

EasyTrace™ Overview



1 EasyTrace™는 어떤 도구인가?

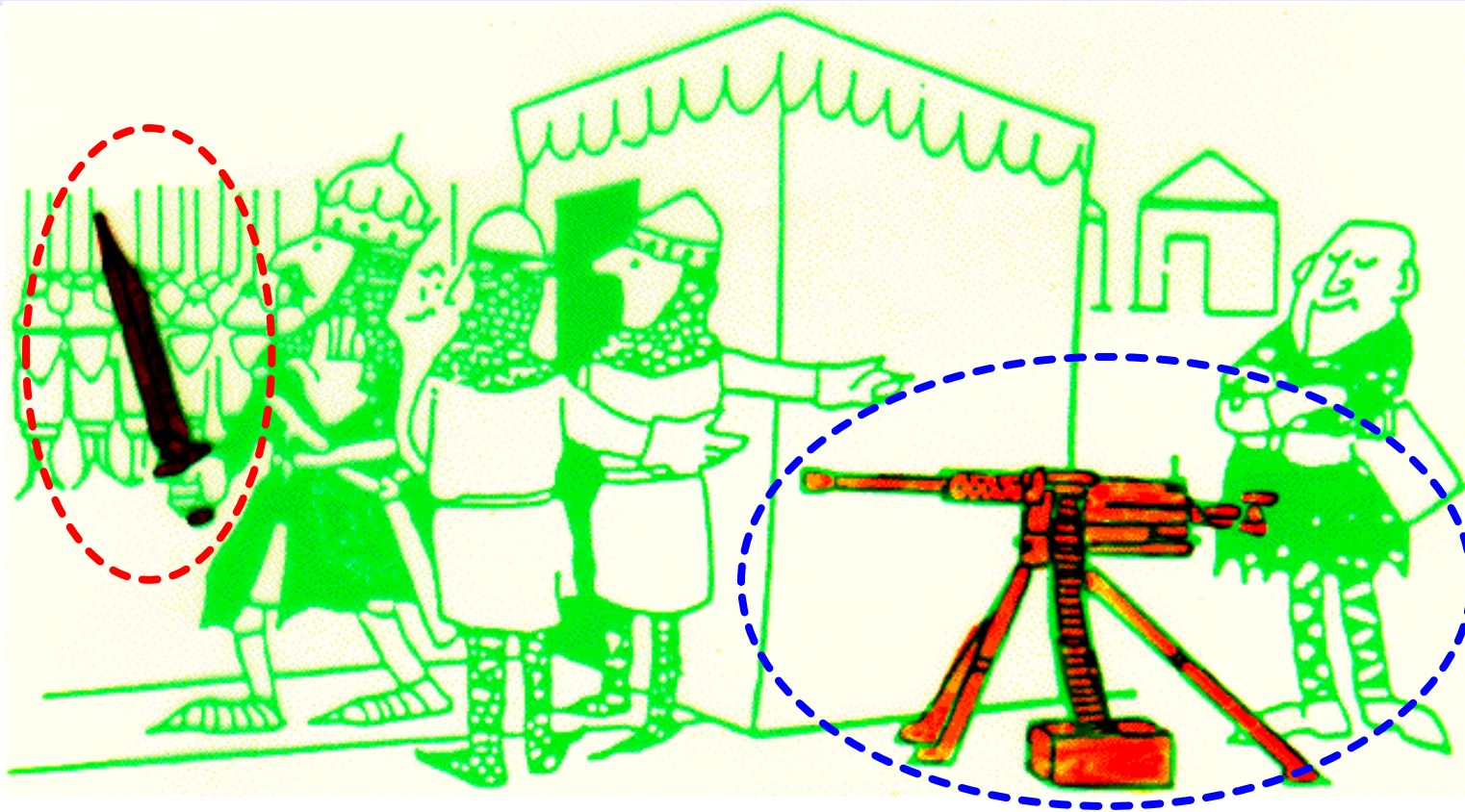
[1]

어느 비행기에 탑승하시겠습니까?



1 EasyTrace™는 어떤 도구인가?

[2]



우린 지금 전쟁터에 나가야 하는데,
무기가 ...

1 EasyTrace™는 어떤 도구인가?

[3]

①시스템 구축 프로젝트의 개발 및 테스트 단계, ②시스템 운영 유지보수 환경에서 Java 어플리케이션의 장애 해결에 특화된 AQI 도구

AQI: Application Quality Insight

인사이트(Insight) 라는 단어는 통찰력이라는 의미로 어떤 문제나 상황에 대해 깊이 있게 이해한다는 의미

장애(기능/성능) 해결 지원

- All Java Error track
- High CPU usage
- Heap Memory Leak
- Dead Lock Thread
- JDBC Connection Leak
- N+1 Query problem
- Slow Service call
- Continuous profiling

강력한 검색 기반 추적 / 근원분석

- 중앙으로 모든 상세거래 프로파일 정보 수집 및 검색 기능 제공
- 검색과 거래의 자연스러운 통합
- 다양한 검색 유형 제공
- Application Log Full Text Search
- Query Statement/Parameter Search
- Distributed trace Search

강력한 코드 레벨 프로파일

- 거래의 내부 실행 흐름 표현
- Request details / Query / Log / Service Call / Async thread 등 흐름 추적
- 거래 내부 흐름 재현 정보
- 내부 발생 사건을 한눈에 파악

