

OPENMARU APM 소개 (Application Performance Management)

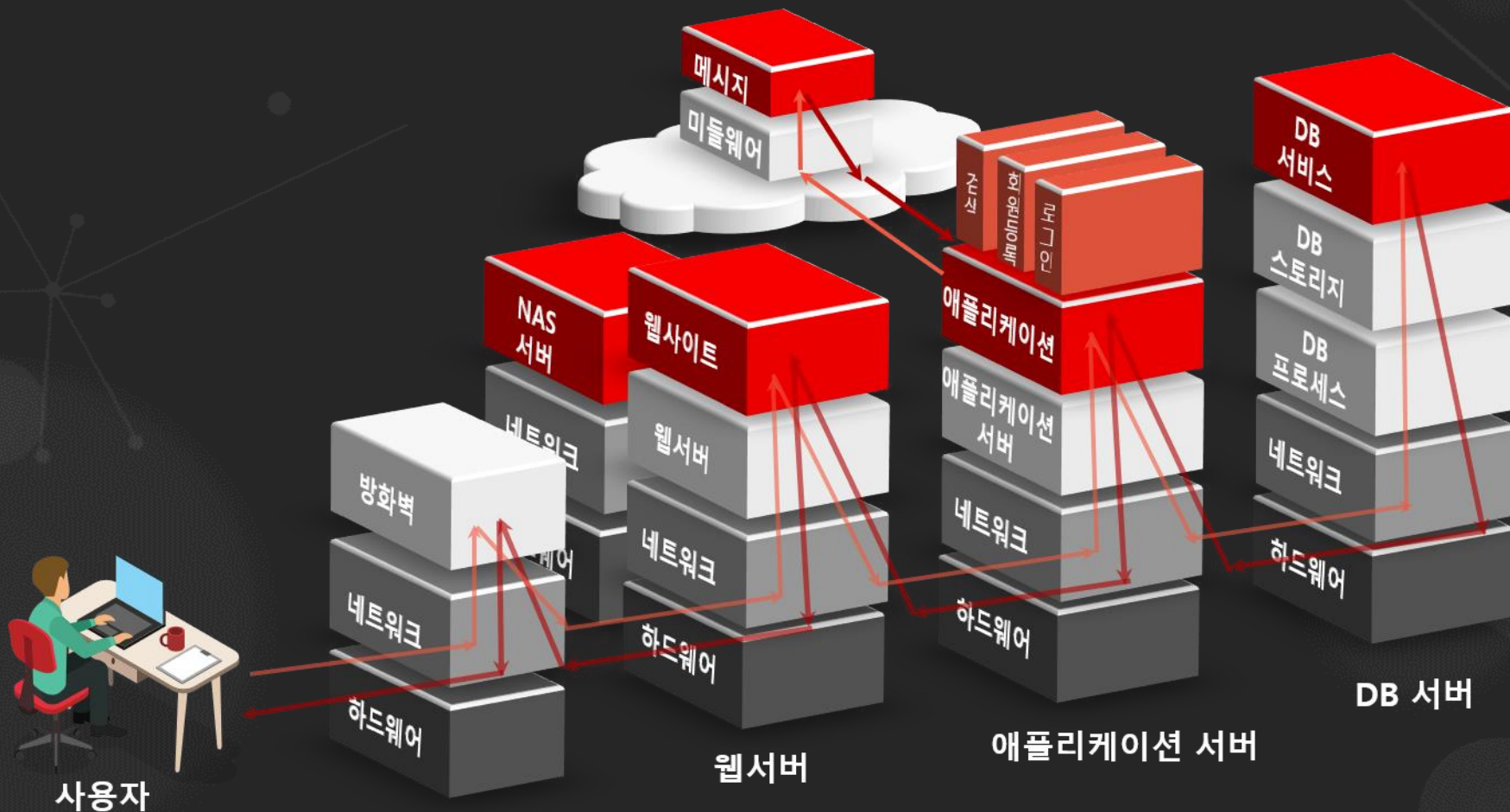
Application Performance Management



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



QA팀

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



openmaru
APM



운영팀

- 성능에 대한 최적화와 서비스 정상 유/무
- 신속한 장애 감지와 원인 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



[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

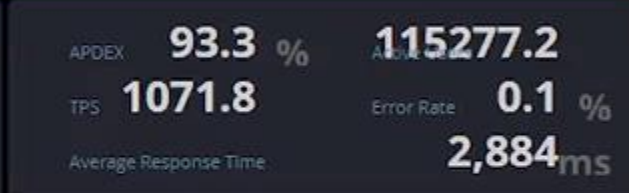
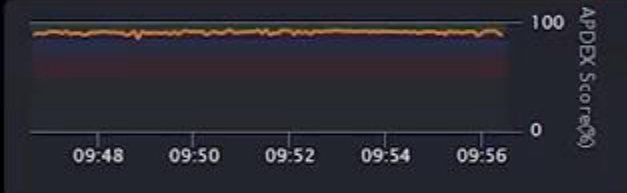
Request Viewer



Request Velocity



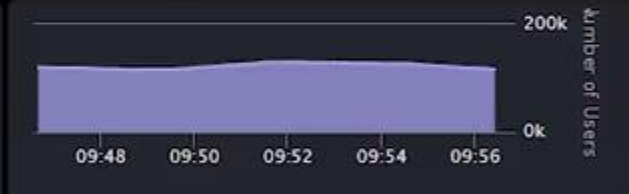
APDEX



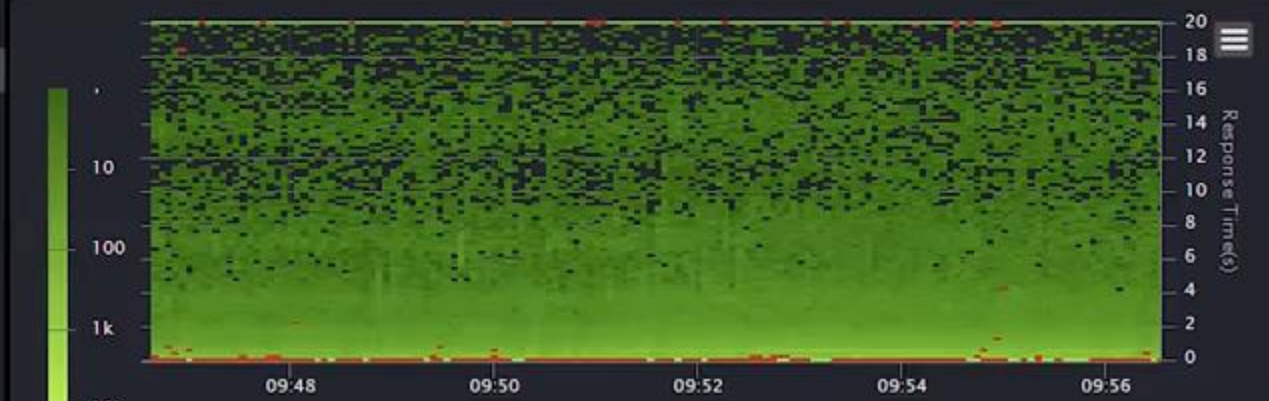
TPS



Active Users



Transaction Heatmap (T-Map)



Avg. Res. Time



Error Rates



JVM Heap Usage

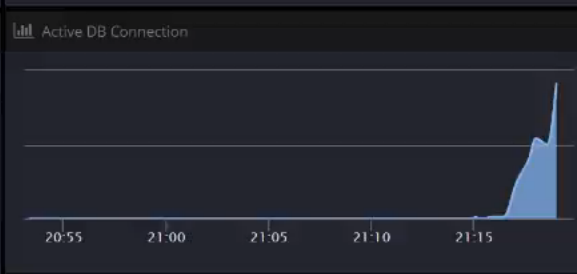
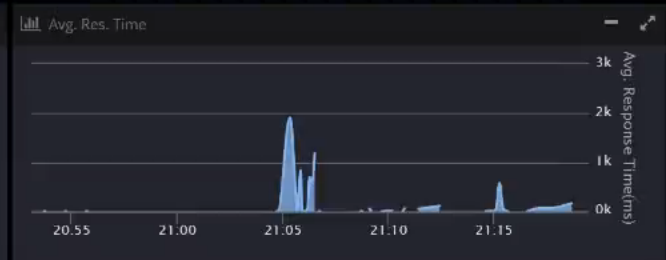
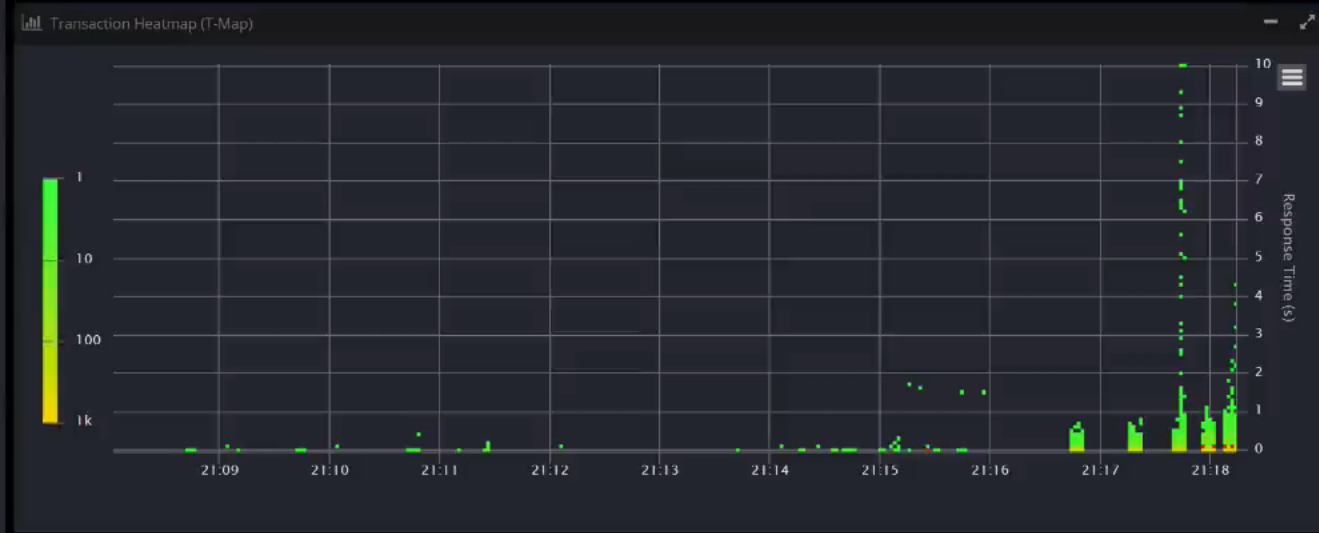
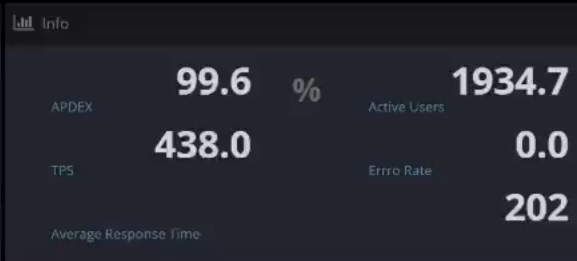
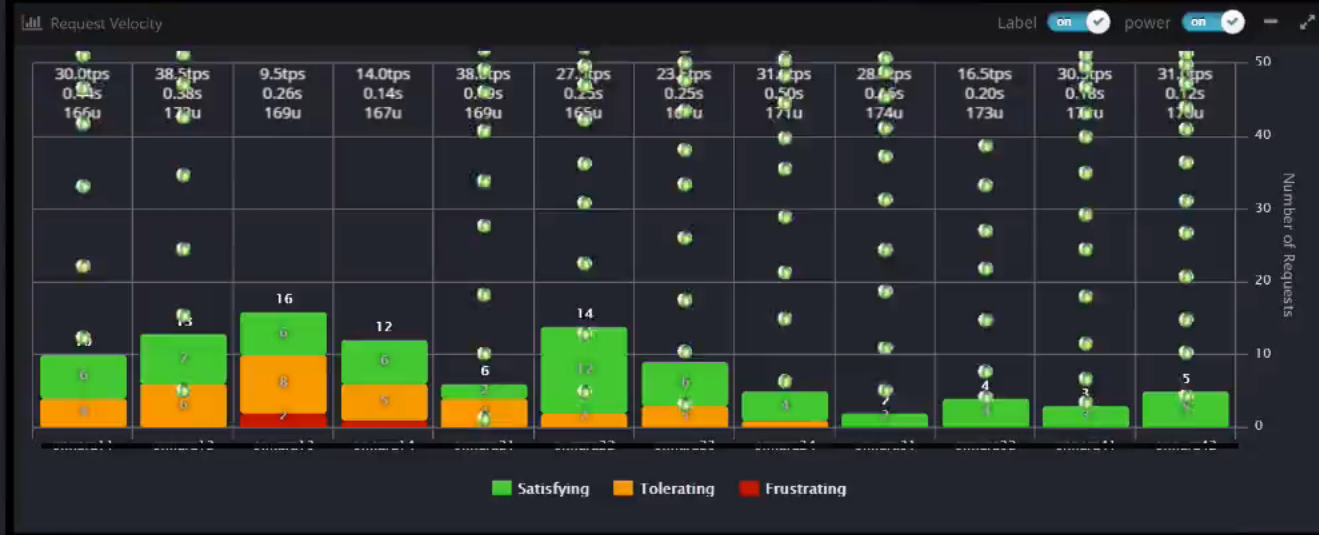
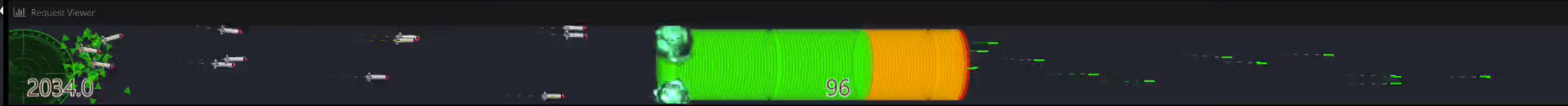


Active DB Connection

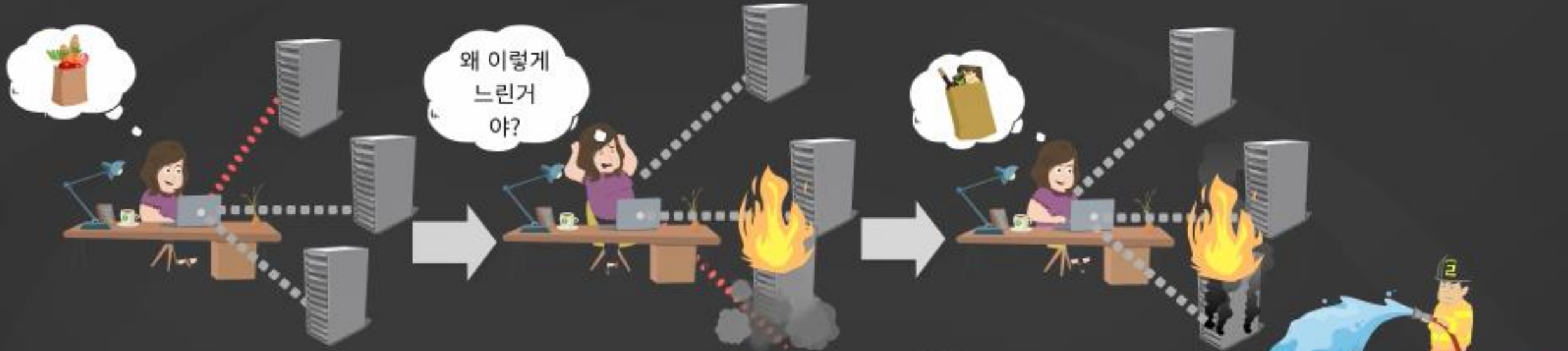


정상 서비스 상황

Off Sum camera1 camera12 camera11 camera21 camera22 camera21 camera24 camera23 camera14 camera12 camera13 camera11 camera51 camera52



화재 경보와 소방관



WAS엔지니어를 통한 문제 해결

APM을 통해 고객보다 먼저 장애와 성능 이슈 파악



openmaru
APM

이것은
부채군요.



이것은
창이군요.



이것은
벽이군요.



이것은
끈이군요.



이것은
나무네요.



이것은
뱀이군요.



시스템 구축 후 - 고민들



- ✓ 오늘 온라인방문자 수가 급격히 줄었는데요?
- ✓ 다음달에 온라인 광고를 해야 하는데 문제는 없겠죠?
- ✓ 주기적으로 장애가 발생하는데 대책은 없나요?
- ✓ 사용자가 조금만 늘어도 시스템이 다운이니 불안해서 뭘 할 수가 없네요.

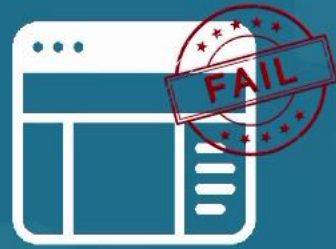
- ✓ 확인해 봤는데 소스는 문제가 없어요. WAS 문제 아닌가요?
- ✓ 느린 페이지에서 사용하는 DB 쿼리를 보고 싶는데요.
- ✓ 개발할 땐 빠르는데 운영시스템에서만 느린 이유가 무엇인가요?



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



openmaru
APM



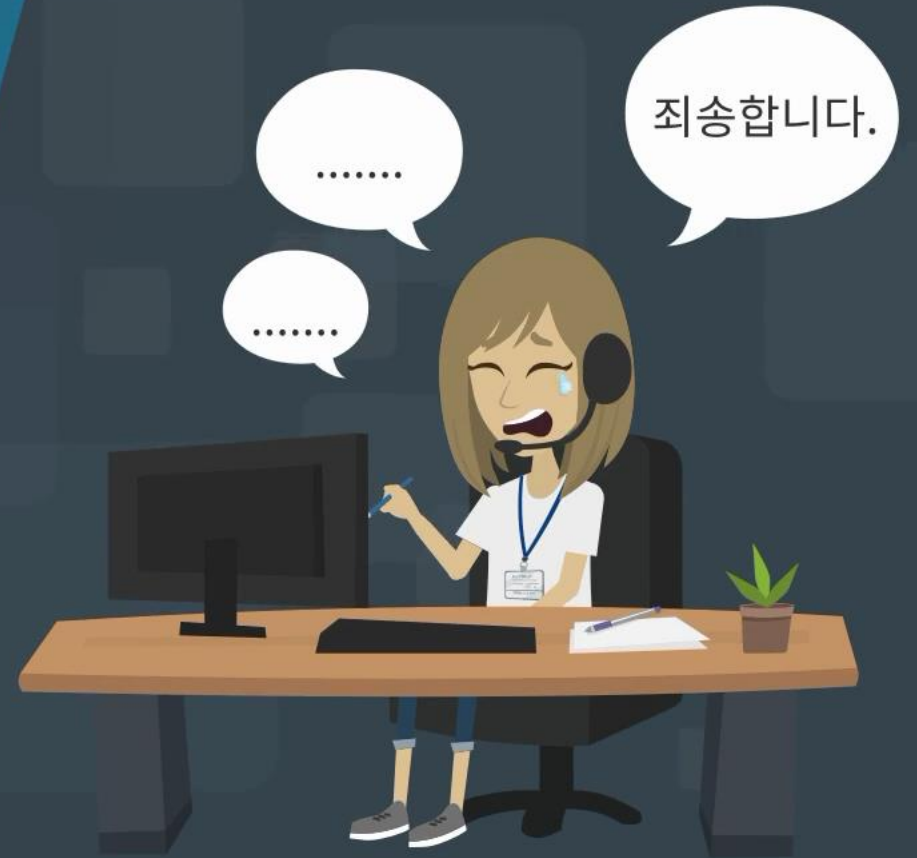
특정페이지 오류



느린 응답속도



검색 오류/ 페이지 찾기 오류



네트워크, 서버 모두 잘 운영되고 있습니다.
심지어 어제보다도 성능이 좋은데요.

