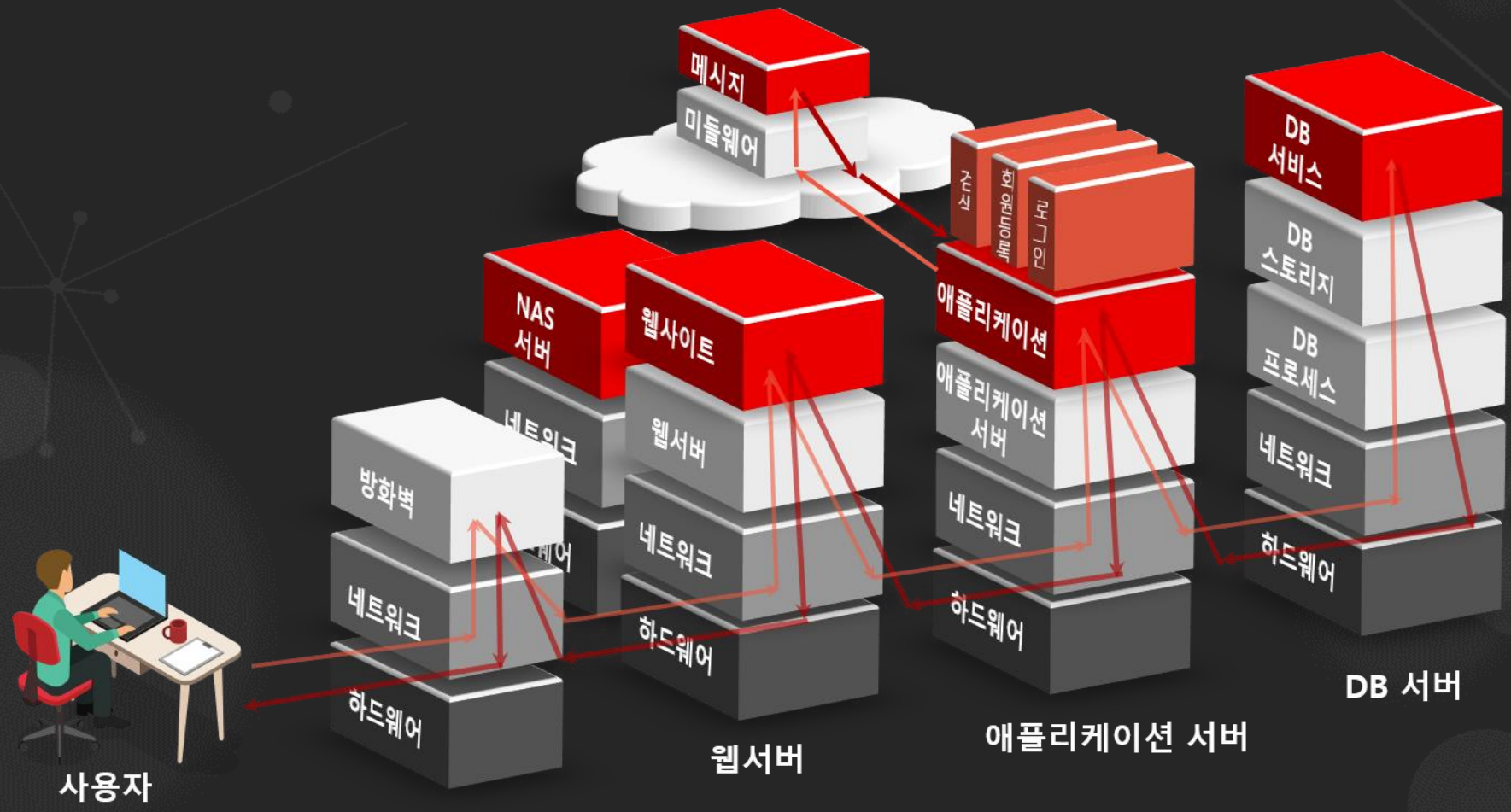
OPENMARU APM 소개
(Application Performance Management)

Application Performance Management

APM 이란?

미들웨어는 시스템 장애의 관문이자 시작점

- 데이터베이스가 50% 느려진다면 사용자 응답시간은 어떻게 될까요?
- DB 테이블 변경으로 SQL 에서 오류로 인하여 페이지가 오류가 난다면?



APM 필요성

개발팀
- 고성능 애플리케이션 개발
- 신속한 버그 픽스
- 이슈 재발 방지

openmaru APM

QA팀
- 부하테스트 시 성능 보장
- 서비스 오류 감지
- 최소 응답시간 유지

운영팀
- 성능에 대한 최적화와 서비스 정상 유/무
- 신속한 장애 감지와 원이 파악 그리고 조치
- Over/Under 사이징인가?

비즈니스 담당자
- 고객의 서비스 만족도는 충분한가?
- 고객 이탈 방지 및 브랜드 이미지 손상 방지
- 경쟁사 대비 성능은 충분한가?

3 Tier Architecture

- **어떤 이슈가 가장 난이도가 높은 문제 일까요?**
  - Network / Storage / Hardware 장애 발생
  - OS / Database / 웹서버 / WAS 에서 장애 발생
  - Application 장애 발생


EBS 온라인 클래스
APM01
[긴급] 비정상 상황 발생 – 대시보드
[심각-CRITICAL] 'Worker Usage %' (평균값: 98.9)이 심각(CRITICAL) 임계값 '95'을 넘었습니다.
발생에이전트 : apache@EBS-OC-PROD2-WEB05[172.17.11.35]
클릭하여 상세한 정보를 확인하세요.... 10s.
[심각-CRITICAL] 'Worker Usage %' (평균값: 99.95)이 심각(CRITICAL) 임계값 '95'을 넘었습니다.
발생에이전트 : apache@EBS-OC-PROD2-WEB02[172.17.11.5]
클릭하여 상세한 정보를 확인하세요.... 10s.
Request Viewer
115098.0
3115
Request Velocity
Label
Ball
Satisfying
Tolerating
Frustrating
APDEX
APDEX 93.3 %
TPS 1071.8
Error Rate 0.1 %
Average Response Time 2,884 ms
TPS
Active Users
Transaction Heatmap (T-Map)
Avg. Res. Time
Error Rates
JVM Heap Usage
Active DB Connection
openmaru APM


KHAN [apm]
한국어
홈 / WAS / 대시보드
정상 서비스 상황
WAS Dashboard
Off
Sum
Request Viewer
2034.0
96
Request Velocity
Label
power
Satisfying
Tolerating
Frustrating
Number of Requests
APDEX
Excellent
Good
Fair
Poor
Unacceptable
APDEX Score(%)
Info
APDEX
99.6
%
TPS
438.0
Active Users
1934.7
Errro Rate
0.0
Average Response Time
202
TPS
Active Users
Transaction Heatmap (T-Map)
Response Time (s)
Avg. Res. Time
Avg. Response Time(ms)
Error Rates
JVM Heap Usage
Heap (%)
Active DB Connection

화재 경보와 소방관

openmaru APM

APM을 통해 고객보다
먼저 장애와 성능 이슈 파악

WAS엔지니어를
통한 문제 해결


openmaru
APM
이것은
부채군요.
이것은
창이군요.
이것은
벽이군요.
이것은
나무네요.
이것은
끈이군요.
이것은
뱀이군요.

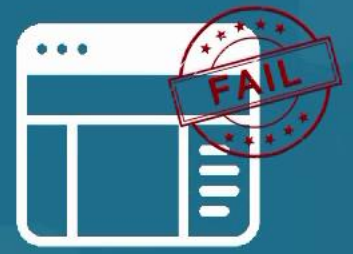
시스템 구축 후 - 고민들

openmaru APM

- ✓ 오늘 **온라인방문자 수가 급격히 줄었는데**요?
- ✓ 다음달에 **온라인 광고**를 해야 하는데 문제는 없겠조?
- ✓ **주기적으로 장애**가 발생하는데 대책은 없나요?
- ✓ 사용자가 조금 만 늘어도 **시스템이 다운이니 불안해서 뭘 할 수가 없네요.**

- ✓ 확인해 봤는데 **소스는 문제가 없어요**. WAS 문제 아닌가요?
- ✓ 느린 페이지에서 사용하는 **DB 쿼리를 보고 싶은데요.**
- ✓ 개발할 땐 빠른데 **운영시스템에서만 느린 이유**가 무엇인가요?

- ✓ 서비스가 갑자기 느려졌는데 **WAS 문제일까요?**
- ✓ 실시간 동시 사용자수나 TPS 등의 **기본 운영 정보를 알고 싶은데요.**
- ✓ 서비스가 느려질 때 **WAS를 Restart 하면 정상적으로 동작해요.**
- ✓ **특정페이지 나 서비스가 느린 이유는** 무엇일까요?

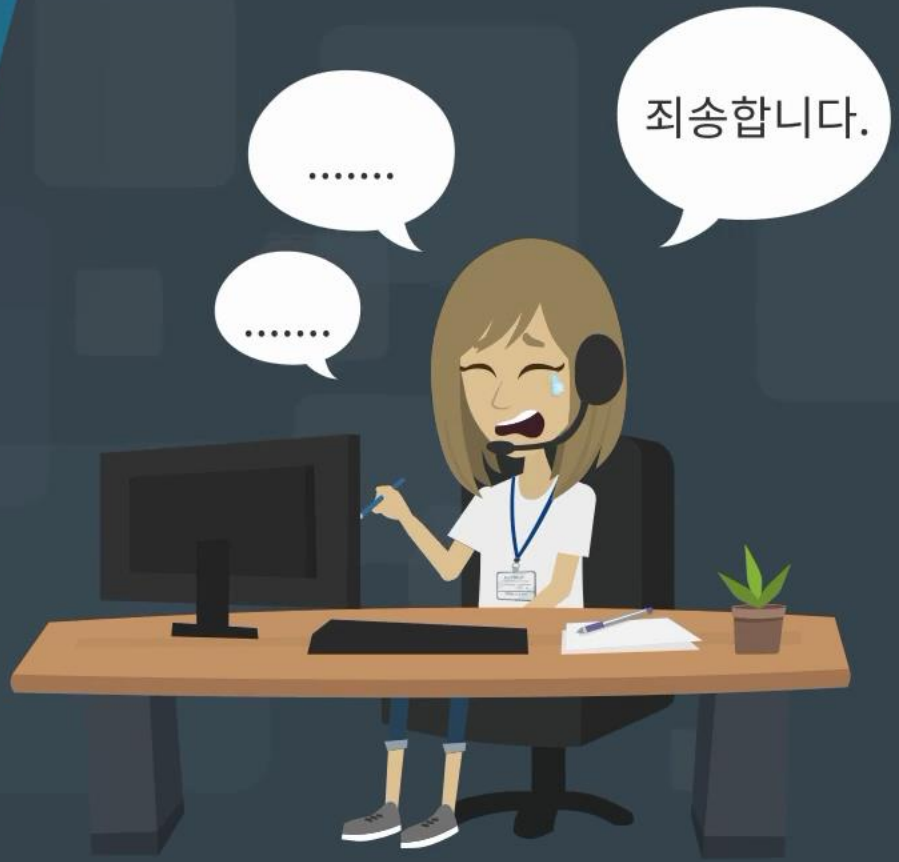


특정페이지 오류

느린 응답속도

검색 오류/ 페이지 찾기 오류

openmaru
APM




네트워크,서버 모두 잘 운영되고 있습니다.
심지어 어제보다도 성능이 좋은데요.
openmaru
APM
네트워크
OK
서버
OK
스토리지
OK

Log Monitoring vs. APM

openmaru APM

DAYS

With Logs

End User Complains → Operations Logs Ticket → Identify → Isolate → Repair

MINUTES

With APM

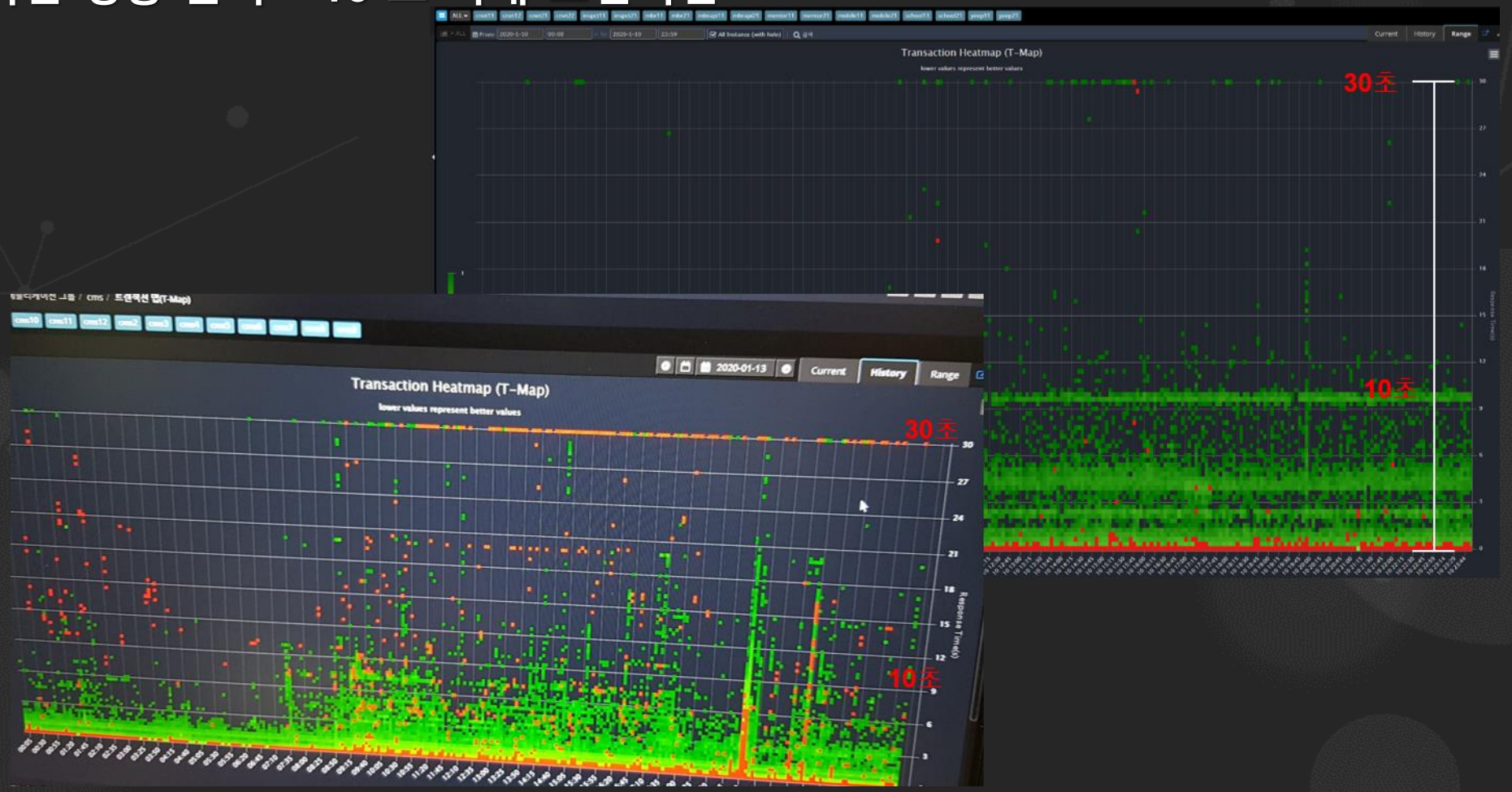
Operations Receive Alert → Identify → Isolate → Repair

Problem Starts

Business Impact

Application 안정화

- 지속적인 품질 관리 – 버그 픽스 (500 에러/ 400 에러)
- 지속적인 성능 관리 – 10 초 이내 트랜잭션



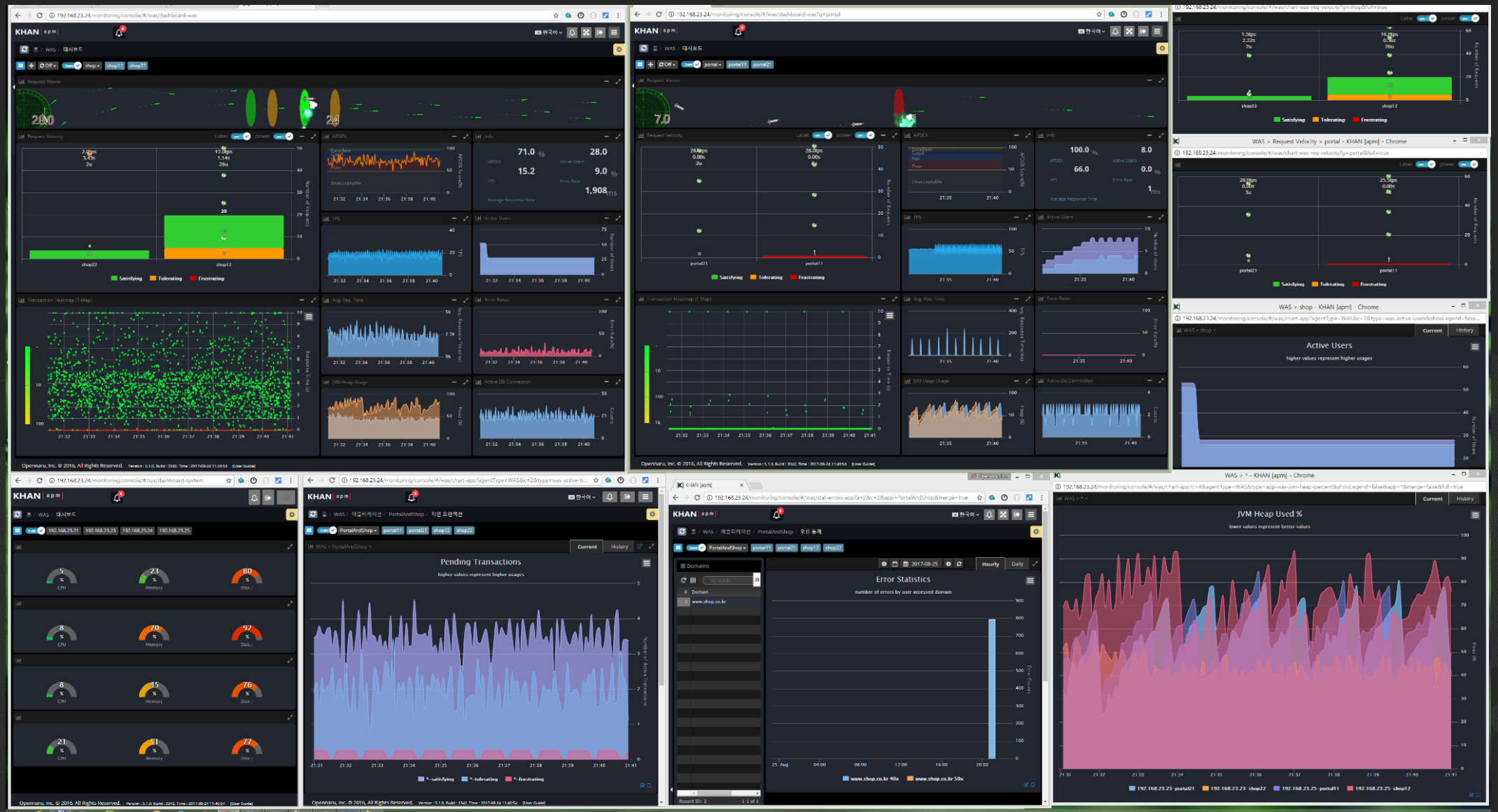

Micro Service Archtecture
OPENMARU APM 제품 개요
openmaru
APM

장애와 성능 이슈에 대한 즉각적인 대응 체계 수립

openmaru APM

OPENMARU APM – Method Tracer

OPENMARU APM :: 9 – HTML5



OPENMARU APM Use Case

openmaru

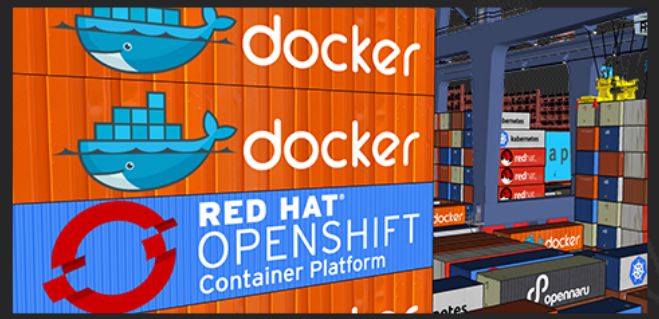
01

Unix To Linux 전환

amazon web services™

02

Amazon 클라우드 전환



03

Openshift PaaS 도입

Red Hat

04

Red Hat Business

국내 주요 실적

- 오픈나루는 최신 IT 기술인 Amazon AWS, Container 그리고 오픈소스 기반 환경에서 다양한 고객 레퍼런스를 가지고 있다.
- OPENMARU APM 은 도커. 클라우드 , 오픈소스 환경의 웹시스템 환경에서는 APM 제품 중 국내 점유율 1위 제품

구분	고객사	프로젝트 명	운영환경	수행시기
01 U2L전환	건국대학교병원	**병원정보시스템 주전산기 성능 모니터링**	Red Hat Linux	2017년
02 클라우드환경	adidas	**eCOM / POS 시스템 성능관리**	Amazon AWS	2018년
	Johnson&Johnson	MyACUVUE 모바일 및 홈페이지 서비스 성능관리	Amazon AWS	2018년
	KERIS 한국교육학술정보원	디지털교과서 플랫폼 서비스 클라우드	G-Cloud	2018년
03 PaaS환경	LOTTECARD	**Life Platform 프로젝트 WAS모니터링 솔루션**	OpenShift	2018년
	행정안전부	**온-나라 문서2.0 확산 및 고도화**	OpenShift	2018년
	SAMSUNG 삼성SDS	**SET시스템 PaaS 플랫폼**	SDS PaaS	2017년
	DOOSAN	PaaS IT Infra 모니터링	OpenShift	2017년
04 오픈소스환경	Finnq	**머니트레이너 시스템 (계정계, 정보계)**	RHEL/JBoss	2017년

SAMSUNG Finnq 해피머니 DOOSAN LOTTECARD SDU 행정안전부 국립환경과학원 우리은행 MEDIAWILL 건국대학교병원 KD 운송그룹 광명시 CJ KERIS SPC NETWORKS 한국환경공단 CareerNet 커리어넷 ACUVUE KEB 하나은행 수협은행 NPS 국민연금 HYUNDAI MOBIS 대한산업안전협회 현대해상 HF 한국주택금융공사 adidas LH 한국토지주택공사 DAEKYO 서울특별시교육청

다양한 WAS 환경 모니터링

모니터링 지원 WAS 환경

- **APM은 다양한 WAS(Web Application Server) 와 Java Daemon으로 실행되는 애플리케이션에 대해 모니터링을 지원해야 한다.**

▶ 모니터링 가능 한 WAS 목록

WAS

지원 WAS 버전 리스트

제품	버전
JBoss by Red Hat	• JBoss EAP 5 ~ 7

제품	버전
ORACLE WebLogic	• 9 버전 이상

제품	버전
IBM WebSphere	• 17 버전 이상

제품	버전
WildFly	• JBoss AS 5 • 8 버전 이상

제품	버전
Tmax JEUS	• 6 버전 이상

제품	버전
resin	• 3 버전 이상

제품	버전
Apache Tomcat	• 5.5 버전 이상

제품	버전
jetty://	• 8 버전 이상

제품	버전
spring boot	• 1 버전 이상

제품	버전
Java	• 1.5 버전 이상

왜 OPENMARU APM 인가?

