이 문제인지 지정해줘."

AI 동적 분석

트랜잭션 Trace 추적, 외부 연동 API 응답 속도 분석

AI 답변

"'/slowRandom API' 응답 지연이 직접 원인. 외부 호출이 전체 지연의 대부분."

4 실행 권장사항 제시

"지금 어떤 조치를 취해야 해?"

AI 실행 가이드

외부 API 상태 점검 · 부하 확인 / WAS 외부 호출 타임아웃 검토 / 지표 추가 모니터링

AI 답변

"즉시 외부 시스템 상태 확인 및 타임아웃 조정 필요. 추가 모니터링 이벤트를 설정하세요."

26 외부 API 에러에 따른 장애 발생

AIOPS Boost | WAS 장애 발생

APM 기반 AIOPS 서비스

운영자 질문

"에러율이 올라가는데 원인을 분석해줘"

감지된 상황

이상 지표

60~65%

에러율 급등

이상 지표

35~45

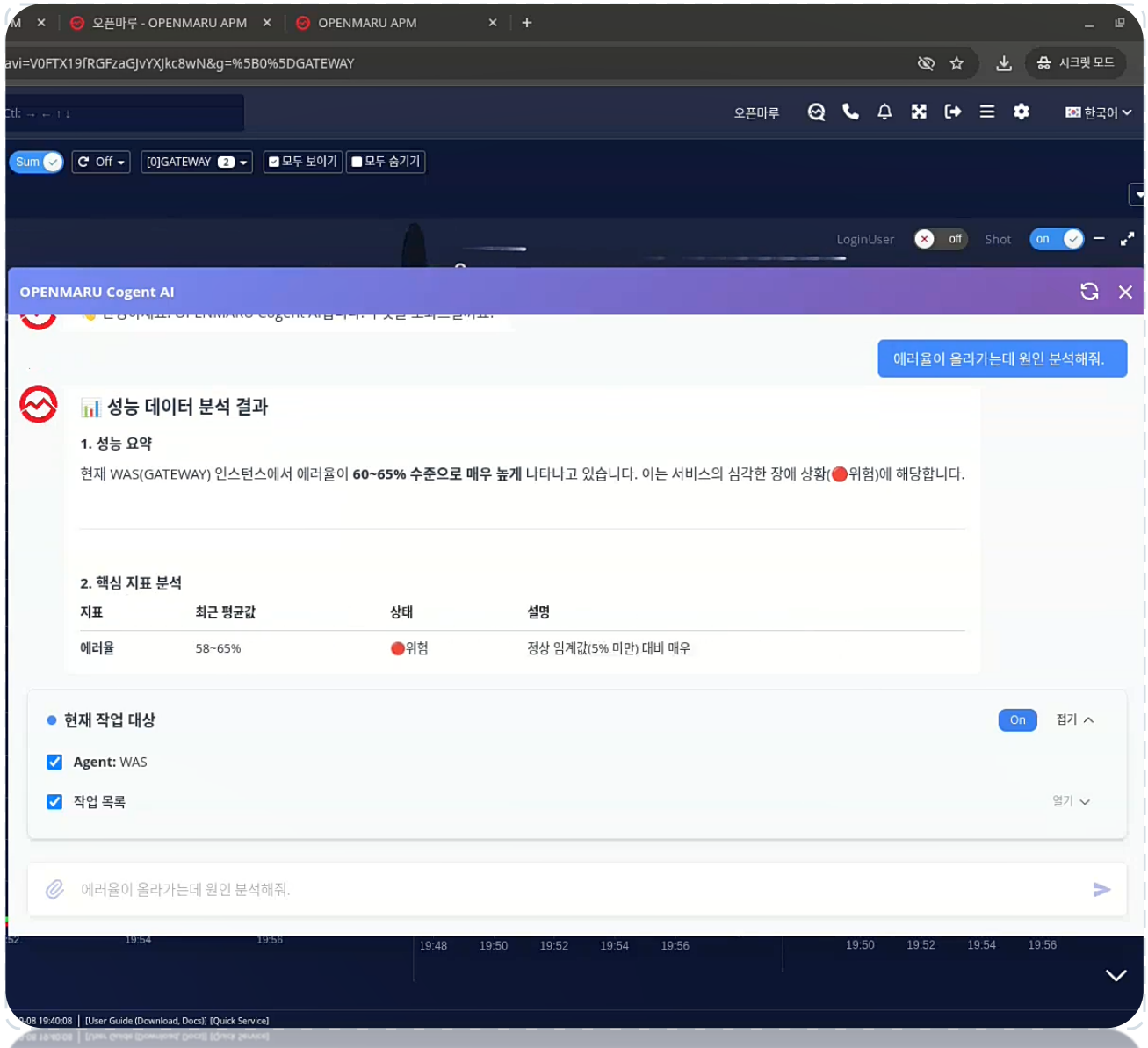
Apdex 급락

정상 범위로 확인된 항목

TPS · 평균 응답시간 · Heap 사용률

원인 특정

WAS 자체 자원 문제가 아닌
외부 API 장애(HTTP 500)



알람만 주던 자리에 원인과 조치를 함께

외부 연동 API 장애 · 자원 부족 · 네트워크 오류 — 기존 방식은 단순 알람만 제공해 로그와 지표 분석은 운영자의 몫

1 운영자 질문

"에러율이 급격히
올라가는데 원인 분석해줘."

→

2 AI 데이터 수집

APM 지표 수집 →
외부 API 호출 상태 점검 →
에러 유형 통계 분석

→

3 자동 원인 분석

LLM 기반 AIOPS 가
/errorTest 트랜잭션의 외부
API 오류(HTTP 500)로 특정

→

4 운영자에게 제공

외부 API 점검
에러 핸들링 로직 강화
네트워크 상태 추가 점검

서비스 효과

데이터 수집

APM 지표(Apdex · TPS ·
응답시간 · Heap)와 외부 API
호출 로그를 자동 연계

신속한 원인 파악

에러율 급등 · 외부 API 장애
여부 · 네트워크 상태를
실시간으로 확인

종합적 자동 분석

단순 지표 확인이 아닌
트랜잭션 추적 · 에러 유형 ·
분산 Trace 기반 원인 도출

대응 시간 단축

운영자가 로그를 직접 분석하지
않고, AI 의 원인–대응 초안으로
즉시 조치

1/10

장애 대응 시간을 1/10 수준으로 단축

운영자는 단순 알람이 아닌 AI 의 원인–대응을 받아 즉시 실행 가능
복잡한 모니터링 지표 속에서도 핵심 원인만 추출해 가용성과 운영 효율성을 향상

28 DB 관련 이슈로 인한 장애

AIOPS Boost　WAS 장애 발생

운영자 질문
"현재 발생하는 지연현상 분석해줘"