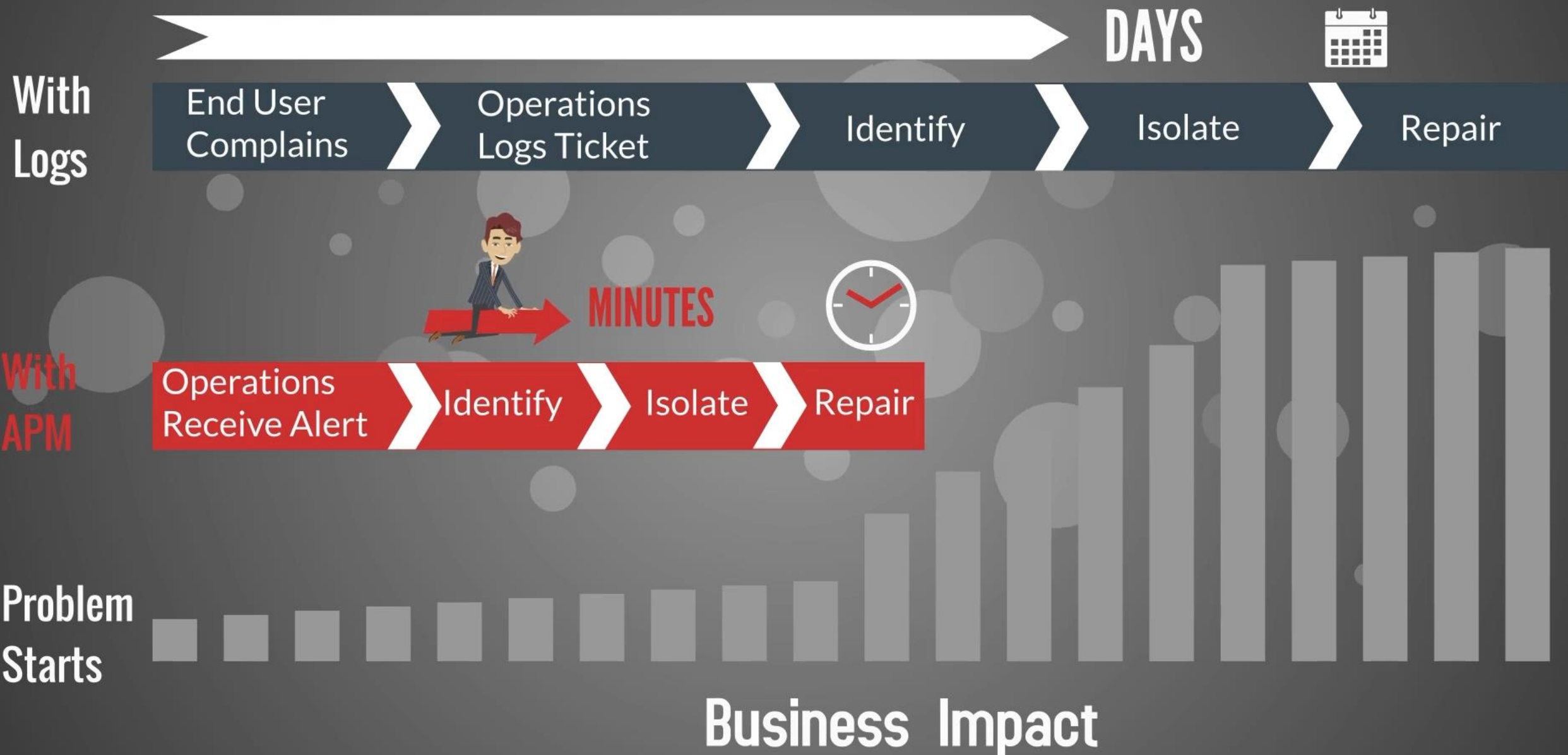


네트워크 OK

서버 OK

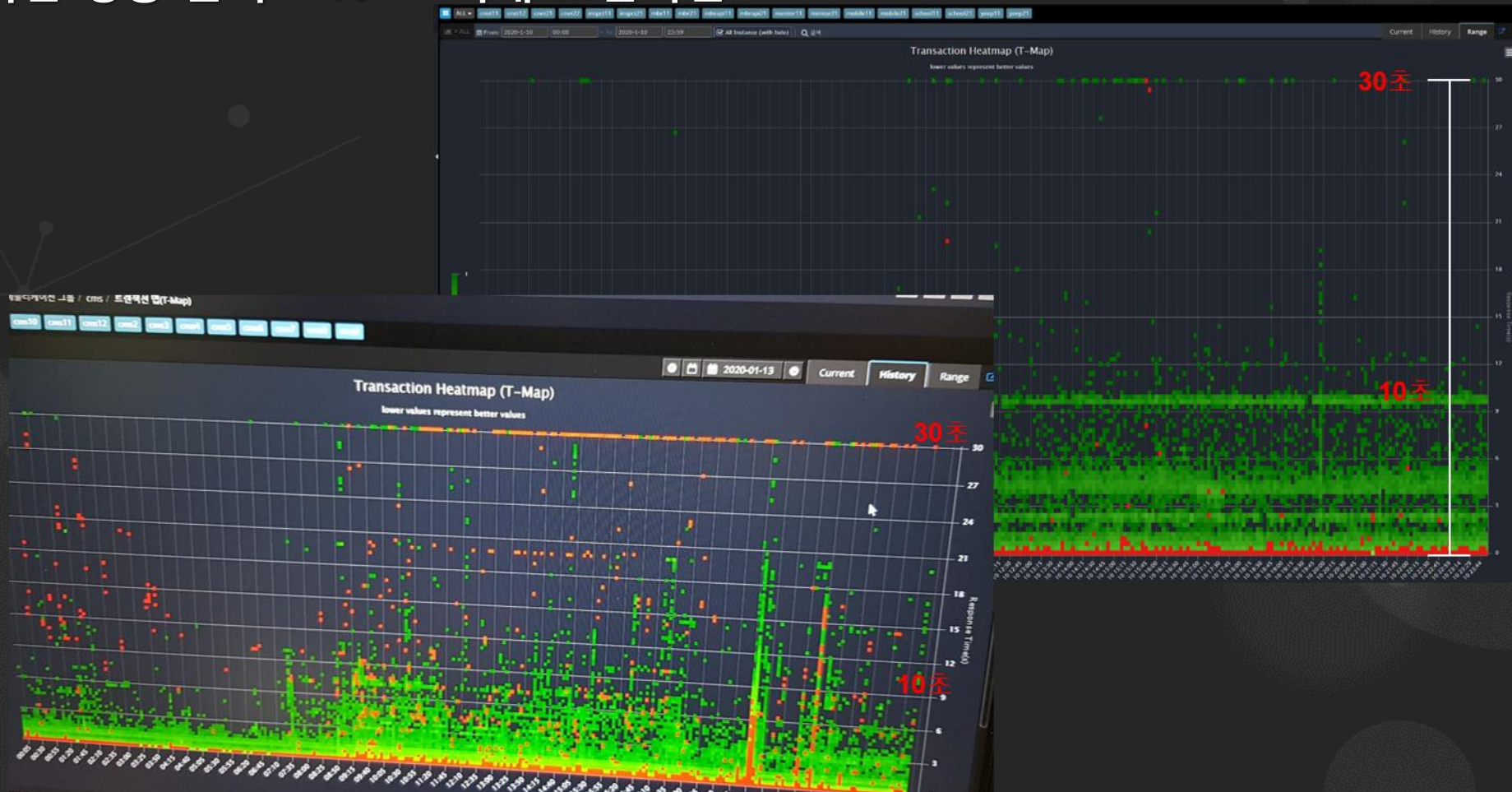
스토리지 OK

Log Monitoring vs. APM



Application 안정화

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



Micro Service Architecture

OPENMARU APM 제품 개요

장애와 성능 이슈에 대한 즉각적인 대응 체계 수립



OPENMARU APM – Method Tracer

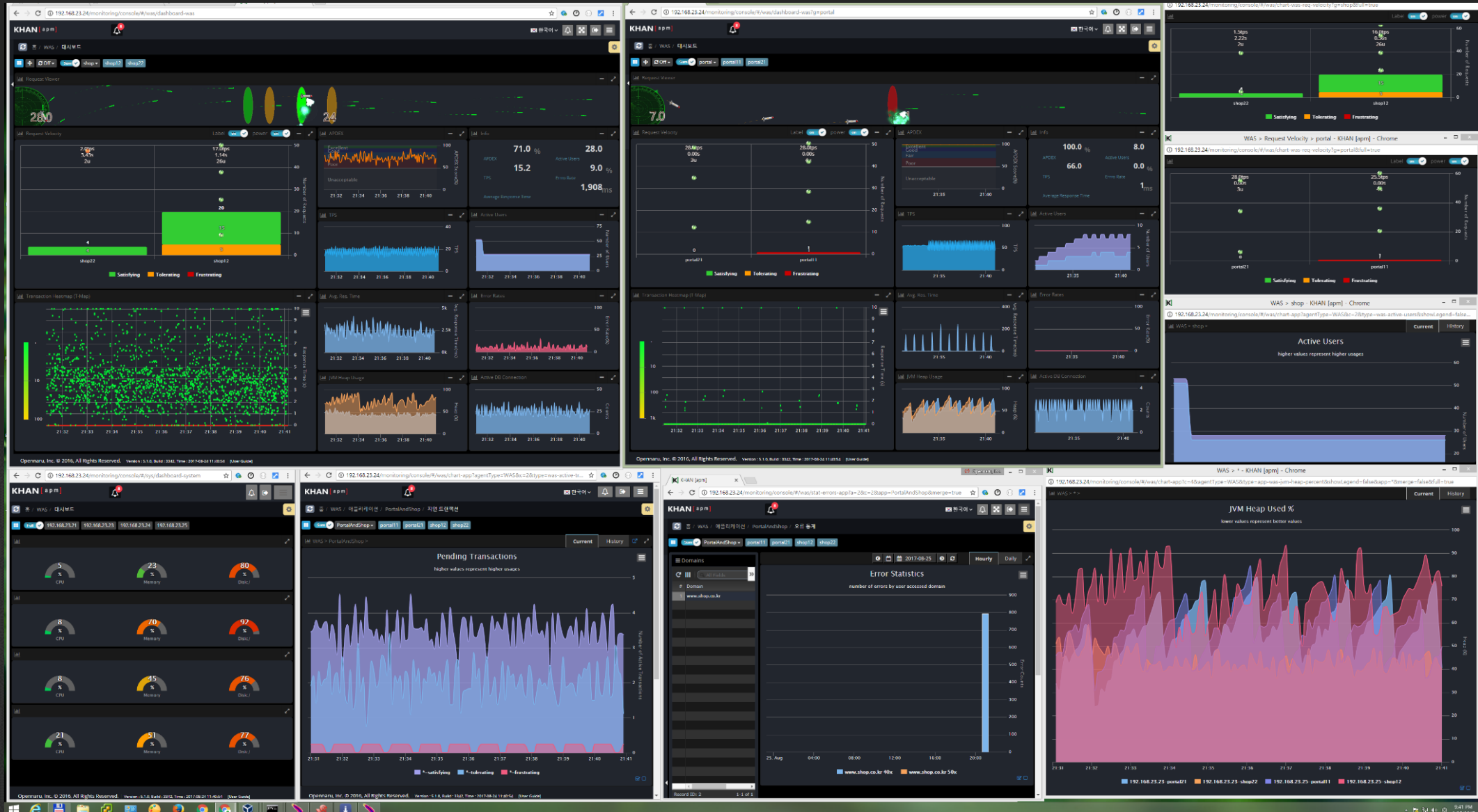


The screenshot displays the Openmaru APM Method Tracer interface. The top section shows a 'WAS Dashboard' with a 'Request Velocity' chart and a 'Transaction Heatmap (T-Map)'. The main area is dominated by a 'Transaction Detail' view, which is magnified by a red circular overlay. This view shows a table of transactions and a detailed method call stack for a specific transaction.

Instance ID	URL	Status	Duration (ms)	SQL Time (ms)	CPU Time (ms)
server11	/session/slow.jsp	200	3,792	0	2.6
server11	/sht/cmm/main/mainPage.do	200	9	3	7.14
server12	/session/	200	1	0	1.47
server11	/session/500.jsp	500	2	0	2.58
server11	/session/	200	2	0	2.43

Num.	Start Time	Elapsed	%	B-Gab	Exclusi	A-Gab	CPUTime	Method Call
[1]	[10:49:49.967]	9]	100]	0]	0]	0]	0.0]	+ org.apache.catalina.connector.CoyoteAdapter.service()
[2]	[10:49:49.967]	9]	100]	0]	0]	9]	0.0]	+ org.apache.logging.log4j.core.web.Log4jServletFilter.doFilter()
[3]	[10:49:49.967]	9]	100]	0]	0]	9]	0.0]	+ org.springframework.web.servlet.FrameworkServlet.service()
[4]	[10:49:49.967]	9]	100]	0]	0]	9]	0.0]	+ org.springframework.web.servlet.FrameworkServlet.doGet()
[5]	[10:49:49.967]	9]	100]	0]	0]	9]	0.0]	+ org.springframework.web.servlet.DispatcherServlet.doService()
[6]	[10:49:49.967]	9]	100]	0]	3]	6]	0.0]	+ org.springframework.web.servlet.DispatcherServlet.doDispatch()
[7]	[10:49:49.968]	0]	0]	0]	0]	0]	0.0]	+ org.apache.commons.dbcp.BasicDataSource.getConnection()
[8]	[10:49:49.968]	0]	0]	0]	0]	0]	0.0]	+ org.apache.commons.dbcp.PoolingDataSource.getConnection()
[9]	[10:49:49.968]	0]	0]	0]	0]	0]	0.0]	+ org.apache.commons.dbcp.DelegatingConnection.prepareStatement()
[10]	[10:49:49.968]	1]	11]	0]	0]	1]	0.0]	+ org.apache.commons.dbcp.DelegatingPreparedStatement.execute()
[11]	[10:49:49.968]	1]	11]	0]	0]	1]	0.0]	+ org.apache.commons.dbcp.DelegatingPreparedStatement.execute()

OPENMARU APM :: 9 – HTML5



OPENMARU APM Use Case



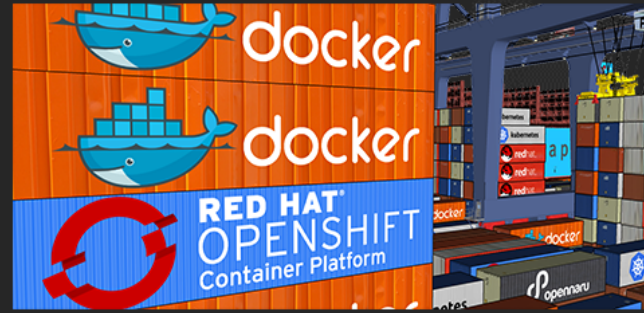
01

Unix To Linux 전환



02

Amazon 클라우드 전환



03

Openshift PaaS 도입



Red Hat 04

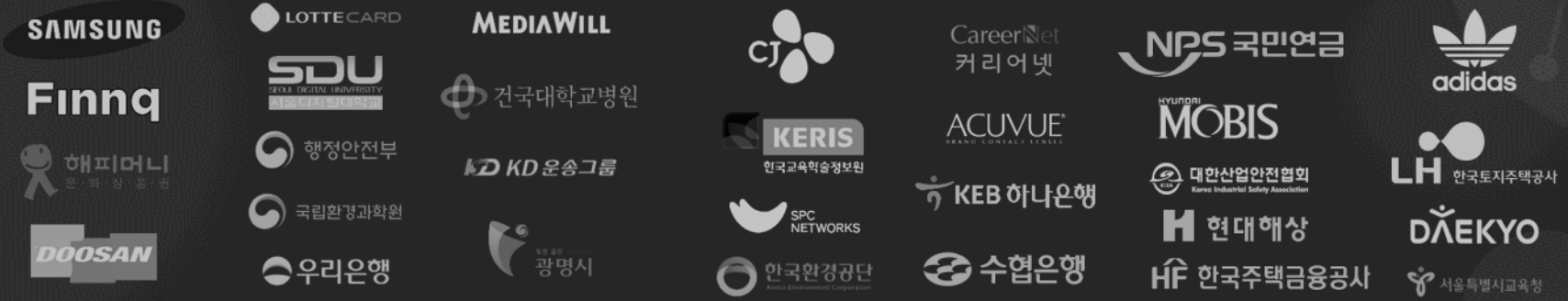
Red Hat Business

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	LOTTE CARD	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



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	제품	버전	제품	버전
 JBoss by Red Hat	• JBoss EAP 5 ~ 7	 ORACLE WebLogic	• 9 버전 이상	 IBM WebSphere	• 17 버전 이상
 WildFly	• JBoss AS 5 • 8 버전 이상	 Tmax JEUS	• 6 버전 이상	 resin.	• 3 버전 이상
 Apache Tomcat	• 5.5 버전 이상	 jetty://	• 8 버전 이상	 spring boot	• 1 버전 이상
 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

openmaru APM	openmaru APM	openmaru APM	openmaru APM	openmaru APM
Active-X	HTML5	HTML5	HTML5	HTML5
Java EE	Spring /e-Gov	Spring /e-Gov	Spring /e-Gov	Spring /e-Gov
독점/고가 WAS	Apache/Tomcat/JBoss	Apache/Tomcat/JBoss	Apache/Tomcat/JBoss	Apache/Tomcat/JBoss
Java	Java	Java	Java	Java
vPar/PowerVM/ OracleVM	Red Hat Virtualization	openstack™	amazon web services™	
HP-UX/AIX/Solaris	Red Hat Linux	vmware	Microsoft Azure	docker
PA-RISC/Power/SPARC	X86	RED HAT ENTERPRISE VIRTUALIZATION	Google Cloud Platform	OPENSIFT
물리 환경	리눅스/가상화	Private Cloud	Public Cloud	컨테이너

첫날부터 'EBS 먹통'...대리 출석에 게임까지 '혼돈' / SBS

SBS

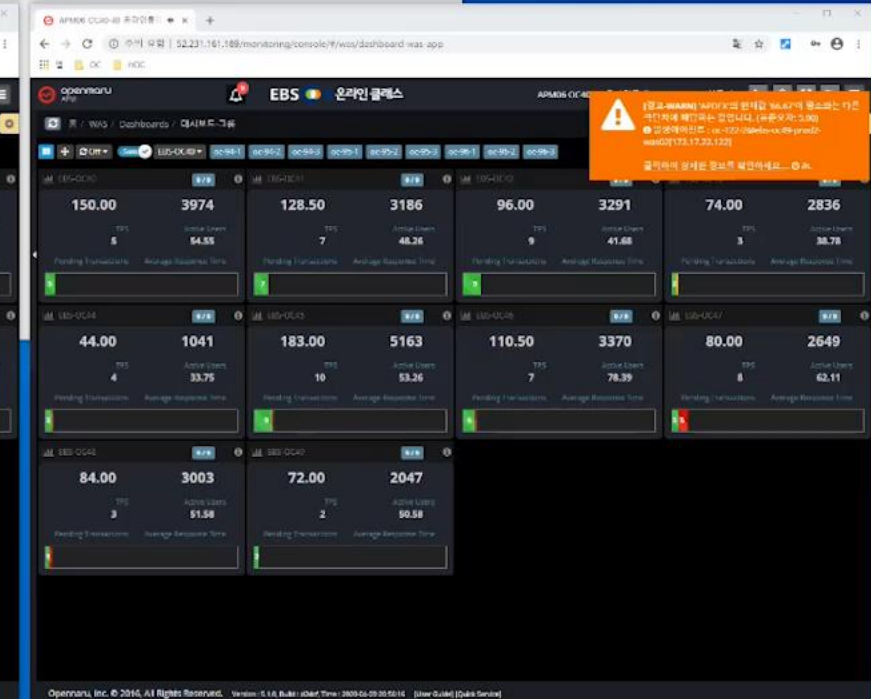
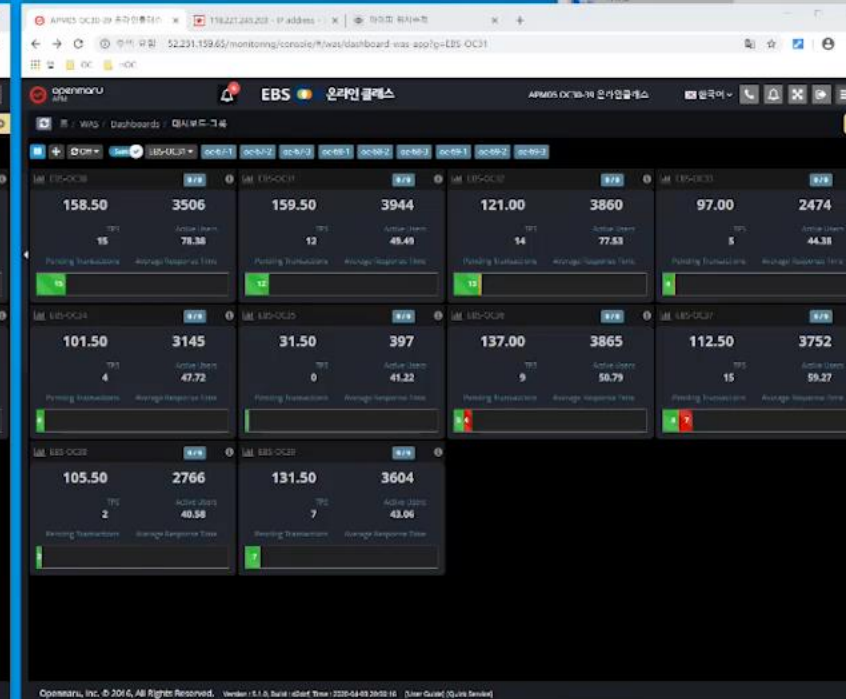
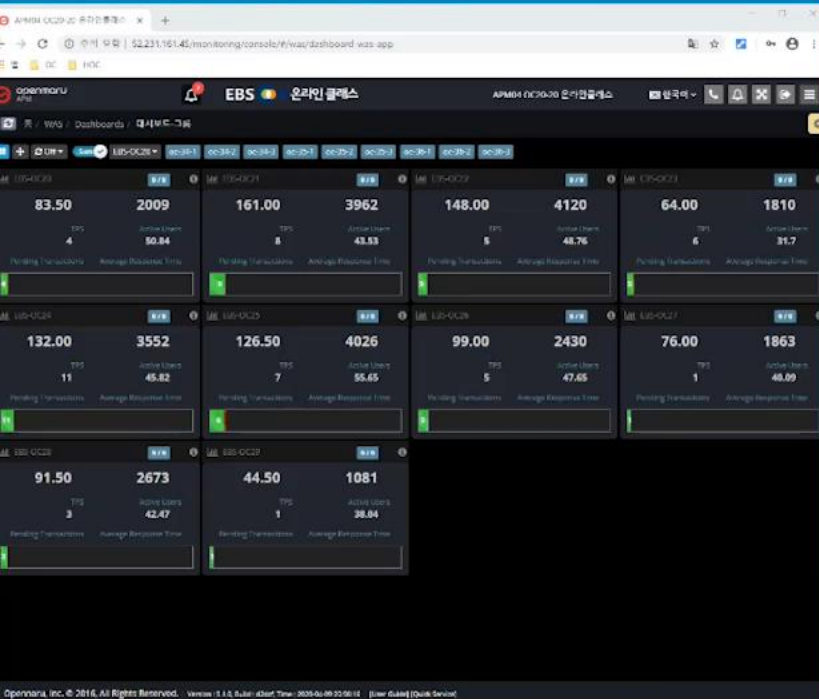
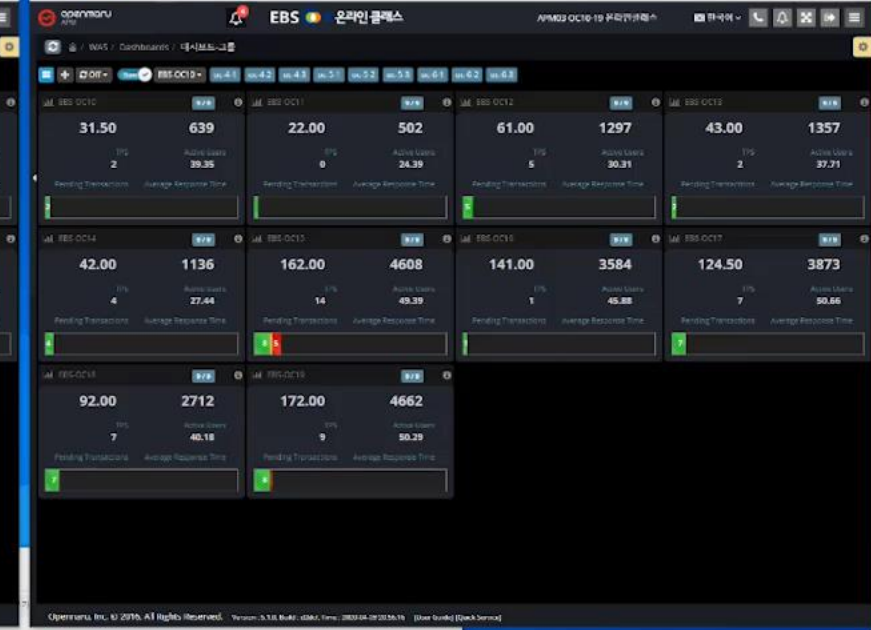


8 SBS NEWS

EBS 한때 '먹통' .. "온라인 게임 하고 학원 가고"

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

[장애 발생] = 비정상 상황 발생 그룹 모니터링



[중요-WARN] 서버의 운영을 위해서 필요한 것은
 - 데이터 백업은 필수입니다. (매주 0:00-0:30)
 - 운영데이팅은 매 0:00-0:05에 backup을 해야
 합니다 (17.12.22.132)
 글쓴이의 글에는 중요한 정보가 있습니다...