1 장애 원인 추적 및 재현 지원 툴의 기능

[4]

□ 장애 원인 추적 및 재현 지원 툴인 EasyTrace™는 JAVA 어플리케이션의 장애해결에 특화된 툴이며, 시스템 구축의 개발/테스트 단계 및 운영유지보수 시 운영환경에서도 가동할 수 있음

장애(오류/성능) 해결 지원

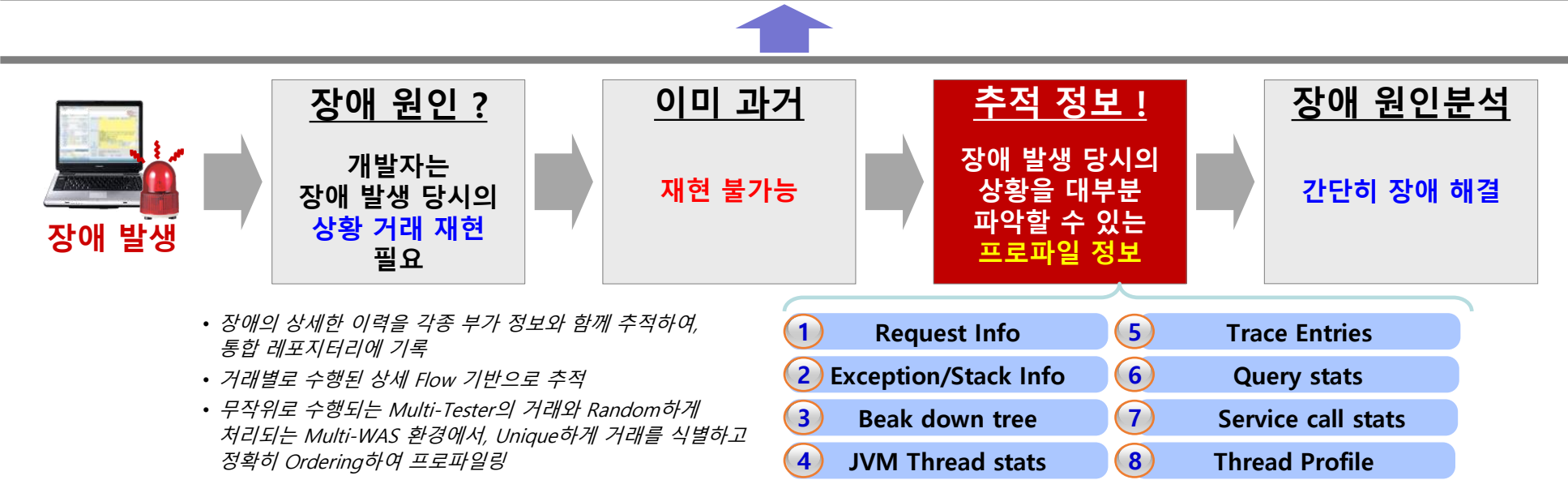
- All Java Error track
- High CPU usage
- Heap Memory Leak
- Dead Lock Thread
- JDBC Connection Leak
- N+1 Query problem
- Slow Service call
- Continuous profiling

강력한 검색 기반 추적 / 근원분석

- 중앙으로 모든 상세거래 프로파일 정보 수집 및 검색 기능 제공
- 검색과 거래의 자연스러운 통합
- 다양한 검색 유형 제공
- Application Log Full Text Search
- Query Statement/Parameter Search
- Distributed trace Search

강력한 코드 레벨 프로파일

- 거래의 내부 실행 흐름 표현
- Request details / Query / Log / Service Call / Async thread 등 흐름 추적
- 거래 내부 흐름 재현 정보
- 내부 발생 사건을 한눈에 파악





- ◆ [기본적] **할당된 개발/테스트케이스 수행(테스팅) ~**
(일정이 뒤로 가더라도 개발/테스트케이스 양이 안 줄어듦
← 비현실적인 개발/테스트 계획 : 일정 대비 동일 수량 분포)
- ◆ [기본적] **발견된 오류 해결 ~**
- ◆ [기본적] **오류 해결된 테스트케이스 재 수행 ~**
(확인테스트/회귀테스트)
- ◆ [기본적] **개발/테스트 진척 등록 ~**
- ◆ [기본적] **고객사 테스트 지원 ~ (테스트 데이터, 환경 등)**
- ◆ [일반적] **미완료 개발분 개발 ~**
- ◆ [일반적] **미완료 산출물 작성 및 보완 ~**
- ◆ ...

➔ 퇴근/출근할 시간이 없을 정도

➔ 최적화할 수 있는 부분이 어디인가 ?