분석 결과

JDBC 요청 중 sleep(5) 함수가 포함된 쿼리 실행

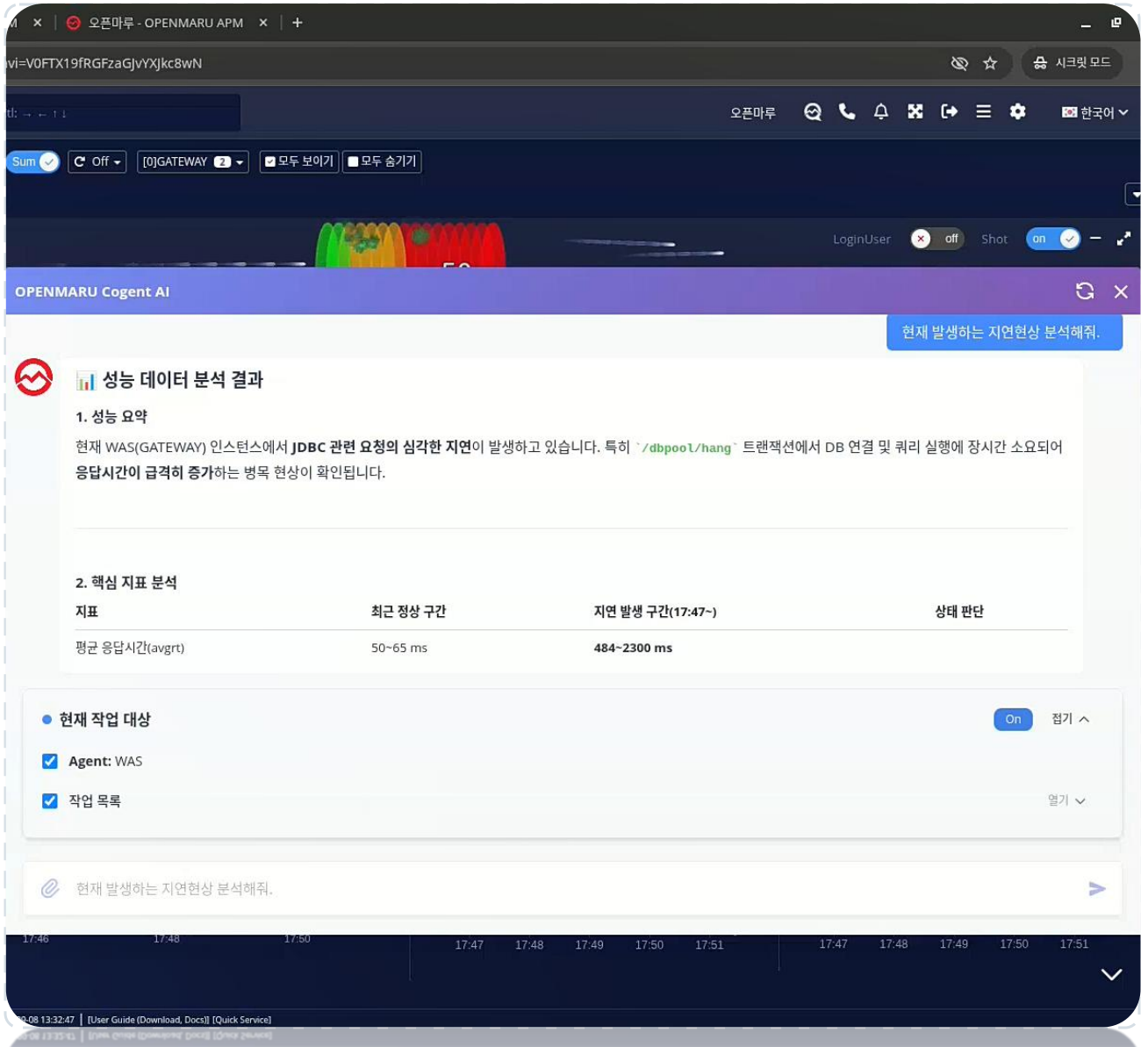
평균 응답시간
500ms → **2,300ms**
약 4.6배 급상승

HikariCP 커넥션 획득
24초 이상
커넥션 풀 고갈 · DB 연결 지연

즉시 조치
쿼리 점검
커넥션 풀 설정 확인
DB 상태 점검

개선 방안
풀 모니터링 강화
불필요한 쿼리 제거
트랜잭션 자동 알림 설정

CogentAI 결론
TPS · 오류율 · Apdex 를 자동 분석해 DB 커넥션 풀 고갈을 원인으로 특정하고 조치와 개선안을 동시 제시



지연의 출처를 쿼리 단위까지 좁히는 세 번의 질문

운영자가 자연어로 질의하면 성능 지표(TPS · 응답시간 · 오류율 · Apdex)를 자동 분석하고 DB 지연 원인을 추적해 대응 방안 제시

자연어 처리 기반 장애 분석 예시

Q1 "현재 발생하는 지연현상 분석해줘."

자연어 처리 → 성능 데이터 수집 → APM 분석 → 병목 구간 식별

질문을 성능 데이터 분석 요청으로 변환하고 TPS · 오류율 · 응답시간 이상을 즉시 확인

Q2 "DB 쿼리 때문에 장애가 발생했는지 확인해줘."

JDBC 지연 추적 → 커넥션 풀 상태 분석 → 문제 쿼리 탐지

DB 커넥션 풀 고갈 · 지연 쿼리 · 외부 서비스 장애 등 원인을 자동 추적

Q3 "지금 바로 조치할 방법을 알려줘."