Micro Service Architecture

OPENMARU APM 특징점

01



차별화된 기능

- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 킷서비스
- 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

01



차별화된 기능

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

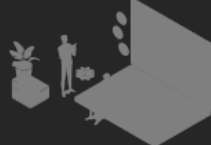
02



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

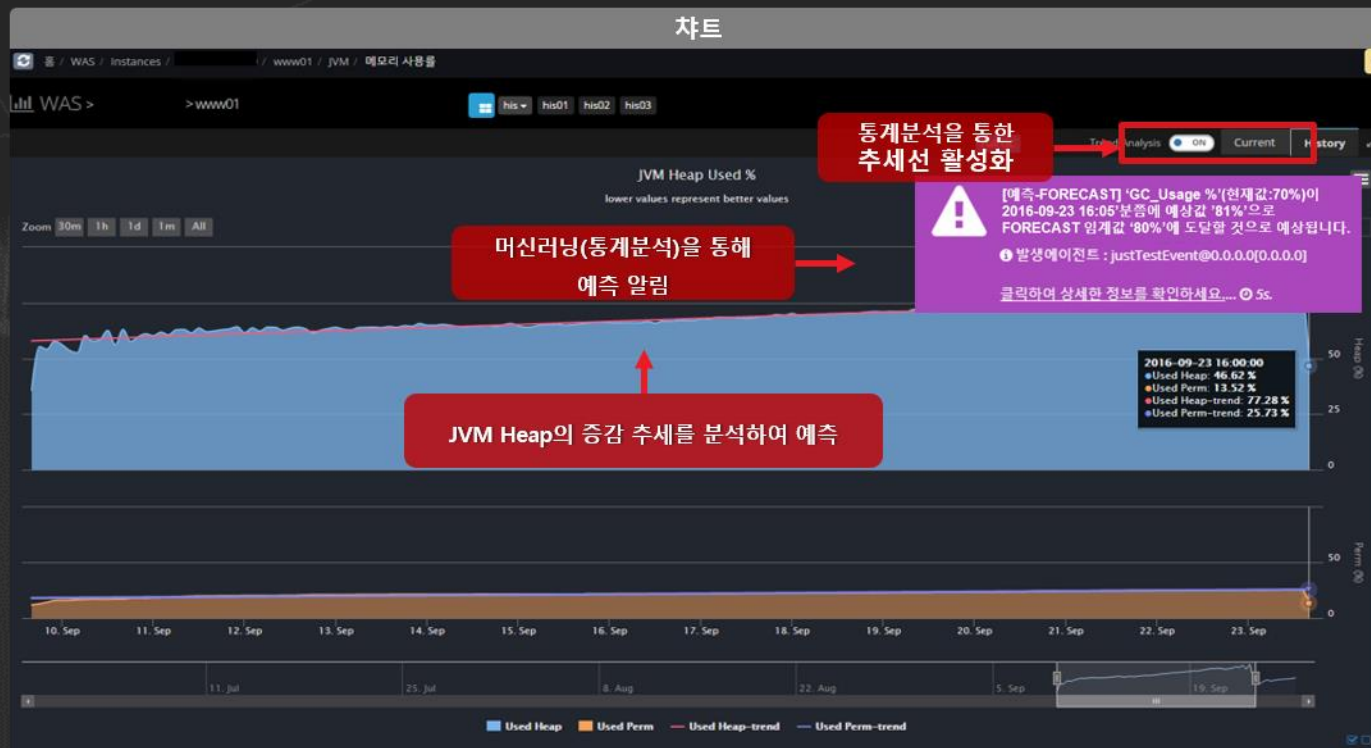
임계치에 도달하기 전에 이상 징후를 사전에 예측



- 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 도착 전 이상 징후 사전 예측
(홈 > WAS > (모든 차트) History)

Web WAS Container System Application

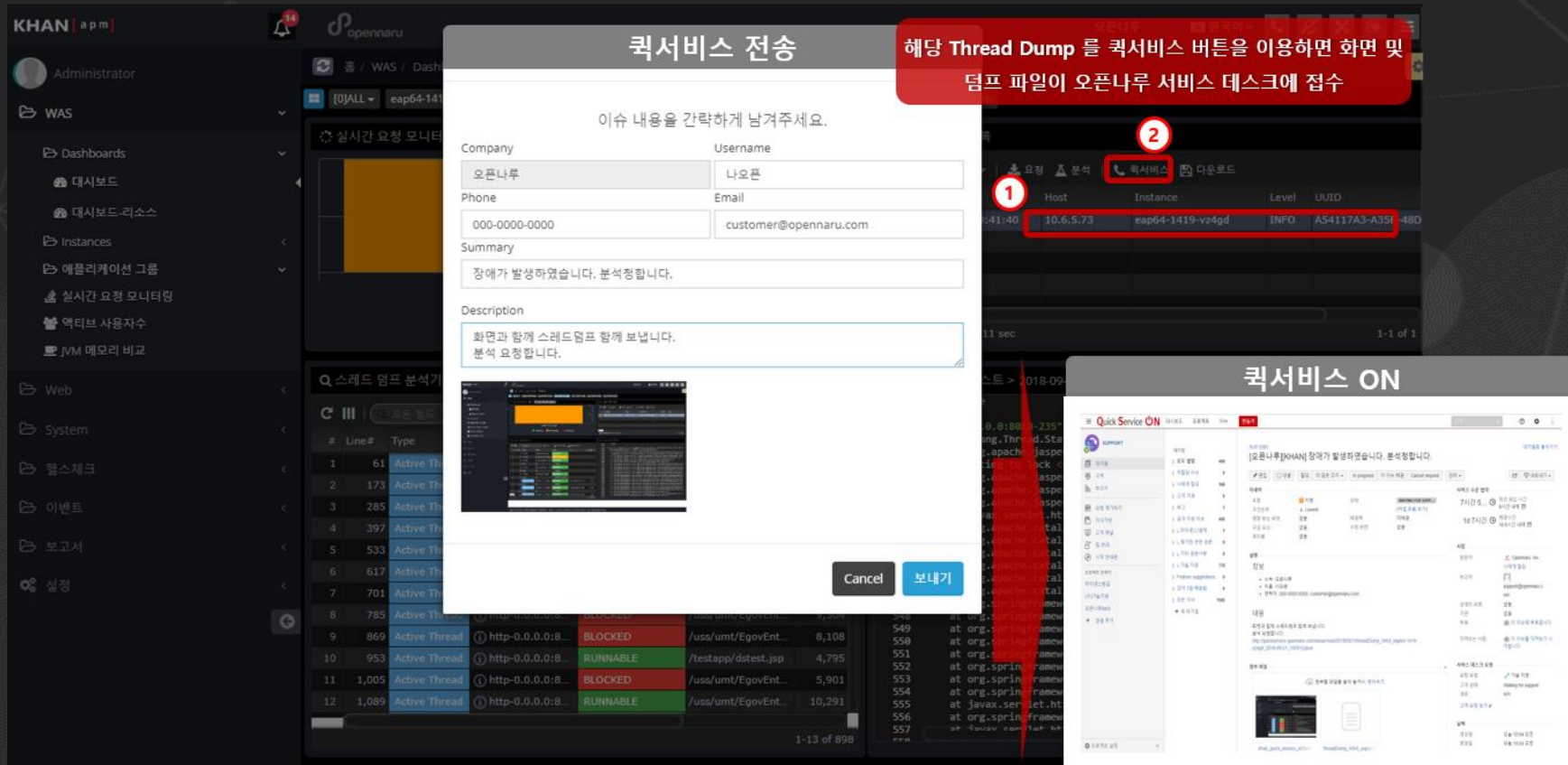


항목	이벤트 발생 항목
WAS	<ul style="list-style-type: none"> JVM Heap Usage APDEX Error Rate JVM Perm Usage Database Response Time Pending Transactions GC Usage Database Connection Pool Usage
WEB	<ul style="list-style-type: none"> WEB Traffic Worker Usage
System	<ul style="list-style-type: none"> Memory Usage Disk Usage Memory Swap Usage CPU Usage Network Packet Error Rate
Cubrid	<ul style="list-style-type: none"> CAS Usage

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.



퀵서비스 전송

이슈 내용을 간략하게 남겨주세요.

Company: 오픈나루 Username: 나오편

Phone: 000-0000-0000 Email: customer@openmaru.com

Summary: 장애가 발생하였습니다. 분석중입니다.

Description: 화면과 함께 스크린샷도 함께 보냅니다. 분석 요청합니다.

Cancel 보내기

해당 Thread Dump 를 퀵서비스 버튼을 이용하면 화면 및 덤프 파일이 오픈나루 서비스 데스크에 접수

1 **2**

퀵서비스 ON

Quick Service ON

[오픈나루]KHAN] 장애가 발생하였습니다. 분석중입니다.

Host	Instance	Level	URLD
10.6.5.73	eap64-1419-v24gd	INFO	A54117A3-A35E-48D

#	Line#	Type	Message	Count
1	61	Active Thread
2	173	Active Thread
3	285	Active Thread
4	397	Active Thread
5	533	Active Thread
6	617	Active Thread
7	701	Active Thread
8	785	Active Thread
9	869	Active Thread	http-0.0.0.0:8... BLOCKED /uss/umt/EgovEnt... 8,108	549
10	953	Active Thread	http-0.0.0.0:8... RUNNABLE /testapp/detest.jsp 4,795	550
11	1,005	Active Thread	http-0.0.0.0:8... BLOCKED /uss/umt/EgovEnt... 5,901	551
12	1,089	Active Thread	http-0.0.0.0:8... RUNNABLE /uss/umt/EgovEnt... 10,291	552

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

대시 보드

특정 Client가 비정상적인 호출시 비정상 거래 감지 및 실시간 알림 이벤트 표시

[경고-WARN] 무차별 공격 유형이 감지되었습니다. 1초 동안 1 사용자가 공격하고 있습니다. 192.168.23.3:217
발생 에이전트 : server11-23-11@khan-dev01[192.168.23.11]
클릭하여 상세한 정보를 확인하세요... &s.

이벤트 알람 기준

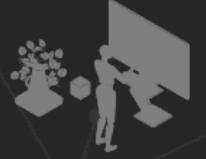
- 동일 사용자가 단위시간에 특정URL을 100회 호출한 경우 (설정을 통해 변경 가능)

이벤트

이벤트 경고 리스트에 해당 비정상 거래의 상세 내용을 출력

#	Created	Level	Name	Type	Description	Link	UUID	Host IP	Hostname	Agent Type	Instance
1	2017-11-25 16:21	경고-WARN	EVENT_RES_UBT_ALERT	User Behavior Tracker	무차별 공격 유형이 감지되었습니다. 1초 동안 1 사용자가 공격하고 있습니다. 192.168.23.3:808	View details	0318	192.168.23.11	khan-dev01	WAS	server
2	2017-11-25 16:20	경고-WARN	EVENT_RES_UBT_ALERT	User Behavior Tracker	무차별 공격 유형이 감지되었습니다. 1초 동안 1 사용자가 공격하고 있습니다. 192.168.23.3:217	View details	0347	192.168.23.11	khan-dev01	WAS	server
3	2017-11-25 13:56	정보-INFO	EVENT_AGENT_DISCONNECTED		WAS가 시작되어 연결되었습니다.	View details	8464	192.168.23.11	khan-dev01	WAS	server
4	2017-11-25 13:56	경고-WARN	EVENT_AGENT_DISCONNECTED		WAS가 시작되어 연결되었습니다.	View details	1806	192.168.23.11	khan-dev01	WAS	server
5	2017-11-25 13:21	정보-INFO	EVENT_AGENT_CONNECTED		WAS가 시작되어 연결되었습니다.	View details	1806	192.168.23.11	khan-dev01	WAS	server
6	2017-11-25 13:08	정보-INFO	EVENT_AGENT_CONNECTED		WAS가 시작되어 연결되었습니다.	View details	1806	192.168.23.11	khan-dev01	WAS	server
7	2017-11-25 13:08	정보-INFO	EVENT_AGENT_CONNECTED		WAS가 시작되어 연결되었습니다.	View details	1806	192.168.23.11	khan-dev01	WAS	server


01



차별화된 기능

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

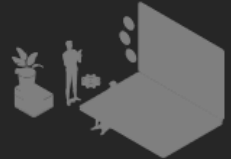
02



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 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

#	Lin.	Type	Name	State	URL	Duration	CPU %	Class	Alt
1	696	Active Thread	ajp-0.0.0.0-8009...	RUNNABLE		1,309	0.0	java...	
2	871	Active Thread	ajp-0.0.0.0-8009...	RUNNABLE		123,355	0.0	java...	
3	936	Active Thread	ajp-0.0.0.0-8009...	RUNNABLE		156,825	10.0	java...	
4	1,556	Active Thread	ajp-0.0.0.0-8009...	RUNNABLE		183,887	0.0	java...	

Lock을 추적가능, URL 정보표시

JVM 메모리 객체 분석기

#	Num	Classname	Bytes [%]	Bytes	Instances
1	1	char[]	35.25%	35.2 MB	258,535
2	2	byte[]	8.62%	8.6 MB	10,388
3	3	java.lang.String	6.11%	6.1 MB	253,859
4	4	java.util.HashMap\$Node	5.53%	5.5 MB	173,073
5	5	java.util.jar.JarFile\$JarFileEntry	5.06%	5.0 MB	52,595

Java 메모리를 점유한 객체 분석/비교

네트워크 상태 분석기

#	Proto	Recv	Send-Q	Local Address	Foreign Address	State	PID/Program name
1	tcp	0	0	127.0.0.1:9999	0.0.0.0:*	LISTEN	509/java
2	tcp	0	0	10.6.3.224:5455	0.0.0.0:*	LISTEN	509/java
3	tcp	0	0	10.6.3.224:7600	0.0.0.0:*	LISTEN	509/java
4	tcp	0	0	0.0.0.0:8080	0.0.0.0:*	LISTEN	509/java

Java 프로세스,시스템이 사용중인 네트워크 분석

오픈파일 분석기

#	Cam...	PID	User	FD	Type	Device	Size/Off	Node	Name
1	java	509	boss	cwd	DIR	253,28	85	4199509	/home/boss
2	java	509	boss	rd	DIR	253,28	261	1027	/
3	java	509	boss	txt	REG	253,28	7376	6295581	/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.1...

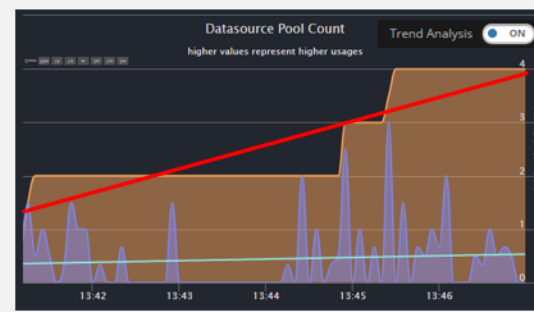
Java 프로세스가 오픈한 파일 분석

시스템 프로세스 분석기

#	Created	Uptime	Load Avg	3.4	3.9	3.6	1m, 5m, 15m
1	2019-01-18 18:41:34	4 days 02:49:11	3.4	3.9	3.6	1m, 5m, 15m	

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

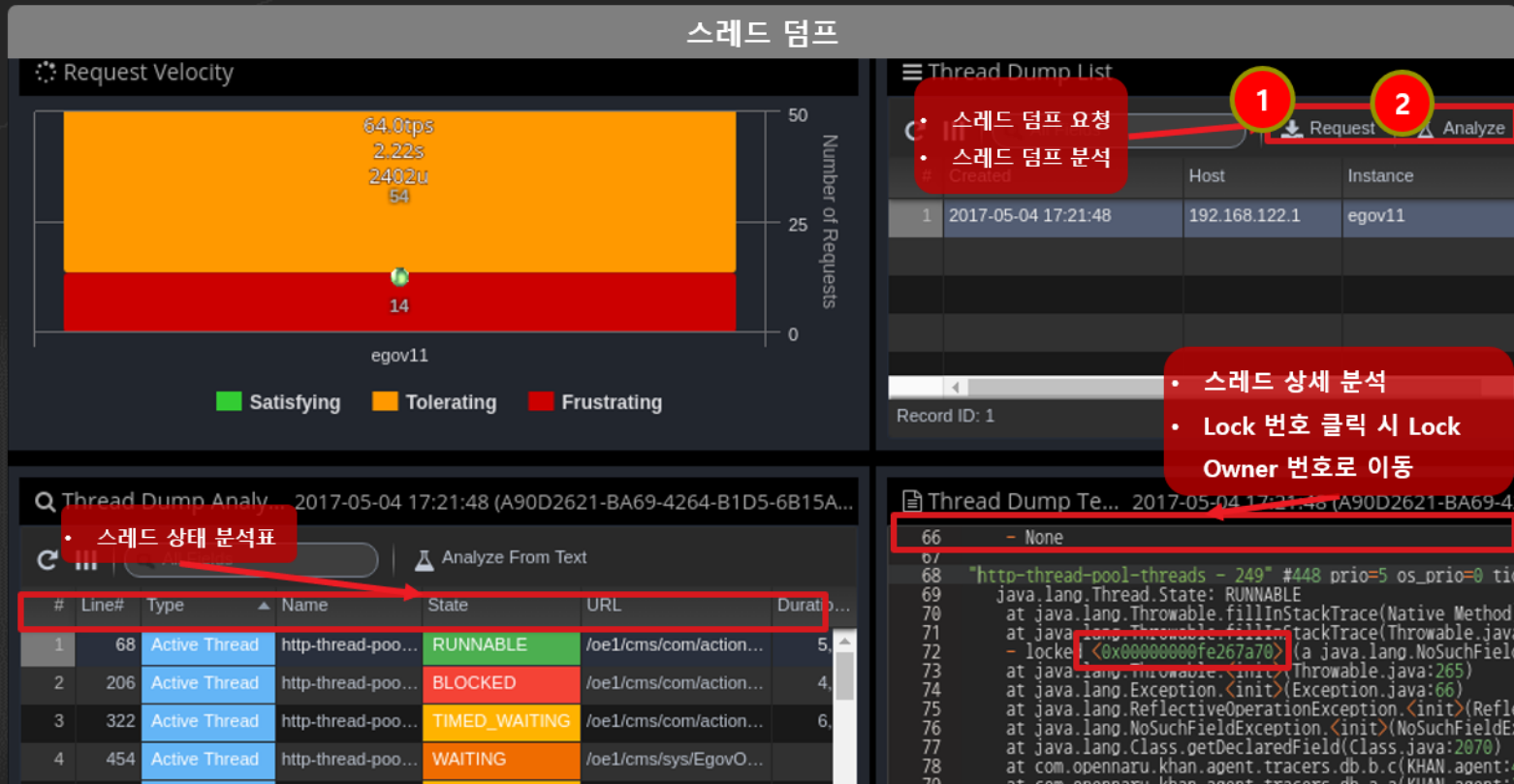


- 시스템에 직접 접속하여 해당 인스턴스를 찾아서 커맨드를 실행하여 스레드 덤프를 생성한다.
- APM에서는 스레드 덤프 분석기를 사용하여 스레드 덤프를 생성하고 시각적으로 쉽게 스레드 분석 한다.