기존 방식의 장애 원인분석

- ◆ 장애가 발생했다는 **신호**만 제시
- ◆ 신호의 의미를 찾기 위해 **산더미 같이 쌓여 있는 로그**를 뒤져야 하고, 관련 있다고 생각되는 **많은 사람들이** 모여 많은 시간을 소비함
 - **Multi-WAS 환경** → 어느 WAS에 내 로그가?
 - 수십명의 개발자/테스터/사용자가 동시 테스트 → 어떤 게 내 거래 로그?
- ◆ **거래 관점**의 SQL 튜닝포인트나 어플리케이션 최적화 요건을 제시하지 못함
- ◆ 관련자(설계자, 타 개발자, AA 등)와의 공유 및 문제해결 **Co-work** 방법 부재
- ◆ 단위테스트 수준에서는 어떤 도구나 수단 없이도 해결이 수월함

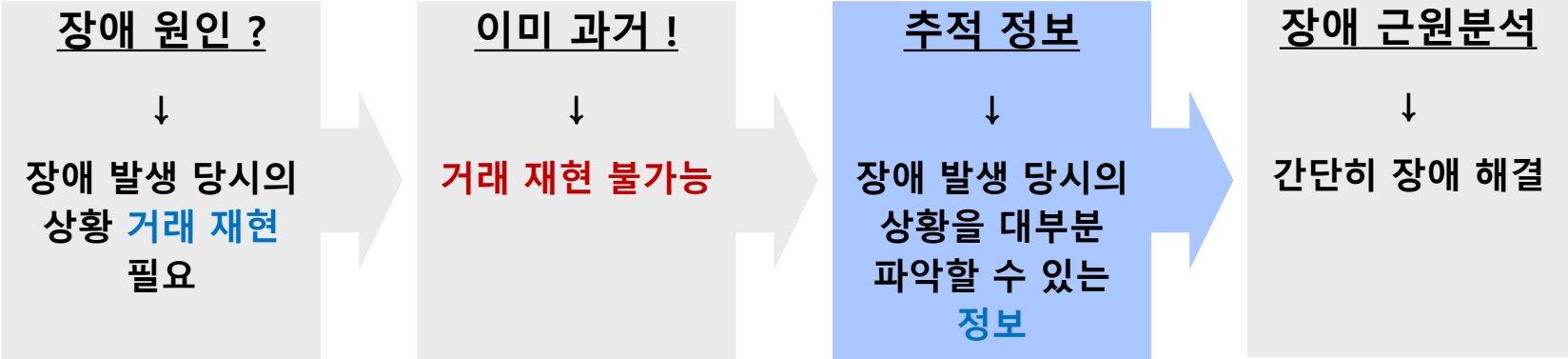
EasyTrace™를 활용한 장애 근원분석

- ◆ 장애 발생 **신호와 함께** 본격적인 해결활동 시작
- ◆ 장애의 원인(Root Cause)분석 **99% 정확도**
- ◆ 복잡한 아키텍처, WAS Clustering, Multi-Node, 수백명의 사용자 등도 문제가 되지 않음
- ◆ 장애 원인분석을 위해서는 많은 정보가 필요하며, EasyTrace™가 이를 제공함 (거래 재현)
 - 누가, 언제, 어디에서, 무엇을, 얼마나, 어떻게
 - 결과는 ...

3 EasyTrace™는 장애를 추적한다 ▶ 장애 해결? 신속한 **근원분석**이 답이다!

[9]