01 모니터링 보다는 장애 진단과 성능 최적화에 집중

- 실시간성 Dashboard 도 중요하지만
- Troubleshooting 에 필요한 전문화된 도구의 필요성
- 지속적인 성능관리와 운영 최적화 지원

02 Cloud Ready Architecture

- 시계열 DB, NoSQL 기반의 저장 구조
- HTML5 / Web-Socket 기반으로 상호운영성 확보
- 클라우드 환경을 위한 Auto-Scaling/Scale-out 지원

03 개방형 플랫폼에서 점유율 1위 제품

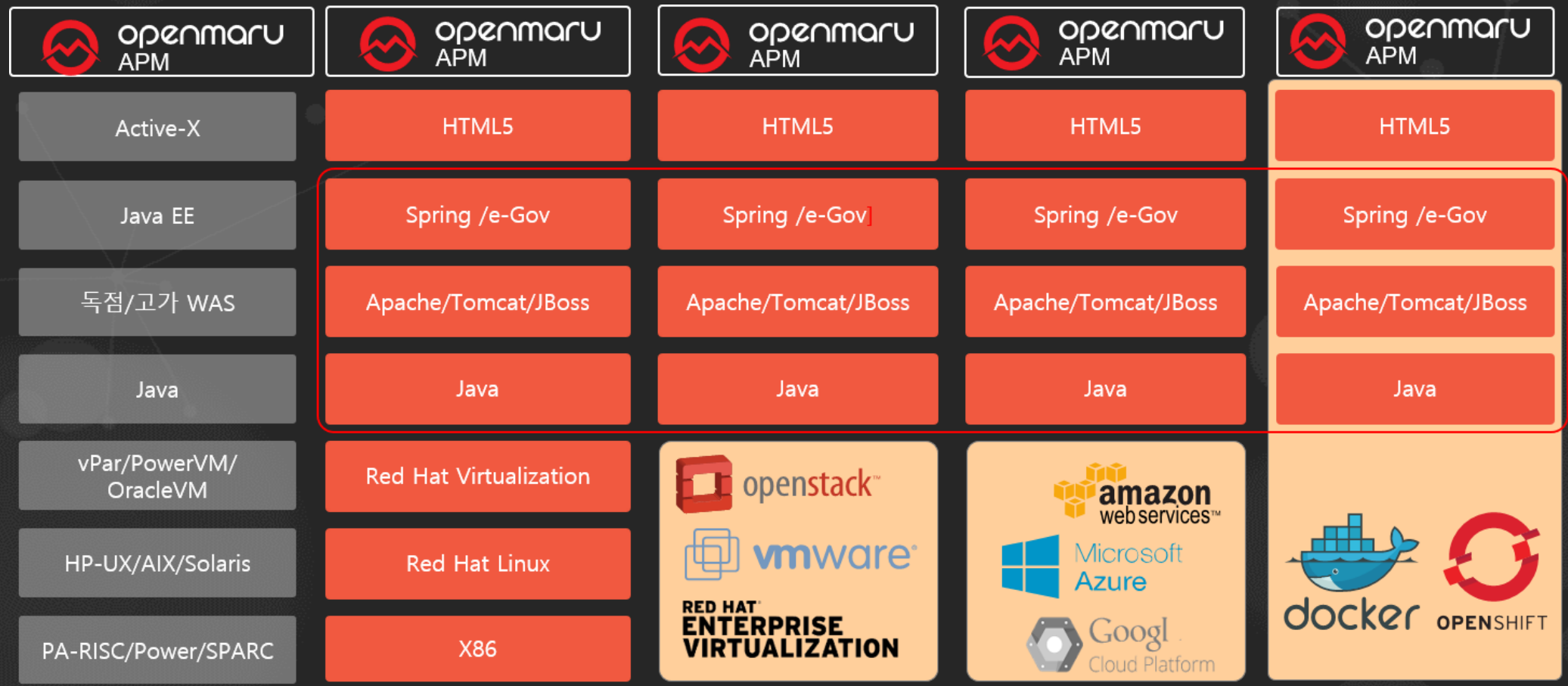
- UNIX-> Linux -> Virtualization -> Container
- Apache/Tomcat/JBoss 환경에서 주로 사용하는 상용 APM
- X86/Linux , 컨테이너, 클라우드 환경에서 점유율 1위

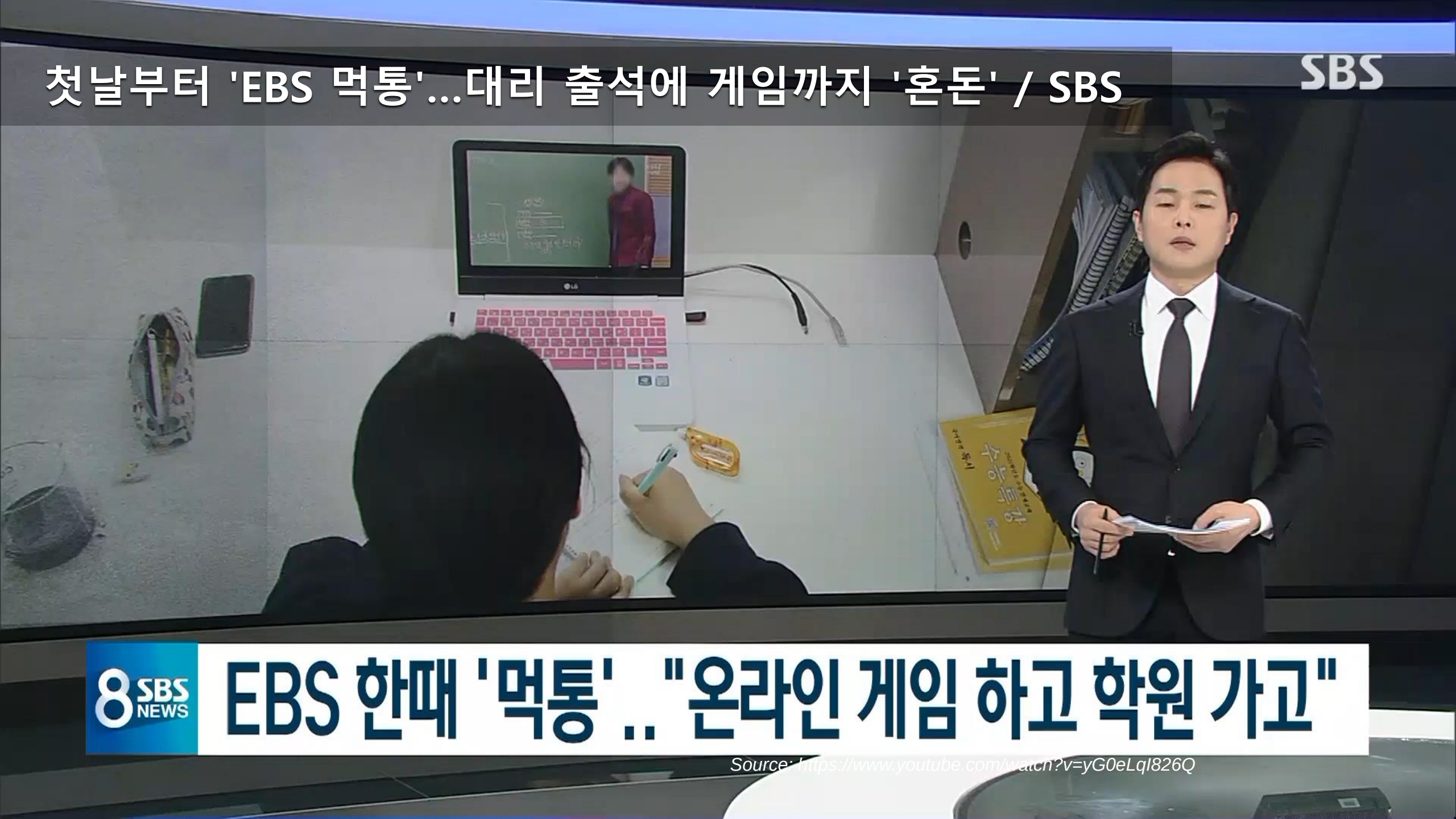
04 국내 최초 컨테이너 및 클라우드 환경 지원 APM

- Docker/Openshift/Openstack – immutable infrastructure 지원
- Amazon Cloud 지원
- Multicast 통신이 불가능한 환경 지원 / 유동IP 환경 지원

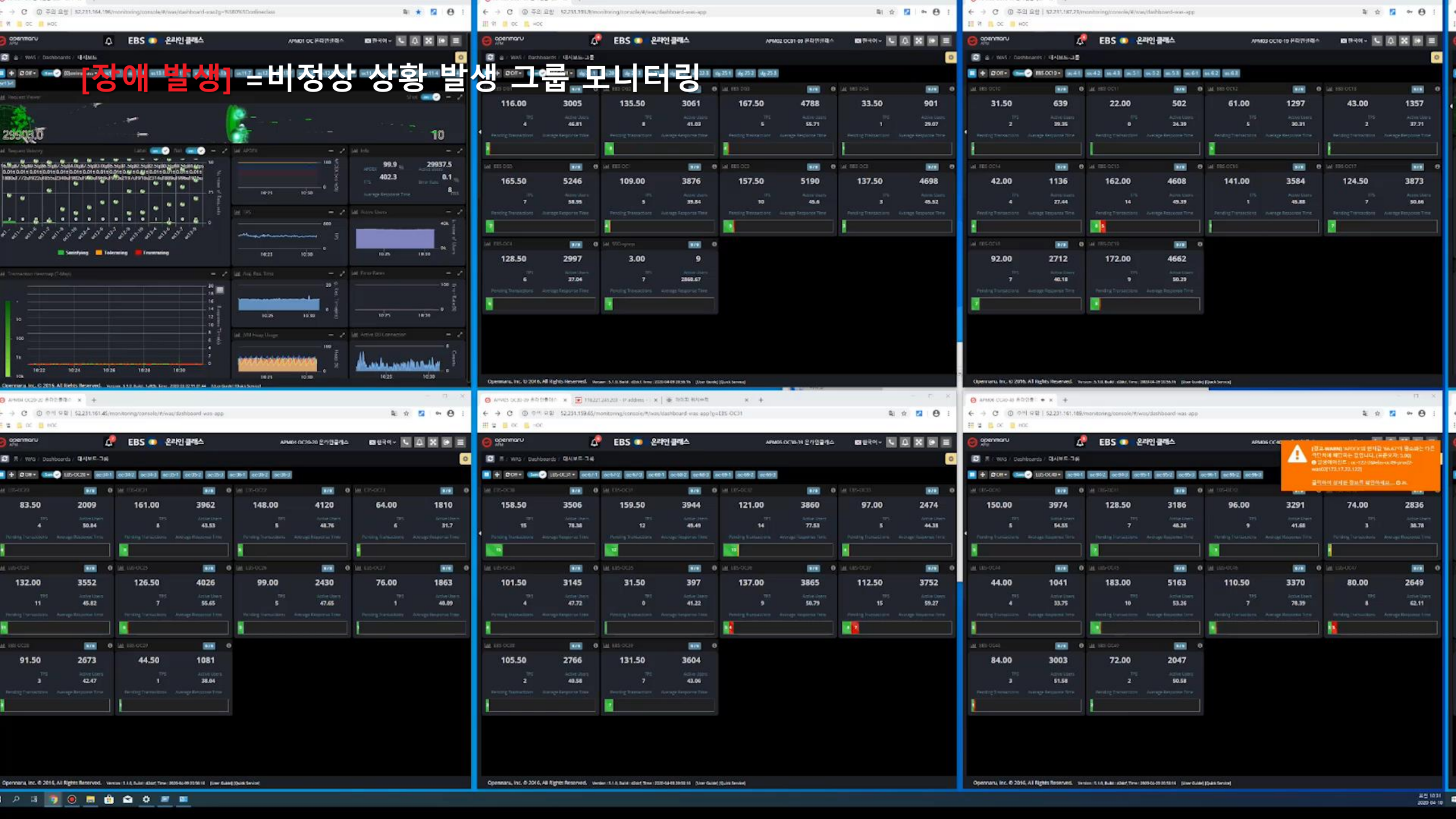
Hybrid Cloud 환경 지원 APM

- 다양한 플랫폼(OS, 가상화, 클라우드)에서 단일한 형태의 모니터링 지원





Source: https://www.youtube.com/watch?v=yG0eLqI826Q


[장애 발생] =비정상 상황 발생 그룹 모니터링

Micro Service Archtecture

OPENMARU APM 특장점

OPENMARU APM 차별성

openmaru

01

차별화된 기능

- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 퀵서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03

오픈소스 S/W에 최적화된 APM

- 오픈소스에 최적화된 APM
- 컨테이너 환경에 최적화된 APM
- AWS 환경에 최적화된 APM

04

미들웨어 전문기술기업

- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능 평가 1위

05

최신 IT 기술 지원

- WebSocket 을 통한 포트 사용 최소화
- 아마존 클라우드/ 컨테이너 환경에서 오토스케일링 지원
- PaaS지원 APM 시장 점유율 1위

OPENMARU APM 차별성

openmaru

01

차별화된 기능

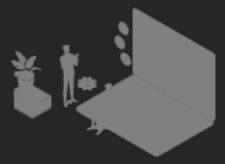
- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 퀵서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체 분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03



오픈소스 S/W 에 최적화된 APM

- 오픈소스에 최적화된 APM
- 컨테이너 환경에 최적화된 APM
- AWS 환경에 최적화된 APM

04

미들웨어 전문 기술기업

- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능평가 1위

05

최신 IT 기술 지원

- WebSocket 을 통한 포트 사용 최소화
- 아마존 클라우드/컨테이너 환경에서 오토스케일링 지원
- PaaS지원 APM 시장점유율 1위

장애예측 및 알림

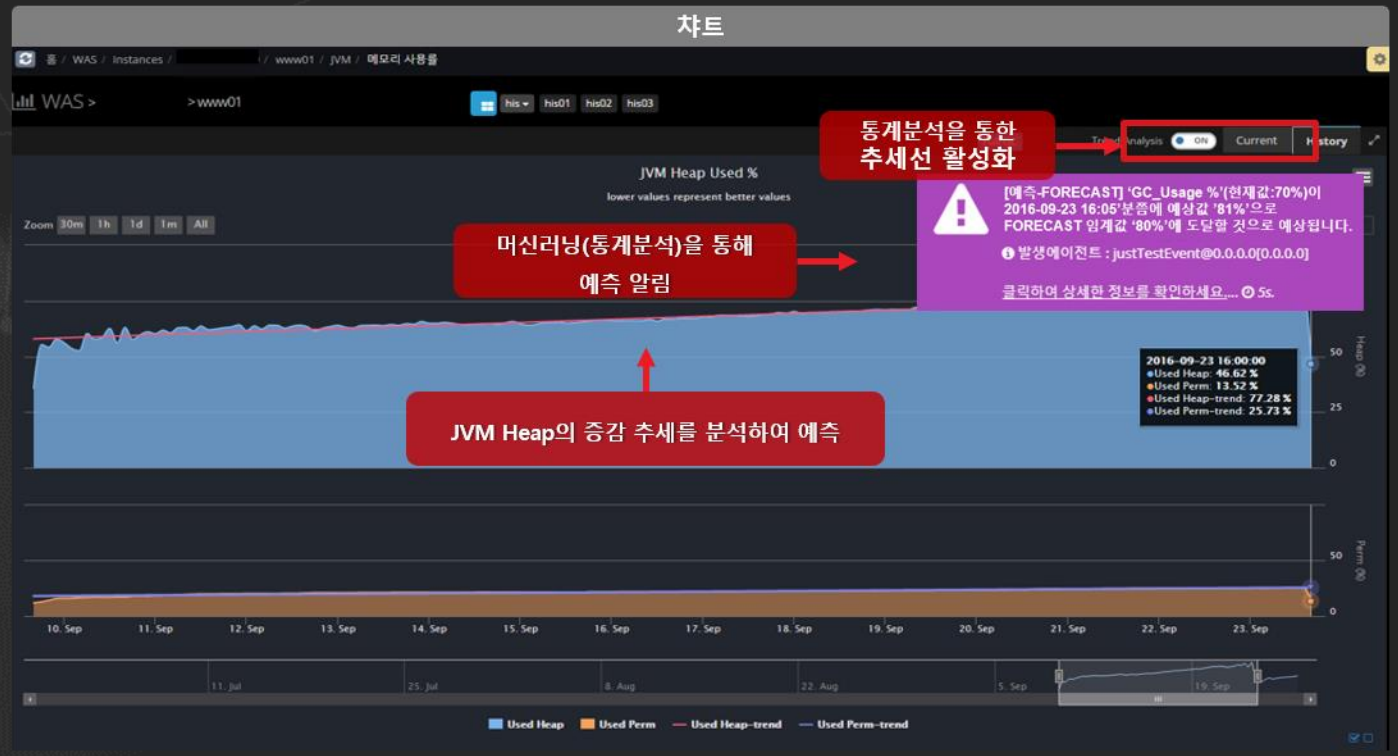
임계치에 도달하기 전에 이상 징후를 사전에 예측

- 실시간 머신러닝(통계분석)을 통해 앞으로 몇 분 후에 관리자가 설정한 임계 값에 도달할 것이라는 예측 이벤트를 통보한다.
- APM은 현재 상태 모니터링 및 추세분석을 통해 임계값 도달 시점을 예측하여 이벤트를 통보 한다.

▶임계치 도작 전 이상 징후 사전 예측
(홈 > WAS > (모든 차트) History)

Web | WAS | Container | System | Application

차트

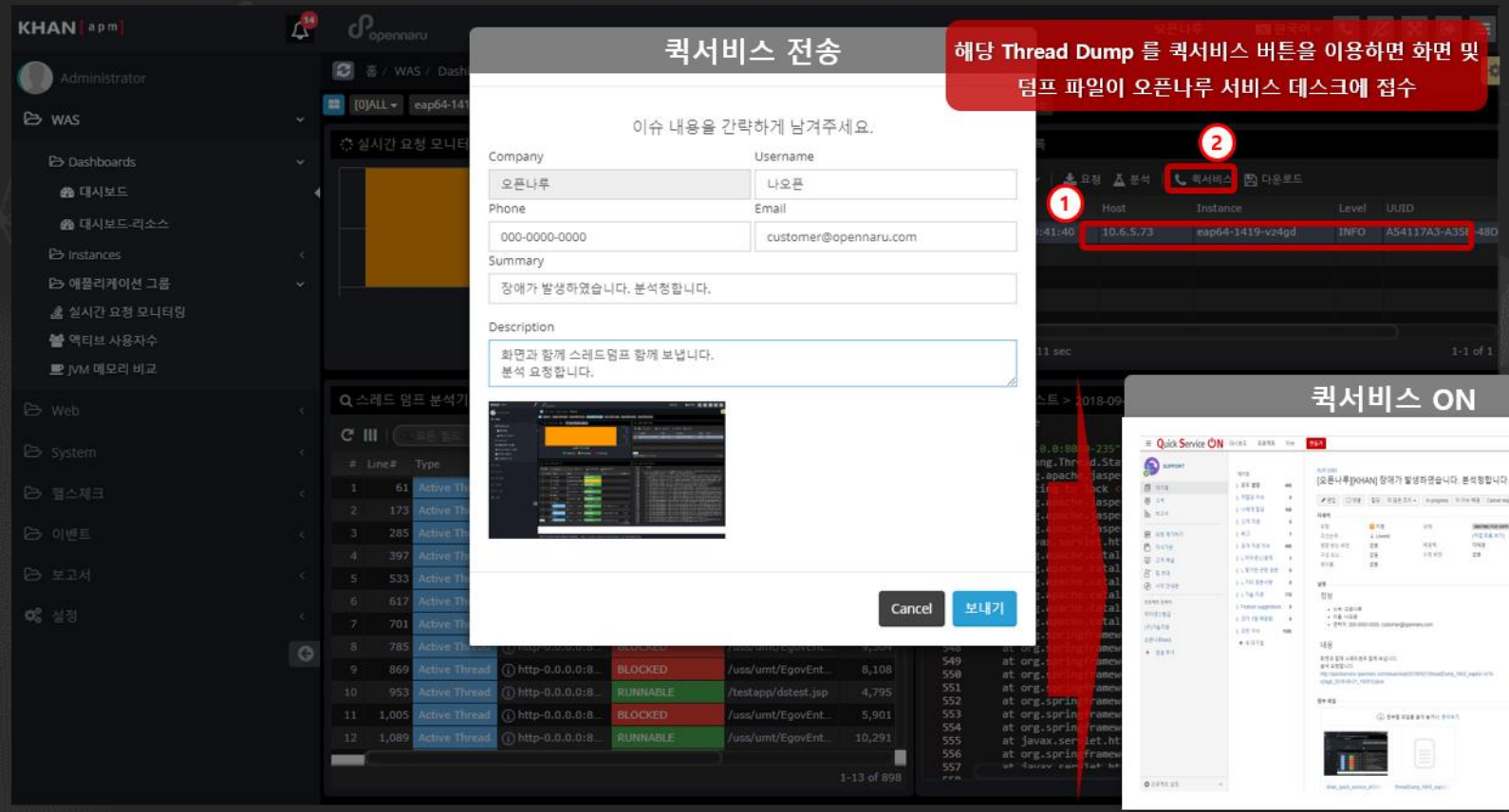


항목	이벤트 발생 항목
WAS	• JVM Heap Usage • APDEX • Error Rate • JVM Perm Usage • Database Response Time • Pending Transactions • GC Usage • Database Connection Pool Usage
WEB	• WEB Traffic • Worker Usage
System	• Memory Usage • Disk Usage • Memory Swap Usage • CPU Usage • Network Packet Error Rate
Cubrid	• CAS Usage

퀵 서비스

기술지원 문의 자동화 > 퀵서비스

- KHAN APM 에 표시된 지표에 대한 분석이 필요할 경우 퀵서비스를 통해서 실시간으로 문의 할 수 있다.
- 실시간으로 문의된 퀵서비스 내용은 신속하게 담당자가 할당되고, 처리 과정을 포탈, 메일등을 통해서 공유한다.



무차별 공격 알림 기능

무차별 공격 알림 기능

- DDoS 공격 패턴과 같은 동일한 Client IP에서 무차별적으로 애플리케이션 URL에 호출하는 현상을 감지 해야 한다.
- APM은 무차별 공격 유형이 감지될 경우 실시간 알림 이벤트와 해당 비정상 거래 상세 내용을 출력한다.