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

WAS

Application



스레드 덤프

Request Velocity

64.0tps
2.22s
2402u
54

Number of Requests

50
25
0

14

egov11

■ Satisfying ■ Tolerating ■ Frustrating

Thread Dump List

- 스레드 덤프 요청
- 스레드 덤프 분석

#	Created	Host	Instance
1	2017-05-04 17:21:48	192.168.122.1	egov11

Record ID: 1

- 스레드 상세 분석
- Lock 번호 클릭 시 Lock Owner 번호로 이동

Thread Dump Analyze

- 스레드 상태 분석표

#	Line#	Type	Name	State	URL	Durat...
1	68	Active Thread	http-thread-poo...	RUNNABLE	/oe1/cms/com/action...	5...
2	206	Active Thread	http-thread-poo...	BLOCKED	/oe1/cms/com/action...	4...
3	322	Active Thread	http-thread-poo...	TIMED_WAITING	/oe1/cms/com/action...	6...
4	454	Active Thread	http-thread-poo...	WAITING	/oe1/cms/sys/EgovO...	

Thread Dump Te...

```
66 - None
67
68 "http-thread-pool-threads - 249" #448 prio=5 os_prio=0 tid
69 java.lang.Thread.State: RUNNABLE
70 at java.lang.Thread.State: RUNNABLE
71 at java.lang.Thread.State: RUNNABLE
72 - locked <0x00000000fe267a70> (a java.lang.NoSuchField
73 at java.lang.Thread.State: RUNNABLE
74 at java.lang.Exception.<init>(Exception.java:66)
75 at java.lang.ReflectiveOperationException.<init>(Refle
76 at java.lang.NoSuchFieldException.<init>(NoSuchFieldEx
77 at java.lang.Class.getDeclaredField(Class.java:2070)
78 at com.openmaru.khan.agent.tracers.db.b.c(KHAN.agent:4
79 at com.openmaru.khan.agent.tracers.db.b.c(KHAN.agent:4
```

애플리케이션 오류 로그 수집 추적 기능



- 애플리케이션에서 Exception이 발생한 경우 이를 추적할 수 있어야 한다.
- APM은 여러 Exception을 추적하여 어떤 종류의 Exception이 발생되었는지 확인할 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

Warning Message

- warn message 1
- warn message 2
- error message 1
- error message 2

Logging Level에 따라
Warning Message 확인
가능

Method Call

```
org.apache.catalina.connector.CoyoteAdapter.service()
org.apache.jasper.servlet.JspServlet.service()
org.apache.jsp.index_jsp._jspService()
```

SQL Message

```
121) as CreatedDate, /* */ Convert(varchar, DATEADD(MONTH, 1, ISNULL(AcuvueChangeDate, A.CreatedDate), GETDATE()), 121) AS CurrentDate, /* */ Convert(varchar, C.EcpEyeInfoDate, 121) AS EcpEyeInfoDate, /* */ Convert(VARCHAR(8), GETDATE(), 112)+CONVERT(varchar(8), GETDATE(), 114), ':') AS SearchStartTime, /* */ CONVERT(VARCHAR(8), GETDATE(), 112)+CONVERT(varchar(8), GETDATE(), 114), ':') AS SearchEndTime FROM contact LIKE N'%'+ + '% ' 0
```

> SQLERR: The query has timed out.
> CallAt:

SQL message 발생시 Trace 하여
Exception 및 Error 메시지 확인

```
at com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.execute(SQLServerPreparedStatement.java:111)
at org.apache.ibatis.executor.statement.PreparedStatementHandler.query(PreparedStatementHandler.java:60)
at org.apache.ibatis.executor.statement.RoutingStatementHandler.query(RoutingStatementHandler.java:111)
at org.apache.ibatis.executor.SimpleExecutor.query(SimpleExecutor.java:60)
at org.apache.ibatis.executor.CachingExecutor.query(CachingExecutor.java:137)
at org.apache.ibatis.executor.CachingExecutor.query(CachingExecutor.java:96)
at org.apache.ibatis.executor.CachingExecutor.query(CachingExecutor.java:77)
```

Out Of Memory Error

JDBC time (ms)	93	Latency
Thread Name	http-bio-80-exec-67	Thread I
IP	192.168.1.25	Instance
Agent Type	WAS	Transact

Out Of Memory Error 발생시
Error Message를 확인 가능

java.lang.OutOfMemoryError: GC overhead limit exceeded

Method Traces

Runtime Exception

Thread Name	http-thread-pool-threads - 104
IP	105.27.137.25
Agent Type	WAS

Runtime Exception을 Error Message로 확인 가능

JBWEE004038: An exception occurred processing JSP page
< %
int pageCount = 0;
for (int i = 0; i < items.length; i

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)



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

프로세스 상태 분석

홈 / System / 192.168.23.11 / 프로세스 상태 분석

1Test server1 server12

CPU Usage

시스템 상태 정보 개요

Uptime: 3 days 18:54:33 | Load Avg: 2, 2.8, 3 (1m, 5m, 15m)

Processes: 364 Total / 362 Sleeping, 2 Running, 0 Zombie, 0 Stopped

CPU Usage: User: 40.0%, System: 10.0%, Nice: 0.0%, Wait: 0.0%, Idle: 50.0%

Mem Usage: Total: 8.4 GB, Used: 8.2 GB, Free: 143.7 | Swap: Total: 5.3 GB, Used: 700.4 MB, Free: 4.6 GB

#	PID	User	Start Time	Mem	RSS Mem	Shared ...	State	CPU Time	CPU...	CPU...	MEM...	MEM...	Command
1	9,369	jjeon	2017-08-28 12:25...	4.0 GB	581.9 MB	357.7 MB	Sleeping	00:14:01.1	42.4%	+ 42.0%	11.2%	+ 11.0%	/opt/google/chrome/chrome --type...
2	12,596	jjeon	2017-08-25 20:33...	2.0 GB	184.2 MB	63.1 MB	Sleeping	00:11:21.3	39.2%	+ 39.0%	3.0%	+ 3.0%	/opt/google/chrome/chrome
3	12,887	jjeon	2017-08-25 20:33...	700.4 MB	72.0 MB	14.0 MB	Sleeping	00:10:16.8	20.5%	+ 20.0%	1.0%	+ 1.0%	/opt/google/chrome/chrome --type...

Process Snapshot List

All Fields Request Analyse **Diff Analyse**

#	Created	UUID
1	2017-08-28 13:10:31	FE3231EE-8FCF-458C-B4E2-E75...
2	2017-08-24 17:58:12	027BF4FA-FF35-44B5-8B5D-194...

Snapshot 2개를 선택하고 'Diff' 버튼을 클릭

PID : Process ID
User : 프로세스 Owner
Start Time : 프로세스 시작 시간
RSS Mem : 프로세스 점유 메모리
State : 프로세스 상태
CPU : Snapshot 중 최근 CPU 사용량
CPU Diff : 1번째와 2번째 Snapshot의 CPU 사용량 증감
MEM : Snapshot 중 최근 메모리 사용량
MEM Diff : 1번째와 2번째 Snapshot의 메모리 사용량 증감
Command : 프로세스 시작 Command

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


네트워크 상태 분석

Administrator

홈 / WAS / Instances / 192.168.23.11 / server11 / 네트워크 상태 분석

1Test server11 server12

Netstats



Number of Connections

established syn_sent syn_rcv fin_wait1 close fin_wait2 time_wait close_wait last_ack listen closing idle bound unknown

송수신 버퍼에 쌓여있는 데이터(Byte)

List Of Netstats > 2017-08-11 01:05:40

Grid Editor

All Fields Search

#	Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID
1	tcp	0	0	127.0.0.1:26605	0.0.0.0:*	LISTEN	
2	tcp	0	0	192.168.23.11:8109	0.0.0.0:*	LISTEN	
3	tcp	0	0	0.0.0.0:10190	0.0.0.0:*	LISTEN	
4	tcp	0	0	127.0.0.1:63342	0.0.0.0:*	LISTEN	
5	tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	
6	tcp	0	0	0.0.0.0:12689	0.0.0.0:*	LISTEN	
7	tcp	0	0	127.0.0.1:5857	0.0.0.0:*	LISTEN	
8	tcp	0	0	127.0.0.1:4205	0.0.0.0:*	LISTEN	
9	tcp	0	0	0.0.0.0:81	0.0.0.0:*	LISTEN	

Grid Table로 출력하여 Sorting 및 검색이 가능

Listen 되어있는 Port 번호

Netstat dump created

#	Created	UUID
1	2017-08-11 01:05:40	B78A55E2-9FCS-41E...
2	2017-07-28 18:47:01	8497D286-04BD-444...

과거 데이터 추세 분석 기능

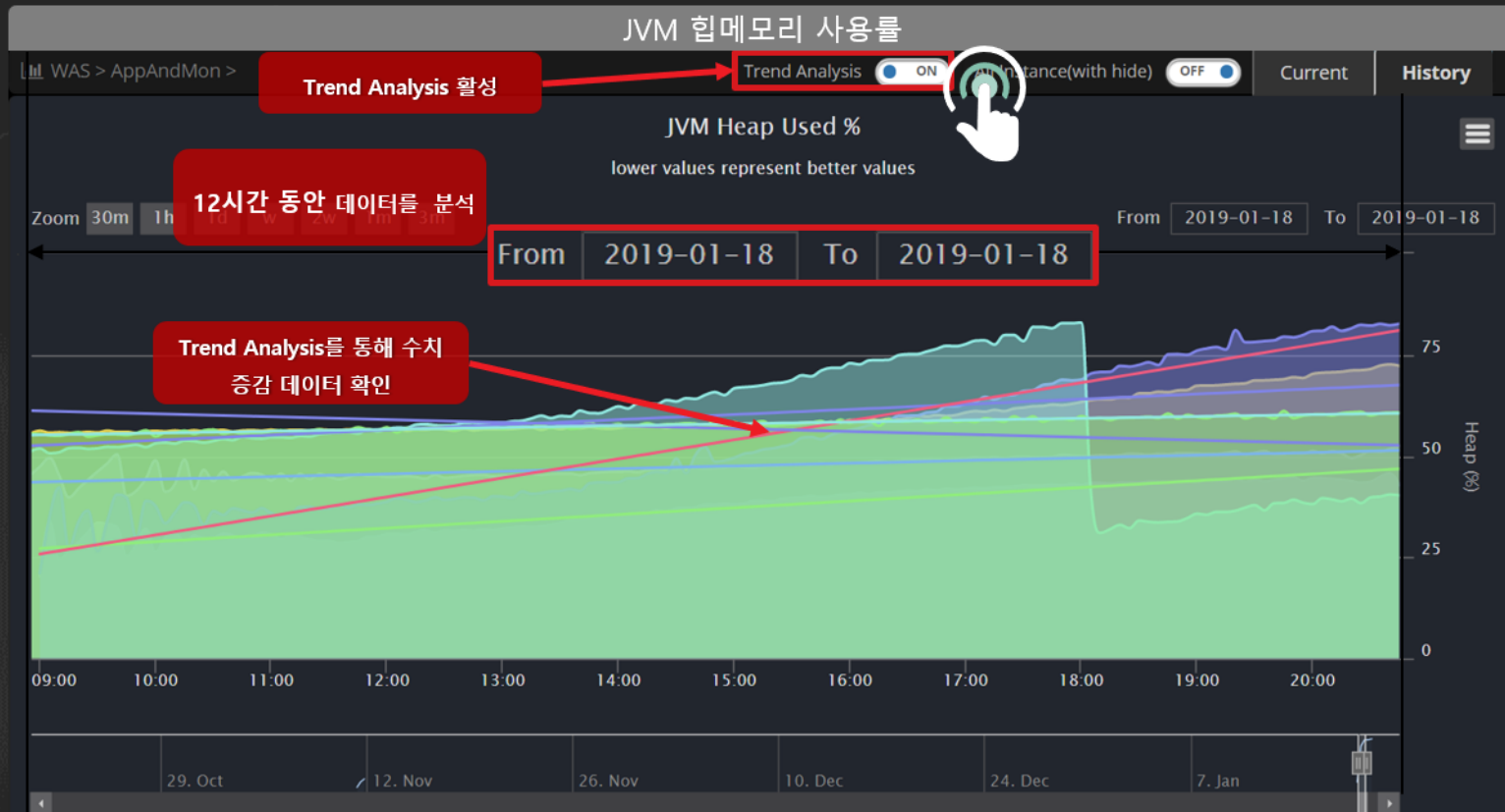
과거 데이터 추세 분석 기능



- 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



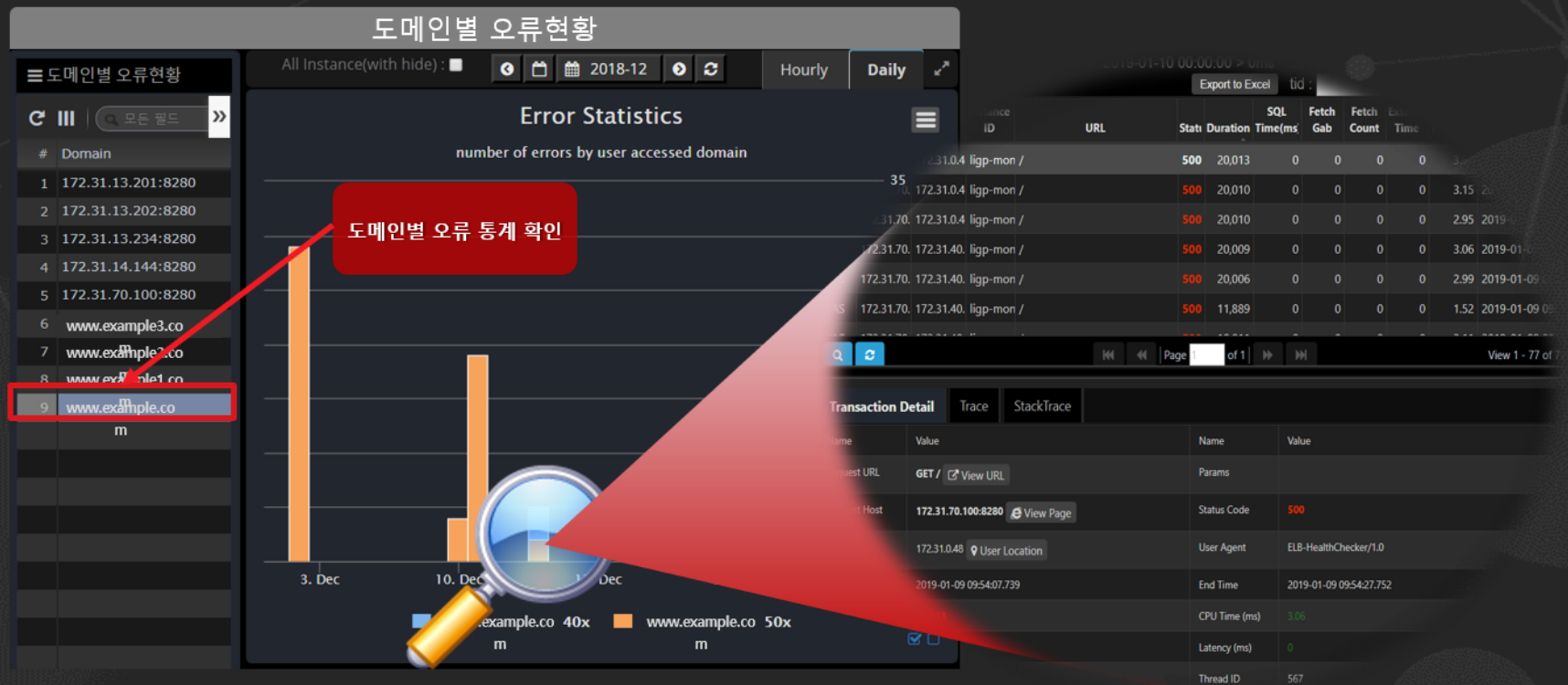
사용자 접속 도메인 별 오류 분석기능

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



WAS 시작 옵션의 기록 및 비교 분석



- 서버 옵션을 변경하여 문제가 발생하는 경우, 옵션 변경기록이 남지 않는다.
- APM에서는 Agent 가동시에 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

JVM 가동옵션 비교

Server Start Time


#	Started	UU
1	2017-08-28 12:26:31	E1C7E966-1618-4761-955F-CA083ED65376
2	2017-08-21 21:43:32	D666EC75-00C4-4832-A950-C3E725386BB2
3	2017-08-21 20:51:51	FCE3CBE4-FAE8-4F...
4	2017-08-21 20:50:44	D9249D63-D84A-40...
5	2017-08-21 20:27:01	6D91C02B-E620-4B...
6	2017-08-21 20:24:38	6AB159A4-A5D6-4B...
7	2017-08-21 20:18:02	63747A7B-937B-43...
8	2017-08-21 13:06:47	9CE65238-F369-43...
9	2017-08-21 10:46:16	8D8138B5-1373-4F...
10	2017-08-21 05:08:54	59C8370C-DFC7-46...
11	2017-08-21 02:25:30	33A38818-893C-4F...
12	2017-08-20 23:57:23	3B546292-62F5-49...
13	2017-08-20 01:40:44	BAA6618F-33F5-40...
14	2017-08-20 01:37:06	2750E928-8AE3-4F...
15	2017-08-18 21:16:19	4C94110C-8D4A-4B...
16	2017-08-18 20:49:16	E226415B-EE87-4F...
17	2017-08-18 20:49:03	9DB04F23-ES2D-43...
18	2017-08-18 20:49:00	11A79864-B547-44...
19	2017-08-18 20:48:48	ADA53E60-4789-43...
20	2017-08-18 20:48:44	19209A1D-D988-4A...
21	2017-08-18 20:48:43	0BAF2C0D-9C4E-47...
22	2017-08-18 20:48:39	95A8DA55-B692-4A...
23	2017-08-18 20:48:37	17450178-E1CF-42...

Agent 실행 시 수집된 서버 정보를 Diff 분석

WAS 시작 시 변경된 환경 변수, Java Option, PID등을 Diff로 파악 가능

```
Instance Info: "E1C7E966-1618-4761-955F-CA083ED65376" - 2017-08-21 21:43:32 (D666EC75-00C4-4832-A950-C3E725386BB2)
{
  "agentConfigInfo": {
    "sql_stacktrace_threshold": 30000,
    "sql_parameterize": false,
    "trace_enabled": true,
    "apdex_threshold": 1,
    "transaction_trace_threshold": 0,
    "last_modified": 1503314369000,
    "sql_capture_enabled": true,
    "transaction_trace_sampling_interval": 1,
    "userCharsetEncoding": "EUC-KR",
    "userInterceptorFile": "user-interceptor.conf",
    "userInterceptors": [
      "com.microsoft.sqlserver.*"
    ]
  },
  "databaseFetchWarnings": [
    100000,
    200000,
    300000,
    400000,
    500000,
    600000,
    700000
  ],
  "databaseConnLeakWarning": false
},
  "applicationInfo": {
    "port": 0,
    "agentVersion": "Version : 5.1.0, Build : 3333, Time : 2017-08-21 21:43:32",
    "agentType": "WAS",
    "hostName": "khan-dev01",
    "appType": "Unknown",
    "ipAddress": "192.168.23.11",
    "appName": "my Application",
    "instanceId": "server11",
    "compressType": 1
  },
  "jvmInfo": {
    "javaVersion": "1.7.0_79",
    "agentVersion": "5.1.0",
    "javaVendor": "Oracle Corporation",
    "agentPath": "/data/iboss/intel1111/worksapce/khan-monitor/agent.jar",
    "javaPid": "9917",
    "javaBootClassPath": "/home/jjeon/iboss/eap/iboss-eap-6.3.1.jar",
    "javaClassPath": "/home/jjeon/iboss/eap/iboss-eap-6.3.1.jar"
  }
}
```


01



차별화된 기능

- 머신러닝 기반의 장애 경고 알람
- 헬스체크
- 킷서비스
- 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

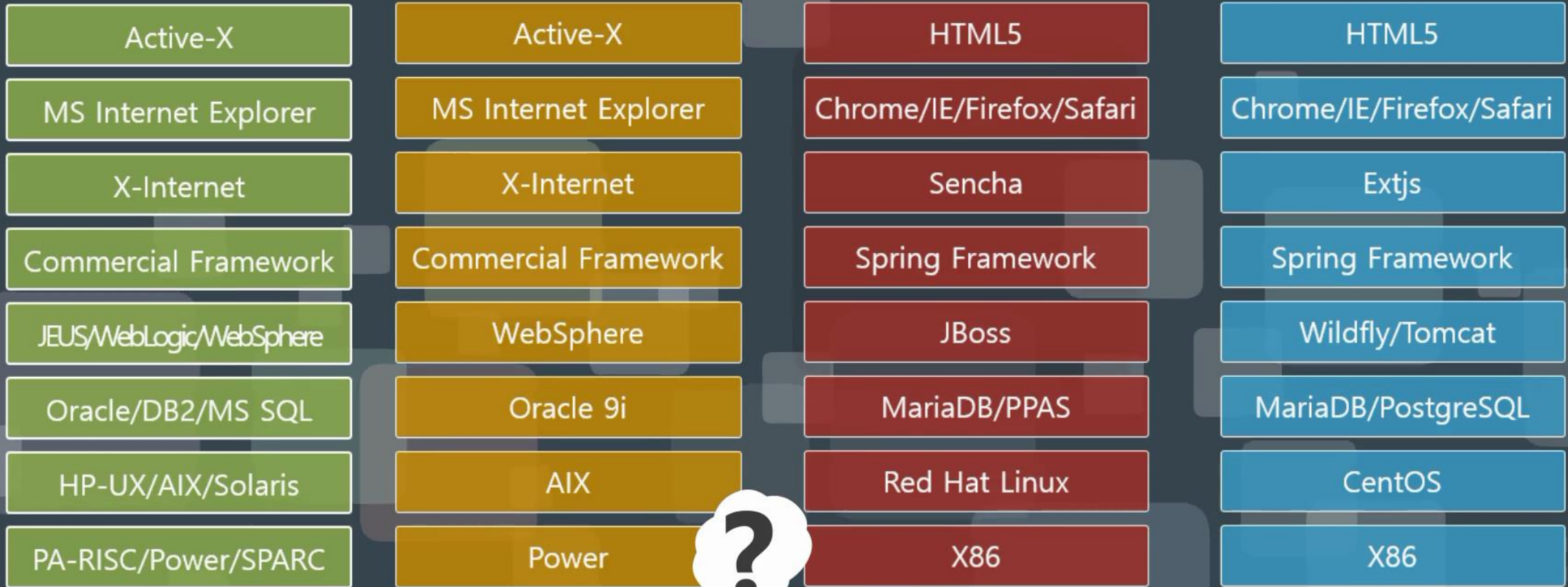
- GS 인증 1등급
- 오픈소스 전문기업이 만든 APM
- WAS 장애 지원 및 튜닝 지원
- 행정안전부 온-나라 BMT 성능 평가 1위