거래 추적/프로파일링



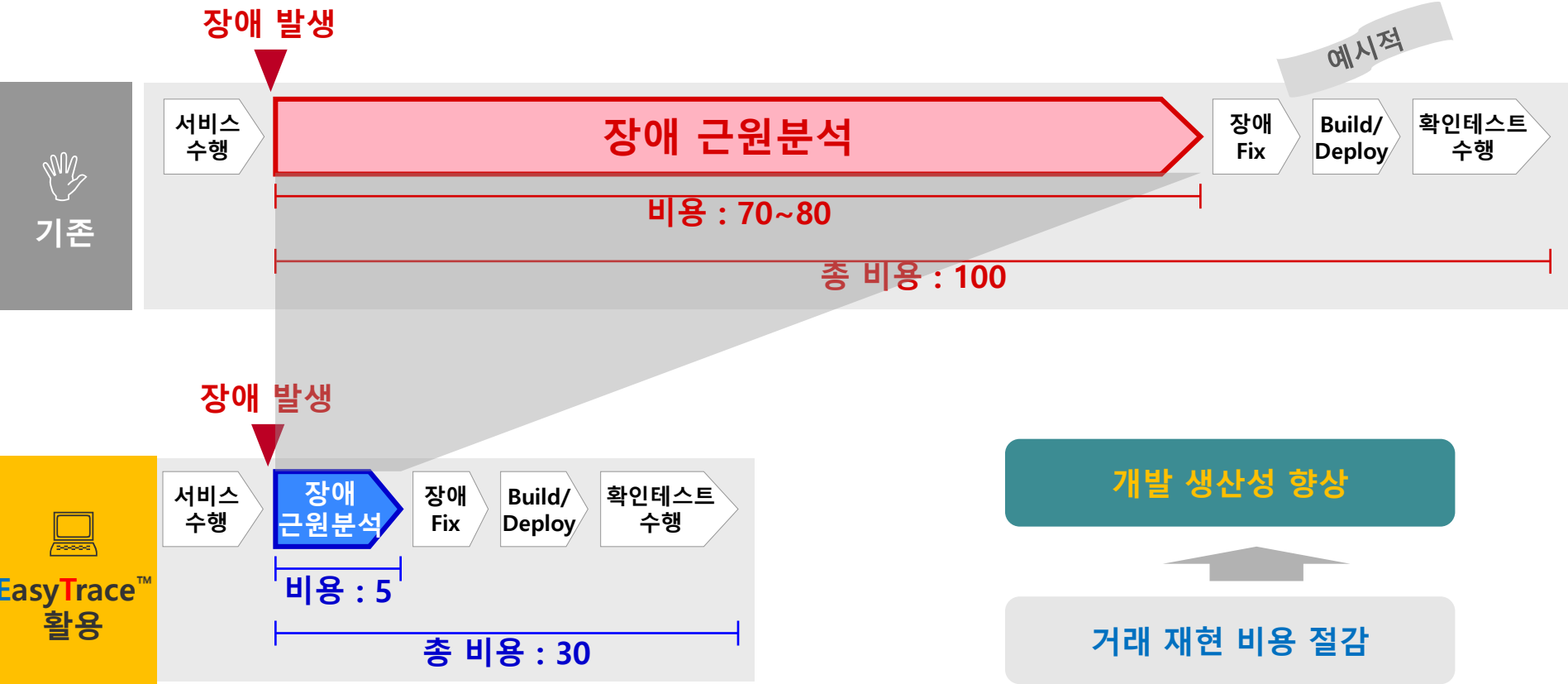
거래별

- | | |
|------------------------|----------------------|
| 1 Request Info | 5 Trace Entries |
| 2 Exception/Stack Info | 6 Query stats |
| 3 Break down tree | 7 Service call stats |
| 4 JVM Thread stats | 8 Thread Profile |

3 EasyTrace™는 장애를 추적한다 ▶ 장애 해결? 신속한 **근원분석**이 답이다!

[10]

장애 해결 프로세스의 최적화

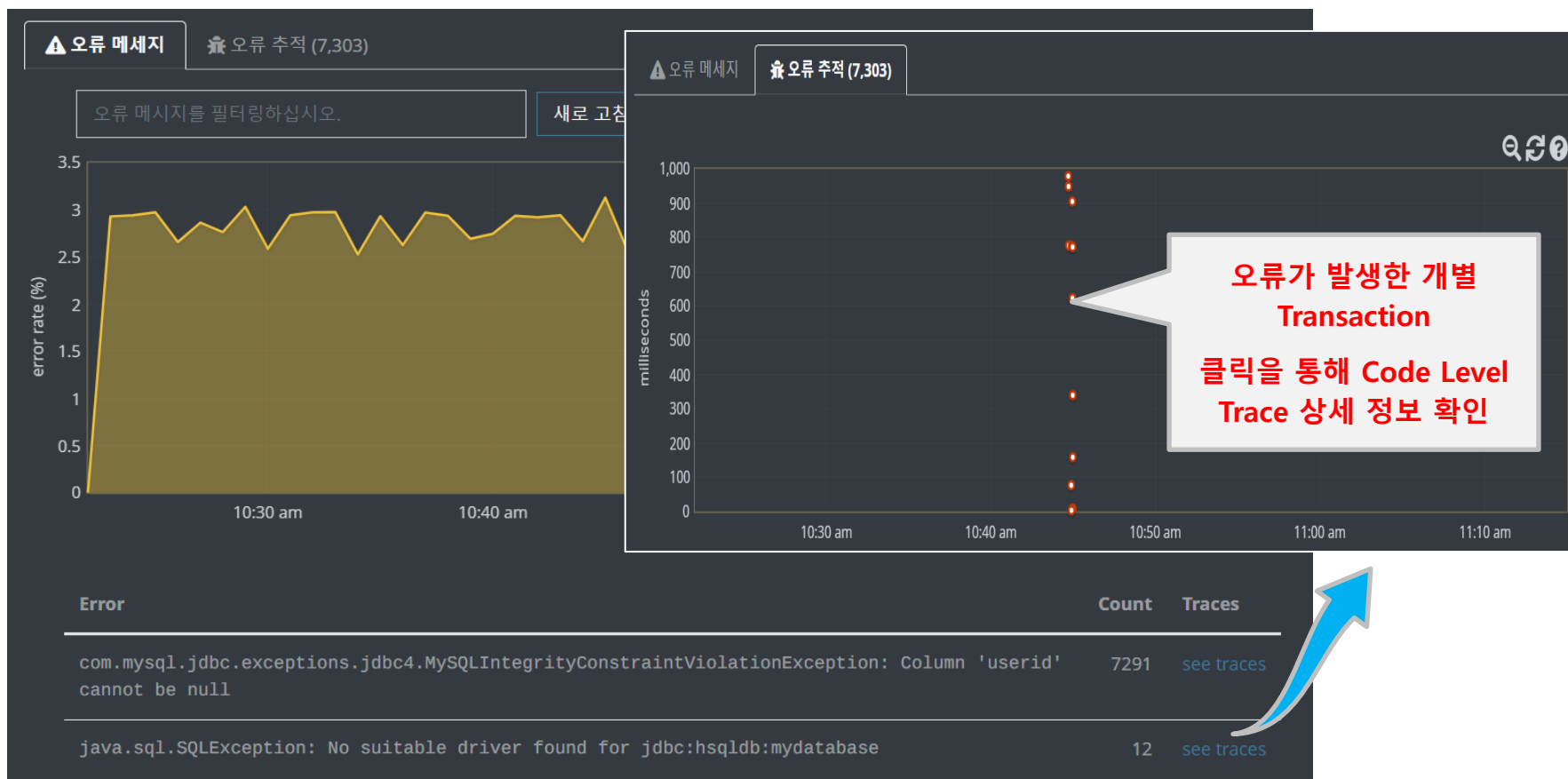


4 EasyTrace™ 기능 ▶ 1) Track All Error

[11]