즉시 조치	불필요한 쿼리 제거, 커넥션 풀 파라미터 점검
개선 제안	풀 모니터링 강화, 자동 알림 설정, 쿼리 최적화

주요 기능 및 효과

자연어 처리

운영자의 질문을 성능 데이터 분석 요청으로 변환
쿼리 문법이나 도구 사용법을 몰라도 사용 가능

실시간 성능 분석

APM 수집 데이터와 연계해 TPS · 오류율 ·
응답시간의 이상 여부를 즉시 확인

자동 병목 탐지

DB 커넥션 풀 고갈 · 지연 쿼리 · 외부 서비스
장애 등 원인을 자동 추적

조치 가이드 제시

즉시 대응(쿼리 점검 · 커넥션 풀 설정 확인)과 장기 개선 방안을 동시 제공

SECTION 02

T-Map 장애 패턴 감지

히트맵 모양 기반 장애 유형 자동 진단 — DEMO 01 · APM

T-Map 의 장애 패턴을 시스템이 먼저 지목

현장의 문제

운영 중에는 수천~수만 건의 트랜잭션이 쏟아집니다. "무엇이, 어떤 유형으로" 잘못되는지 표로 찾기는 느리고 놓치기 쉽습니다. 특히 쿼리 타임아웃은 징후가 여러 트랜잭션에 흩어져 직관적으로 보이지 않습니다.

표를 훑는 방식으로는 유형별 진단에 도달하기 어려움

모양을 보고 자동 진단

시간 × 응답시간 평면에 그린
히트맵

장애 유형마다 고유한 시각
패턴 → 원인 방향 즉시 파악

T-Map 히트맵

16종 패턴 자동 분류

T-Map 의 '모양'만으로 장애를
16종 패턴으로 자동 분류

세로줄 · 느린 응답 · 파도치기 ·
단계적 성능 저하 · 연쇄 장애 등

유형 판정 자동화

타임아웃 진단

DB 지연 · 타임아웃 특유의
패턴을 자동 감지

→ 느린 SQL · 외부 호출까지
자동 진단으로 연결

진단 체인 자동 연결

핵심

"트랜잭션의 문제 패턴을 시스템이 먼저 짚어 줍니다."

트랜잭션 트리의 자동 진단과 AI — 표를 훑지 않아도 유형이 먼저 드러남


1
order12 ···
0
order13 ···
1
product11 ···
0
product12 ···
2
product13 ···
위험 패턴감지
주의 세로줄
특정 순간에 응답시간이 위아래로 한꺼번에 퍼짐 — 메모리 정리 멈춤 / 데이터베이스 락 의심
심각 5xx 군집 (느린 응답)
서버 오류 다발 — 응답이 느림 (응답 대기 초과 연쇄 / 연결 자원 부족 의심)
AI 즉시 AI 버튼을 클릭하여 진단
활성 사용자
168
정보
사용자 만족도
99.8%
TPS
357.3
오류율
0.2%
평균 응답시간
8ms
TPS
515
사용자 만족도
97.4
35.9
평균 응답시간
304.1
오류율
1.6
2,722

SECTION 03

APM 연계 AIOps 활용

지연 트랜잭션 분석 · 사용자 활동 추적 · 보고서 자동 생성

32 APM 연계 지연 트랜잭션 원인 분석

AIOPS Boost | APM | Kubernetes AIOPS 서비스

운영자 질문

"[homepage] 애플리케이션에 문제는 없는지 현재 상황을 분석해줘"