05



최신 IT 기술 지원

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



독점
소프트웨어

현재 구성

유상
오픈소스

무상
오픈소스



물리 환경



Private/Public
Cloud



설치와 구성

진단과 조치



Provisioning

openmaru
APM

Monitoring



튜닝

모니터링



전문가 수준의 구성 품질

장애 사전 예방

성능 보장

안정된 운영

설치

- 리눅스만 설치되어 있으면 WAS 와 웹서버 등 웹시스템 구축에 필요한 소프트웨어를 자동설치

구성

- WAS 클러스터링, 데이터그리드, 운영 스크립트 등 난이도 높은 구성을 지원
- 개발자/관리자를 위한 운영 가이드 자동 생성

튜닝

- 전문가 수준의 튜닝 (리눅스 커널, 웹서버/ Java/WAS 튜닝)

모니터링

- 시스템과 OS 에 대한 모니터링 뿐만 아니라 Java 나 WAS 까지 포괄적으로 지원
- Openshift/Docker 컨테이너 모니터링

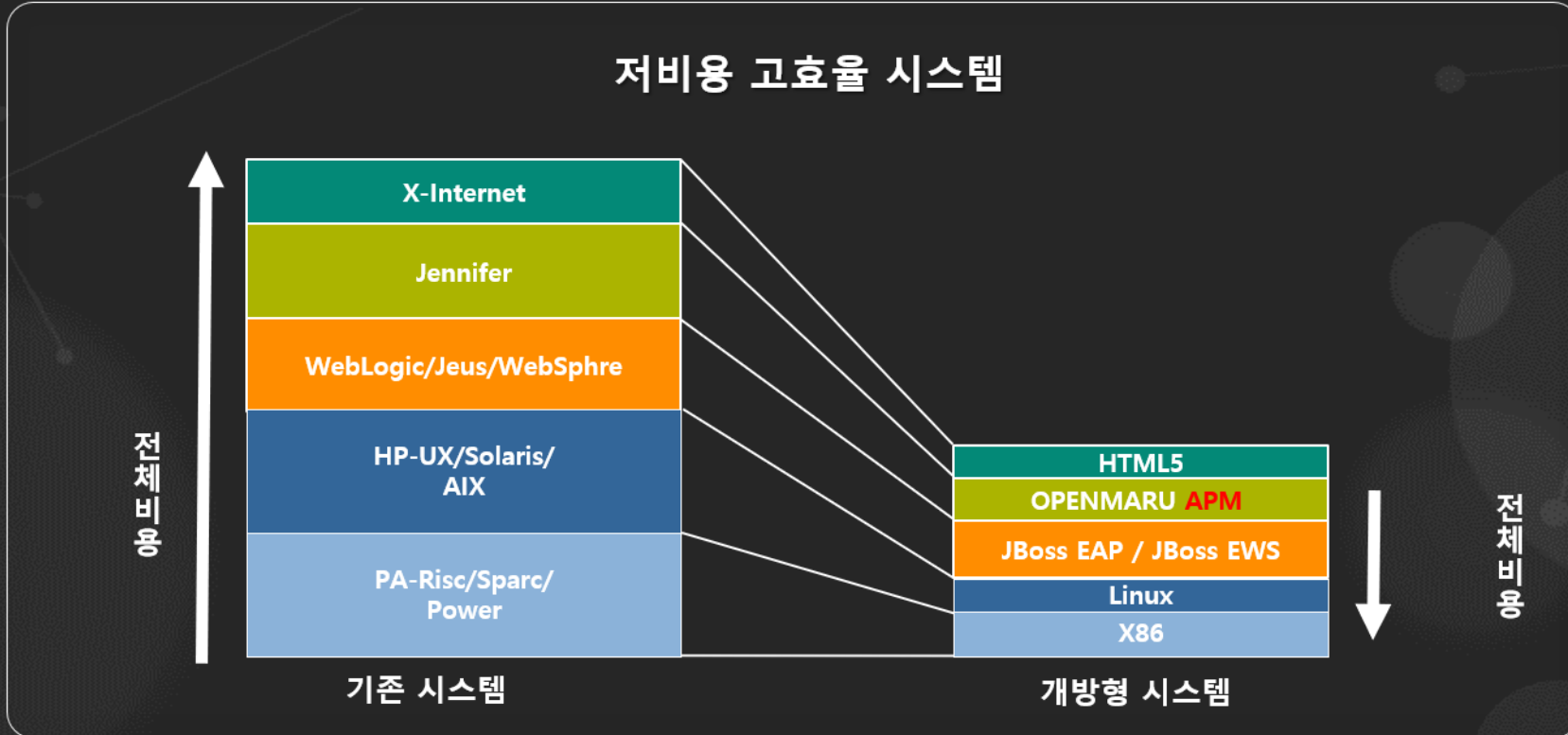
진단/조치

- Threaddump , 네트워크 상태, 웹서버 상태를 분석할 수 있는 진단 도구 제공
- OS/Container 시스템 에 대한 연관 분석

OPENMARU APM 제품 비교 - 비용 측면

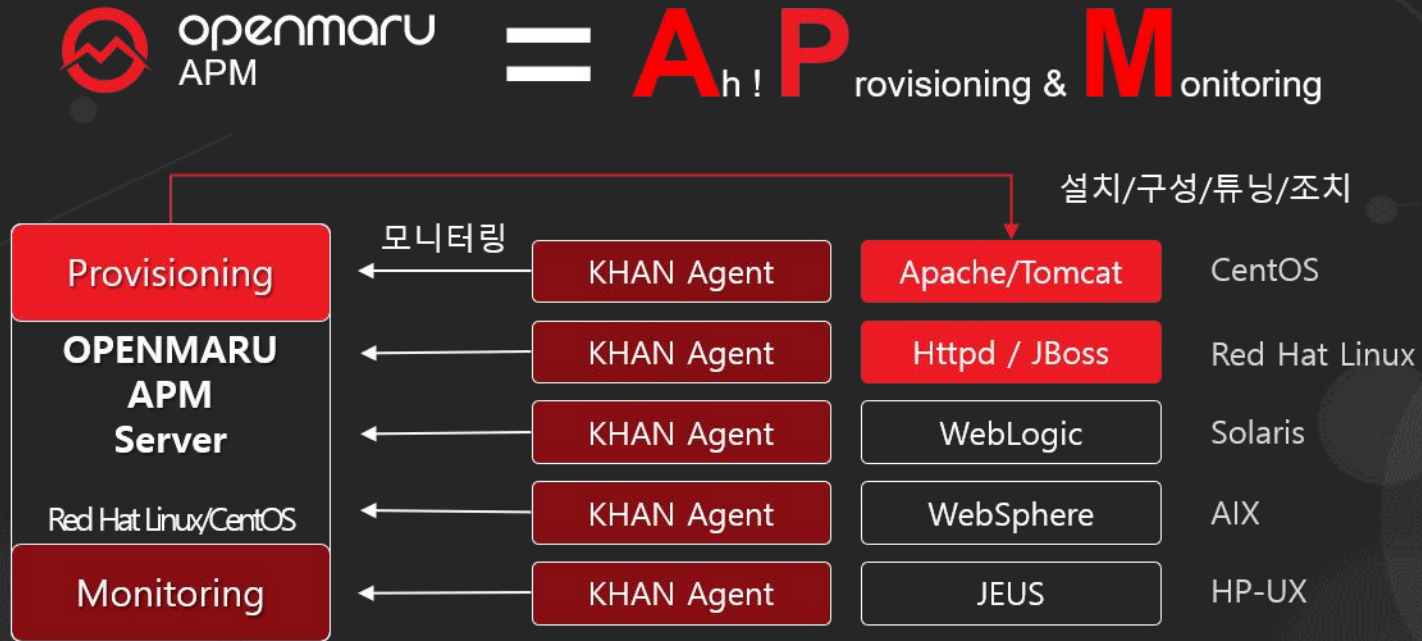


- 오픈 소스 기반의 저비용 고효율 웹시스템 구축
- APM 을 통한 운영 효율성과 안정성 확보
- 서브스크립션 구매형태의 비용 절감과 벤더 종속성 탈피
- 웹시스템에 대한 일원화된 기술 지원 체계



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



고객 요구사항에 따른 웹 시스템 작업



OPENMARU APM - Provisioning 기능은?



OS 만 설치되어 있으면
수분 이내에 설치 환경을
테스트하고 웹서버
와 WAS서버를 설치하
고 즉시 서비스할 수 있
는 환경 제공



미들웨어 전문가가 아니
어도 전문가 수준의 시
스템 튜닝이나 난이도
높은 구성을 할 수 있도
록 기능 제공



서버 구성에 대한 정보
만 입력하면 한대에서
수 십대까지 규모에 상
관없이 자동으로
웹시스템 운영환경을
구성

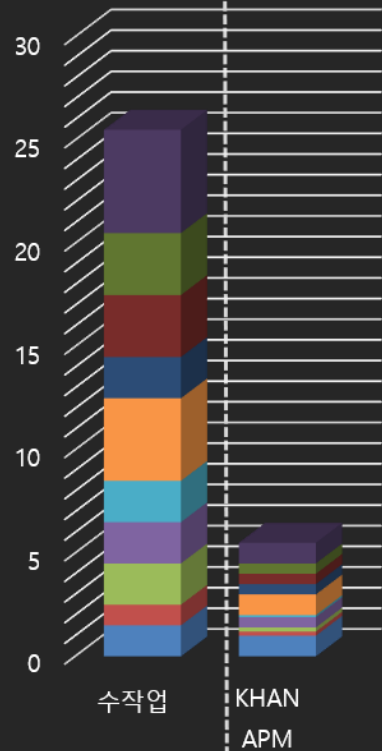
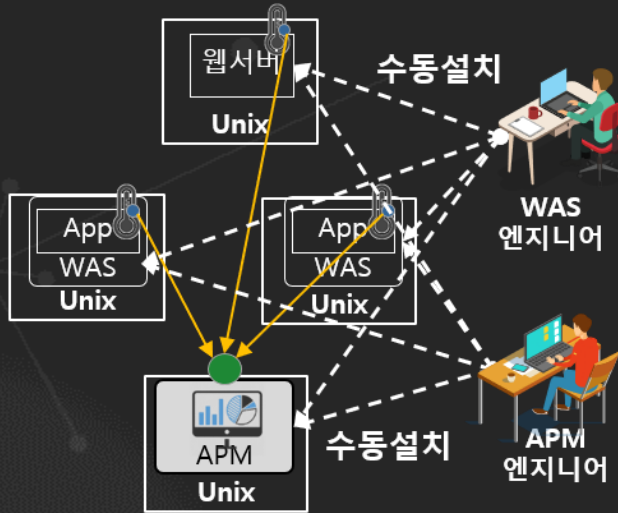


웹시스템 설치/구성
보고서를 시스템에 맞
게 자동으로 생성하여
개발팀과 운영 팀에게
제공

수 분 내 튜닝된 웹 서버와 WAS 서버로 웹 시스템 구축

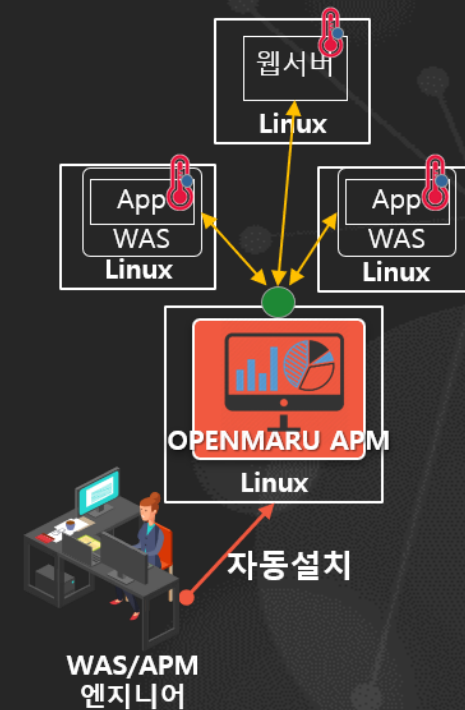
OPENMARU APM 제품 비교 - 개요

기존 웹시스템 구축 - 수작업



OPENMARU APM - 자동화

- 장애지원
- 운영교육
- 안정화
- 오픈지원
- 성능및 통합 테스트
- APM설치
- 단위테스트
- 구성
- 설치
- App 배포



- ✓ 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

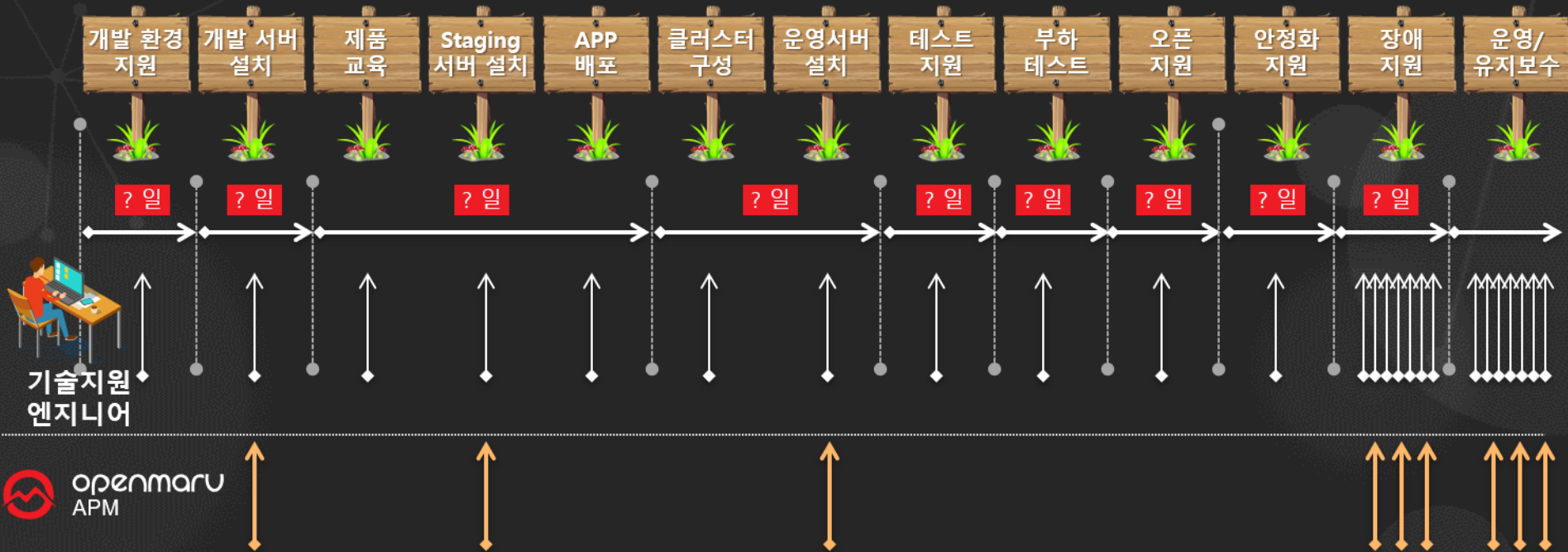


항목	수작업에 의한 웹시스템 구축	OPENMARU Installer 을 통한 웹시스템 구축
설치 방법		
구축 기간	<ul style="list-style-type: none"> 수일 ~ 수 주 	<ul style="list-style-type: none"> 수 시간 ~ 수일
안정성	<ul style="list-style-type: none"> 담당자의 기술 수준에 따른 품질의 차이 수작업으로 인한 Human Error 	<ul style="list-style-type: none"> JBoss 전문가에 의해서 구성되는 최적화된 환경으로 웹서버/WAS 환경 구성 자동화 구성으로 설치/구성의 오류 발생률 최소화
튜닝	<ul style="list-style-type: none"> 설치 담당자의 기술력에 따라 튜닝 적용의 차이점 발생 표준화하더라도 해당 가이드 적용에 대하여 지속적인 관리 필요 	<ul style="list-style-type: none"> 전문가에 의한 각 부분별 튜닝 적용 자동화된 구성으로 인하여 최적화된 표준환경 구성
유지보수	<ul style="list-style-type: none"> 구성 변경 시 마다 수작업 	<ul style="list-style-type: none"> 구성 파일을 수정하여 자동화된 재설치
생산성	<ul style="list-style-type: none"> 수작업에 의한 다운로드/튜닝 OS/웹서버/WAS 서버 별 수작성 설치 구성 	<ul style="list-style-type: none"> 자동화된 다운로드, 설치, 구성, 튜닝 웹서버/WAS 서버에 최적화된 환경으로 표준화
벤더	<ul style="list-style-type: none"> WebLogic/WebSpehrere/JEUS/Resin Apache httpd / Apache tomcat 	<ul style="list-style-type: none"> 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



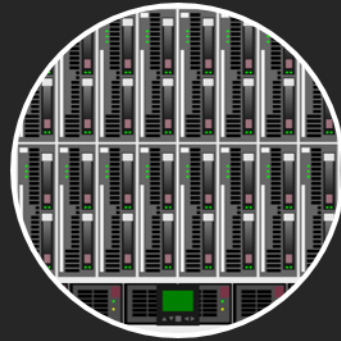
OPENMARU APM - Provisioning 기능은?



OS 만 설치되어 있으면
수분 이내에 설치 환경을
테스트하고 웹서버
와 WAS서버를 설치하
고 즉시 서비스할 수 있
는 환경 제공



미들웨어 전문가가 아니
어도 전문가 수준의 시
스템 튜닝이나 난이도
높은 구성을 할 수 있도
록 기능 제공



서버 구성에 대한 정보
만 입력하면 한대에서
수 십대까지 규모에 상
관없이 자동으로
웹시스템 운영환경을
구성



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

수 분 내 튜닝된 웹 서버와 WAS 서버로 웹 시스템 구축

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

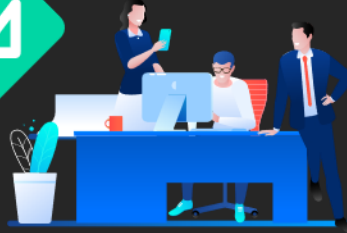
03



오픈소스 S/W에 최적화된 APM

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

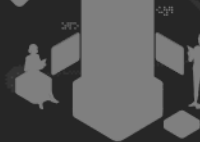
04



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



QA팀

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



openmaru
APM



운영팀

- 성능에 대한 최적화와 서비스 정상 유/무
- 신속한 장애 감지와 원인 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.



Opennaru Quick Service 란?

