**오픈소스분석**

**(중간보고서)**

**2020-01-07**

**배창호**

**백재민**

**전성욱**



1. 분석한 open source
2. Scouter

2-1. application 성능 모니터링

2-2. 주요 모니터링 항목

2-3. 모듈구성

2-4. 특징

2-5. X log

2-6. Performance Counter

# Appdynamics/process-monitoring-extension

* 1. <https://github.com/Appdynamics/process-monitoring-extension>

# 소스 분석중 Appdynamic 이라는 회사에서 쓰는 앱에 사용됨을 식별. 찾다보니 유료인 것들이 있어서 분석을 마침.

1. jProcess
   1. <https://github.com/andrewgp/jProcess>
   2. 모니터링 데이터 다양성 부족 및 기반 언어 ruby
2. sensu
   1. <https://github.com/sensu>
   2. 기반 언어 ruby
3. hyperic
   1. <https://github.com/hyperic>
   2. c와 jsp, java의 혼용으로 보류
4. openNMS
   1. <https://github.com/OpenNMS/opennms>
   2. 모니터링 데이터의 다양성 부족
5. oshi
   1. <https://github.com/oshi/oshi>
   2. Scouter에 비해 성능이 딸려 잠시 보류
6. Scouter
   1. <https://github.com/scouter-project/scouter/releases/>
   2. 데이터의 다양성 충족(thread data 등) 및 server와 client의 연동성에 유리하다고 판단