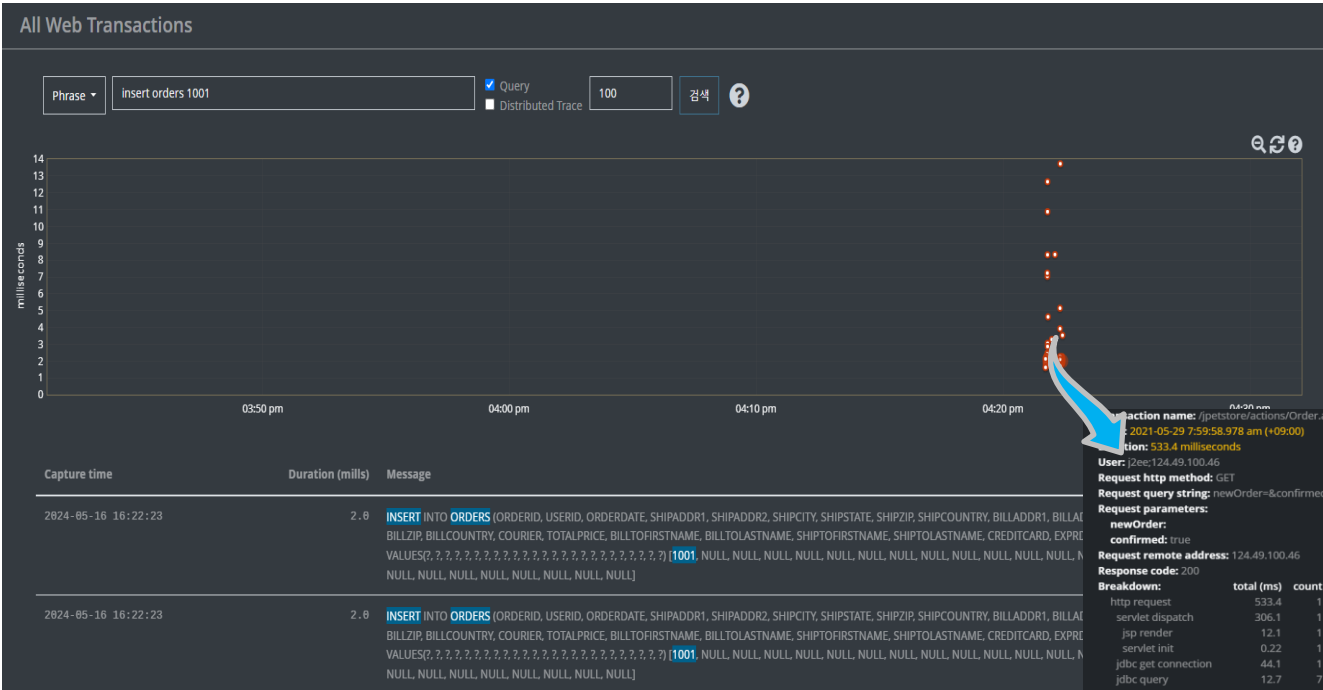
Application 서비스 처리 시 **모든 비즈니스 오류**에 대하여 정보를 수집하여 **Code Level Trace** 수준의 정보 제공

- 기존 APM에서는 일부 오류(Servlet, JDBC, Socket, File)에 대한 정보만 제공(전체 오류 추적하지 않음)



4 EasyTrace™ 기능 ▶ 2) Centralized Log & Search

EasyTrace™에서 중앙으로 수집된 Application Log 및 Query Parameter 에 대하여 강력한 Full-Text Search 기능을 통해 Filter 및 검색이 가능하며 Transaction Trace와 통합되어 빠르게 문제의 원인을 거래 재현으로 파악함



검색된 Application Log 및 Query Parameter 에서 관련된 Transaction Trace(거래 재현) 정보 View

강력한 Full-Text Search 기능을 통해 Application Log 및 Query Parameter 통합 검색

Transaction name: /petstore/actions/Order.action
Transaction: 2021-05-29 7:59:58.978 am (+09:00)
Duration: 533.4 milliseconds

User: j2ee:124.49.100.46
Request http method: GET
Request query string: newOrder=&confirmed=true
Request parameters:
newOrder:
confirmed: true
Request remote address: 124.49.100.46
Response code: 200

Breakdown:	total (ms)	count	list
http request	533.4	1	
servlet dispatch	306.1	1	
jsp render	12.1	1	
servlet init	0.22	1	
jdbc get connection	44.1	1	
jdbc query	12.7	7	
logging	0.81	8	
jdbc commit	5.7	1	
logging	2.6	16	
jdbc connection close	0.51	1	

JVM Thread Stats
CPU time: 302.0 milliseconds
Blocked time: 0.0 milliseconds
Waited time: 0.0 milliseconds
Allocated memory: 33.2 MB

Trace entries (37)

- jdbc get connection
- log debug: o.m.j.persistence.SequenceMapper - Cache Hit Ratio [org.mybatis.jpstore.persistence.SequenceMapper]: 0.0
- log debug: o.m.j.p.SequenceMapper.getSequence - ==> Preparing: SELECT name, nextid FROM SEQUENCE WHERE NAME = ?