고객사

① 시스템 이슈발생



② 쓰레드 덤프와 모니터링 화면 전달



③ 오픈나루 서비스 데스크 접수



④ 장애 분석



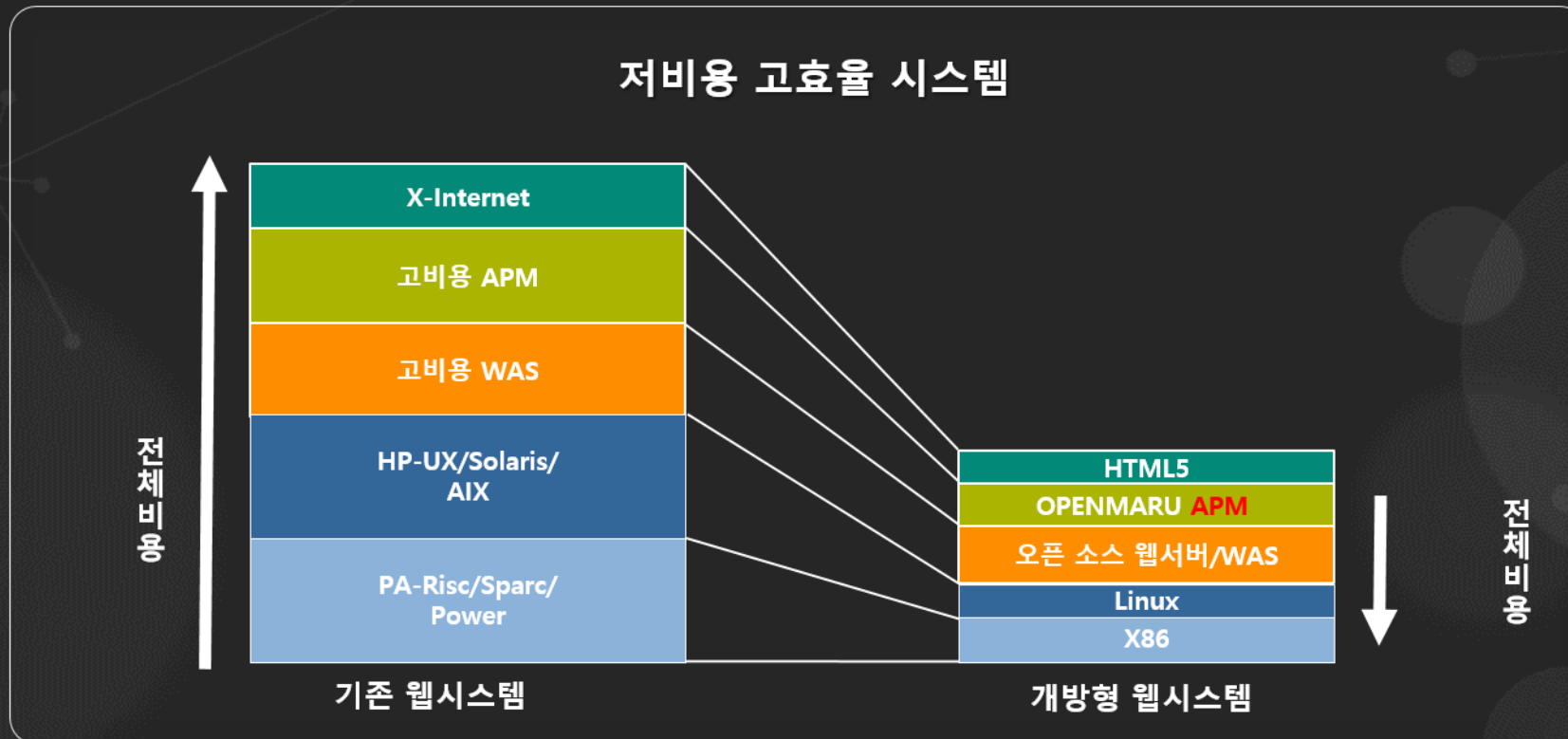
⑤ 장애 분석 보고서 전달




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




01



차별화된 기능

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


02



전문 장애분석 도구 제공

- 스택드 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

Development Process



WATERFALL



AGILE



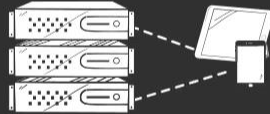
DEVOPS



Application Architecture



MONOLITHIC



N-TIER



MICROSERVICES



Deployment & Packaging



PHYSICAL SERVERS



VIRTUAL SERVERS



CONTAINERS



Application Infrastructure



DATA CENTER



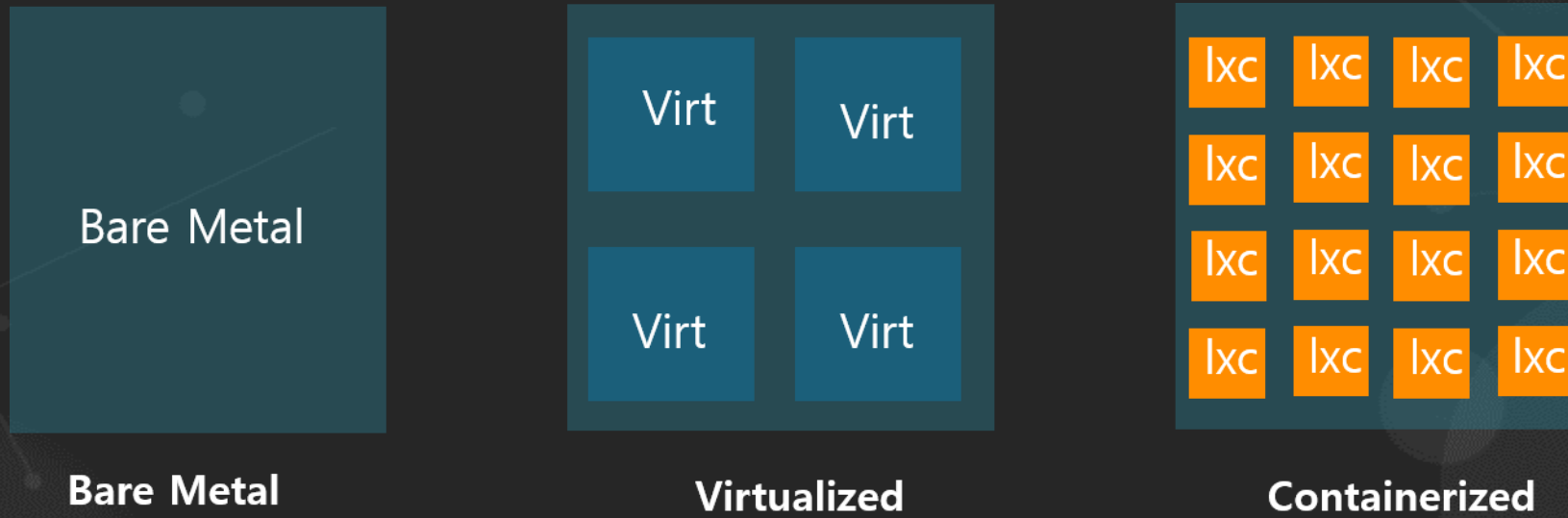
HOSTED



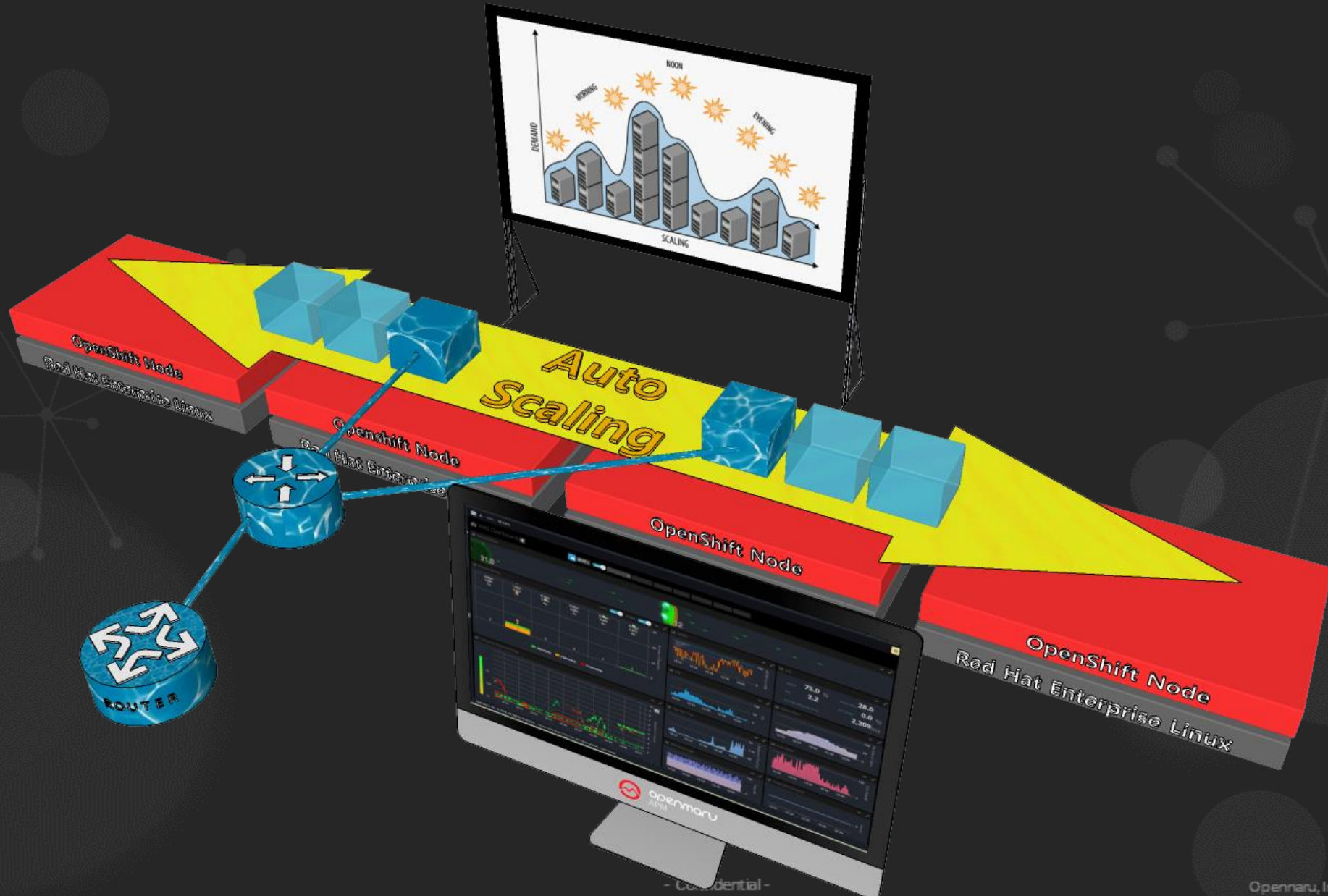
CLOUD



Evolution of Infrastructure Architectures



OpenShift Auto Scaling 과 OPENMARU APM





Red Hat OpenShift Container Platform

오토스케일링 데모

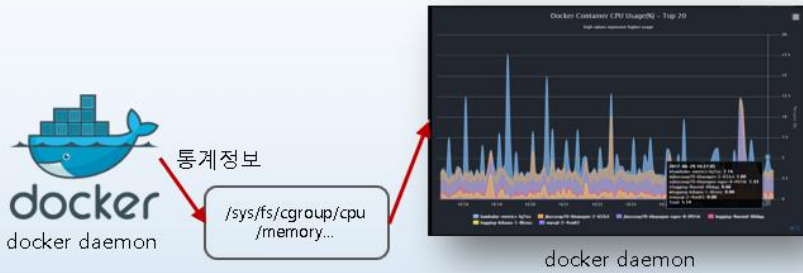
한국레드햇
김현수이사

Paas 환경에서 OPENMARU APM이 필요한 이유는?

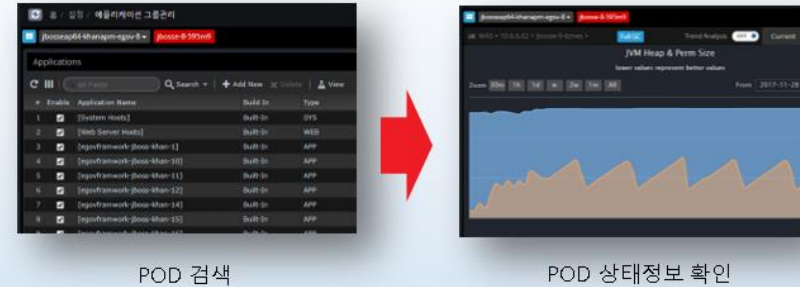


- Docker 환경은 가상OS환경이기 때문에 시스템 리소스관리가 어려움
- 유동IP이며, 멀티캐스트를 지원하지 않는 환경

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	<ul style="list-style-type: none"> • 모니터링 지원 	<ul style="list-style-type: none"> • 모니터링 지원
	WAS	<ul style="list-style-type: none"> • 모니터링 지원 	<ul style="list-style-type: none"> • 모니터링 지원
클라우드 온-나라 시스템	PaaS 환경	<ul style="list-style-type: none"> • Docker Daemon 과 관련 이미지 정보 제공 • POD 에 대한 리소스 정보 제공 (CPU/Memory/Disk/Network) 	<ul style="list-style-type: none"> • 모니터링 지원 불가
	오토스케일링	<ul style="list-style-type: none"> • 오토스케일링 관련 리소스 정보 제공 • WAS 상태 정보 제공 	<ul style="list-style-type: none"> • 모니터링 지원 불가
	PaaS 에서 Java 정보	<ul style="list-style-type: none"> • POD 상에서 실행되는 Java 가상 머신 정보 제공 (Heap , Java 상태 정보 등) 	<ul style="list-style-type: none"> • 모니터링 지원 불가
	PaaS 에서 WAS 정보	<ul style="list-style-type: none"> • POD 상에서 실행된 WAS 에 대한 정보 	<ul style="list-style-type: none"> • 모니터링 지원 불가
	POD 상태 정보 제공	<ul style="list-style-type: none"> • 폐기된 POD 에 대한 검색 지원 • 과거 POD 에 대한 정보 제공 	<ul style="list-style-type: none"> • 모니터링 지원 불가

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

Docker

도커 정보

컨테이너

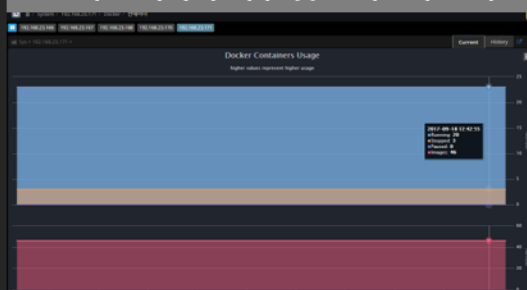
CPU

메모리

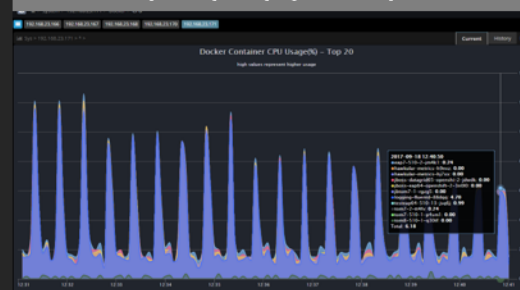


도커 모니터링 정보

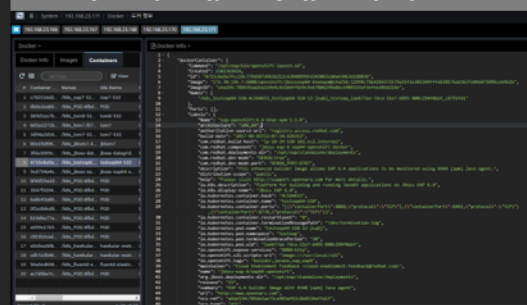
도커 이미지의 개수 및 컨테이너 상태



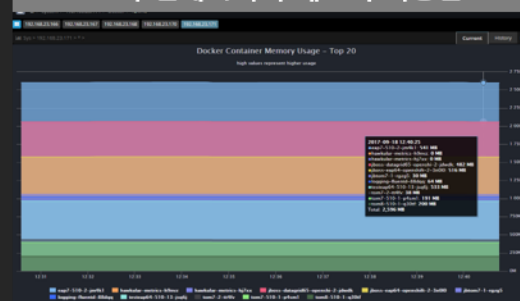
도커 컨테이너의 CPU 사용률



도커 컨테이너, 이미지, 도커 데몬 정보



도커 컨테이너의 메모리 사용률



컨테이너에서 운영되는 WAS의 성능 모니터링 기능

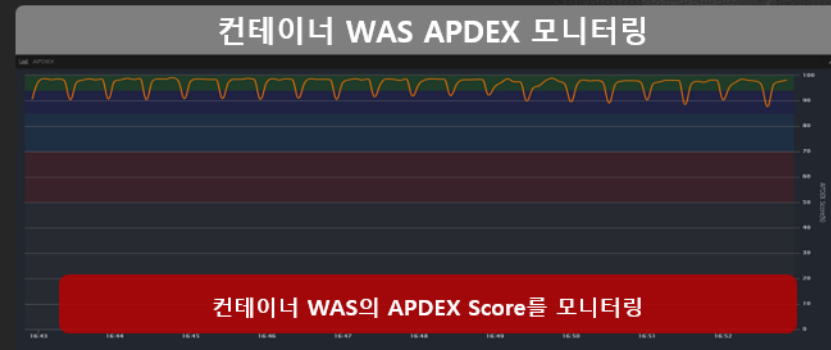
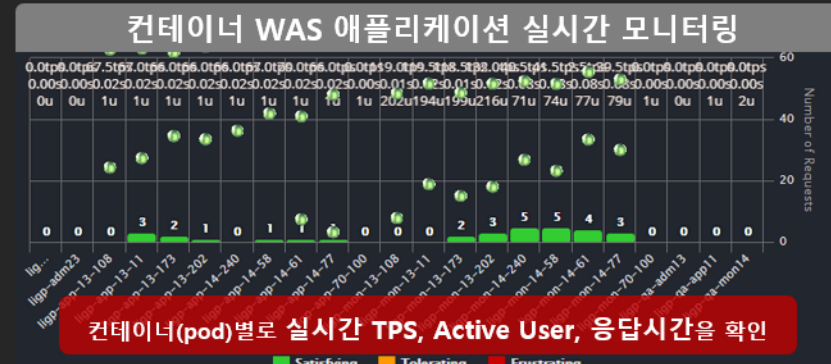
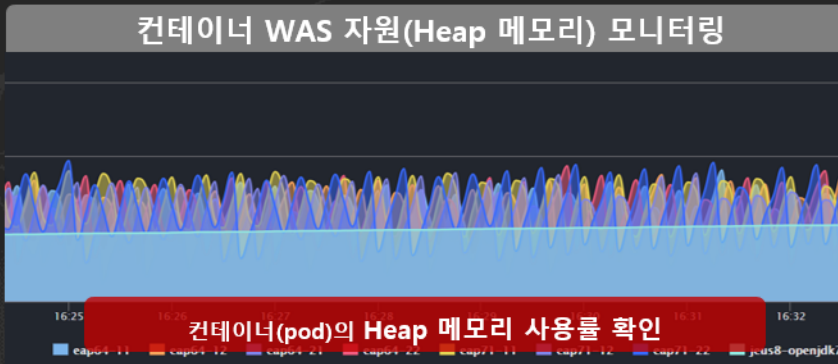
컨테이너 내의 WAS 자원 및 애플리케이션 모니터링



- 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



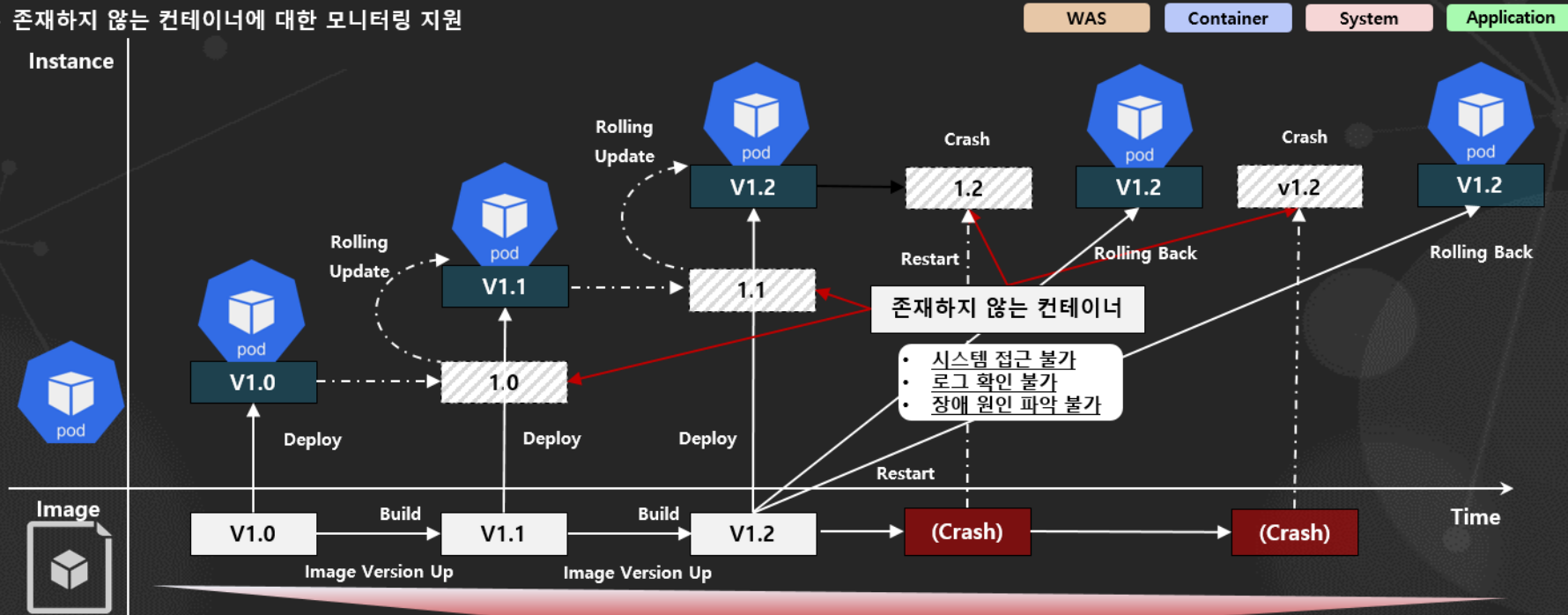
비상태 WAS 인스턴스에 대한 모니터링 기능

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.