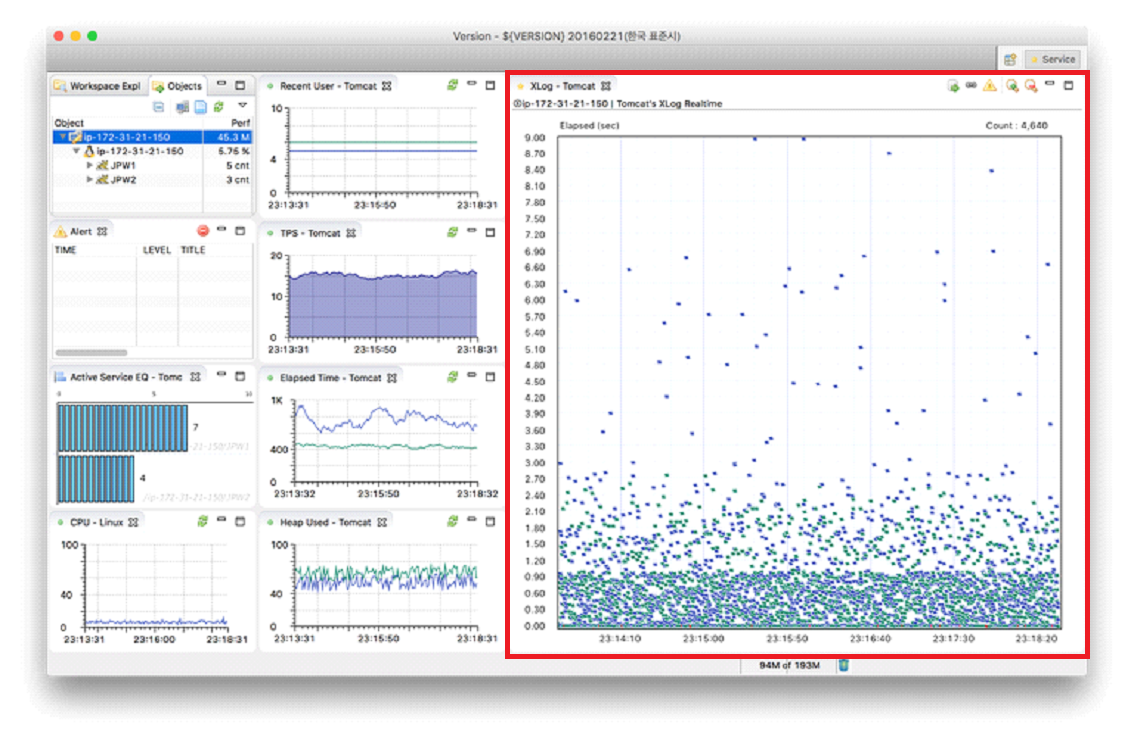
## 

## **S**[**couter**](https://github.com/scouter-project/scouter)

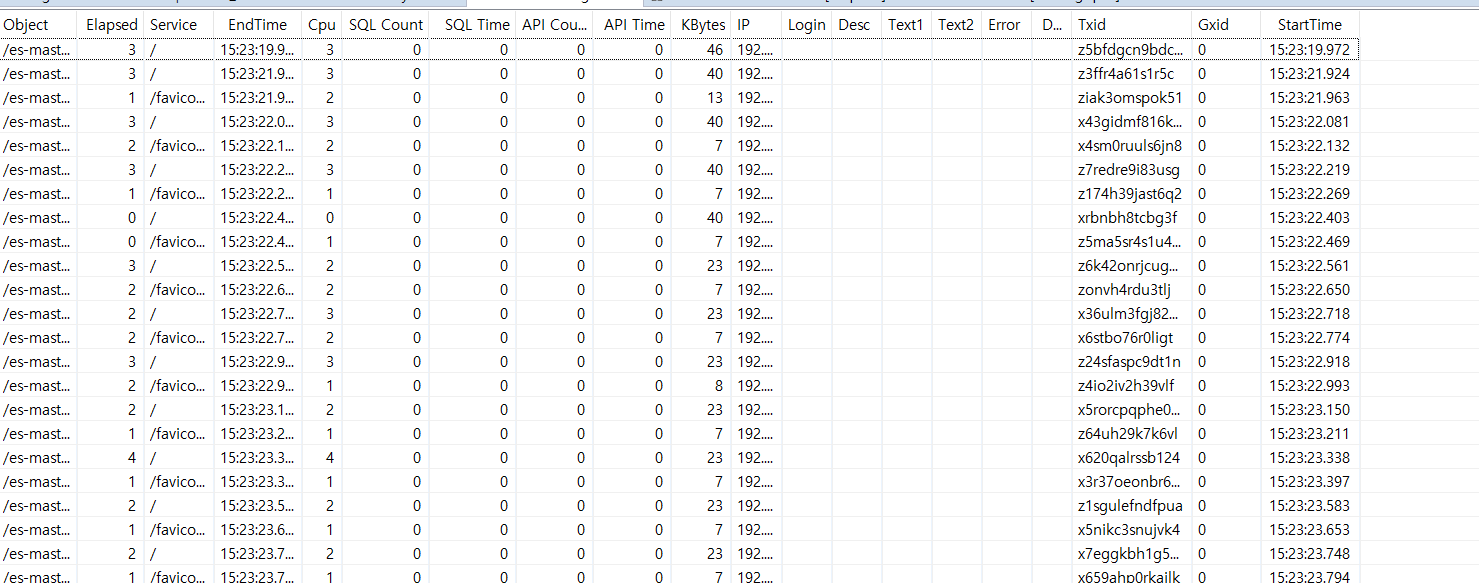
<https://github.com/scouter-project/scouter/releases/>