4 EasyTrace™ 기능 ▶ 3) Lightweight Code Profiling

[13]

EasyTrace™에서 하나의 Transaction(API Service)에 대하여 내부에서 일어난 사건을 강력한 Code Level 로 보여줌으로 거래 재현을 쉽게 파악할 수 있음

서비스 실행 흐름 재현 코드

Trace entries (79)

- struts action: org.apache.struts.beanaction.BeanAction.execute()
- jdbc query: SELECT SIGNON.USERNAME, ACCOUNT.EMAILFAVCATEGORY = BANNERDATA.FAVCATEGORY ['j2ee', 'j2ee'] => 1 row
- jdbc rollback
- jdbc query: SELECT PRODUCTID, NAME, DESCN a ... ryId FROM PRODUCT WHERE CATEGORY
- jdbc rollback
- auxiliary thread (Async Thread)
- guid : 6a8f44ffe43a2a41363a1e909dc1cdf
- log debug: c.i.jpctest.FactorialCalculator - 2::: guid = 6a8f44ffe43a2a41363a1e909dc1cdf
- jdbc query: SELECT PRODUCTID, NAME, DESCN ... yId FROM PRODUCT WHERE CATEGORY = ? ['DOGS'] => 6 rows
- jdbc rollback
- jdbc query: SELECT CATID AS categoryId, NAME, ... iption FROM CATEGORY WHERE CATID = ? ['DOGS'] => 1 row
- jdbc rollback
- log debug: c.i.jpctest.FactorialCalculator - CatalogBean.viewCategory() result => success
- auxiliary thread (SQL Queries)
- guid : 6a8f44ffe43a2a41363a1e909dc1cdf
- log debug: c.i.jpctest.FactorialCalculator - 1::: guid = 6a8f44ffe43a2a41363a1e909dc1cdf
- jdbc query: SELECT PRODUCTID, NAME, DESCN ... yId FROM PRODUCT WHERE CATEGORY = ? ['DOGS'] => 6 rows
- jdbc rollback
- jdbc query: SELECT CATID AS categoryId, NAME, ... iption FROM CATEGORY WHERE CATID = ? ['DOGS'] => 1 row
- jdbc rollback
- log debug: c.i.jpctest.FactorialCalculator - CatalogBean.viewCategory() result => success
- auxiliary thread (Logging)
- guid : 6a8f44ffe43a2a41363a1e909dc1cdf
- log debug: c.i.jpctest.FactorialCalculator - 1::: guid = 6a8f44ffe43a2a41363a1e909dc1cdf
- jdbc query: SELECT PRODUCTID, NAME, DESCN ... yId FROM PRODUCT WHERE CATEGORY = ? ['DOGS'] => 6 rows
- jdbc rollback
- jdbc query: SELECT CATID AS categoryId, NAME, ... iption FROM CATEGORY WHERE CATID = ? ['DOGS'] => 1 row

5

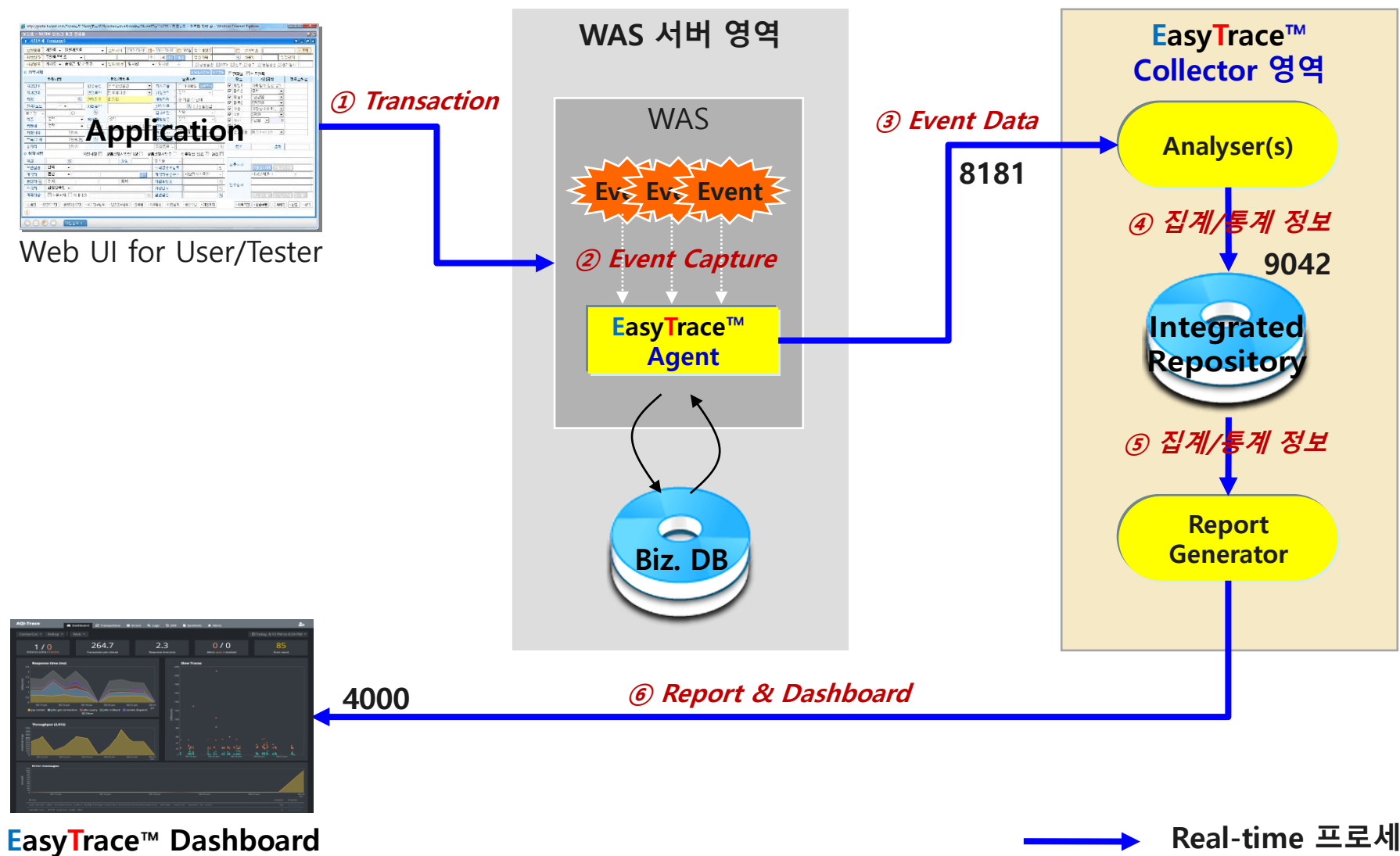
AQI for DevOps 도구 (EasyTrace™)