▶ 무차별 공격 알림 및 제한 기능 (홈 > 이벤트 > 이벤트 목록)

WAS | Container | System | Application

이벤트 알람 기준

- 동일 사용자가 단위시간에 특정URL을 100회 호출한 경우 (설정을 통해 변경 가능)

OPENMARU APM 차별성

openmaru

01

차별화된 기능

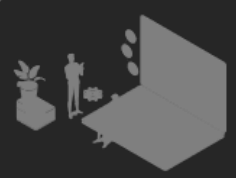
- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 쿽서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체 분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03



오픈소스 S/W 에 최적화된 APM

- 오픈소스에 최적화된 APM
- 컨테이너 환경에 최적화된 APM
- AWS 환경에 최적화된 APM

04

미들웨어 전문 기술기업

- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능 평가 1위

05

최신 IT 기술 지원

- WebSocket 을 통한 포트 사용 최소화
- 아마존 클라우드/컨테이너 환경에서 오토스케일링 지원
- PaaS지원 APM 시장점유율 1위

장애 원인 분석 도구 제공

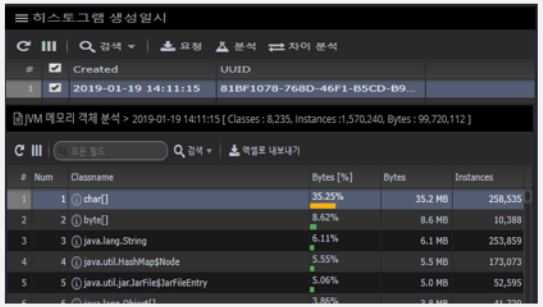
트러블 슈팅에 특화된 OPENMARU APM

- KHAN [apm]은 애플리케이션 성능 모니터링 및 진단, 장애 원인 분석을 통해 서비스를 최적의 상태로 운영할 수 있는 모니터링 솔루션이다.

스레드 덤프 분석기

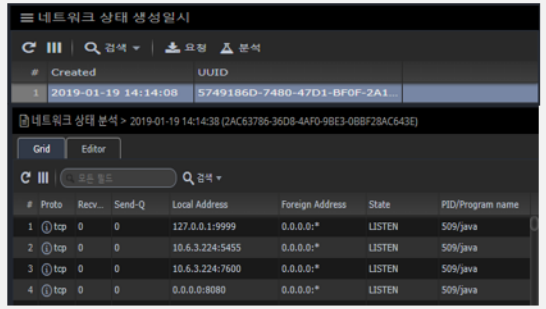
Lock을 추적가능, URL 정보표시

JVM 메모리 객체 분석기



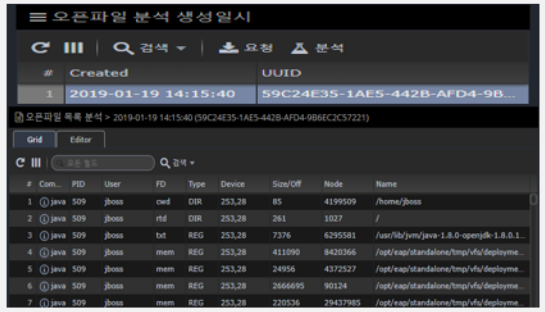
Java 메모리를 점유한 객체 분석/비교

네트워크 상태 분석기



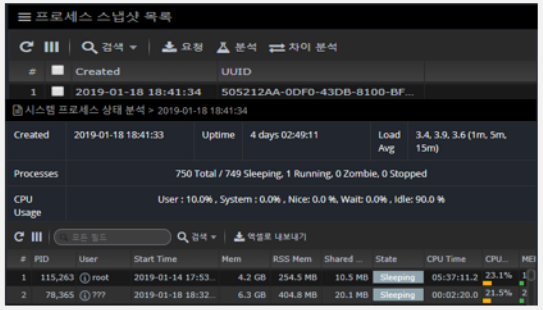
Java 프로세스,시스템이 사용중인 네트워크 분석

오픈파일 분석기



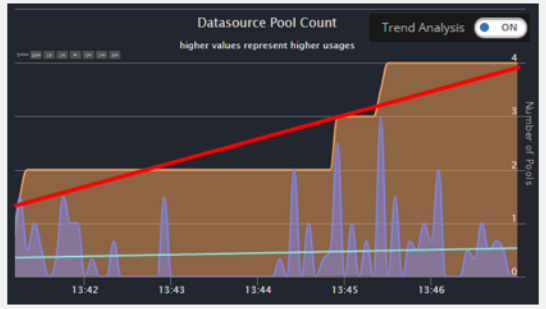
Java 프로세스가 오픈한 파일 분석

시스템 프로세스 분석기



시스템의 프로세스 CPU, 메모리 사용량 분석/비교

데이터 추세 분석



과거 데이터의 증감 추세를 분석하는 기능

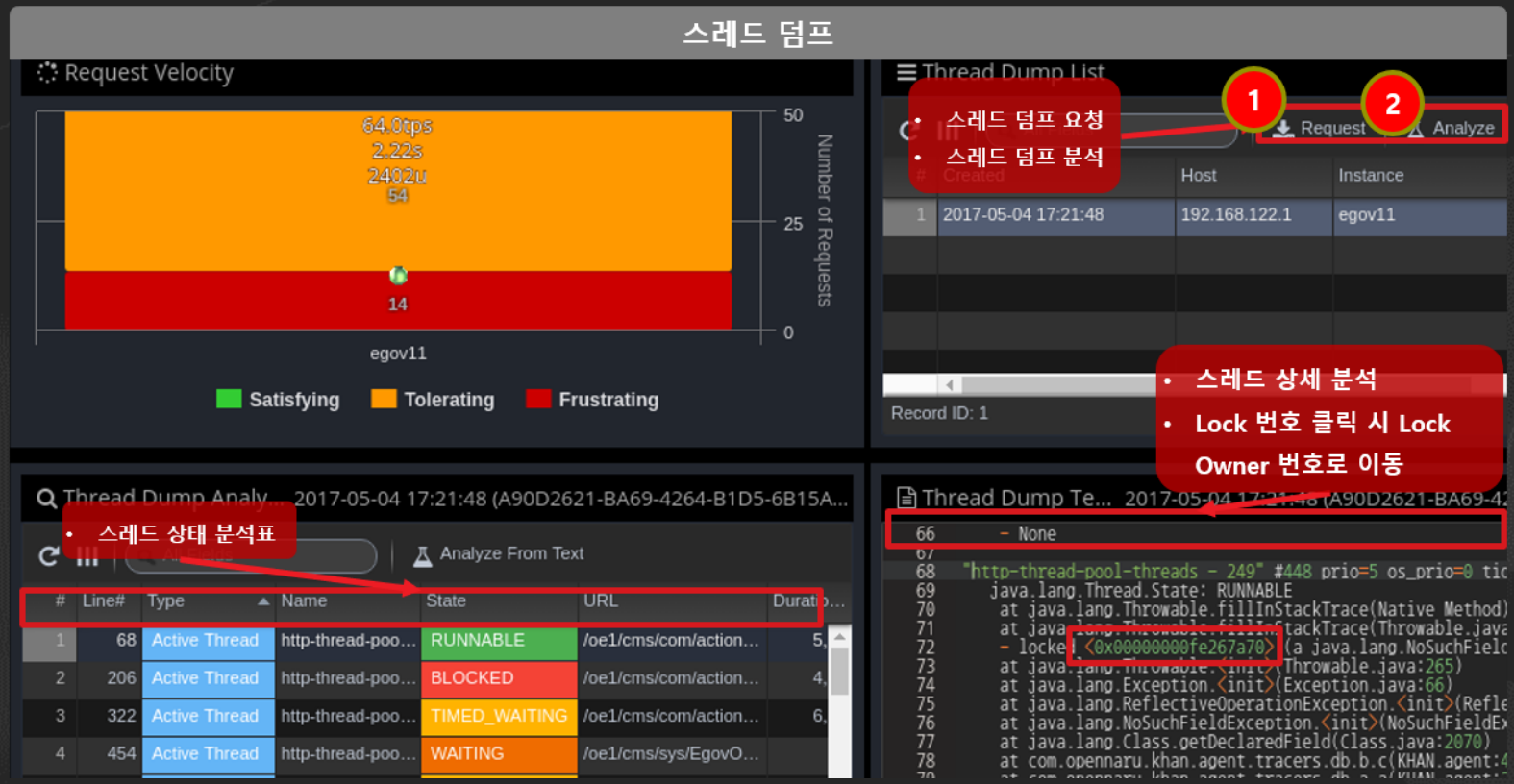
Java 스레드 덤프 생성 및 분석도구

장애 원인 분석기능을 위한 스레드 덤프 분석도구

- 시스템에 직접 접속하여 해당 인스턴스를 찾아서 커맨드를 실행하여 스레드 덤프를 생성한다.
- APM에서는 스레드 덤프 분석기를 사용하여 스레드 덤프를 생성하고 시각적으로 쉽게 스레드 분석 한다.

▶ Java 스레드(Thread) 덤프 분석 기능 (홈 > WAS > 애플리케이션 그룹 > 인스턴스 > 스레드 덤프 분석)

WAS Application



애플리케이션 오류 모니터링 기능

애플리케이션 오류 로그 수집 추적 기능

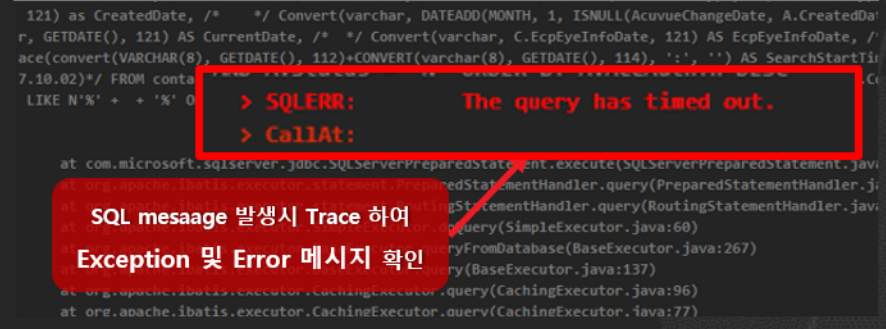
- 애플리케이션에서 Exception이 발생한 경우 이를 추적할 수 있어야 한다.
- APM은 여러 Exception을 추적하여 어떤 종류의 Exception이 발생되었는지 확인 할 수 있다.
- APM은 로깅 라이브러리에서 출력한 WARN, Error Level 값도 추적할 수 있다.

▶ 애플리케이션 예외사항(Exception) 추적 기능 (홈 > WAS > 애플리케이션 그룹 > 트랜잭션 맵)

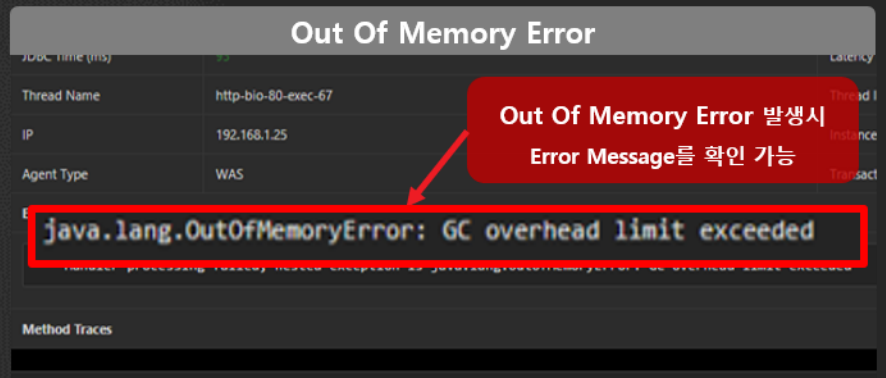
WAS Application

Logging Level Message

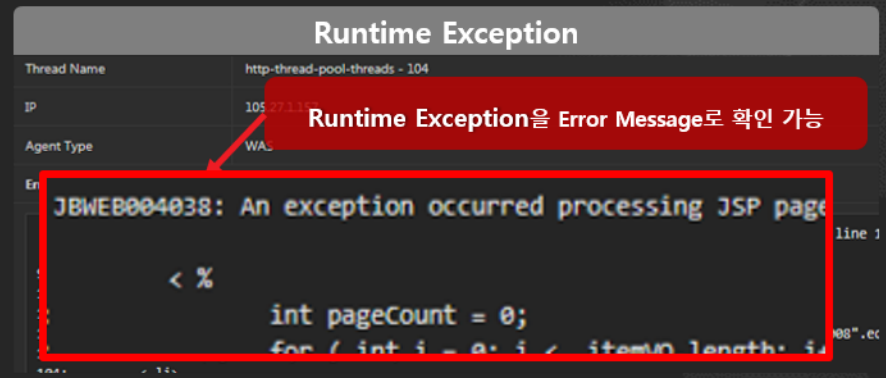
SQL Message


> SQLERR: The query has timed out.
> CallAt:
SQL mesaage 발생시 Trace 하여 Exception 및 Error 메시지 확인

Out Of Memory Error



Runtime Exception



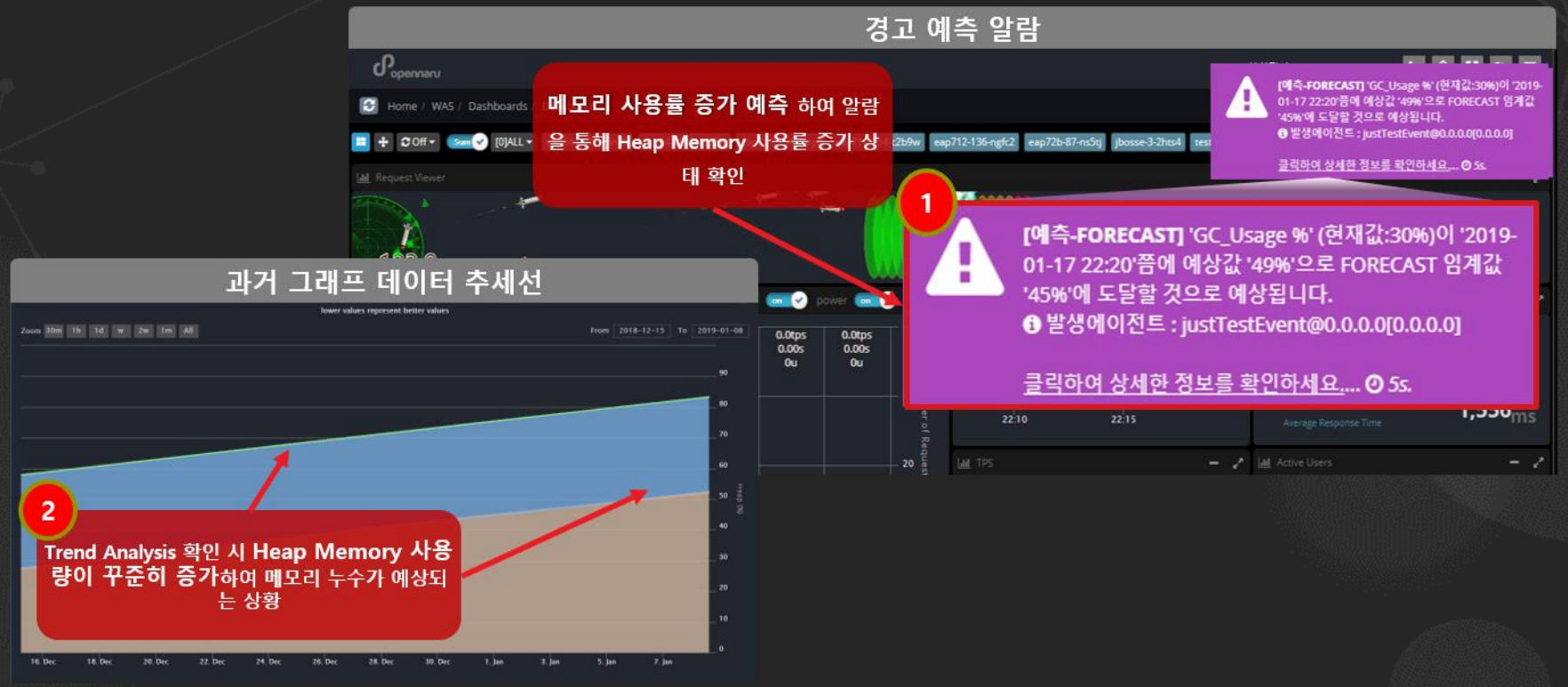
장애 원인 분석 기능

메모리 누수 감지기능

- JVM에서 GC를 실행 하더라도 애플리케이션에서 객체의 레퍼런스를 잡고 있다면 메모리의 공간을 지속적으로 점유하여 Out Of Memory가 발생하며 WAS가 종료 된다.
- APM은 메모리 누수 현상을 사전에 감지하여 관리자에게 경고를 해야 한다.

▶ 메모리 누수 감지 기능 (홈 > WAS > 애플리케이션 그룹 > JVM 메모리 비교)

WAS Application



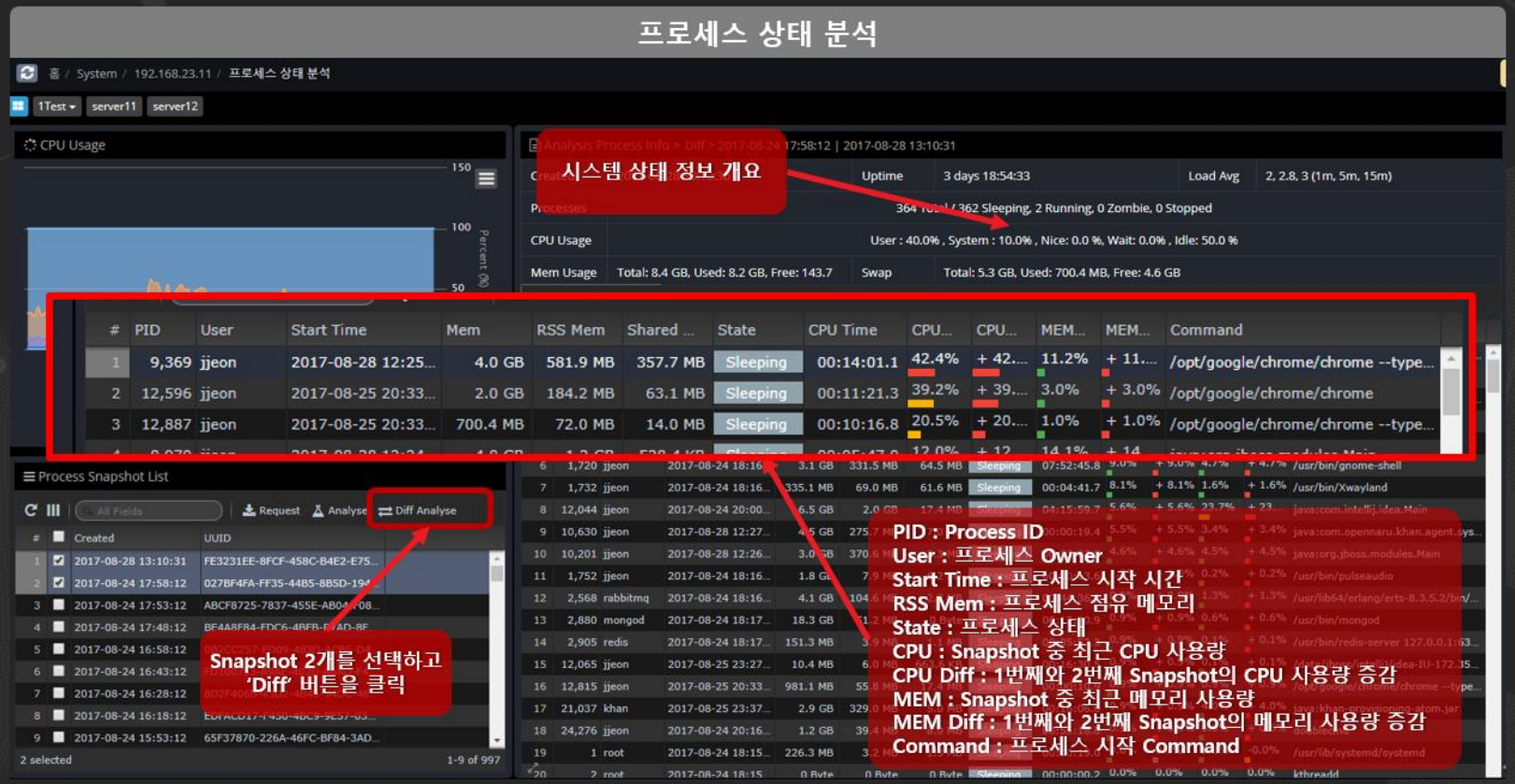
프로세스 상태 분석 기능

프로세스 상태 분석기능

- 기존에는 시스템에서 프로세스 상태(실행중인 프로세스의 resource) 분석을 하기 위해서 직접적으로 서버에 접속하여 command tool을 사용하여 분석 했다.
- APM 에서는 현재 실행중인 프로세스의 리소스 사용량 및 과거의 데이터와 비교해서 예전과 현재 상태를 비교 할 수 있어야 한다.