1. application 성능 모니터링(Scouter는 JVM을 사용하는 application 및 OS 자원에 대한 모니터링 기능을 제공함)
   1. 모니터링 대상 (exclusive agent)
      1. Java Agent : Web application, Standalone java application
      2. Host Agent : Linux, Windows, Unix
   2. 모니터링 대상 (Telegraf support)
      1. Redis, nginX, apache httpd, haproxy, Kafka, MySQL, MongoDB, RabbitMQ, ElasticSearch, Kube, Mesos 등등
   3. 모니터링 대상 (Zipkin-Scouter storage)
      1. zipkin instrumentations를 XLog 차트를 통해 디스플레이.
2. 주요 모니터링 항목
   1. User : Active User, Recent User, Today Visitor
   2. Service : Active Service, TPS, Response Time, Transaction Profile
   3. Resource : Cpu, Memory, Network and Heap usage, Connection pool.
3. 모듈구성
   1. Agent : 성능 데이터를 수집하여 수집 서버로 전송
      1. Tomcat Agent (Java Agent) : JVM 과 Tomcat WAS 성능 수집
      2. Host Agent (OS Agent) : Linux, Windows 및 OSX 성능
   2. Server (Collector) : Agent가 전송한 데이터를 저장하고 Client 요청시 Client에게 적절한 데이터를 전송
   3. Client (Viewer) : 수집된 데이터를 보기 위한 RCP 기반 Client Program
   4. Web API : 성능 카운터, XLog, 프로파일등의 정보를 HTTP 프로토콜을 통해 제공
4. 특징
   1. scouter의 저장소는 파일기반의 저장소이기 때문에 한대의 서버에만 데이터를 저장할수있는 구조가 기본이다 -> zipkin(아파치 분산저장소) 활용가능 저장소(Cassandra,Elasticsearch)
   2. Scouter 를 사용하기 위해선 모니터링 대상 시스템과 같은 내부망에 수집 서버를 설치해야 한다. 이렇게 구성할 경우 SaaS 형 모니터링 솔루션에 비해 보다 상세한 데이터를 수집할 수 있다. SaaS 형은 설치가 필요없기에 쉽게 사용할 수 있는 반면, Scouter 와 같은 설치형은 수집된 데이터를 보다 상세하기 때문에 분석이 효율적이다.
   3. 다양한 Plugin들을 제공할 수 있다.
      1. 수집데이터를 단순히 출력해 주는 sample plugin
         * <https://github.com/scouter-project/scouter-plugin-server-null>
      2. 수집데이터를 단순히 출력해 주는 sample plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-null>
      3. Scouter에서 발생하는 alert를 email로 전송하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-alert-email>
      4. Sconter에서 발생하는 alert를 telegram으로 전송하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-alert-telegram>
      5. Scouter에서 발생하는 alert를 slack으로 전송하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-alert-slack>
      6. Scouter에서 발생하는 alert를 line으로 전송하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-alert-line>
      7. Scouter에서 발생하는 alert를 dingtalk으로 전송하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-alert-dingtalk>
      8. Scouter의 성능 counter 데이터를 시계열 DB인 influxDB로 연동하는 plugin
         * <https://github.com/scouter-contrib/scouter-plugin-server-influxdb>
5. XLog

* 각각의 트랜잭션을 분석하고 시스템의 전반적인 상황을 한눈에 볼 수 있는 산포도
* Scouter의 메인 기능이자 핵심 항목
* Xlog 실행화면(기본모니터링화면)