[14]

AQI 도구인 EasyTrace™는 어플리케이션 상태 및 가용성을 실시간 점검하며 기능/성능 장애 발생 시 빠르게 문제를 인지하고 문제의 근원분석(특히 **마이크로 서비스**)을 통해 해결을 지원하는 도구임

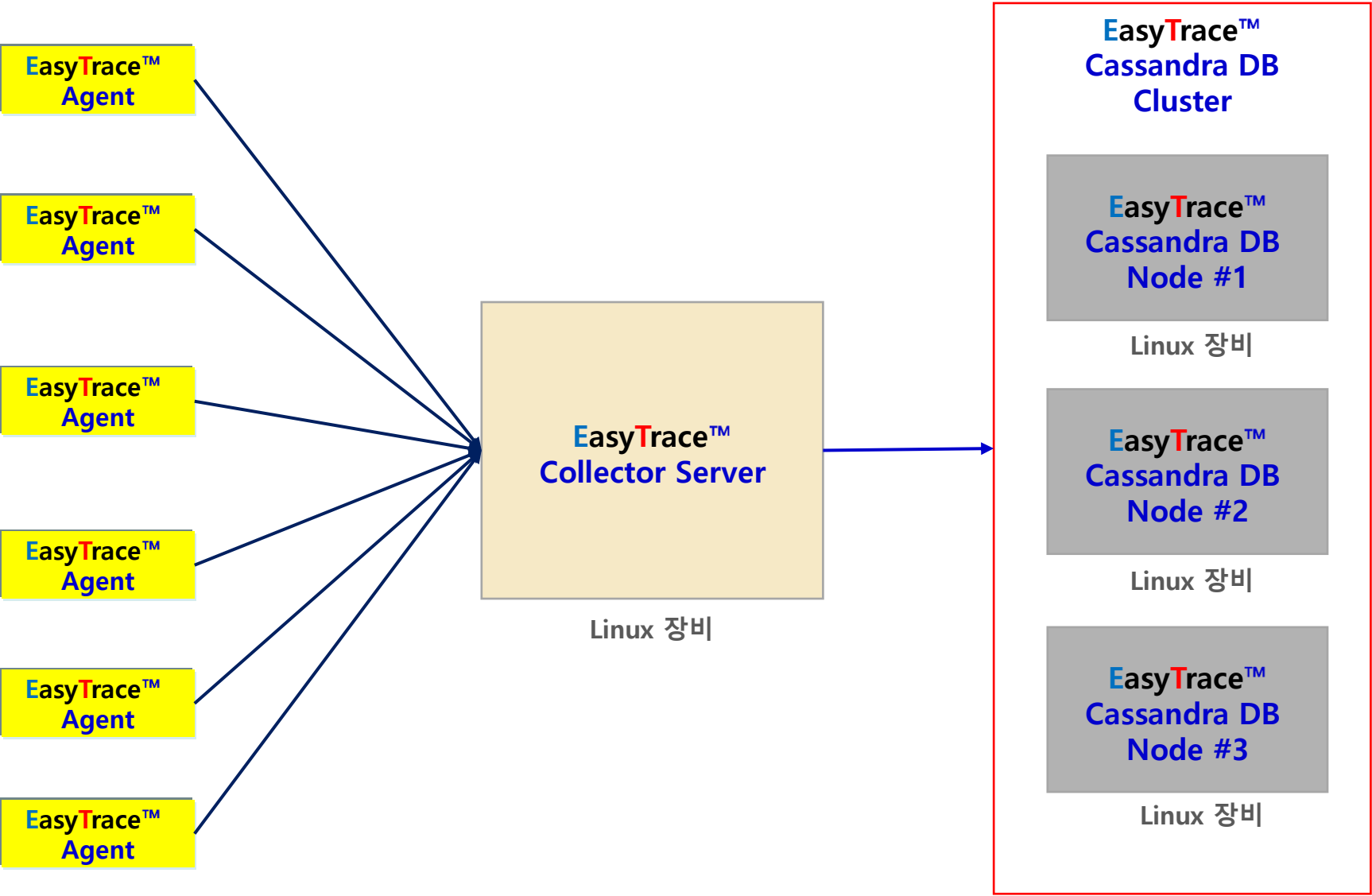
필요성	주요 기능	활용 영역	특장점
- 모든 개발자 로그 및 Query Parameter가 한곳으로 수집되고 강력한 검색기능 및 거래 서비스 통합 연계 - 마이크로 서비스인 경우 성능 문제 또는 어플리케이션 오류가 발생할 경우 원인분석이 굉장히 어려움 - 응답시간이 느린 경우 어떤 컴포넌트가 차지한 시간 비율이 많은지를 한눈에 파악이 어려움 - 어플리케이션의 상태 및 가용성을 한눈에 파악 - High CPU 또는 Memory Leak 을 발생시킨 원인을 쉽게 파악이 어려움 - Synthetic Monitor 를 통해 외부에서 서비스 상태를 파악이 어려움	❖ 모든 Error 추적 - 어플리케이션에서 발생하는 모든 Error 를 수집하고 검색 기능 제공 - 에러 메시지, 에러 비율, 에러 필터 등을 통해 쉽게 접근 ❖ Centralized Logging - 모든 어플리케이션 개발자 로그 및 Query Parameter를 중앙에 통합 수집하며 강력한 검색기능을 통해 쉽게 서비스 거래 파악 ❖ 한눈에 응답시간 지연 파악 - 응답시간 스택 차트를 통해 한눈에 응답시간을 구성하는 컴포넌트 시간 비율을 쉽게 파악 ❖ 마이크로 서비스 분산 추적 - 성능 및 Error 발생 거래에 대하여 쉽게 마이크로 서비스 분산 추적을 제공 ❖ Synthetic Monitor - 외부에서 서비스 요청을 통해 어플리케이션 상태를 파악	• 장애 해결 지원 - EasyTrace™는 모든 시스템 오류를 식별하고, 오류 해결시간을 1/10로 줄일 수 있음 - 장애 해결 프로세스 중 가장 많은 비용 (시간+인력)이 소요되는 장애 원인 분석 시간을 획기적으로 줄임으로써 가능함 - 화면 상에서 오류를 처리하지 않아 식별되지 않는 많은 자바 시스템 오류(WAS 상 식별 오류) 정보 100% 제공 • 개발자 Log/Query Parameter 통합 검색 - EasyTrace™는 개발자가 디버깅을 위해 남긴 Log 및 Query Parameter를 모두 한곳으로 수집 - 