분석 결과

/api/orders/statistics 트랜잭션이 6초 이상 지연되며, 그 원인이 느린 SQL 과 JDBC 커넥션 풀 고갈임을 특정

AI 분석을 실측으로 검증하는 3단계

원인 분석

JDBC 실행 시간이 전체 지연의 대부분(6,651ms)임을 지목

중첩 SELECT 풀 스캔과 커넥션 풀 10개 전량 점유 확인

개선 방안

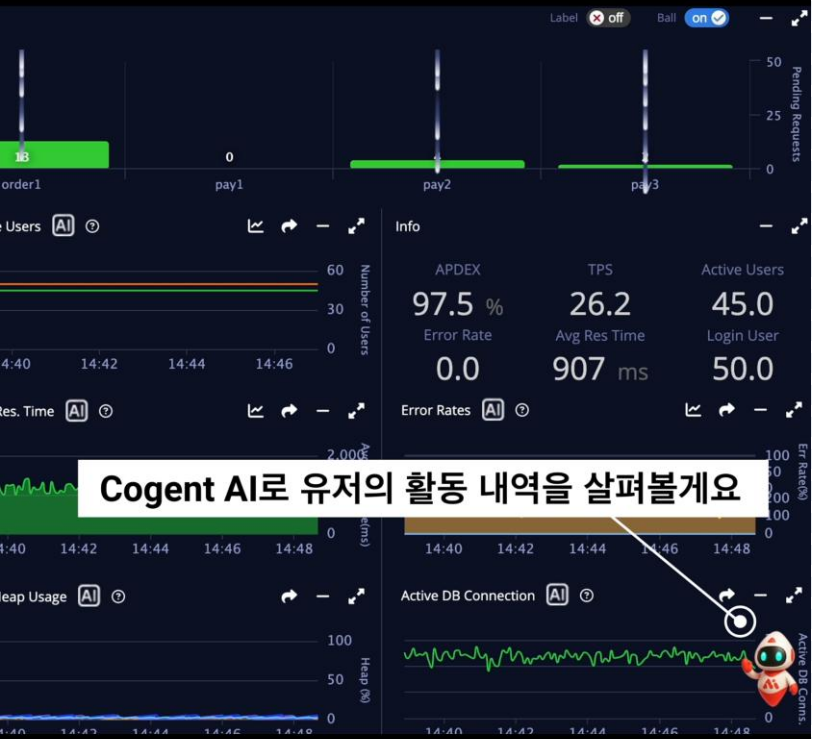
인덱스 추가, 서브쿼리 JOIN 전환, 커넥션 풀 15~20 확장 제시

영역 · 조치 · 기대 효과를 표로 정리해 3~5초 이하 단축 목표

실측 검증

트랜잭션 추적에서 Duration 6,714ms 중 SQL Time 6,711ms 확인

콜 트리로 AI가 지목한 쿼리가 실제 호출됨을 증명



33 APM 기반 접속 사용자 활동 추적

AIOPS Boost | APM | Kubernetes AIOPS 서비스

운영자 질문

"지금 접속한 사용자와 활동 내역을 보여줘"

처리 방식

APM 세션 · 트랜잭션 데이터를 조회해 사용자가 어떤 화면과 API 를 어떤 순서로 호출했는지 정리

접속자 수 나열을 넘어선 세 가지

활동 내역

세션 ID · 로그인 시각 · 요청 경로를 타임라인으로 제시

사용자 한 명의 여정을 시간순으로 재구성

성능 진단

트랜잭션 응답 시간과 오류 구간을 함께 표시해 지연 지점 식별

어느 화면에서 느려졌는지가 바로 보임

운영 활용

이상 호출 패턴 탐지와 사용자 기준 장애 재현 지원

사용자 관점의 서비스 상태를 즉시 파악

34 LLM 기반 APM 보고서 자동 생성과 출력

AIOPS Boost | APM | OPENMARU APM AIOPS 서비스

운영자 질문

"APM 성능 데이터를 정리해 보고서로 만들어줘"