▶ 존재하지 않는 컨테이너에 대한 모니터링 지원



실시간 데이터 수집
비상태 정보 기록
OPENMARU APM Server



Web Console

- 비상태 인스턴스 조회가능
- 사라진 장애 시점의 데이터 조회가능
- 인스턴스 장애 원인 파악가능

휘발성 인스턴스 모니터링 기록 추적 기능

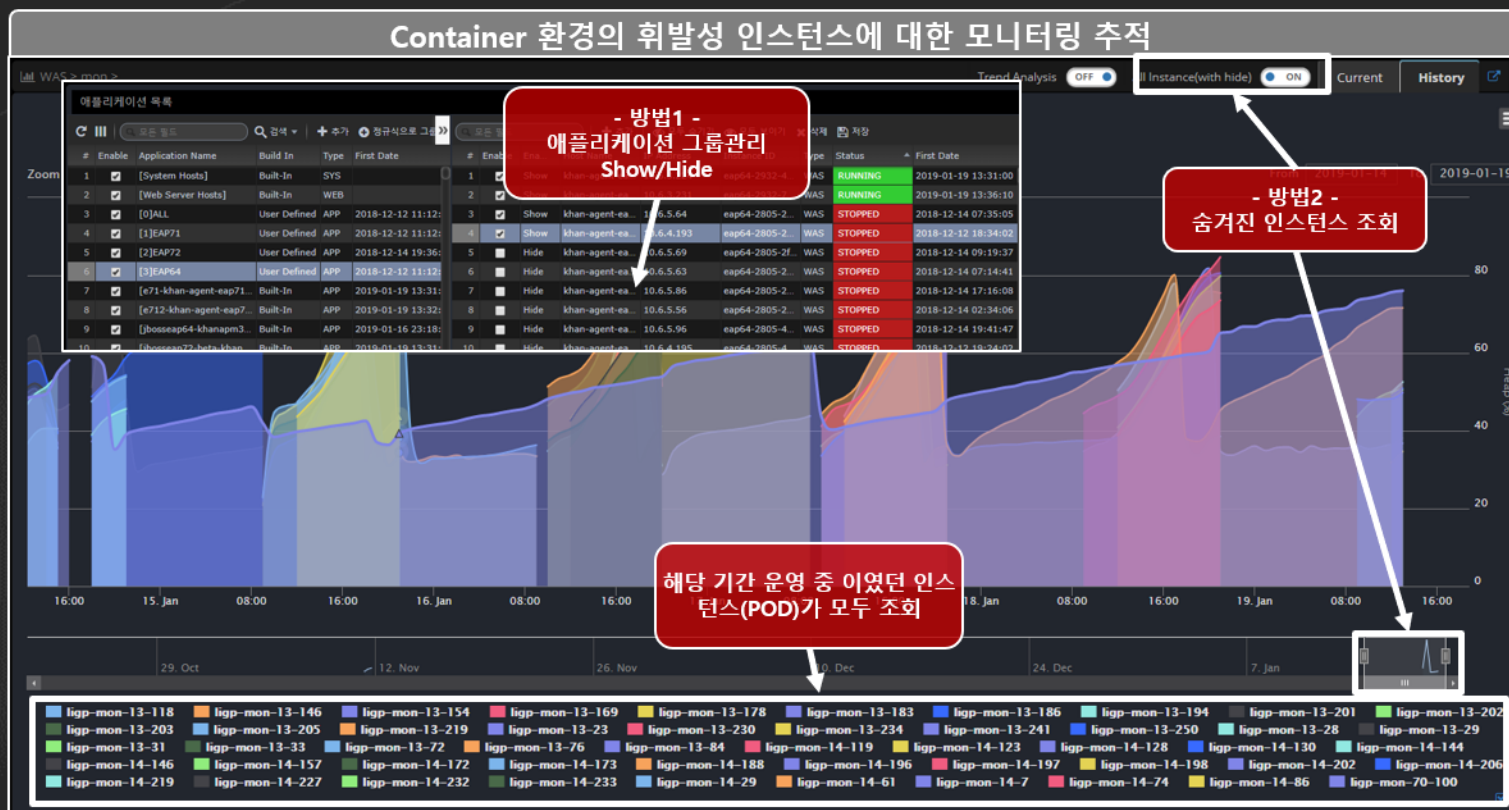
중지된 컨테이너에 대한 추적 기능



- 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



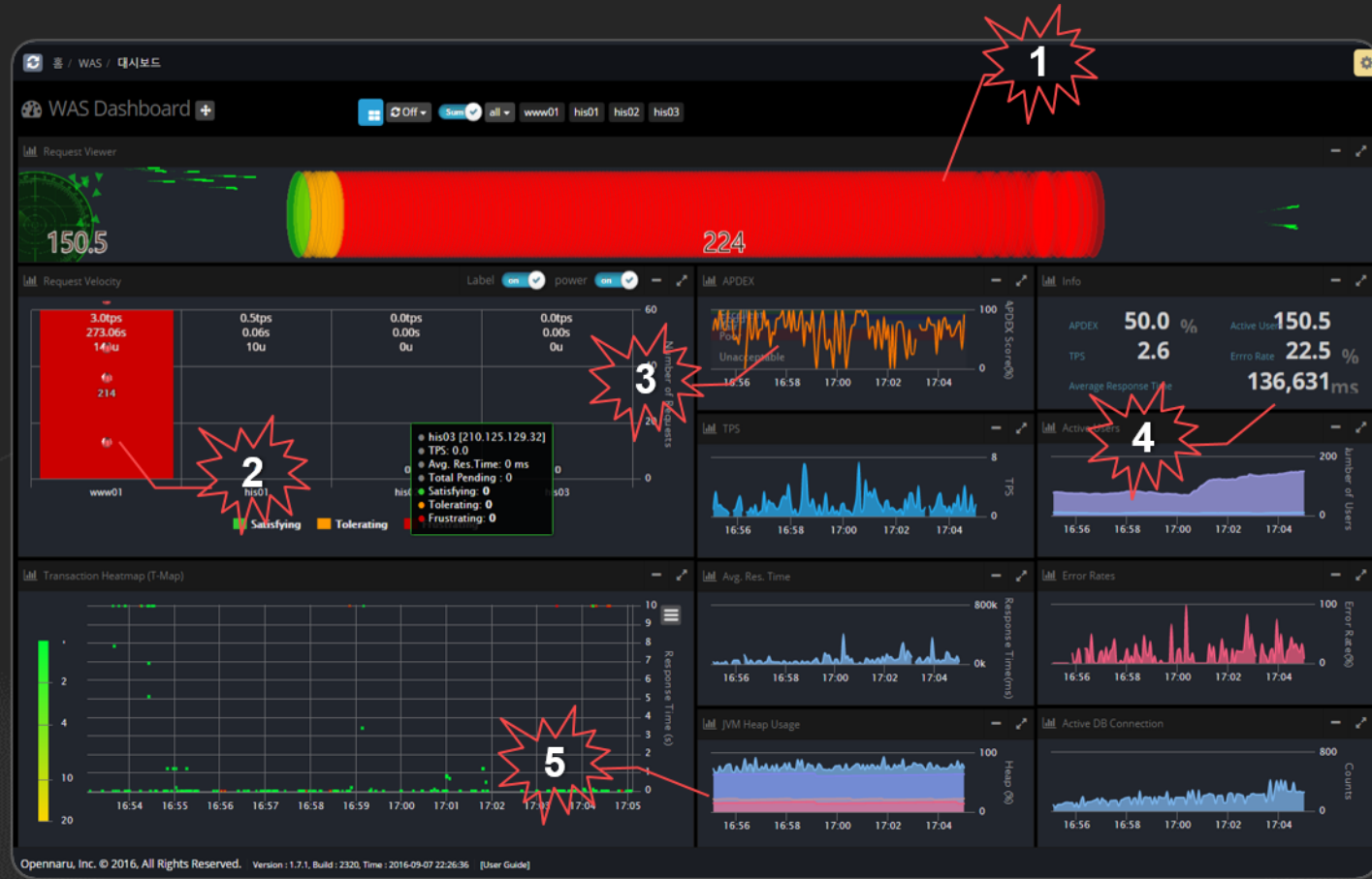
Micro Service Architecture

OPENMARU APM Use Case

Application Performance Management

사건#1
웹사이트가 멈추었습니다.

사건#1 현장 – 웹사이트가 멈추었습니다.



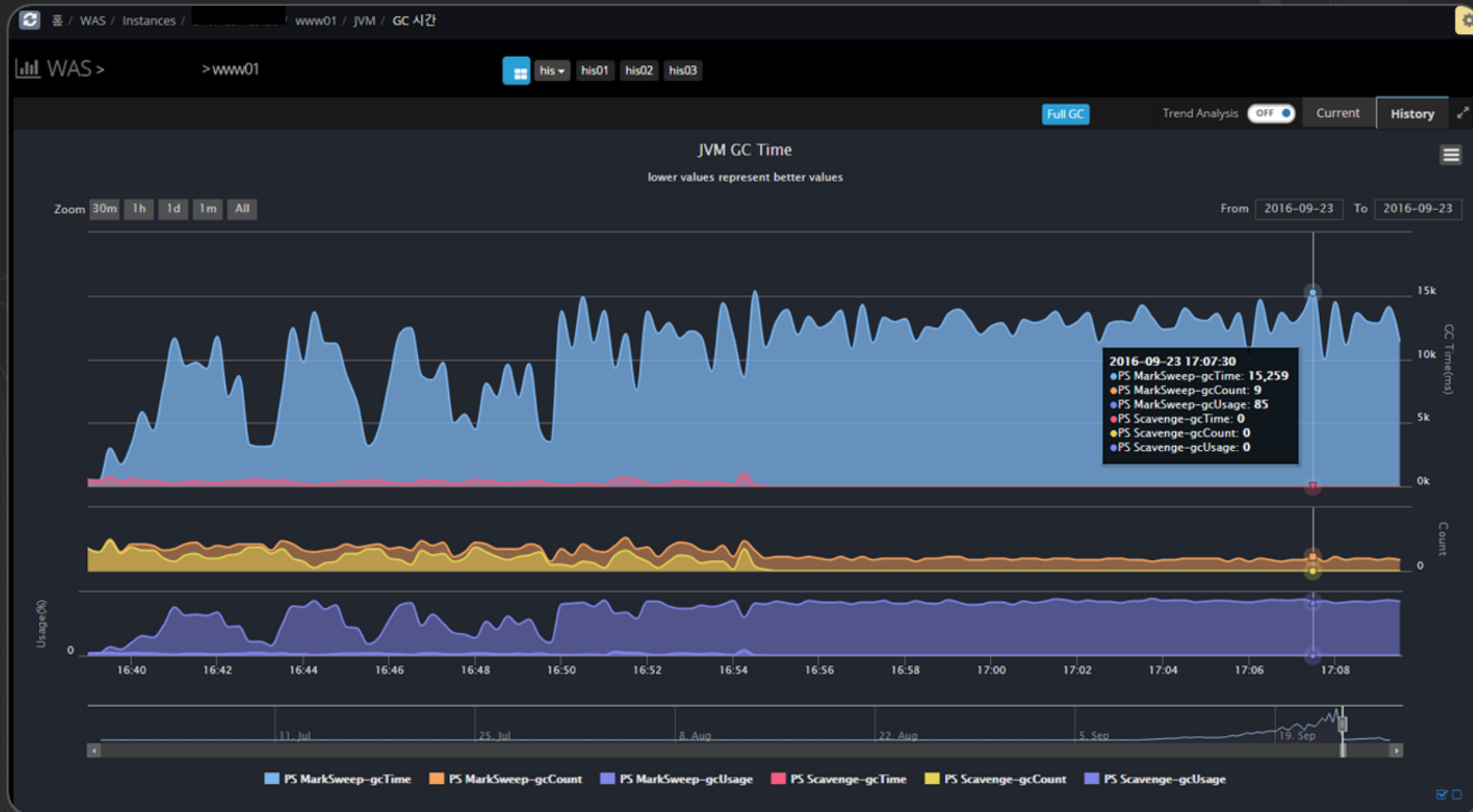
사건#1 용의자 – JVM Heap Memory

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



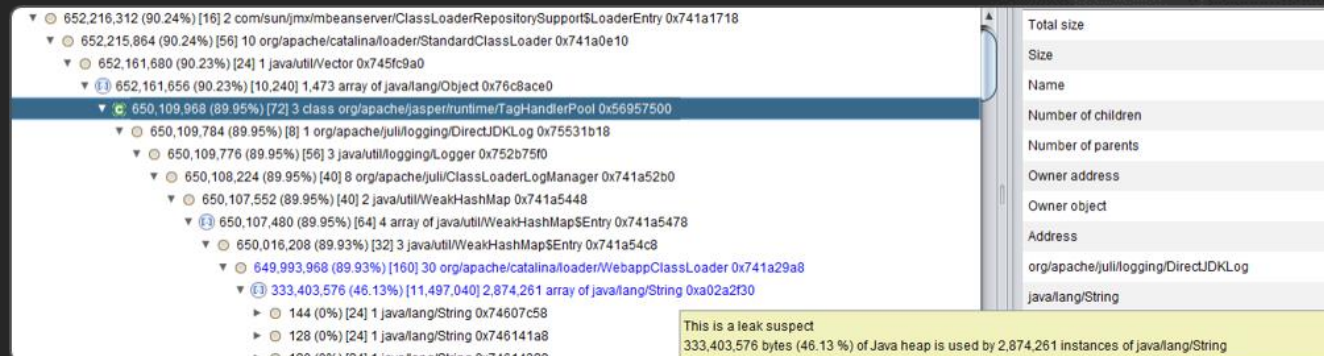
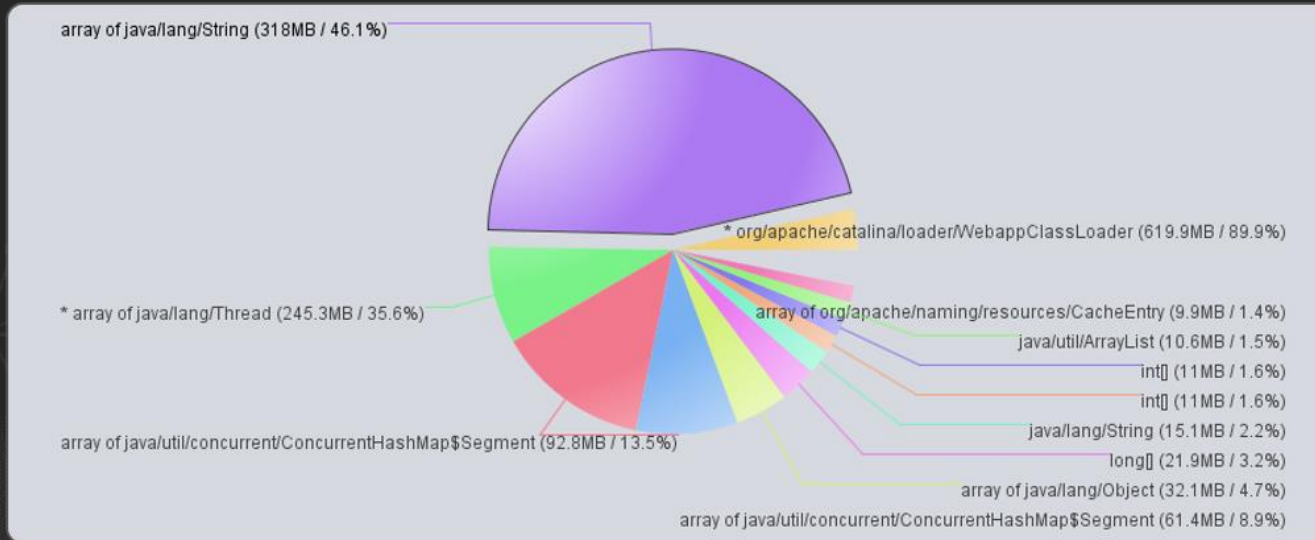
사건#1 용의자 – JVM Heap Memory

- 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

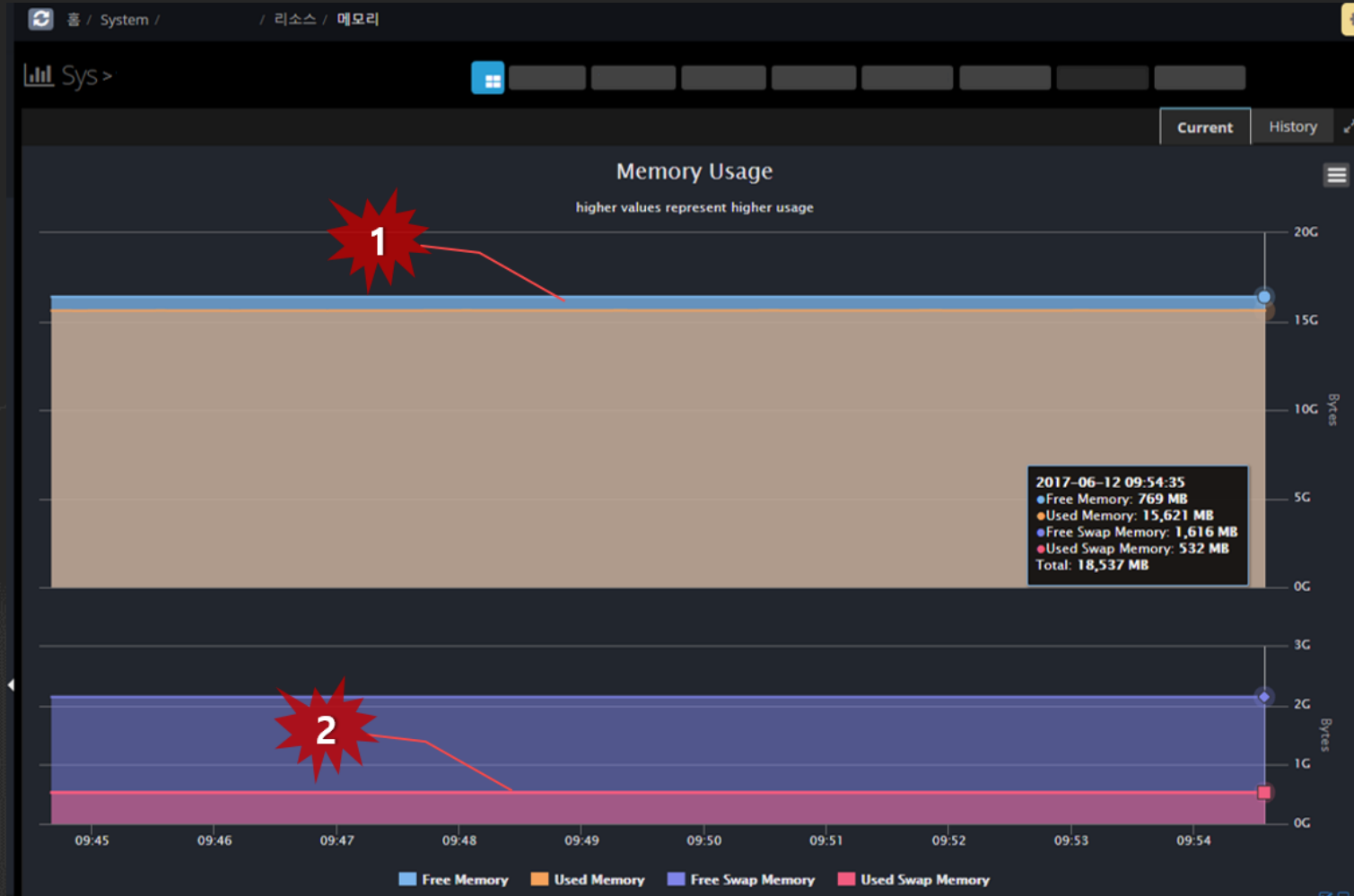
- 88.95%의 메모리를 Tomcat의 TagHandlerPool에서 점유하고 있음.
- Tomcat Logger에서 사용하는 287만개의 String 객체가 46.13%의 메모리를 점유



Application Performance Management

Episode #2 시스템 메모리 고갈

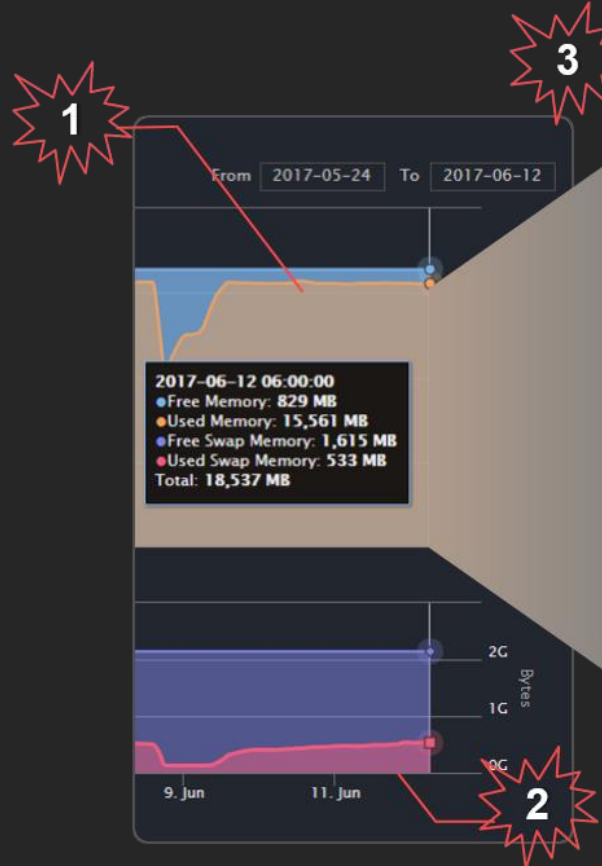
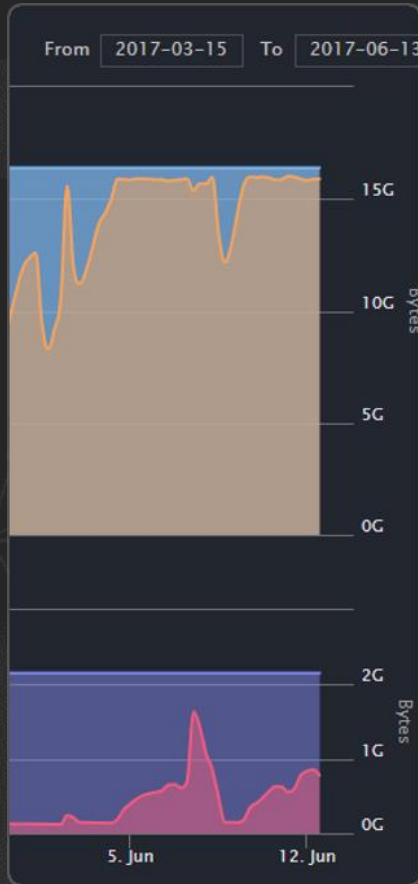
Episode #2 현장 – System 메모리