▶ 프로세스 상태 분석 기능 (홈 > System > 호스트 > 프로세스 상태 분석)

System



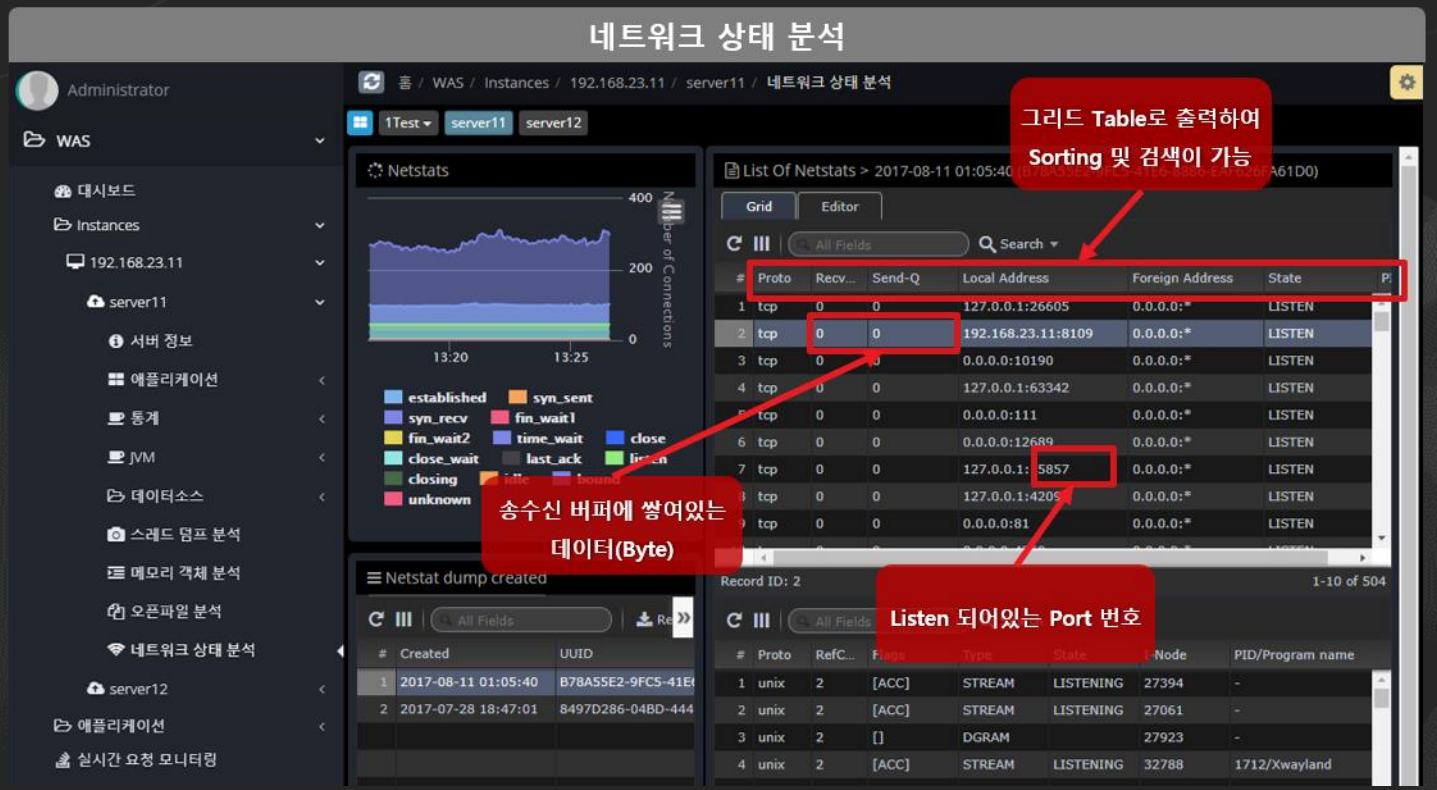
네트워크 상태 분석 기능

OS 네트워크 상태 분석 분석기능

- 기존에는 프로세스가 어떤 포트를 사용하고 네트워크 상태를 확인시에 직접적으로 서버에 접속해서 command tool을 사용하여 분석 했다.
- APM에서는 시스템의 네트워크 상태를 분석 할 수 있어야 한다.

▶ 네트워크 상태 분석 기능 (홈 > WAS > 인스턴스 > 네트워크 상태 분석)

WAS Application



과거 데이터 추세 분석 기능

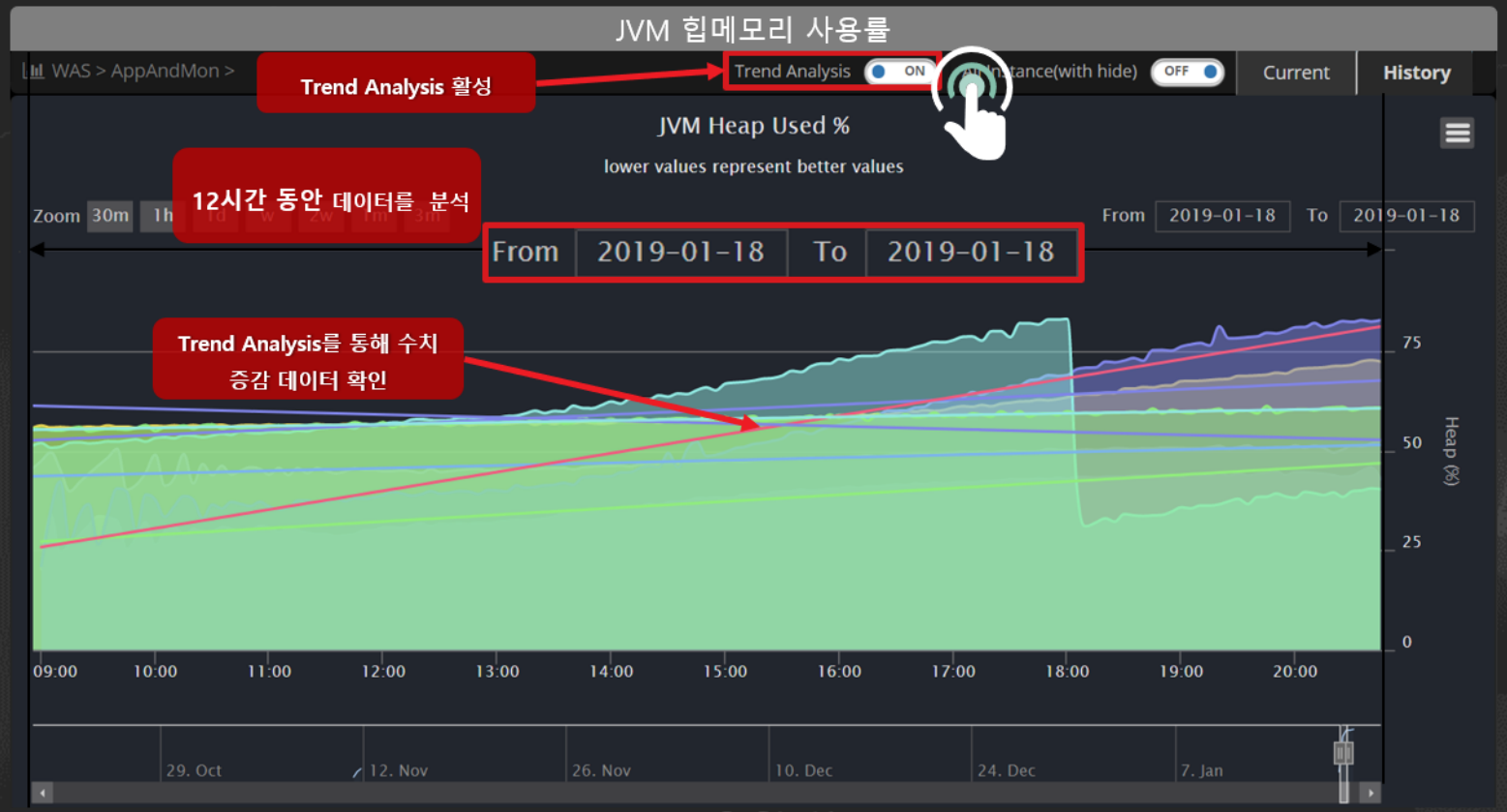
openmaru

과거 데이터 추세 분석 기능

- 과거와 현재 통계 데이터를 비교하여 분석하기 위해서는 Trend 분석 기능이 필요하다.
- APM에서는 저장된 통계 데이터의 Trend 분석을 통해 수치의 증감 상태를 확인할 수 있다.

▶ 과거 데이터에 대한 Trend 분석 기능 (홈 > WAS > 애플리케이션 그룹 > JVM 메모리 비교)

WAS Application

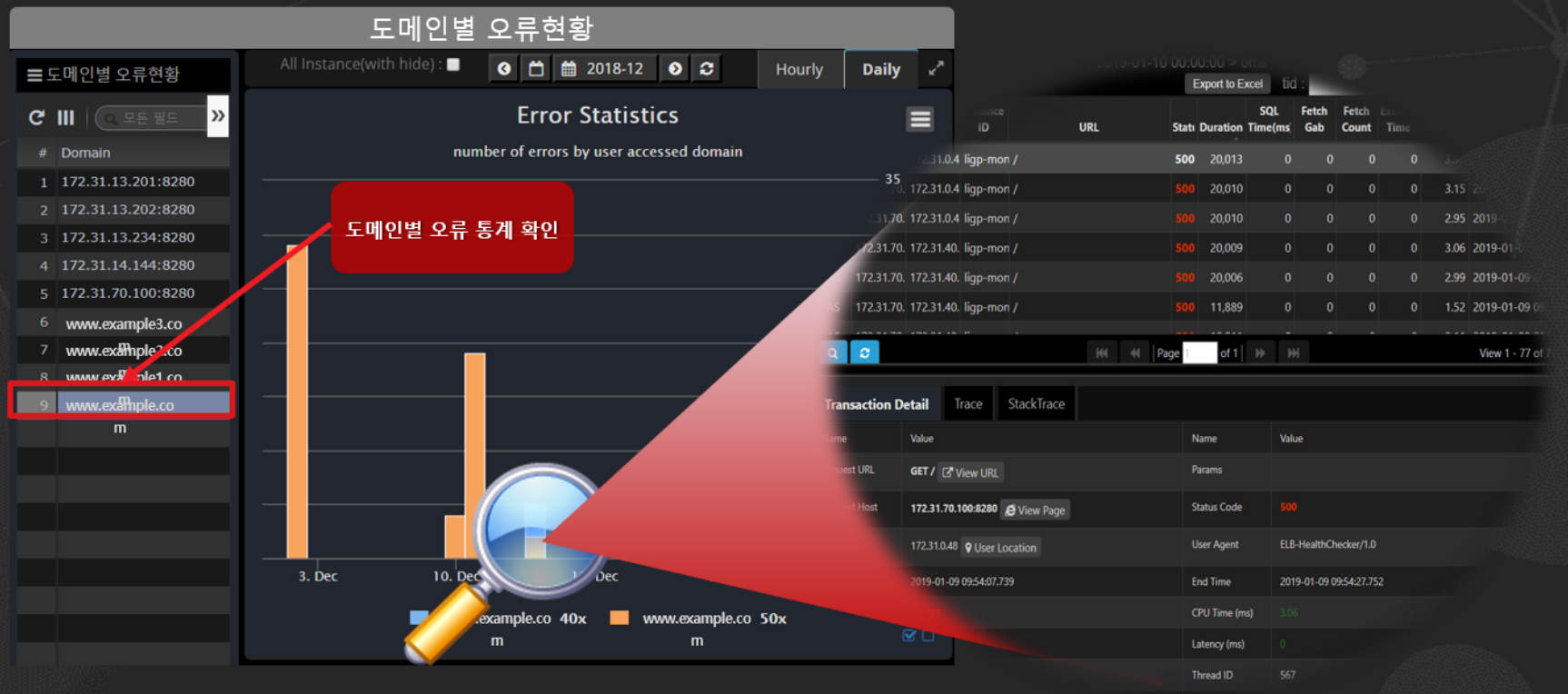


사용자 접속 도메인 별 오류 분석기능

사용자 접속 도메인별 오류 통계 및 상세 내용 파악

- 여러 도메인 운영 중 오류 발생시 도메인별로 어떤 오류가 발생했는지, 여러 시스템 직접 접근하여 문제의 원인 확인해야 한다.
- APM에서는 도메인 별로 오류(4xx, 5xx)에 대하여 통계 분석 차트를 제공하고 어떤 URL에서 오류가 발생했는지 분석한다.

▶ 사용자 도메인별 오류(400,500)에 대한 통계 분석 차트 제공 및 오류 상세 검색 기능 (홈 > WAS > 애플리케이션 그룹 > 오류 통계) WAS Application



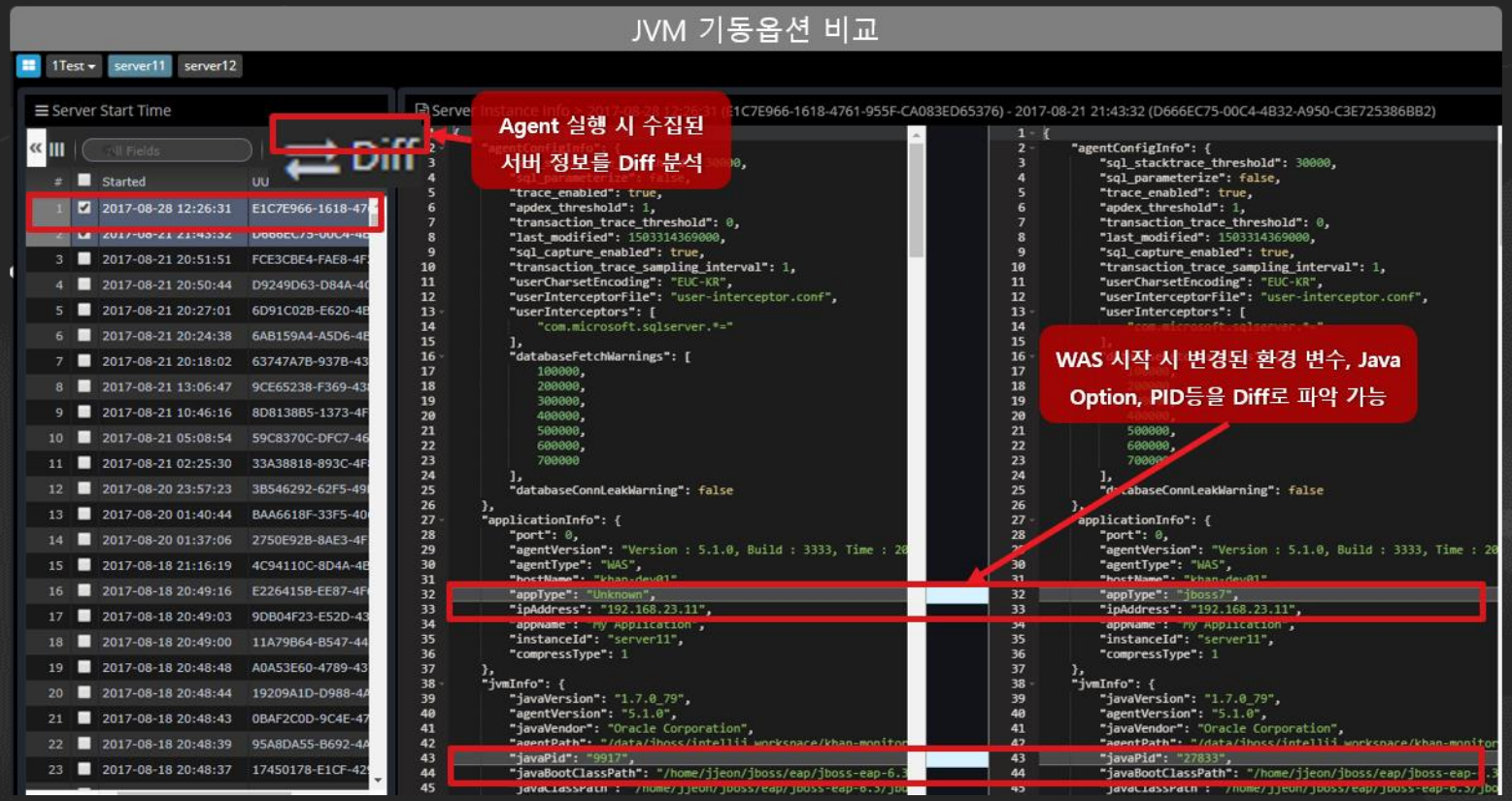
Java 시작옵션 비교 분석 기능

WAS 시작 옵션의 기록 및 비교 분석

- 서버 옵션을 변경하여 문제가 발생하는 경우, 옵션 변경기록이 남지 않는다.
- APM에서는 Agent 기동시에 상세 정보를 수집하고 보관하여, 특정 시점과 비교하여 어느 부분이 언제 변경되었는지 확인한다.

▶ JVM 시작시 상세 옵션 및 특정 시점과 비교 분석 기능 (홈 > WAS > 인스턴스 > 서버 정보)

WAS Application



OPENMARU APM 차별성

openmaru

01

차별화된 기능

- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 퀵서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체 분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03

오픈소스 S/W에 최적화된 APM

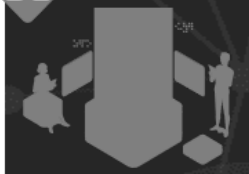
- **오픈소스에 최적화 APM**
- **컨테이너 환경에 최적화된 APM**
- **AWS 환경에 최적화 APM**

04

미들웨어 전문 기술기업

- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능 평가 1위

05



최신 IT 기술 지원

- WebSocket 을 통한 포트 사용 최소화
- 아마존 클라우드/컨테이너 환경에서 오토스케일링 지원
- PaaS지원 APM 시장점유율 1위

openmaru APM

독점 소프트웨어	현재 구성	유상 오픈소스	무상 오픈소스
Active-X	Active-X	HTML5	HTML5
MS Internet Explorer	MS Internet Explorer	Chrome/IE/Firefox/Safari	Chrome/IE/Firefox/Safari
X-Internet	X-Internet	Sencha	Extjs
Commercial Framework	Commercial Framework	Spring Framework	Spring Framework
JEUS/WebLogic/WebSphere	WebSphere	JBoss	Wildfly/Tomcat
Oracle/DB2/MS SQL	Oracle 9i	MariaDB/PPAS	MariaDB/PostgreSQL
HP-UX/AIX/Solaris	AIX	Red Hat Linux	CentOS
PA-RISC/Power/SPARC	Power	X86	X86

설치와 구성

진단과 조치

openmaru
APM

Provisioning

Monitoring

튜닝

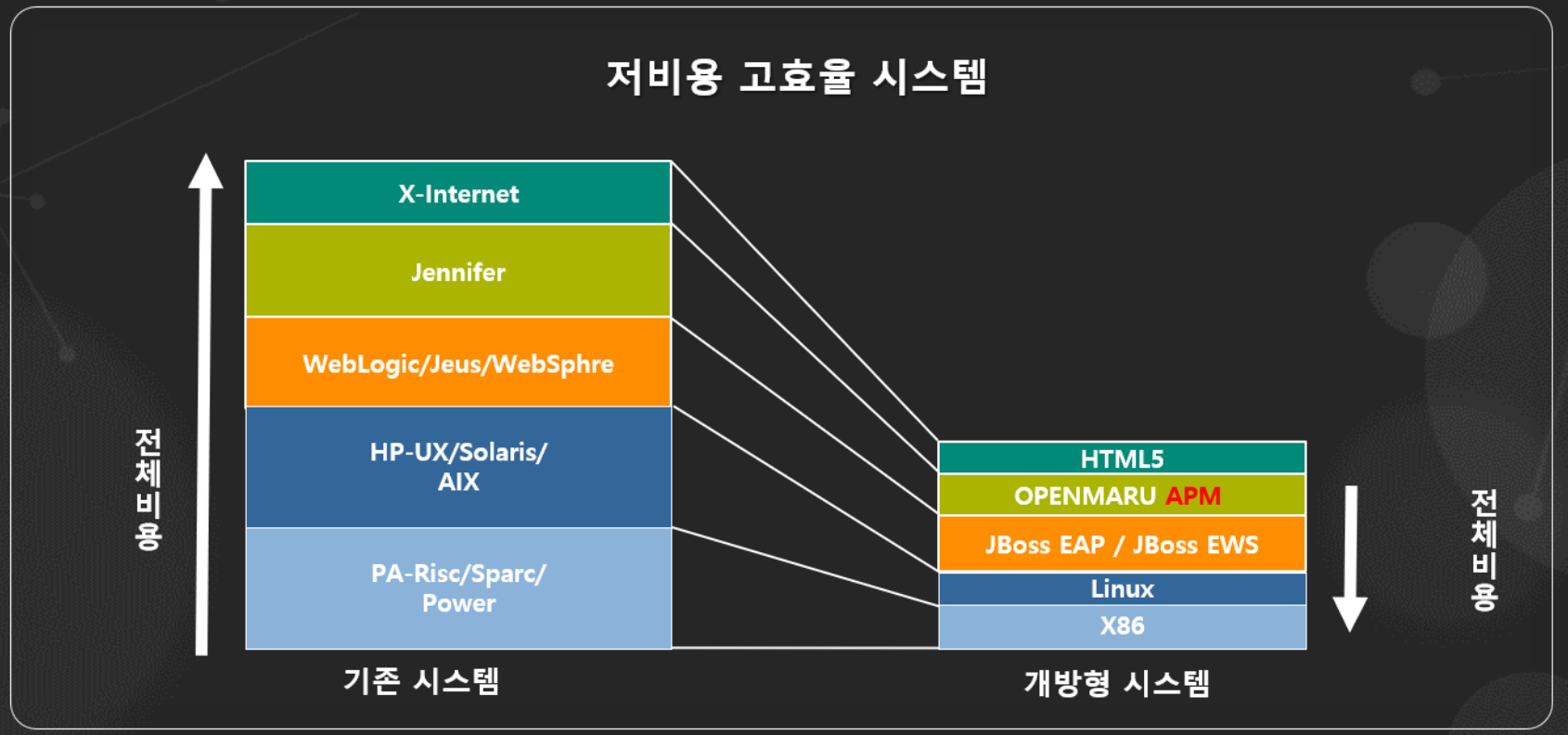
모니터링

전문가 수준의 구성 품질 › 장애 사전 예방 › 성능 보장 › 안정된 운영

설치	구성	튜닝	모니터링	진단/조치
•리눅스만 설치되어 있으면 WAS 와 웹서버 등 웹시스템 구축에 필요한 소프트웨어를 자동설치	•WAS 클러스터링, 데이터그리드 , 운영 스크립트 등 난이도 높은 구성을 지원 •개발자/관리자를 위한 운영 가이드 자동 생성	•전문가 수준의 튜닝 (리눅스 커널, 웹서버/ Java/WAS 튜닝)	•시스템과 OS 에 대한 모니터링 뿐만 아니라 Java 나 WAS 까지 포괄적으로 지원 • Openshift/Docker 컨테이너 모니터링	•Threaddump , 네트워크 상태, 웹서버 상태를 분석할 수 있는 진단 도구 제공 • OS/Container 시스템에 대한 연관 분석

OPENMARU APM 제품 비교 – 비용 측면

- 오픈 소스 기반의 저비용 고효율 웹시스템 구축
- APM 을 통한 운영 효율성과 안정성 확보
- 서브스크립션 구매형태의 비용 절감과 벤더 종속성 탈피
- 웹시스템에 대한 일원화된 기술 지원 체계



OPENMARU APM 구성 아키텍처

- OPENMARU APM 구성 아키텍처
- 모니터링 기능은 대부분의 OS나 주요 WAS를 지원
- 프로비저닝 기능은 Apache/Tomcat/JBoss와 레드햇 계열 리눅스 환경을 지원

openmaru APM = Ah ! Provisioning & Monitoring

고객 요구사항에 따른 웹 시스템 작업

고객

수 일 이상
기술 지원

- 1 일: 개발 환경 지원, 개발 서버 설치
- 1 일: 웹 서버 설치, WAS서버 설치, 웹 서버/WAS서버연결
- 1 일: 클러스터링 구성, 애플리케이션 배포 지원, 테스트 지원
- 1 일: 부하테스트 지원
- ? 일: 오픈 지원
- ? 일: 웹 시스템 표준 구성
- ? 일: 장애 지원

openmaru APM

수작업
단순 설치

최적화된 웹시스템 구성

웹서버 설치	웹서버 튜닝	Worker MPM 구성	mod_cluster 설정	웹서버
mod_jk 설정	웹서버 설치 보고서 작성	로깅 디렉터리 설정	웹취약점 조치	
JBoss 설치	도메인 모드 구성	쓰레드 풀 튜닝	애플리케이션 배포 구성	JBoss
	멀티 인스턴스 구성	각종 Dump 디렉터리 구성	Native 라이브러리 설정	
JDK 설치	JDBC 커넥션 풀 튜닝	모듈 설정	JSP 변경 체크 설정	
	모니터링 설정	Log4J 설정	WAS 서버 설치보고서 작성	
운영 계정 설정	추가 rpm 설치	블루투스 off	커널 파라미터 설정	OS
	Huge Page 설정	limits.conf 설정	Cpuspped 설정	
방화벽 설정	OS 설정 보고서 작성	운영을 위한 쉘 구성	로깅 디렉터리 설정	
배포 테스트	기본 네트워크 테스트	클러스터링 테스트	단위 테스트	테스트
	애플리케이션 배포 테스트	통합 테스트	Benchmark 테스트	
	운영관리 테스트	멀티캐스트 테스트	JDBC 테스트	

OPENMARU APM - Provisioning 기능은?