자동 취합

APM 이 수집한 응답시간 · 처리량 · 에러율 · 자원 사용률을 자동 취합하고,
LLM 이 해석해 사람이 읽을 수 있는 보고서로 작성

분석 · 요약

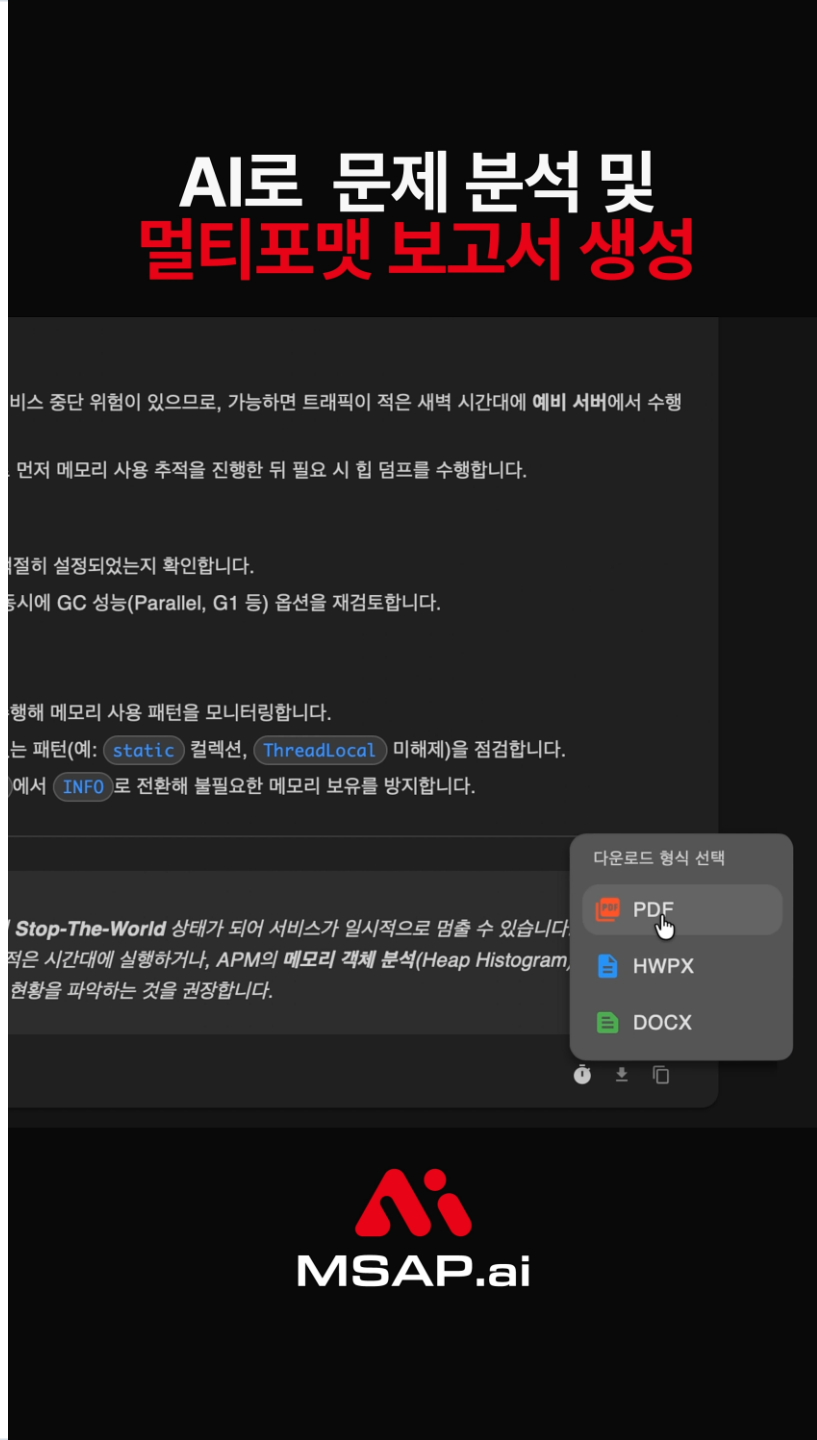
임계치 초과 구간과 이상 징후를 식별하고
성능 저하의 추정 원인을 서술

개선 권고

GC 설정 · 메모리 사용 패턴 · 로그 레벨 등
구체적인 조치 사항 제시

멀티포맷 출력

완성된 보고서를 PDF · HWPX · DOCX 형식으로 즉시 다운로드
수작업 집계나 문서 편집 없이 운영 · 경영 보고용 보고서 완성



SECTION 04

느린 쿼리 진단과 개선

실행 계획 해석부터 수정 제안까지 — DEMO 02 · APM

느린 쿼리를 찾는 데서 끝내지 않고, 고치는 법까지

현장의 문제

개별 쿼리는 임계치 미만(8~18ms)이라 슬로우 감지에 안 잡히지만 N+1 · 대량 Fetch 로 전체가 느려집니다.
원인을 찾아도 실행 계획 해석과 인덱스 설계는 DBA 영역이라, 개발자가 매번 묻는 병목이 생깁니다.

감지 임계치 미만이라 잡히지 않고, 잡아도 해석은 다른 사람의 일

자동 진단

트랜잭션을 여는 즉시 느린 SQL
Top N(100ms 초과 정렬 · 비율)을
상단에 표시

N+1 · 대량 Fetch 포착

Top N 자동 정렬

AI 슬로우 SQL

버튼 클릭만으로 AI 에 전달
→ 실행 계획 해석 + 인덱스 추천
(스키마 · 통계 기준)

Oracle · PostgreSQL · MySQL · MSSQL

DBA 의존도 감소

쿼리 원형 보존

주석과 메타데이터를 그대로
AI 에 전달

쿼리의 의도까지 함께 읽혀
분석 정확도가 높아짐

정확도 향상

핵심

"느린 쿼리를 찾는 것에서 끝나지 않고, 쿼리 플랜을 AI 가 분석합니다."