1 사용 가능한 메모리를 거의 소진

2 Swap 메모리를 500MB 사용

Episode #2 현장 – System 메모리



```
[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:      1657880 kB
Active(anon):  9907128 kB
Inactive(anon): 1524664 kB
Active(file):  151016 kB
Inactive(file): 132716 kB
Unevictable:   0 kB
Mlocked:       0 kB
SwapTotal:     2097148 kB
SwapFree:      1461308 kB
Dirty:         180 kB
Writeback:     0 kB
AnonPages:     11389316 kB
Mapped:        22120 kB
Shmem:         5652 kB
Slab:          3937136 kB
SReclaimable: 3903356 kB
SUnreclaim:    33380 kB
```

- 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

```

top - 13:18:19 up 1 day, 15:17, 5 users, load average: 2.26, 2.04, 1.93
Tasks: 554 total, 1 running, 541 sleeping, 0 stopped, 12 zombie
Cpu(s): 28.3% user, 0.0% nice, 65.5% idle, 0.0% wa, 0.0% hi, 0.4% si
Mem: 5764320k total, 57279644k used, 364012k free, 4860000k buffers
Swap: 16777212k total, 43076k used, 16734136k free, 14568280k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 26803 topasibe  20   0 10.2g 2.5g 10m  S 185.2  4.6 270:23.91 java
 27292 topasibe  20   0  9.9g 2.5g 10m  S 150.4  4.6 270:20.61 java
 28969 topasibe  20   0  9918m 2.4g 10m  S 45.4  4.4  86:25.93 java
 28415 topasibe  20   0  9984m 2.4g 10m  S 43.1  4.3  86:52.44 java
 29459 topasibe  20   0  9919m 2.4g 10m  S 39.1  4.3  87:40.68 java
 27824 topasibe  20   0  9919m 2.4g 10m  S 30.8  4.3  85:39.91 java
 2233 topasibe  20   0  8803m 1.5g 10m  S 28.5  2.6   7:05.96 java
 1132 topasibe  20   0  8948m 1.5g 10m  S  9.9  2.7   7:21.24 java
 5389 khan      20   0  2360m 167m 9952  S  3.6  0.3 120:28.31 java
 29700 daemon   20   0  2171m 1.9g 10m  S  1.5  3.3  6:03.02 httpd
 9606 daemon   20   0  2011m 10m 2170  S  1.3  0.0  6:03.02 httpd
 32692 topasibe  20   0  7984m 1.7g 10m  S  1.3  3.1 22:24.71 java
 28496 daemon   20   0  2619m 1.0g 10m  S  1.0  0.6  6:49.77 httpd
 729 topasibe  20   0  10572m 2.1g 10m  S  1.1  3.7  6:51.70 java
 1667 topasibe  20   0  9208m 1.4g 10m  S  0.7  2.6  6:51.70 java
 29329 root     20   0 15424 1596 928  R  0.7  0.0  0:00.09 top
 30208 topasibe  20   0  9099m 620m 9856  S  0.7  1.1 17:25.62 java
 30483 topasibe  20   0  9057m 313m 9828  S  0.7  0.6 13:00.58 java
 30818 topasibe  20   0  9059m 622m 9864  S  0.7  1.1 16:46.74 java
 31036 topasibe  20   0  9099m 422m 9832  S  0.7  0.7 14:54.65 java
 13 root     20   0  0 0 0  S  0.3  0.0  0:23.32 ksoftirqd/2
 1282 topasibe  20   0  7254m 1.6g 10m  S  0.3  2.9  9:28.78 java
 2180 root     20   0  0 0 0  S  0.3  0.0  5:12.36 kondemand/0
 2181 root     20   0  0 0 0  S  0.3  0.0  3:51.14 kondemand/1
 2182 root     20   0  0 0 0  S  0.3  0.0  3:28.49 kondemand/2
 2183 root     20   0  0 0 0  S  0.3  0.0  3:10.82 kondemand/3
 2184 root     20   0  0 0 0  S  0.3  0.0  3:06.81 kondemand/4
 2185 root     20   0  0 0 0  S  0.3  0.0  1:53.11 kondemand/5

root@was:/root# free -m
              used        free      shrd  buffers  cached
Mem:           56292          322           1          4746          14251
-/+ buffers/cache: 36972          19320
Swap:          16383             42          16341
root@was:/root#
  
```

리눅스 운영체제의 가용 메모리는
Free + Buffers + Cached로 파악해야 함

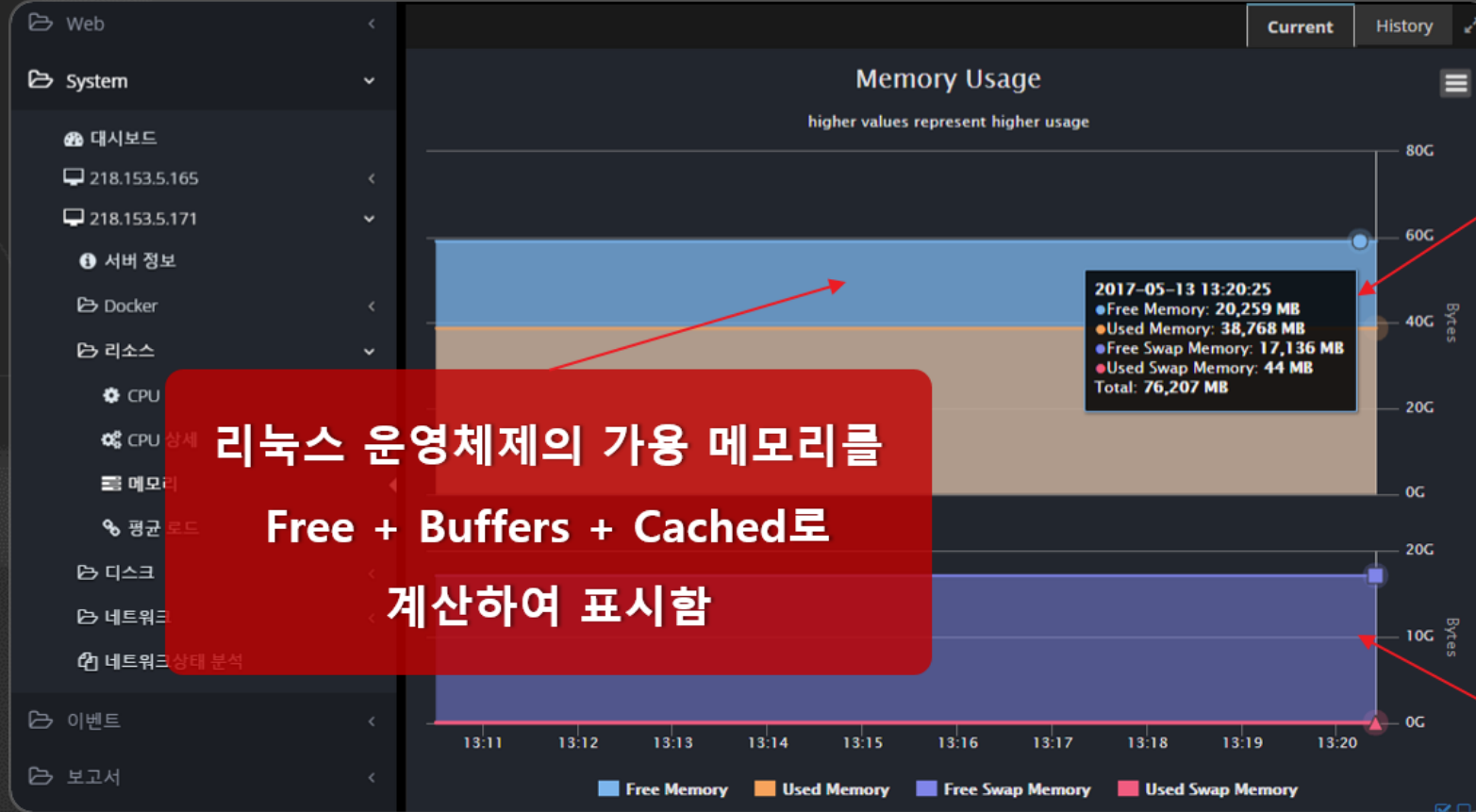
Linux 메모리 관리방식

- 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



Episode #2 현장 – System 메모리



Episode #2 현장 – System 메모리



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



```
[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: 0 kB
Mlocked: 0 kB
SwapTotal: 0 kB
SwapFree: 0 kB
Dirty: 100 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
```

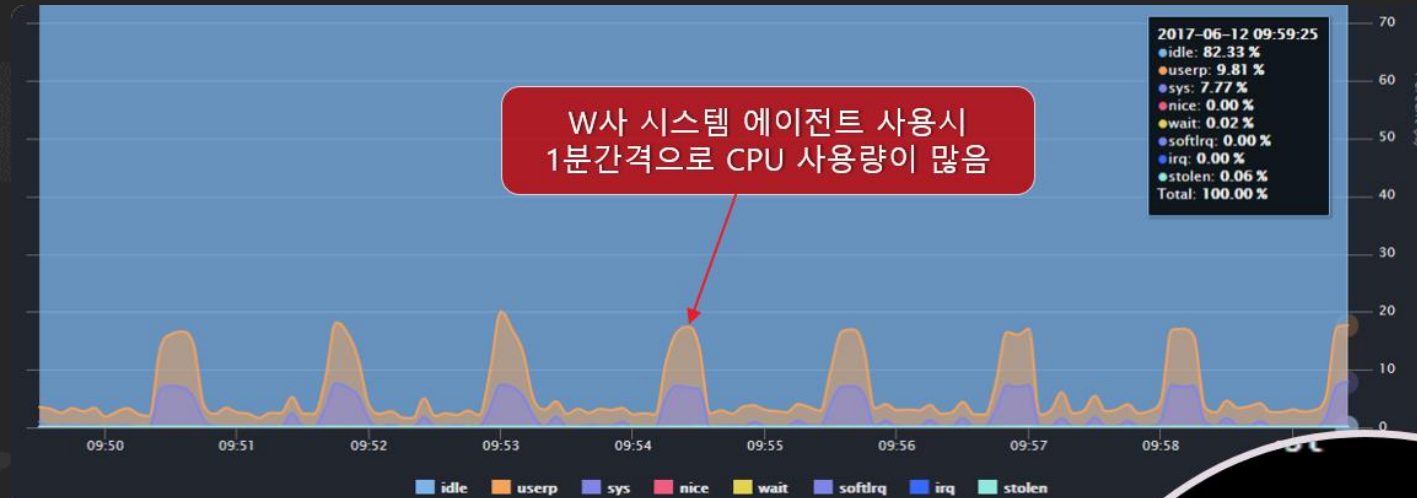
리눅스 커널 메모리중 slab 메모리를 3.9GB 사용하고 있음
→ slabtop으로 분석 필요

```
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	51	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	5007	55%	1.03K	243	31	10076K	nfs_inode_cache	
10323	9118	91%	0.31K	378	37	3333K	vm_area_struct	
9690	9690	100%	0.03K	114	9	496K	shared_policy_node	
8364	8264	99%	0.03K	114	9	496K	ext4_extent_status	
8184	5195	63%	1.03K	293	31	8448K	ext4_inode_cache	
7911	7790	98%	0.50K	293	27	4688K	inode_cache	
7552	5679	75%	0.06K	293	64	472K	anon_vma	
7488	6487	86%	0.25K	234	32	1872K	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 840MB와 proc_inode_cache 2960MB를 사용하고 있음
slab

Episode #2 검거 – Linux 커널 캐시 메모리 상태



```
ser. load average: 0.37, 0.32, 0.32
mping. 0 stopped, 0 zombie
.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
d. 201700 free, 11472 buffers
d. 1885372 free, 729988 cached Mem
```

SHR	S	%CPU	MEM	TIME+	COMMAND
18736	S	11.9	34.7	159:13.35	/usr/java/latest/bin/java -D[St...
18596	S	10.9	33.4	162:10.35	/usr/java/latest/bin/java -D[St...
5864	S	0.0	0.0	87:39.61	/usr/java/latest/bin/java -D[St...
1428	S	100.2	0.0	0:01.40	/usr/whatap/monitoring/wps
4000	S	0.0	0.0	0:00.06	sshd: root@pts/0
4204	S	0.0	0.0	0:05.40	/usr/lib/systemd/systemd-journ...
1280	S	0.0	0.0	0:14.55	/usr/lib/systemd/systemd-logind
1612	S	0.0	0.0	0:00.00	-bash
1264	R	0.0	0.0	0:00.87	top
1532	S	0.0	0.0	0:01.02	/usr/sbin/rsyslogd -n
596	S	0.0	0.0	1:59.74	/bin/bash /usr/sbin/xm-damon
1296	S	0.0	0.0	0:00.01	/bin/sh /svc/cnet/was/jboss-map...
1296	S	0.0	0.0	0:00.01	/bin/sh /svc/cnet/was/jboss-map...
812	S	0.0	0.0	0:00.48	/usr/lib/polkit-1/polkitd --no-de...
664	S	0.0	0.0	7:05.36	/usr/bin/python -Es /usr/sbin/tun...
820	S	0.0	0.0	0:04.80	/usr/lib/systemd/systemd-logind
624	S	0.0	0.0	10:01.96	/usr/whatap/monitoring/whatap_agend...
828	S	0.0	0.0	0:33.26	/usr/whatap/monitoring/whatap_agend...
488	S	0.0	0.0	0:02.82	/bin/dbus-daemon --system --address=system...
584	S	0.0	0.0	0:03.33	/usr/sbin/chronyd -u chrony
536	S	0.0	0.0	0:01.80	avahi-daemon: running [cnet-naru-was03.local]
384	S	0.0	0.0	11:05.91	/usr/sbin/irqbalance --foreground
464	S	0.0	0.0	0:11.14	/usr/sbin/crond -n
460	S	0.0	0.0	0:00.16	/usr/sbin/sshd -D
380	S	0.0	0.0	0:00.15	/usr/lib/systemd/systemd-udev
276	S	0.0	0.0	0:00.00	sleep 60
396	S	0.0	0.0	0:00.01	/usr/sbin/rpc.statd --no-notify

Application Performance Management

Episode #3 응답속도가 갑자기 느려 짐

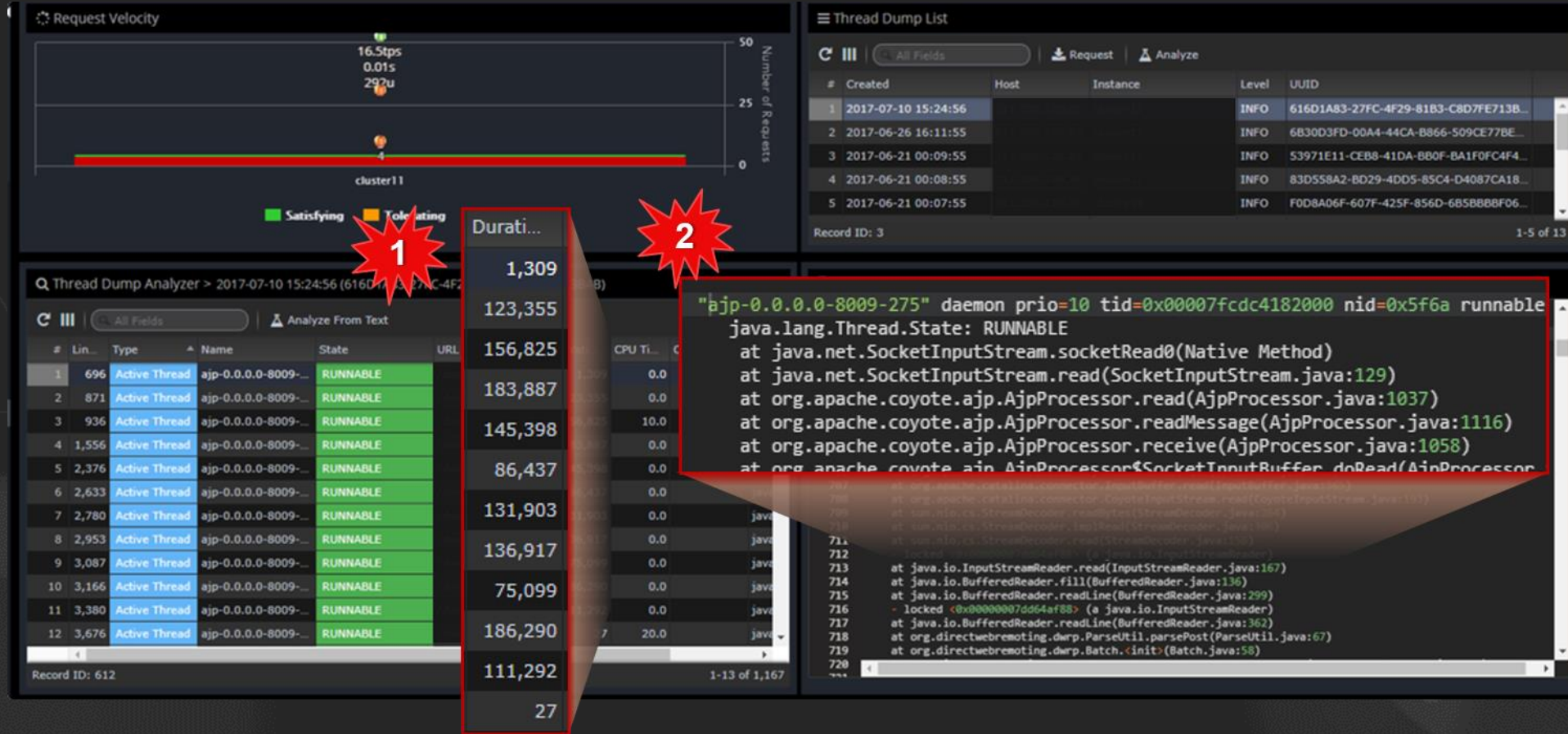
Episode #3 현장 – Pending Transaction이 급증



1 거의 모든 인스턴스에서 Queue 에 쌓임

2 리눅스 운영체제의 SWAP 메모리를 533MB가량 사용하고 있음

Episode #3 용의자 – 인스턴스의 스레드 덤프 분석



Request Velocity

16.Stps
0.01s
292u

Number of Requests

cluster 1 I

■ Satisfying ■ Tolerating

Thread Dump List

#	Created	Host	Instance	Level	UUID
1	2017-07-10 15:24:56			INFO	616D1A83-27FC-4F29-81B3-C8D7FE713B...
2	2017-06-26 16:11:55			INFO	6B30D3FD-00A4-44CA-B866-509CE77BE...
3	2017-06-21 00:09:55			INFO	53971E11-CEB8-41DA-BB0F-BA1F0FC4F4...
4	2017-06-21 00:08:55			INFO	83D558A2-BD29-4DD5-85C4-D4087CA18...
5	2017-06-21 00:07:55			INFO	F0D8A06F-607F-425F-856D-6B5B88BF06...

Record ID: 3

1-5 of 13

Thread Dump Analyzer

Q Thread Dump Analyzer > 2017-07-10 15:24:56 (616D1A83-27FC-4F29-81B3-C8D7FE713B)

Analyze From Text

#	Lin...	Type	Name	State	URL	Durati...	CPU TL...
1	696	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		1,309	
2	871	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		123,355	
3	936	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		156,825	
4	1,556	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		183,887	
5	2,376	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		145,398	10.0
6	2,633	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		86,437	0.0
7	2,780	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		131,903	0.0
8	2,953	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		136,917	0.0
9	3,087	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		75,099	0.0
10	3,166	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		186,290	7 20.0
11	3,380	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		111,292	
12	3,676	Active Thread	ajp-0.0.0.0-8009-...	RUNNABLE		27	

Record ID: 612

1-13 of 1,167

```
"ajp-0.0.0.0-8009-275" daemon prio=10 tid=0x00007fcdc4182000 nid=0x5f6a runnable
java.lang.Thread.State: RUNNABLE
  at java.net.SocketInputStream.socketRead0(Native Method)
  at java.net.SocketInputStream.read(SocketInputStream.java:129)
  at org.apache.coyote.ajp.AjpProcessor.read(AjpProcessor.java:1037)
  at org.apache.coyote.ajp.AjpProcessor.readMessage(AjpProcessor.java:1116)
  at org.apache.coyote.ajp.AjpProcessor.receive(AjpProcessor.java:1058)
  at org.apache.coyote.ajp.AjpProcessor$SocketInputBuffer.doRead(AjpProcessor
711
712
713
714
715
716
717
718
719
720
721
```

- 1 • 해당 인스턴스의 스레드 덤프를 분석
- 1 • 현재 수행중인 스레드가 최대 180초 가량 수행중인 상태
- 2 • 지연되고 있는 트랜잭션(Active Thread)는 대부분 AJP
- 2 • 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 openmaru.com



JVM 옵션

GC 옵션 명시 필요

Tomcat Native(APR)

거래량이 많은 경우 APR 을 적용하는 것이 효율적임

9. 점검 총평

ICP 파라미터	관부 및 규정 검토
JVM 옵션	GC 옵션 명시 필요
Tomcat Native(APR)	거래량이 많은 경우 APR 을 적용하는 것이 효율적임
멤버버 설정	Static Content 분리 필요(별론트, 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