OS 만 설치되어 있으면 수분 이내에 설치 환경을 테스트하고 웹서버와 WAS서버를 설치하고 즉시 서비스할 수 있는 환경 제공

미들웨어 전문가가 아니어도 전문가 수준의 시스템 튜닝이나 난이도 높은 구성을 할 수 있도록 기능 제공

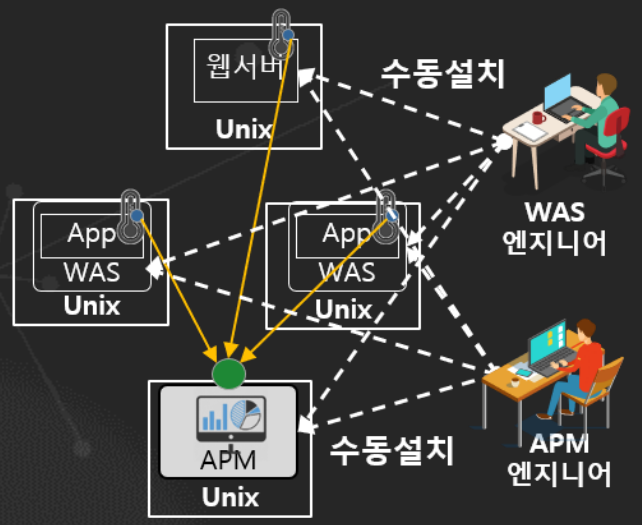
서버 구성에 대한 정보만 입력하면 한대에서 수 십대까지 규모에 상관없이 자동으로 웹시스템 운영환경을 구성

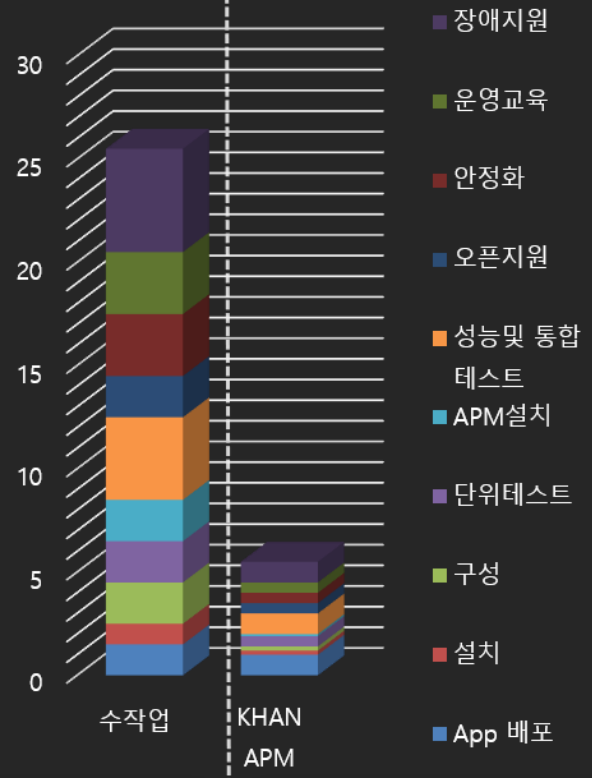
웹시스템 설치/구성 보고서를 시스템에 맞게 자동으로 생성하여 개발팀과 운영 팀에게 제공

수 분 내 튜닝된 웹 서버와 WAS 서버로 웹 시스템 구축

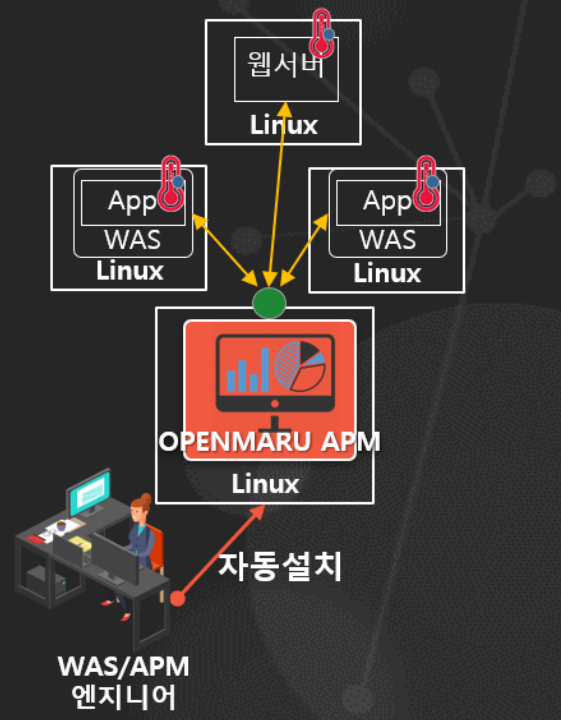
OPENMARU APM 제품 비교 - 개요

기존 웹시스템 구축 - 수작업





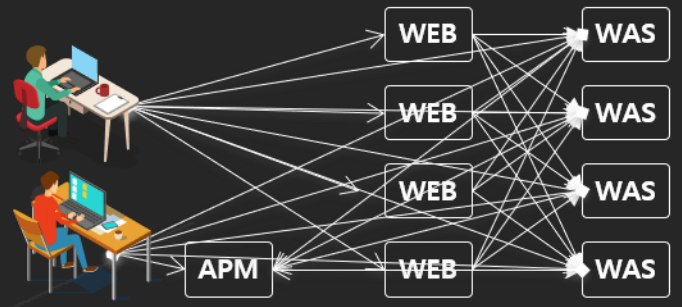
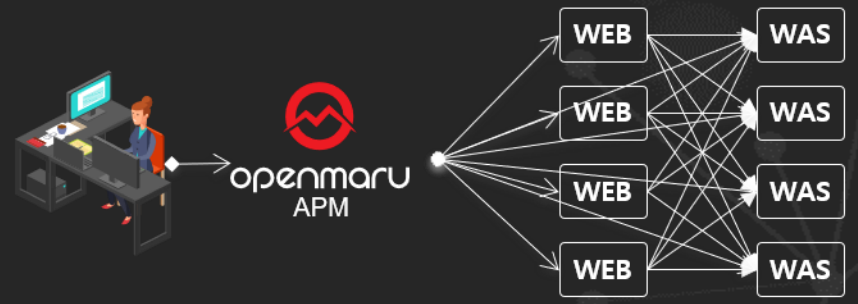
OPENMARU APM - 자동화



- ✓ WAS 엔지니어와 APM엔지니어 분리
- ✓ 수작업 중심/ 튜닝과 운영 지원 환경 별도

- ✓ WAS/APM 에 대한 단일 기술 지원 체계
- ✓ 웹서버/WAS/APM 를 자동 설치/구성/튜닝

OPENMARU APM 제품 비교 – 상세 비교

항목	수작업에 의한 웹시스템 구축		OPENMARU Installer 을 통한 웹시스템 구축	
설치 방법				
구축 기간	●	• 수일 ~ 수 주	●	• 수 시간 ~ 수일
안정성	●	• 담당자의 기술 수준에 따른 품질의 차이 • 수작업으로 인한 Human Error	●	• JBoss 전문가에 의해서 구성되는 최적화된 환경으로 웹서버/WAS 환경 구성 • 자동화 구성으로 설치/구성의 오류 발생률 최소화
튜닝	●	• 설치 담당자의 기술력에 따라 튜닝 적용의 차이점 발생 • 표준화하더라도 해당 가이드 적용 에 대하여 지속적인 관리 필요	●	• 전문가에 의한 각 부분별 튜닝 적용 • 자동화된 구성으로 인하여 최적화된 표준환경 구성
유지보수	●	• 구성 변경 시 마다 수작업	●	• 구성 파일을 수정하여 자동화된 재설치
생산성	●	• 수작업에 의한 다운로드/튜닝 • OS/웹서버/WAS 서버 별 수작성 설치 구성	●	• 자동화된 다운로드, 설치, 구성, 튜닝 • 웹서버/WAS 서버에 최적화된 환경으로 표준화
벤더	●	• WebLogic/WebSpehrere/JEUS/Resin • Apache httpd / Apache tomcat	●	• 오픈 소스 Web/WAS + OPENMARU Installer • Red Hat JBoss + OPENMARU Installer

● 최적 ● 적합 ● 개선필요

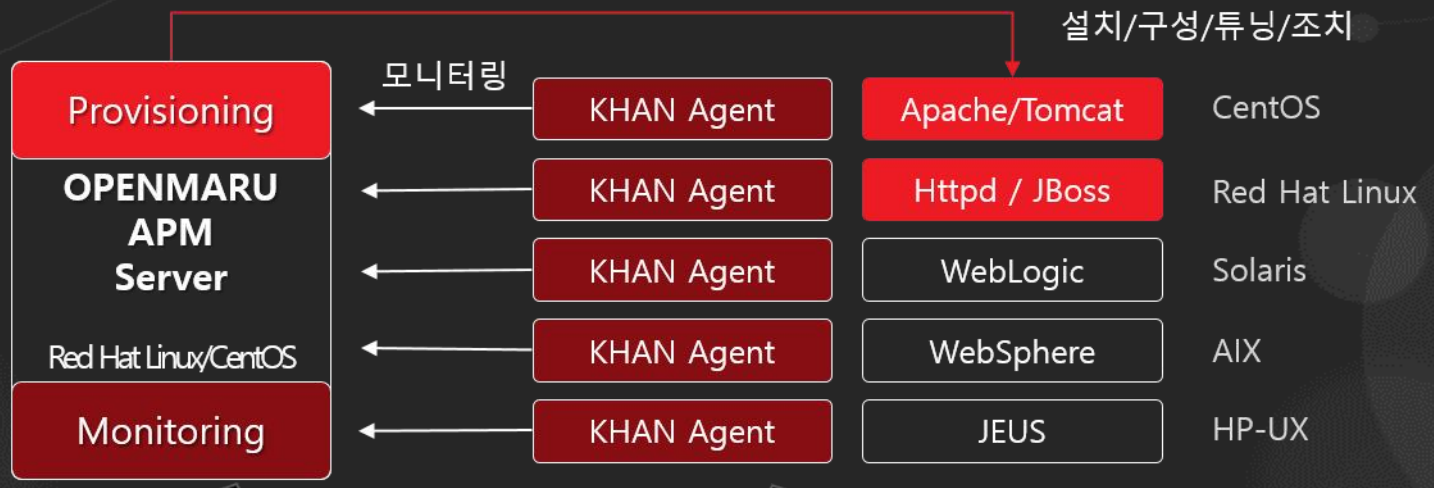
OPENMARU APM 제품 비교 – 설치/구성 단계

- 웹시스템 구축은 일정에 따라 다양한 기술 지원 요청이 발생
- 각 단계별로 전문적인 기술 지원이 필수적임
- OPENMARU Installer 을 사용할 경우 기존 방법 대비 On-Site 지원 횟수 감수
- 온사이트 장애 지원에 대한 횟수 감수

OPENMARU APM 구성 아키텍처

- OPENMARU APM 구성 아키텍처
- 모니터링 기능은 대부분의 OS나 주요 WAS를 지원
- 프로비저닝 기능은 Apache /Tomcat/JBoss와 레드햇 계열 리눅스 환경을 지원

openmaru APM = **A**h ! **P**rovisioning & **M**onitoring



OPENMARU APM - Provisioning 기능은?

OS 만 설치되어 있으면 수분 이내에 설치 환경을 테스트하고 웹서버와 WAS서버를 설치하고 즉시 서비스할 수 있는 환경 제공

미들웨어 전문가가 아니어도 전문가 수준의 시스템 튜닝이나 난이도 높은 구성을 할 수 있도록 기능 제공

서버 구성에 대한 정보만 입력하면 한대에서 수 십대까지 규모에 상관없이 자동으로 웹시스템 운영환경을 구성

웹시스템 설치/구성 보고서를 시스템에 맞게 자동으로 생성하여 개발팀과 운영 팀에게 제공

수 분 내 튜닝된 웹 서버와 WAS 서버로 웹 시스템 구축

OPENMARU APM 차별성

openmaru

01

차별화된 기능

- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 쿼서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체 분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03

오픈소스 S/W 에 최적화된 APM

- 오픈소스에 최적화된 APM
- 컨테이너 환경에 최적화된 APM
- AWS 환경에 최적화된 APM

04

미들웨어 전문 기술기업

- **GS 인증 1등급**
- **오픈소스 전문기업이 만든 APM**
- **WAS 장애 지원 및 튜닝 지원**
- **행정안전부 온-나라 BMT 성능평가 1위**

05

최신 IT 기술 지원

- WebSocket 을 통한 포트 사용 최소화
- 아마존 클라우드/컨테이너 환경에서 오토스케일링 지원
- PaaS지원 APM 시장점유율 1위

APM 필요성

개발팀
- 고성능 애플리케이션 개발
- 신속한 버그 픽스
- 이슈 재발 방지

openmaru APM

QA팀
- 부하테스트 시 성능 보장
- 서비스 오류 감지
- 최소 응답시간 유지

운영팀
- 성능에 대한 최적화와 서비스 정상 유/무
- 신속한 장애 감지와 원이 파악 그리고 조치
- Over/Under 사이징인가?

비즈니스 담당자
- 고객의 서비스 만족도는 충분한가?
- 고객 이탈 방지 및 브랜드 이미지 손상 방지
- 경쟁사 대비 성능은 충분한가?


Quick Service
OPENMARU APM
화면에서 원클릭으로
Quick Service를
요청하시면 장애원인을
분석하여 결과보고서를
드립니다.
openmaru
APM

Opennaru Quick Service 란?

openmaru

고객사

① 시스템 이슈발생

② 쓰레드 덤프와 모니터링 화면 전달

③ 오픈나루 서비스 데스크 접수

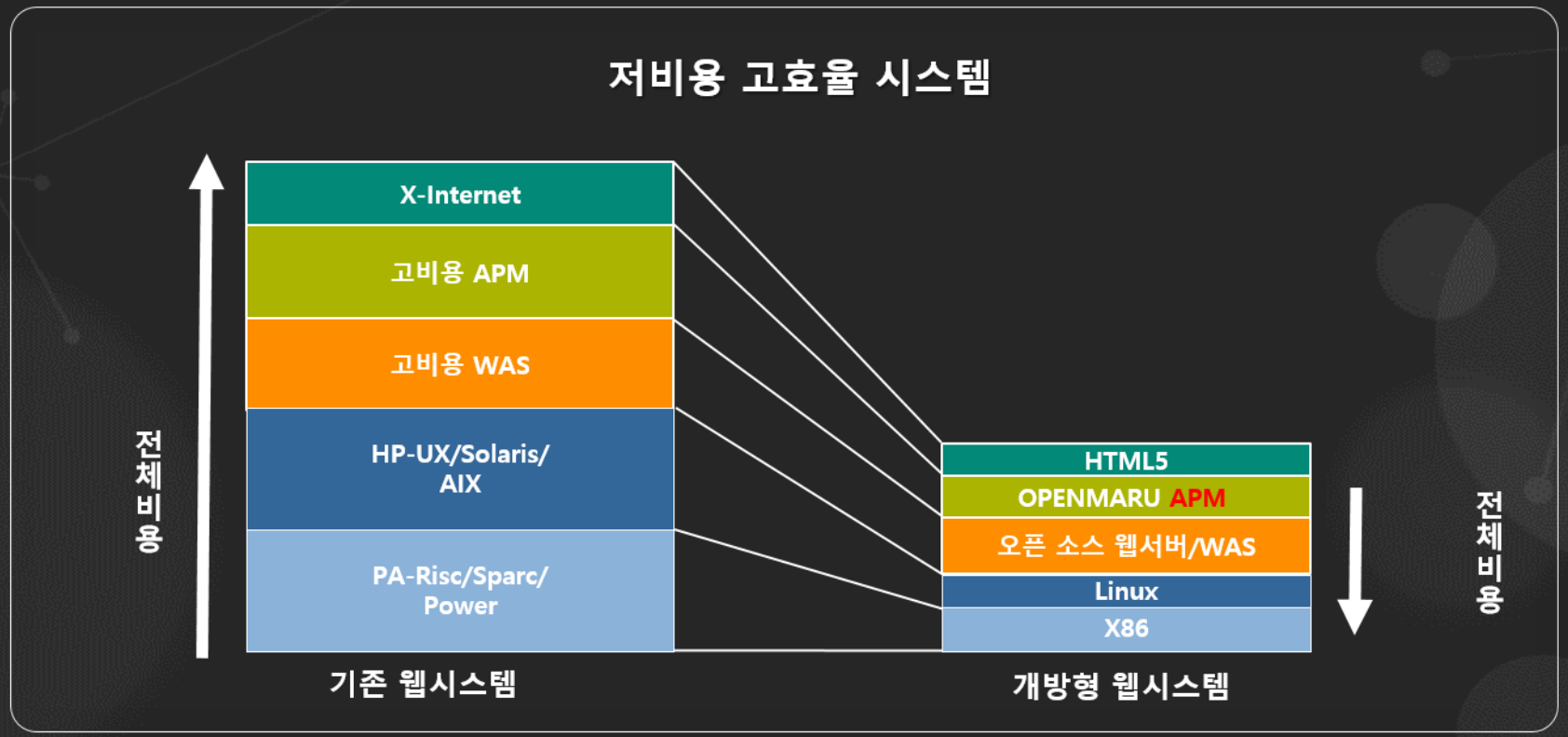
④ 장애 분석

⑤ 장애 분석 보고서 전달

오픈나루

OPENMARU APM 제품 비교 – 비용 측면

- 오픈 소스 기반의 저비용 고효율 웹시스템 구축
- APM 을 통한 운영 효율성과 안정성 확보
- 서브스크립션 구매형태의 비용 절감과 벤더 종속성 탈피
- 웹시스템에 대한 일원화된 기술 지원 체계



OPENMARU APM 차별성

01

차별화된 기능

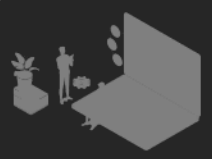
- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 쿽서비스
- 프로비저닝
- 웹 서버 모니터링
- 시스템 모니터링

02

전문 장애분석 도구 제공

- 스레드 덤프 분석
- JVM 메모리 객체 분석
- 네트워크 상태 분석
- 오픈파일 분석
- 시스템 프로세스 분석
- 데이터 추세 분석

03



오픈소스 S/W 에 최적화된 APM

- 오픈소스에 최적화된 APM
- 컨테이너 환경에 최적화된 APM
- AWS 환경에 최적화된 APM

04

미들웨어 전문 기술기업

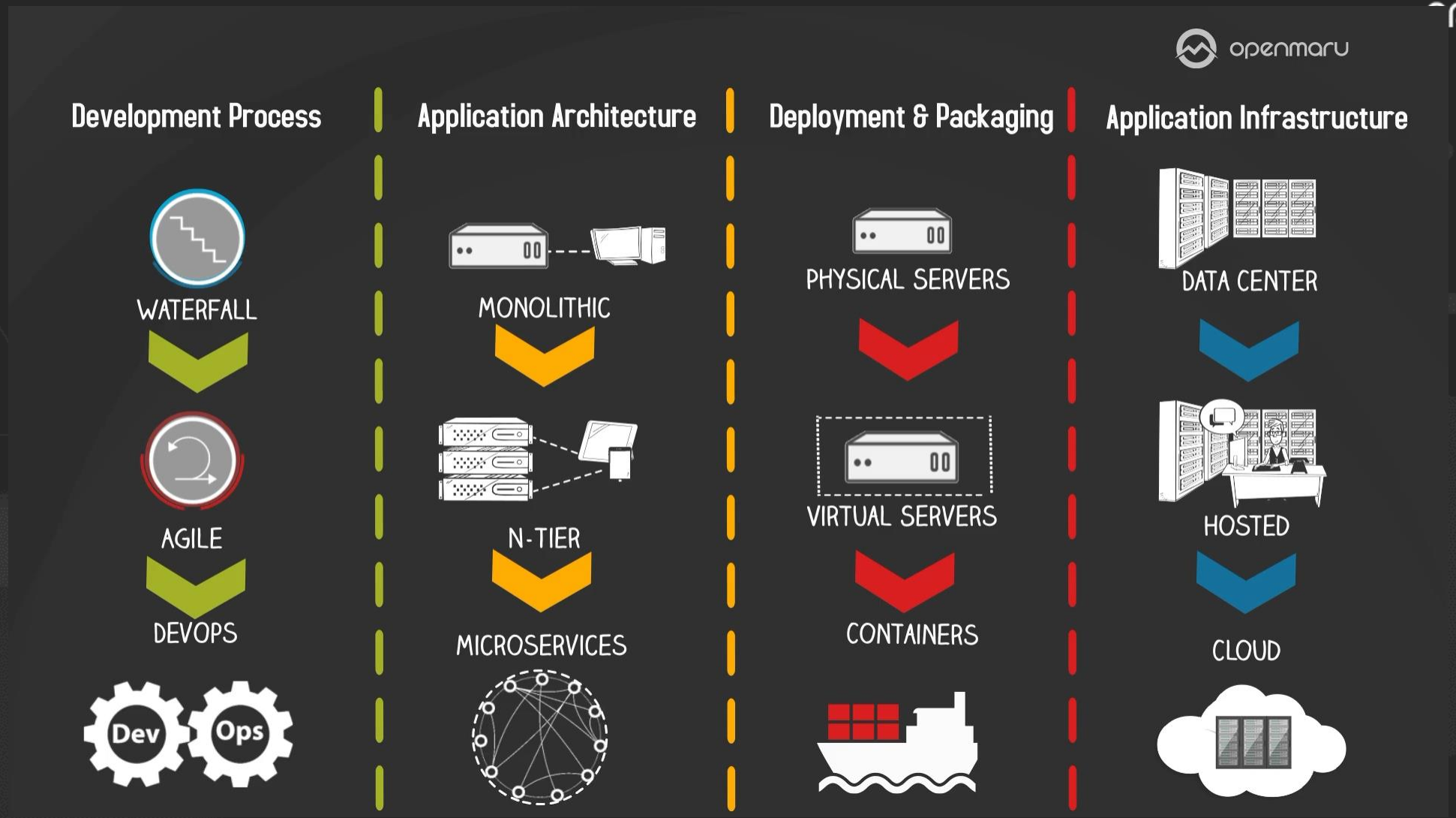
- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능 평가 1위