왜 느린지, 어떻게 고칠지까지 — 탐지에서 처방으로

CogentAI

MSAP APM

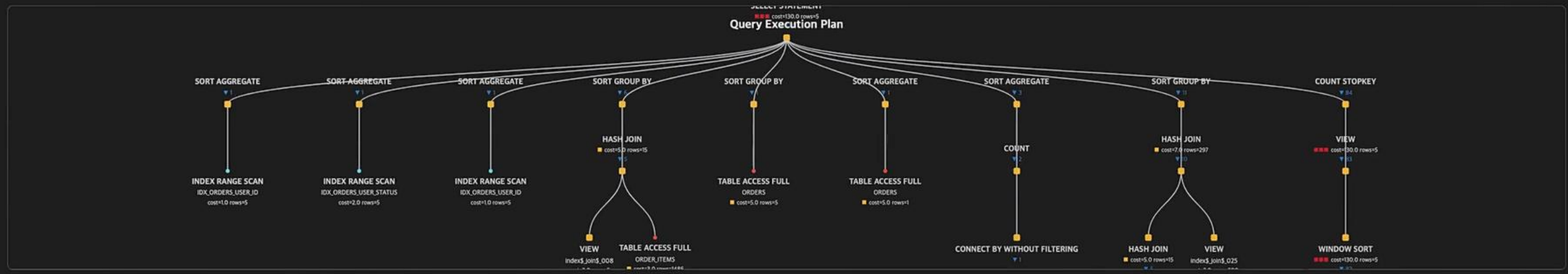
안녕하세요!

AIOps를 위한 CogentAI입니다. 무엇을 도와드릴까요?

느린쿼리 분석 요청:
SELECT base.* FROM (SELECT /*+ NO_PARALLEL */ CUSTOMER_ID, USERNAME, CUSTOMER_FULL_NAME, COMPANY_NAME, INDUSTRY, CUSTOMER_TIER, TOTAL_ORDER_COUNT, TOTAL_ORDER_AMOUNT, AVG_ORDER_AMOUNT, ...

CogentAI가 생각한 내용 (3,384 문자)