강력한 검색기능을 통해 개발자 로그 및 연계된 거래도 함께 통합 • 업무성능 최적화 지원 - EasyTrace™는 모든 거래의 업무성능 측정이 가능하고, 개별거래가 왜 느린지 어디가 병목인지 확실한 정보를 제공함 - SQL 튜닝은 단순히 수행 소요시간 뿐만 아니라, 거래별 SQL 수행 Flow, 개별 거래 내에서 얼마나 많은 SQL이 사용되었는가 등 종합적으로 고려하여 불량 SQL을 추출/튜닝 해야 함	❑ 로그 연계 분산 추적 - 마이크로 서비스 : 성능이나 Error 가 아닌 로직 오류의 경우 마이크로 서비스 분산 시스템에서 디버깅하기는 매우 어려움 - 검색 및 연계 : EasyTrace™는 강력한 개발자 로그 및 Query Parameter 검색기능을 통해 로직 오류를 발생시킨 거래를 쉽게 파악하고 연관된 여러 서비스를 동시에 파악할 수 있음 ❑ High CPU / Memory Leak 감지 - Thread Dump 및 거래단위 Profile 그리고 지속적 Profile 기능을 통해 High CPU 서비스 및 클래스를 쉽게 파악 - 거래단위 할당 Memory 정보 및 Heap Histogram 등을 통해 OOM 또는 Memory Leak 을 쉽게 파악

5 EasyTrace 아키텍처



* EasyTrace™의 Agent 는 JDK 1.6 에서 동작하며 Collector 는 JDK 1.8 에서 동작

5 Large Transaction Site Architecture



5 적용 사례

[17]

금융기관, 통신사 및 공공기관에 적용하였습니다.



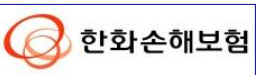
- ◆ 농수산물통합정보시스템 구축
- ◆ 식물검역정보시스템 구축
- ◆ 디지털가축방역시스템 품질 진단
- ◆ 농림수산물부 교육포털 품질 진단



- ◆ 차세대(n.GiANT) 인터넷뱅킹시스템 구축



- ◆ 차세대시스템 구축



- ◆ 한화손보&제일화재 시스템 통합 구축



- ◆ B2B 시스템 구축
- ◆ MVIOS 시스템 구축



- ◆ 인터넷서비스시스템 구축



- ◆ 신 차세대 시스템(AFIS) 구축



- ◆ 차세대 B.I.T OSS 시스템 구축



- ◆ 현대 커머셜 차세대 시스템



- ◆ 현대 캐피탈 차세대 시스템



- ◆ 씨젠 의료재단 차세대 시스템