05

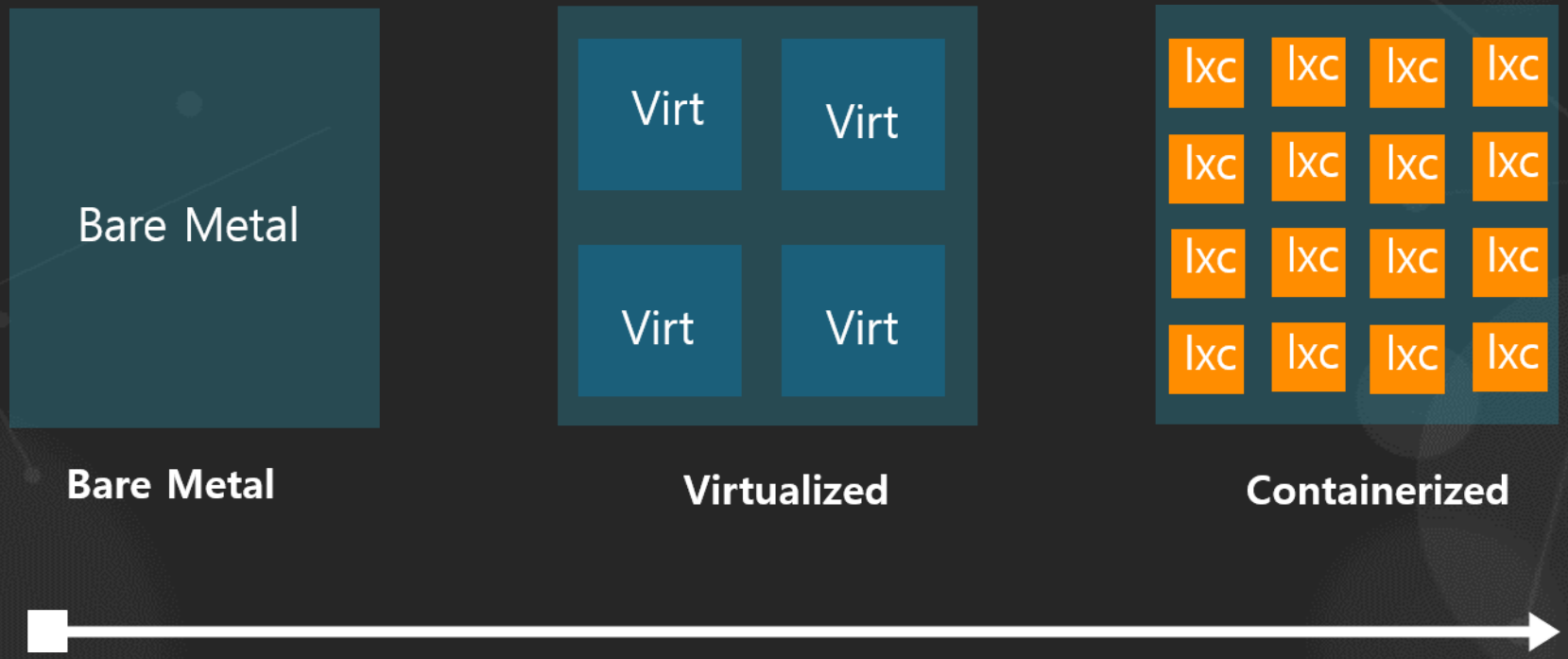
최신 IT 기술 지원

- **WebSocket 을 통한 포트 사용 최소화**
- **아마존 클라우드/ 컨테이너 환경에서 오토스케일링 지원**
- **PaaS지원 APM 시장점유율 1위**

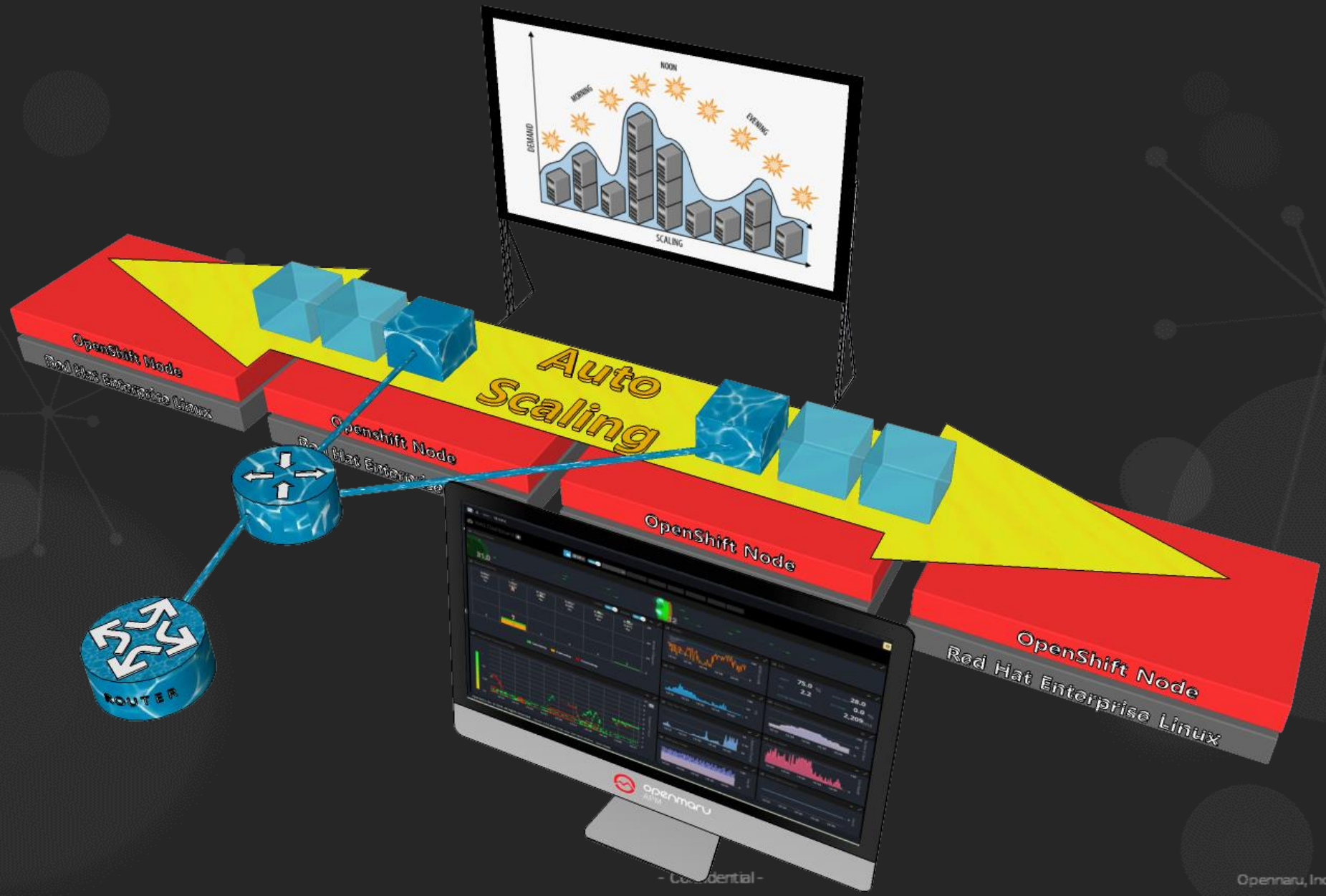
IT Evolution



Evolution of Infrastructure Architectures



OpenShift Auto Scaling 과 OPENMARU APM



redhat.

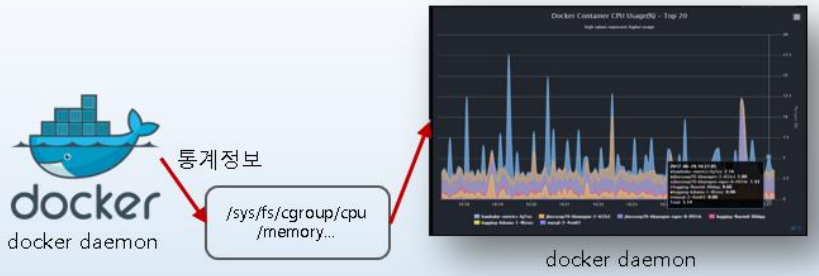
Red Hat OpenShift Container Platform
오토스케일링 데모

한국레드햇
김현수 이사

Paas 환경에서 OPENMARU APM이 필요한 이유는?

- Docker 환경은 가상OS환경이기 때문에 시스템 리소스관리가 어려움
- 유동IP이며, 멀티캐스트를 지원하지 않는 환경

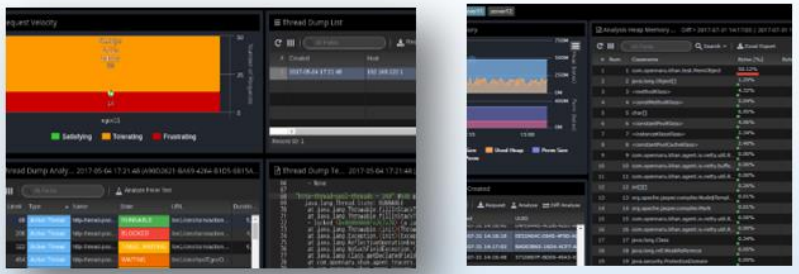
▪ 가상 OS 환경에서 CPU/Memory/Disk/Network 정보 실시간 제공



▪ 폐기된 POD 인스턴스에 대한 검색 및 상태 정보 제공

▪ 오토스케일링 시에 부하 분산과 **pod 별 처리 현황 파악**

▪ WAS 장애 상황에서 필수적으로 필요한 분석 도구 제공



OPENMARU APM – PaaS 형 APM

- PaaS 형 APM 과 기존 APM 기능 비교

구 분	세부 항목	PaaS 형 APM (OPENMARU APM)	기존 APM 제품
기존 온-나라 시스템	Java	• 모니터링 지원	• 모니터링 지원
	WAS	• 모니터링 지원	• 모니터링 지원
클라우드 온-나라 시스템	PaaS 환경	• Docker Daemon 과 관련 이미지 정보 제공 • POD 에 대한 리소스 정보 제공 (CPU/Memory/Disk/Network)	• 모니터링 지원 불가
	오토스케일링	• 오토스케일링 관련 리소스 정보 제공 • WAS 상태 정보 제공	• 모니터링 지원 불가
	PaaS 에서 Java 정보	• POD 상에서 실행되는 Java 가상 머신 정보 제공 (Heap , Java 상태 정보 등)	• 모니터링 지원 불가
	PaaS 에서 WAS 정보	• POD 상에서 실행된 WAS 에 대한 정보	• 모니터링 지원 불가
	POD 상태 정보 제공	• 폐기된 POD 에 대한 검색 지원 • 과거 POD 에 대한 정보 제공	• 모니터링 지원 불가

컨테이너 모니터링 기능

컨테이너 자원 모니터링

- 컨테이너 환경에서는 전반적인 정보(분배된 Pod 위치, 컨테이너 리소스 등)를 모니터링 해야 한다.
- APM에서는 컨테이너 환경을 운영자가 편리하게 컨테이너(pod) 모니터링을 할 수 있다.

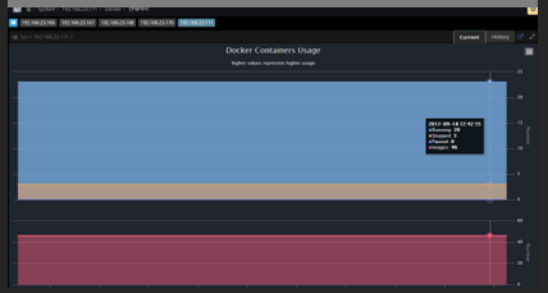
▶ 컨테이너(pod)에 대한 모니터링 (System > 호스트 > Docker > CPU)

Container

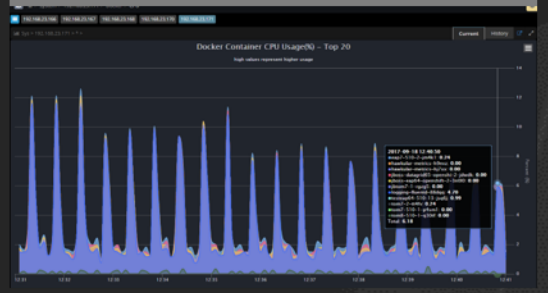
Container 환경 모니터링 메뉴

도커 모니터링 정보

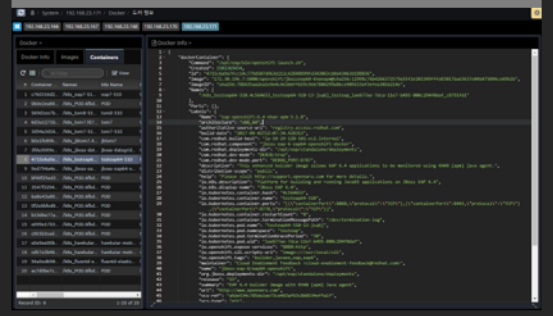
도커 이미지의 개수 및 컨테이너 상태



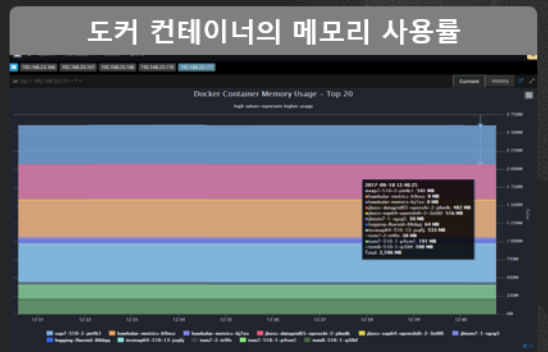
도커 컨테이너의 CPU 사용률



도커 컨테이너, 이미지, 도커 데몬 정보



도커 컨테이너의 메모리 사용률



컨테이너에서 운영되는 WAS의 성능 모니터링 기능

컨테이너 내의 WAS 자원 및 애플리케이션 모니터링

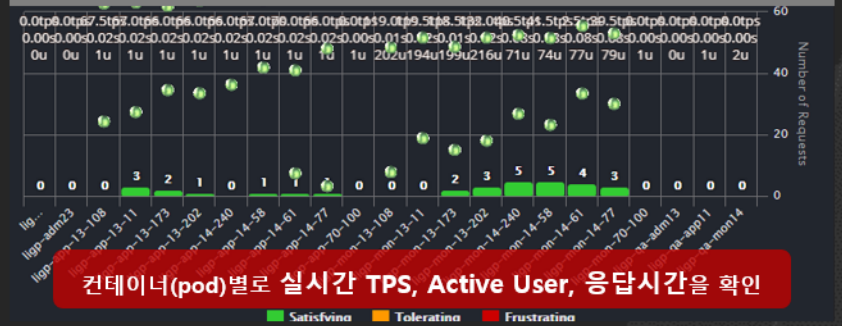
- 컨테이너 환경에서 WAS 자원 및 WAS 애플리케이션 모니터링시 특정 도구를 사용 하거나 도커 컨테이너에 접근하여 로그를 확인해야 한다.
- APM에서는 특정 그룹별로 연결되어 있는 WAS 자원 및 애플리케이션을 모니터링 한다.

▶ 컨테이너(pod) 내의 WAS 자원 및 WAS 애플리케이션 모니터링 기능 제공

WAS | Application | Container

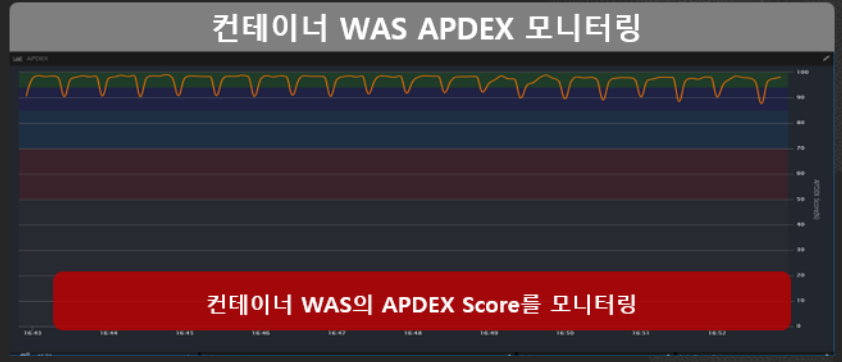
컨테이너 WAS 자원(Heap 메모리) 모니터링

컨테이너 WAS 애플리케이션 실시간 모니터링



컨테이너 WAS 오류율 모니터링

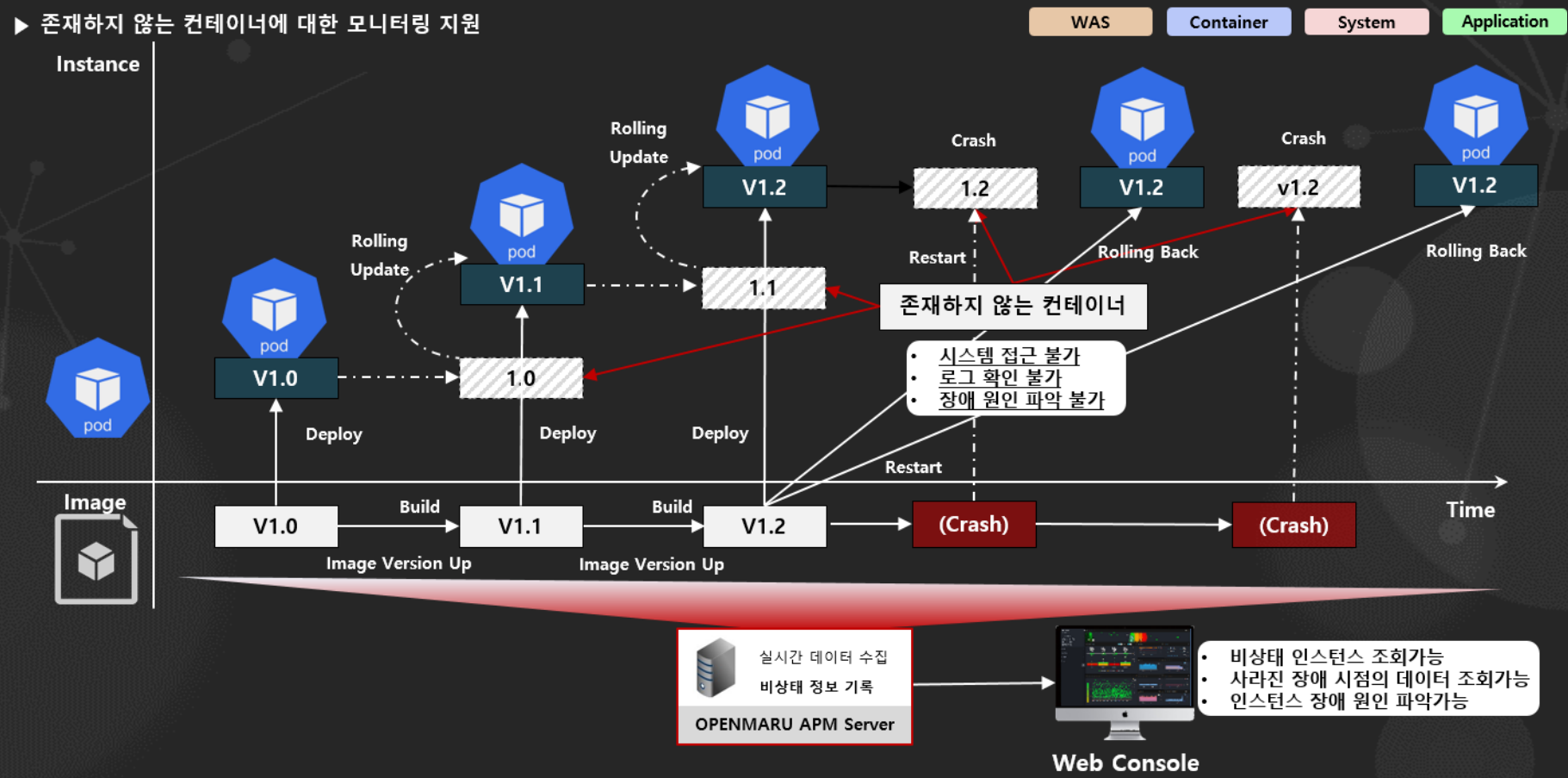
컨테이너 WAS APDEX 모니터링



비상태 WAS 인스턴스에 대한 모니터링 기능

PaaS 환경의 비상태 WAS 인스턴스에 대한 모니터링 기능

- APM은 PaaS 환경에서 존재하지 않는 컨테이너가 중지된 후 장애원인 파악을 위한 정보를 파악할 수 있는 방법을 제공해야 한다.
- APM에서는 존재하지 않는 컨테이너에 대한 정보를 보관하고 있어 장애원인을 파악하여 정확한 조치를 취할 수 있다.