시각화 차트



분석 대상 SQL(Oracle):

```
SELECT base.*
FROM (
  SELECT /*+ NO_PARALLEL */ CUSTOMER_ID, USERNAME, CUSTOMER_FULL_NAME, COMPANY_NAME, INDUSTRY, CUSTOMER_TIER, TOTAL_ORDER_COUNT, TOTAL_ORDER_AMOUNT, AVG_ORDER_AMOUNT, COMPLETION_RATE, DISTINCT_PRODUCT_CNT, ACTIVE_ORDER_MONTHS, ORDER_STDEV, AVG_PRODUCT_MARGIN, ORDERS_LAST_30D, REGION_AVG_ORDER_AMT, CALL_VALUE, TOTAL_PO_COUNT,
TOTAL_PO_AMOUNT, TOTAL_PO_QTY, RANK() OVER ( ORDER BY TOTAL_ORDER_AMOUNT DESC NULLS LAST ) AS GLOBAL_ORDER_RANK, NTILE(5) OVER ( ORDER BY TOTAL_ORDER_AMOUNT DESC NULLS LAST ) AS ORDER_AMOUNT_QUINTILE, RANK() OVER ( PARTITION BY INDUSTRY ORDER BY COMPLETION_RATE DESC NULLS LAST ) AS INDUSTRY_COMPLETION_RANK, ROUND( TOTAL_ORDER_AMOUNT
/ NULLIF(SUM(TOTAL_ORDER_AMOUNT) OVER (), 0) * 100, 4 ) AS REVENUE_CONTRIBUTION_PCT, SYSDATE AS REPORT_GENERATED_AT
  FROM ( SELECT c.ID AS CUSTOMER_ID, u.USERNAME, u.FULL_NAME AS CUSTOMER_FULL_NAME, c.COMPANY_NAME, c.INDUSTRY, c.TIER AS CUSTOMER_TIER, ( SELECT COUNT(*) FROM ORDERS o WHERE o.USER_ID = u.ID ) AS TOTAL_ORDER_COUNT, ( SELECT SUM(o.TOTAL_AMOUNT) FROM ORDERS o WHERE o.USER_ID = u.ID ) AS TOTAL_ORDER_AMOUNT, ( SELECT AVG(o.TOTAL_AMOUNT)
FROM ORDERS o WHERE o.USER_ID = u.ID ) AS AVG_ORDER_AMOUNT, ROUND( ( SELECT COUNT( CASE WHEN o.STATUS = 'COMPLETED' THEN 1 END ) FROM ORDERS o WHERE o.USER_ID = u.ID ) / NULLIF( ( SELECT COUNT(*) FROM ORDERS o WHERE o.USER_ID = u.ID ), 0 ) * 100, 2 ) AS COMPLETION_RATE, ( SELECT COUNT(DISTINCT oi.PRODUCT_ID) FROM ORDERS o JOIN
ORDER_ITEMS oi ON oi.ORDER_ID = o.ID WHERE o.USER_ID = u.ID ) AS DISTINCT_PRODUCT_CNT, ( SELECT COUNT(DISTINCT TO_CHAR(o.CREATED_AT, 'YYYY-MM')) FROM ORDERS o WHERE o.USER_ID = u.ID ) AS ACTIVE_ORDER_MONTHS, ( SELECT ROUND(STDDEV(o.TOTAL_AMOUNT), 2) FROM ORDERS o WHERE o.USER_ID = u.ID ) AS ORDER_STDEV, ( SELECT ROUND(AVG(pr.PRICE -
pr.COST), 2) FROM ORDERS o JOIN ORDER_ITEMS oi ON oi.ORDER_ID = o.ID JOIN PRODUCTS pr ON pr.ID = oi.PRODUCT_ID WHERE o.USER_ID = u.ID ) AS AVG_PRODUCT_MARGIN, ( SELECT COUNT(*) FROM ORDERS o WHERE o.USER_ID = u.ID AND TRUNC(o.CREATED_AT) >= TRUNC(SYSDATE) - 30 ) AS ORDERS_LAST_30D, ( SELECT ROUND(AVG(o_r.TOTAL_AMOUNT), 2) FROM ORDERS
o_r JOIN USERS u_r ON o_r.USER_ID = u_r.ID JOIN CUSTOMERS c_r ON c_r.USER_ID = u_r.ID WHERE c_r.REGION_ID = c.REGION_ID AND o_r.STATUS = 'COMPLETED' ) AS REGION_AVG_ORDER_AMT, /* 신규 발주(PO) 테이블 2개 조인 연산 구문 */ ( SELECT COUNT(DISTINCT po.ID) FROM PURCHASE_ORDERS po JOIN PURCHASE_ORDER_ITEMS poi ON poi.PURCHASE_ORDER_ID = po.ID
JOIN ORDER_ITEMS oi_b ON oi_b.PRODUCT_ID = poi.PRODUCT_ID JOIN ORDERS o_b ON o_b.ID = oi_b.ORDER_ID WHERE o_b.USER_ID = u.ID ) AS TOTAL_PO_COUNT, ( SELECT SUM(po.TOTAL_AMOUNT) FROM PURCHASE_ORDERS po JOIN PURCHASE_ORDER_ITEMS poi ON poi.PURCHASE_ORDER_ID = po.ID JOIN ORDER_ITEMS oi_b ON oi_b.PRODUCT_ID = poi.PRODUCT_ID JOIN ORDERS
o_b ON o_b.ID = oi_b.ORDER_ID WHERE o_b.USER_ID = u.ID ) AS TOTAL_PO_AMOUNT, ( SELECT SUM(poi.QUANTITY) FROM PURCHASE_ORDER_ITEMS poi JOIN ORDER_ITEMS oi_b ON oi_b.PRODUCT_ID = poi.PRODUCT_ID JOIN ORDERS o_b ON o_b.ID = oi_b.ORDER_ID WHERE o_b.USER_ID = u.ID ) AS TOTAL_PO_QTY, ( SELECT SUM(ROWNUM * DBMS_RANDOM.VALUE) FROM DUAL
CONNECT BY LEVEL <= 500000 ) AS CALL_VALUE FROM CUSTOMERS c JOIN USERS u ON c.USER_ID = u.ID WHERE EXISTS ( SELECT 1 FROM ORDERS o WHERE o.USER_ID = u.ID AND o.TOTAL_AMOUNT > 0 ) )
  WHERE TOTAL_ORDER_AMOUNT IS NOT NULL
  ORDER BY TOTAL_ORDER_AMOUNT DESC
) base
WHERE ROWNUM <= 30 AND 1 = 1
```

Query Plan(Oracle):

쿼리 실행계획 트리 시각화

SELECT → JOIN → INDEX SCAN 연산 흐름을 그래프로 표현

메시지

CogentAI는 실수를 할 수 있습니다. 응답을 다시 확인해 주세요.