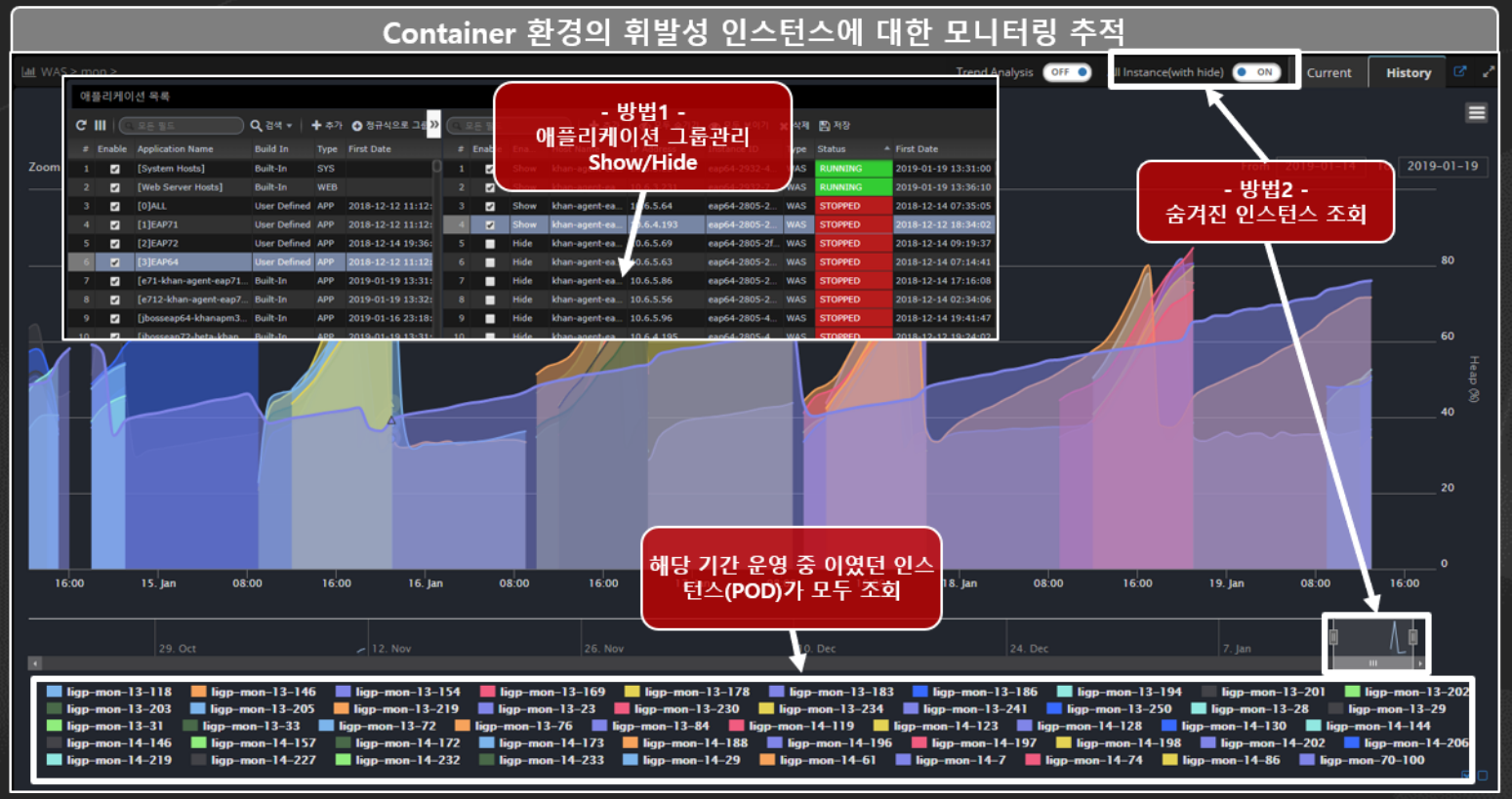
휘발성 인스턴스 모니터링 기록 추적 기능

중지된 컨테이너에 대한 추적 기능

- Container 환경에서 중지되면 로그 분석 및 Container 장애 원인 추적이 불가능하다.
- APM에서는 휘발성 인스턴스의 모든 기록을 남겨놓고 있어, 과거의 Container 데이터를 조회하여 모니터링 기록을 추적, 장애 원인파악이 가능하다.

▶ Container 환경에서의 휘발성 인스턴스에 대한 모니터링 추적(홈 > WAS > 애플리케이션 그룹 관리)

WAS | Container | System | Application



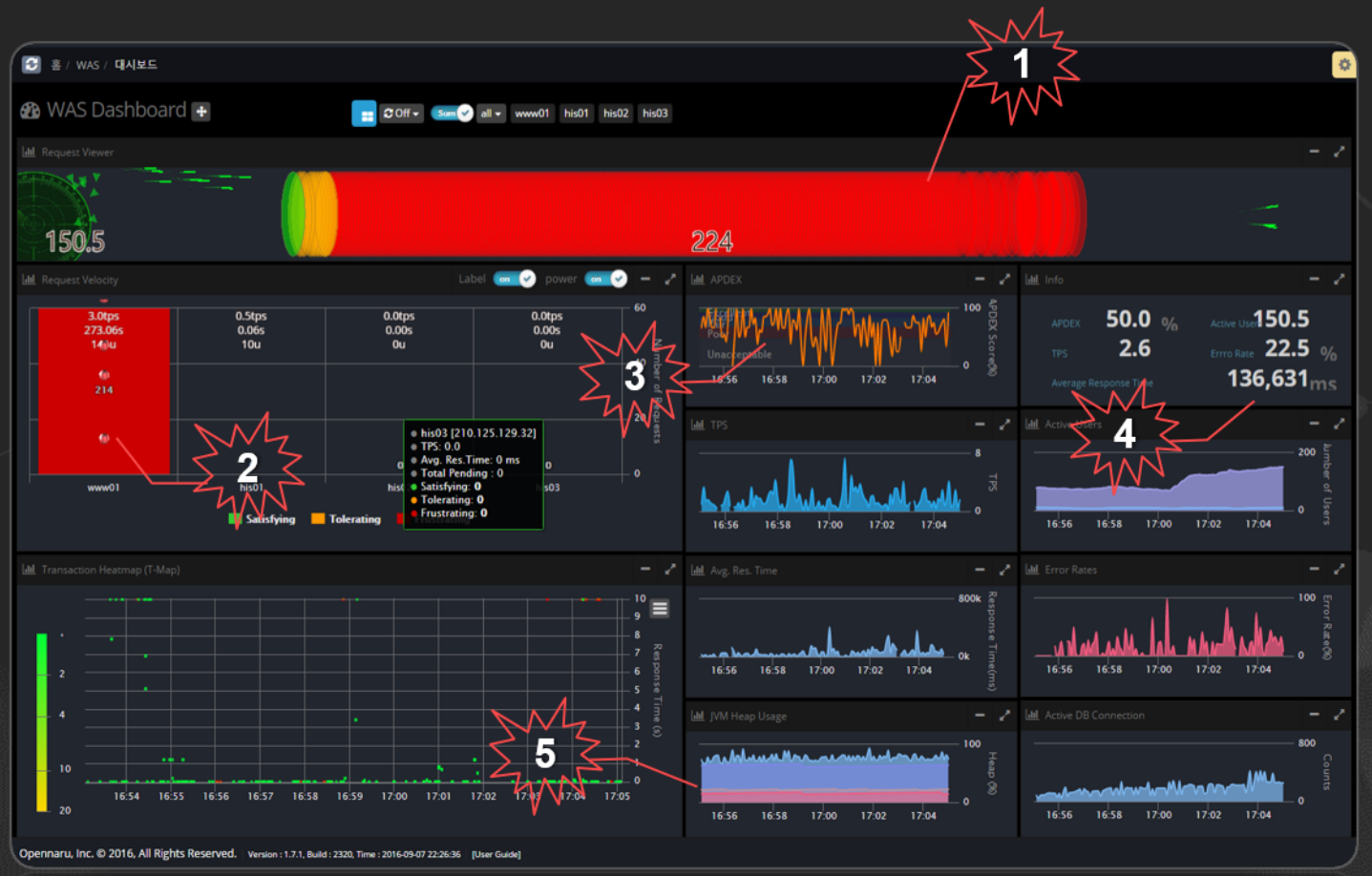

Micro Service Archtecture
OPENMARU APM Use Case
openmaru
APM

Application Performance Management

사건#1
웹사이트가 멈추었습니다.

openmaru
APM

사건#1 현장 – 웹사이트가 멈추었습니다.



1 Queue 에 224 개 쌓임

2 www01 인스턴스에 쌓임

3 APDEX 지수 50% 이하

4 평균응답시간 136 초

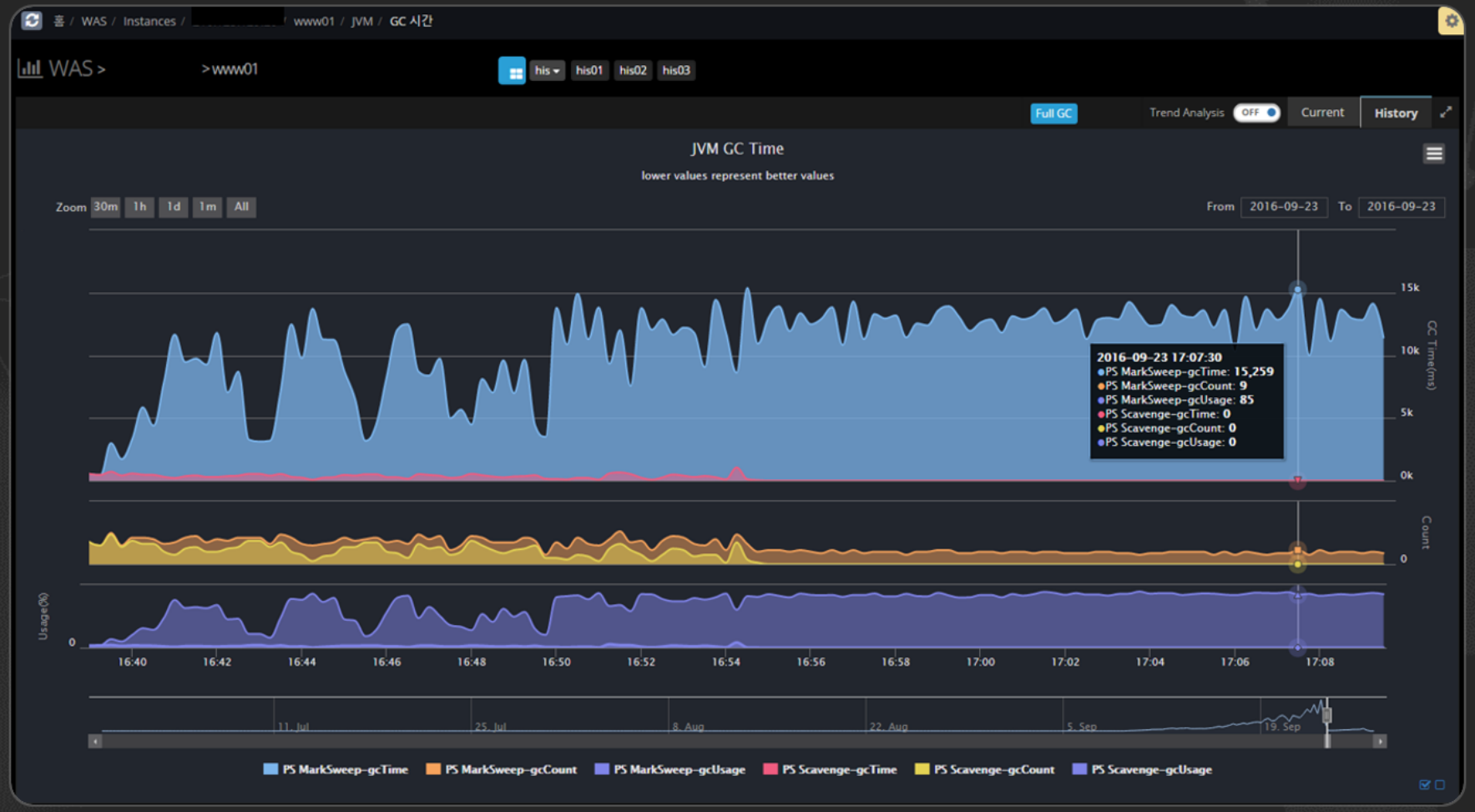
5 JVM Heap Usage 가 거의 100%

사건#1 용의자 – JVM Heap Memory

- 15일간의 메모리 사용량의 추세를 분석하면, 아래와 같이 점차적으로 증가
- JVM Full GC에 걸리는 시간이 점차적으로 늘어나는 상황

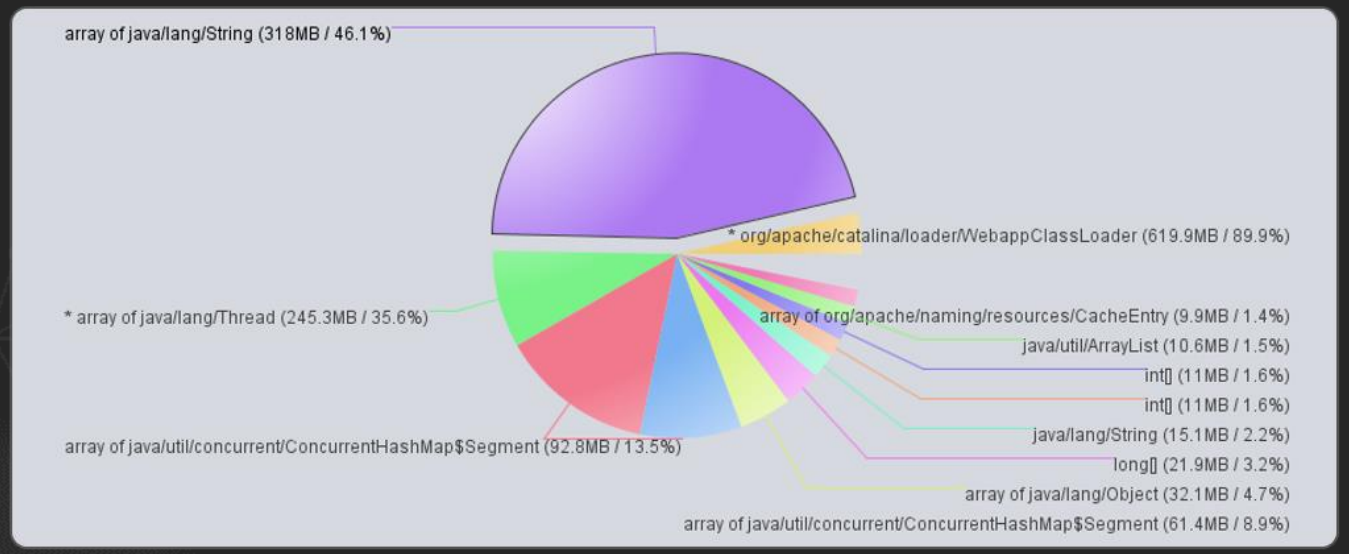
사건#1 용의자 – JVM Heap Memory

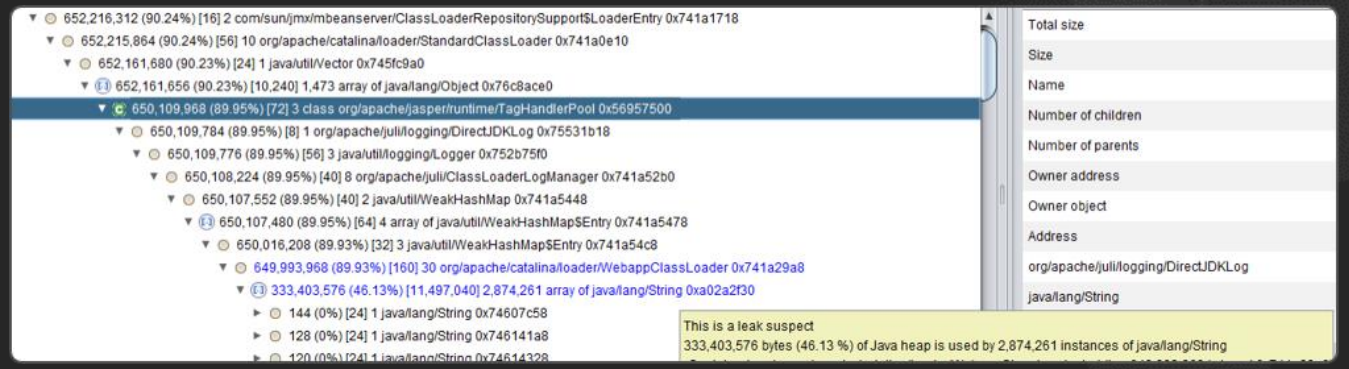
- 장애시점에서는 JVM에서 Full GC만 발생하고 있어, 다른 요청이 전혀 처리되지 못하는 상황으로 Full GC에 만 10~15초 가량 소요.



사건#1 범인 검거 – Tomcat의 TagHandlerPool

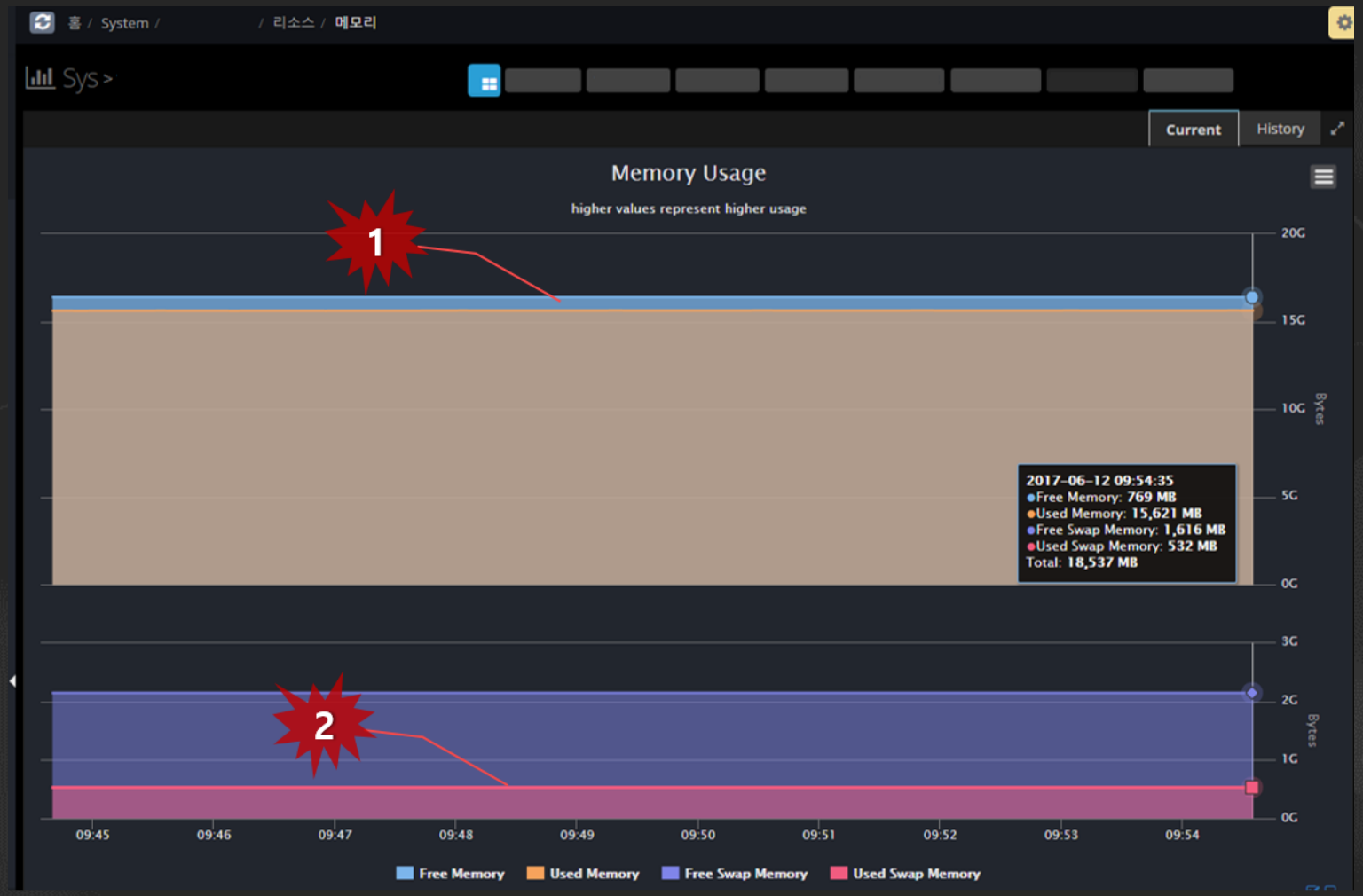
- 88.95%의 메모리를 Tomcat의 TagHandlerPool에서 점유하고 있음.
- Tomcat Logger에서 사용하는 287만개의 String 객체가 46.13%의 메모리를 점유






Application Performance Management
Episode #2
시스템 메모리 고갈
openmaru
APM

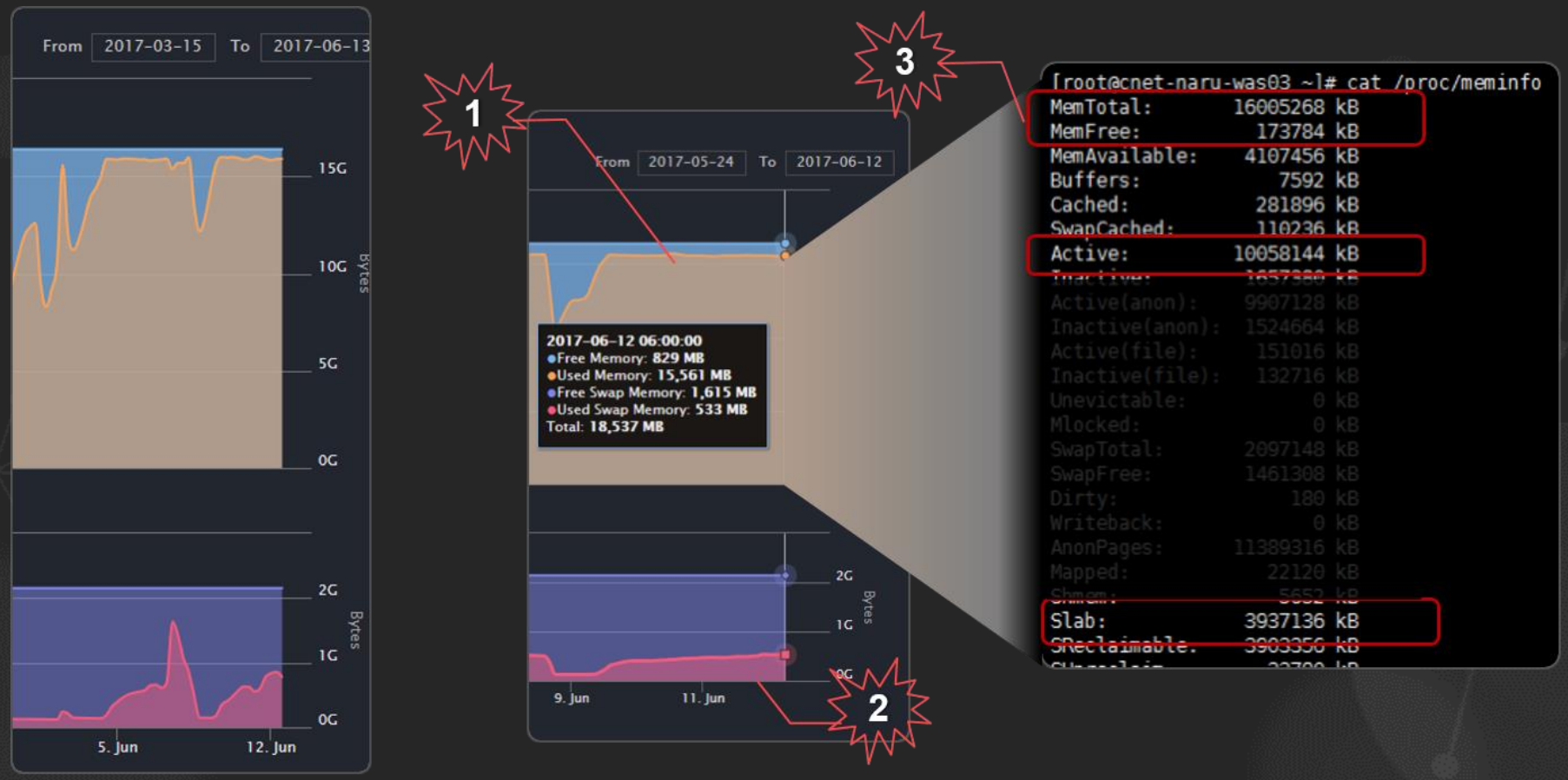
Episode #2 현장 – System 메모리



1 사용 가능한 메모리를 거의 소진

2 Swap 메모리를 500MB 사용

Episode #2 현장 – System 메모리

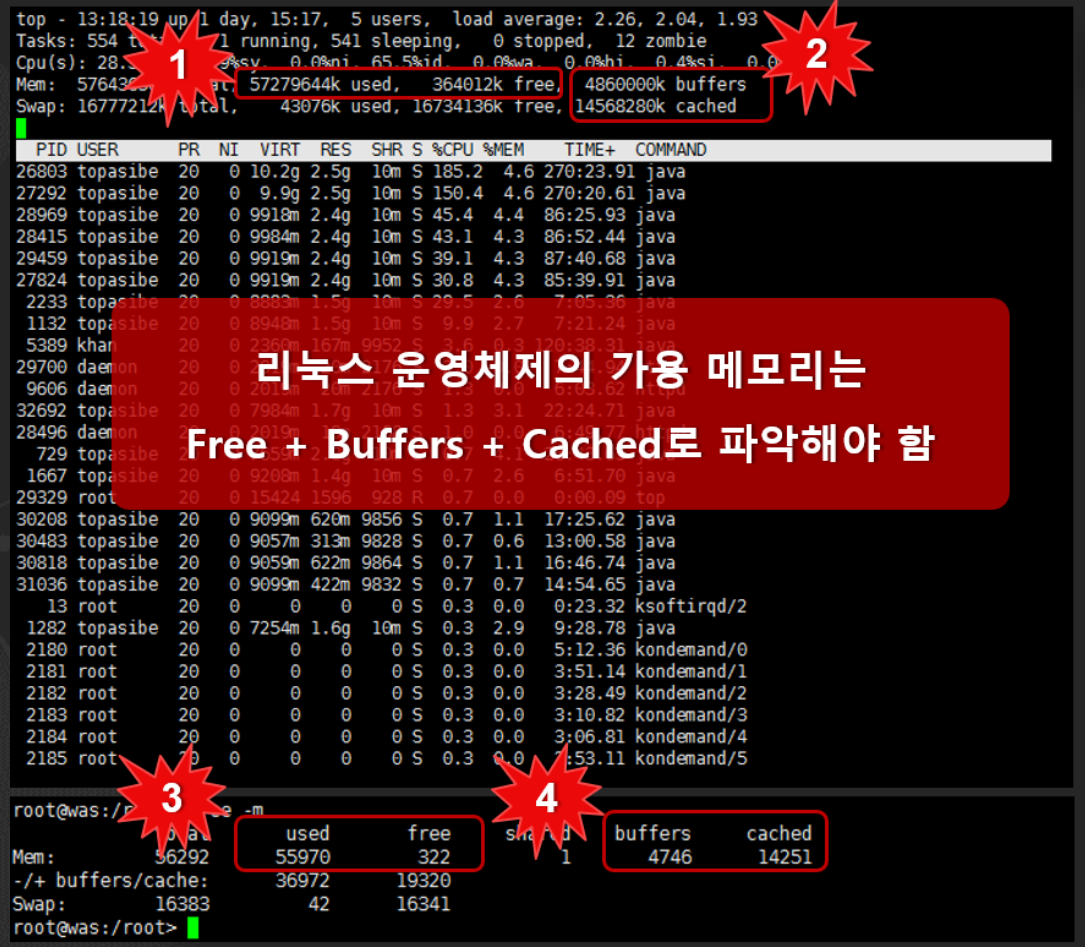


1 리눅스 운영체제의 가용 메모리가 829MB밖에 없는 상황

2 리눅스 운영체제의 SWAP 메모리를 533MB가량 사용하고 있음

3 /proc/meminfo 를 확인하여 상세 메모리 사용량 조사

[참조] 리눅스 top, free 명령으로 메모리 모니터링

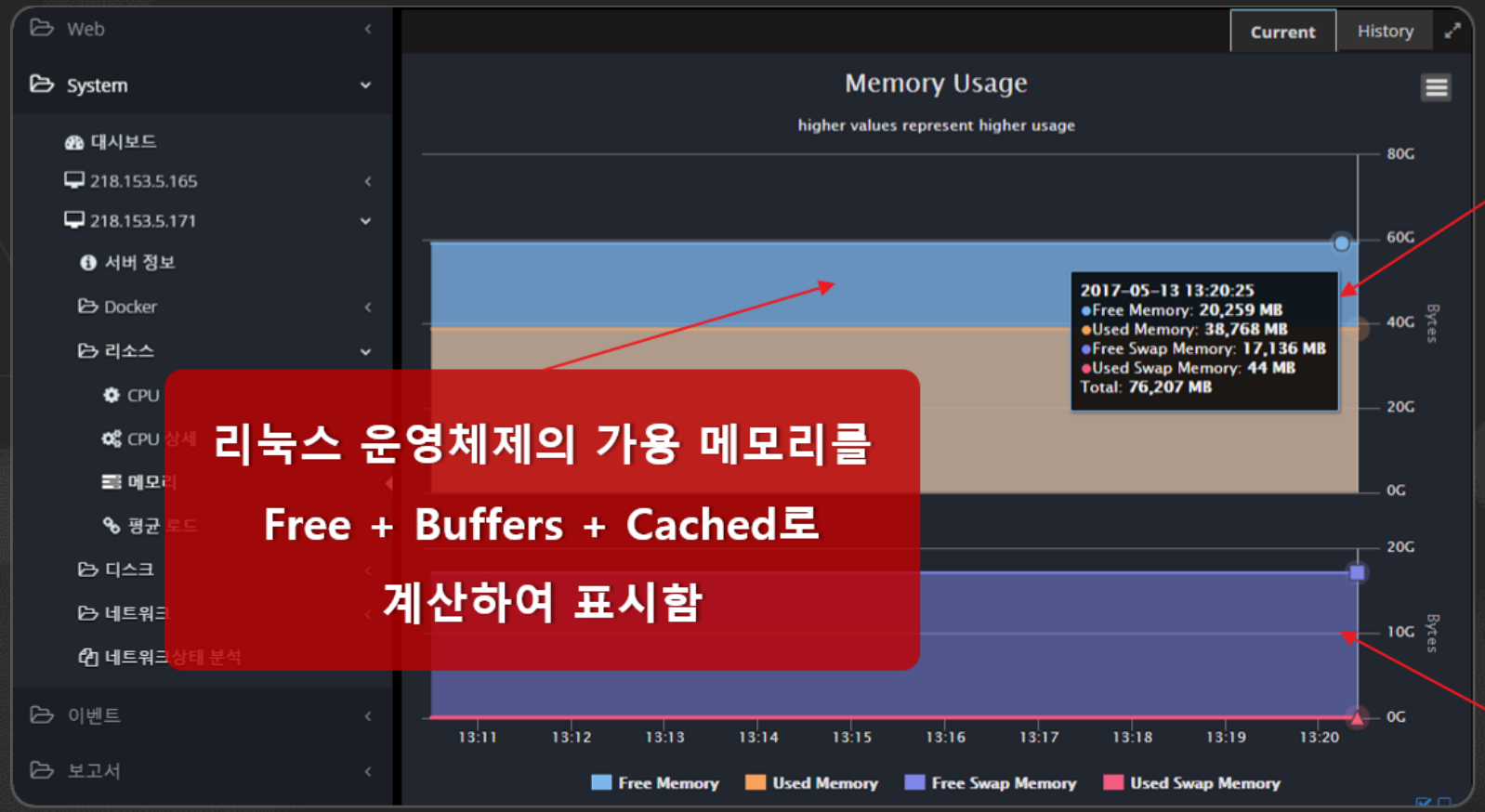


Linux 메모리 관리방식

- Linux 커널에서는 Disk I/O를 줄여 성능을 향상시키기 위해 가용한 메모리를 최대한 캐시 용도로 사용함(used)
- used는 필요시 즉시 해제하여 사용할 수 있는 buffers 및 cached등을 포함하여 계산됨
- 리눅스에서는 일정 시간 후 메모리의 95% 이상이 used로 표시됨
- Linux에서의 가용 메모리는 free + buffers + cached로 파악해야 함

1. free 메모리가 약 3.6G 표시됨
2. 사용가능한 buffers, cache가 각각 4.8G, 14.5G로 표시
3. free -m 명령으로 보면 free 메모리가 322M 가량으로 표시됨
4. buffers, cached가 각각 4.7G, 14.251G 표시됨

[참조] 리눅스 top, free 명령으로 메모리 모니터링



리눅스 운영체제의 가용 메모리를
Free + Buffers + Cached로
계산하여 표시함

물리적 메모리를 실제 애플리케이션 사용한 메모리와 가용한 메모리로 표현

Swap 메모리를 사용한 메모리와 가용한 메모리로 표현

Episode #2 현장 – System 메모리

Episode #2 현장 – System 메모리

Episode #2 용의자 – Linux 커널 캐시 메모리 상태(slabtop)

openmaru

```
[root@cnet-naru-was03 ~]# cat /proc/meminfo
MemTotal:        16005268 kB
MemFree:           173784 kB
MemAvailable:     4107456 kB
Buffers:             7592 kB
Cached:            281896 kB
SwapCached:        110236 kB
Active:          10058144 kB
Inactive:         1657380 kB
Active(anon):     9907128 kB
Inactive(anon):   1524664 kB
Active(file):      151016 kB
Inactive(file):    132716 kB
Unevictable:      [illegible]
Mlocked:          [illegible]
SwapTotal:        [illegible]
SwapFree:         [illegible]
Dirty:                180 kB
Writeback:              0 kB
AnonPages:       11389316 kB
Mapped:             22120 kB
Shmem:               5652 kB
Slab:             3937136 kB
SReclaimable:     3903356 kB
SUnreclaim:         33780 kB
KernelStack:        20144 kB
PageTables:         30152 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:     10099780 kB
Committed_AS:    11402420 kB
VmallocTotal:    34359738367 kB
VmallocUsed:        41780 kB
VmallocChunk:    34359690748 kB
HardwareCorrupted:      0 kB
AnonHugePages:    8792064 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
DirectMap4k:        61440 kB
DirectMap2M:     16707584 kB
```

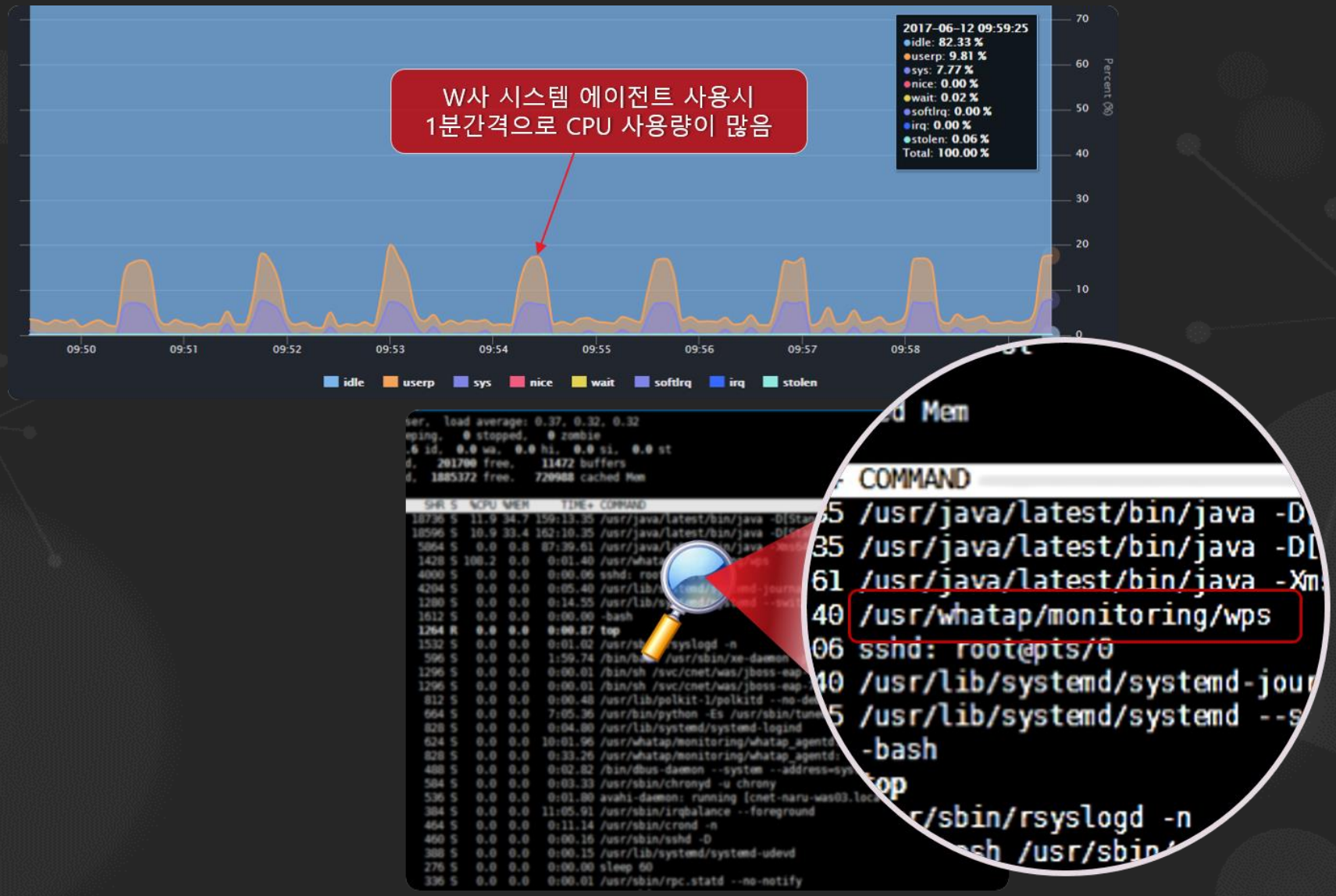
리눅스 커널 메모리중 slab
메모리를 3.9GB 사용하고 있음
→ slabtop으로 분석 필요

```
 Active / Total Objects (% used)    : 9165609 / 9198068 (99.6%)
 Active / Total Slabs (% used)      : 403490 / 403490 (100.0%)
 Active / Total Caches (% used)     : 73 / 99 (73.7%)
 Active / Total Size (% used)       : 3736800.07K / 3753067.05K (99.6%)
 Minimum / Average / Maximum Object : 0.01K / 0.41K / 8.00K

  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
4461219 4460718  99%    0.19K 212439       21    849756K dentry
4442184 4441608  99%    0.64K 185091       24   2961456K proc_inode_cache
 31232  31232 100%    0.01K     61      512       244K kmalloc-8
 25344  25344 100%    0.02K     99      256       396K kmalloc-16
 22032  22032 100%    0.11K    612       36      2448K sysfs_dir_cache
 17920   7855  43%    0.57K    640       28     10240K radix_tree_node
 17472  14212  81%    0.06K    273       64      1092K kmalloc-64
 16000  15165  94%    0.03K    125      128       500K kmalloc-32
 11832  11832 100%    0.08K    232       51       928K selinux_inode_security
 11520  11520 100%    0.03K     90      128       360K jbd2_revoke_record_s
 10633   5897  55%    1.03K    343       31     10976K nfs_inode_cache
 10323 [illegible]                                       vm_area_struct
  9690 [illegible]                                       shared_policy_node
  8364 [illegible]                                       ext4_extent_status
  8184 [illegible]                                       ext4_inode_cache
  7911 [illegible]                                       inode_cache
  7552 [illegible]                                       anon_vma
  7488 [illegible]                                       kmalloc-256
  7182   6580  91%    0.09K    171       42       684K kmalloc-96
  6825   3848  56%    0.10K    175       39       700K buffer_head
  6464   6464 100%    0.06K    101       64       404K ext4_free_data
  6290   6290 100%    0.12K    185       34       740K fsnotify_event
  6272   6272 100%    0.07K    112       56       448K Acpi-ParseExt
  4672   3848  82%    0.12K    146       32       584K kmalloc-128
  4137   3914  94%    0.19K    197       21       788K kmalloc-192
  2560   2329  90%    0.50K     80       32      1280K kmalloc-512
  2448   2448 100%    0.04K     24      102        96K Acpi-Namespace
  2336   2236  95%    1.00K     73       32      2336K kmalloc-1024
  2028   2028 100%    0.10K     52       39       208K blkdev_ioc
  1825   1796  98%    0.62K     73       25      1168K sock_inode_cache
  1440   1440 100%    0.66K     60       24       960K shmem_inode_cache
  1404   1224  87%    0.11K     39       36       156K jbd2_journal_head
```

리눅스 커널 메모리중 dentry 840MB
와 proc_inode_cache 2960MB를
사용하고 있음
slab

Episode #2 검거 – Linux 커널 캐시 메모리 상태



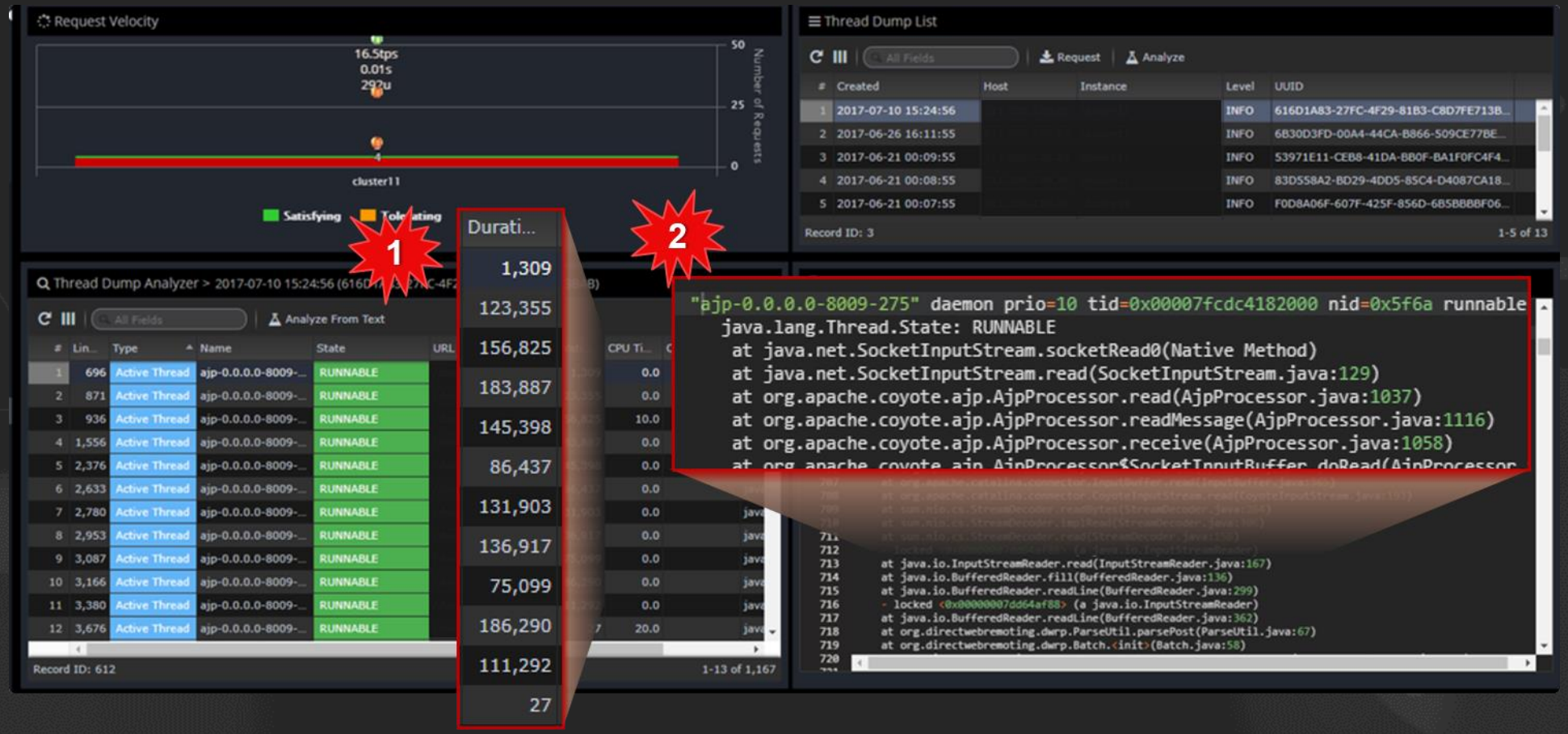
openmaru

Episode #3 현장 – Pending Transaction이 급증

1 거의 모든 인스턴스에서 Queue 에 쌓임

2 리눅스 운영체제의 SWAP 메모리를 533MB가량 사용하고 있음

Episode #3 용의자 – 인스턴스의 스레드 덤프 분석



1
- 해당 인스턴스의 스레드 덤프를 분석
- 현재 수행중인 스레드가 최대 180초 가량 수행중인 상태

2
- 지연되고 있는 트랜잭션(Active Thread)는 대부분 AJP
- 데이터를 읽고 있는 상태(socketRead)임

Episode #3 검거 – 시스템 점검 보고서에서 Native 적용 권장

1

Native 사용 점검

점검 항목	권장값	설명
Tomcat Native Component 적용여부	적용 안됨	lib 디렉토리에 파일들이 설치되어 있는지 확인한다.

※ APR(Apache Portable Runtime)가 설치되면 빠른 성능을 낼 수 있다.

5.2 Thread Pool 점검

파라미터	권장값	현재 설정값	설명
AJP Connector	maxThread=250	기본값 200	AJP 로 사용되는 스레드 풀의 개수
HTTP Connector	maxThread=250	기본값 200	HTTP 연결시 사용되는 스레드 풀의 개수

- 운영환경에서의 처리량에 따라 스레드 풀의 개수를 설정한다.

$SERVER_NAME/conf/server.xml

```
<Connector protocol="HTTP/1.1" port="8080" address="${jboss.bind.address}"
           connectionTimeout="20000" redirectPort="8443" URIEncoding="utf-8" />
<Connector protocol="AJP/1.3" port="8009" address="${jboss.bind.address}"
           redirectPort="8443" URIEncoding="utf-8" />
```

오픈나루 16 opennaru.com

2

JVM 옵션	GC 옵션 명시 필요
Tomcat Native(APR)	거래량이 많을 경우 APR 을 적용하는 것이 효율적임

9. 점검 종평

JVM 옵션	GC 옵션 명시 필요
Tomcat Native(APR)	거래량이 많을 경우 APR 을 적용하는 것이 효율적임
웹서버 설정	Static Content 분리 필요(웹폰트, 영상 파일)
어플리케이션 소스	.xml 404 호출에 대한 어플리케이션 소스 확인
오류율	오류율에 대한 개선 필요
DB Query	느린 쿼리에 대한 개선 포인트 분석 필요


openmaru