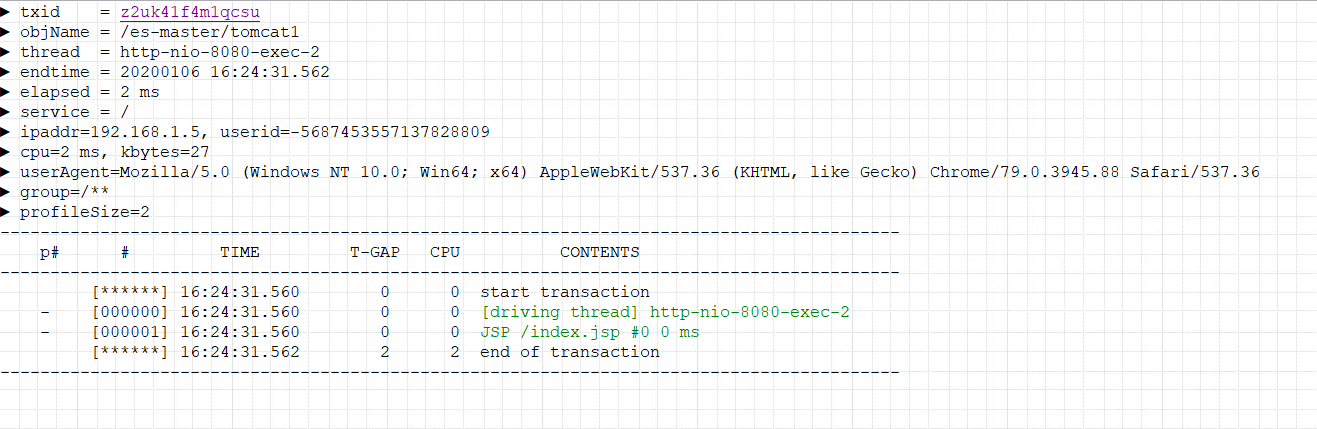


* 특정 시간, 특정 응답시간 에 발생한 서비스에 대한 상세정보

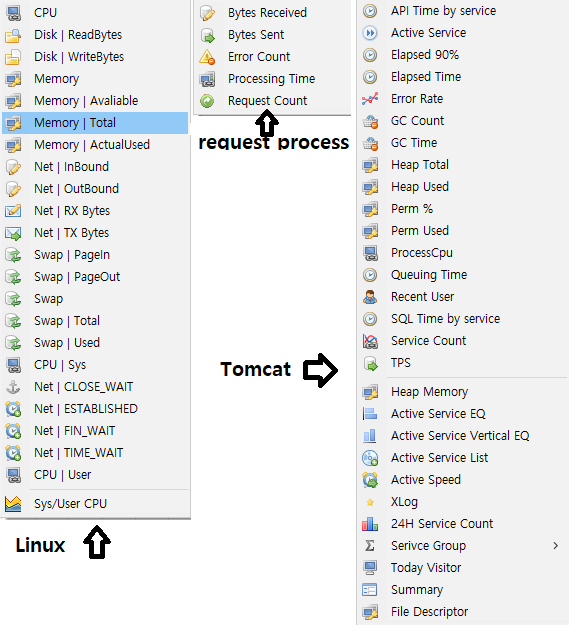


| XLOG 상세정보 표기 항목 | | |
| --- | --- | --- |
| 항목 | 단위 | 설명 |
| Object |  | 서비스가 발생한 서버의 Object Name |
| Elapsed | ms | 서비스 소요시간 |
| Service |  | service URL |
| EndTime | Time | 요청의 종료시간 |
| Cpu Time | ms | 요청을 처리하는 동안 사용한 cpu time |
| SQL Count | cnt | 실행한 SQL 개수 |
| SQL Time | ms | SQL 수행시간 |
| API Count | cnt | 실행한 API 개수 |
| API Time | ms | API 수행시간 |
| KBytes | kbyte | 요청을 처리하는 동안 사용한 heap memory |
| IP | address | 요청자의 IP |
| Login, Text1, Text2, Desc, Dump |  | agent plugin을 통해 사용자가 설정한 정보 |
| Error |  | 서비스에서 발생한 Error |
| Txid | String | 트랜잭션 ID |
| Gxid |  | Global 트랜잭션 ID |
| StartTime | Time | 요청의 시작시간 |
| Thread |  | 처리한 thread명 |

* 세부 서비스 하나에 대한 상세 정보



1. Performance Counter **-** 카테고리별 모니터링 항목



| **Linux** | | |
| --- | --- | --- |
| 항목 | 단위 | 설명 |
| CPU | % | CPU 사용률 |
| Disk | ReadBytes | bytes/sec | 초당 읽은 데이터 |
| Disk | WriteBytes | bytes/sec | 초당 쓴 데이터 |
| Memory | % | 사용 되고 있는 메모리의 백분율(Actual Used/total) |
| Memory | Available | MB | 가용 메모리 |
| Memory | Total | MB | 전체 메모리 |
| Memory | Actual Used | MB | 실제 사용 되고 있는 메모리 |
| Net | InBound | cnt | 송신 횟수 |
| Net | OutBound | cnt | 수신 횟수 |
| Net | RX Bytes | bytes/sec | 초당 송신된 데이터의 크기 |
| Net | TX Bytes | bytes/sec | 초당 수신한 데이터의 크기 |
| Swap | PageIn | page | Page In된 프로그램의 페이지 |
| Swap | PageOut | page | Page Out된 프로그램의 메모리 |
| Swap | % | 사용 되고 있는 Swap 메모리의 백분율(Used/Total) |
| Swap | Total | MB | 전체 Swap Memory |
| Swap | Used | MB | 사용된 Swap Memory |
| CPU | Sys | % | system에서 사용하고 있는 CPU의 사용량 |
| Net | CLOSE\_WAIT | cnt | CLOSE\_WAIT 상태에 있는 프로세스 |
| Net | ESTABLISHED | cnt | ESTABLISHED 상태에 있는 프로세스 |
| Net | FIN\_WAIT | cnt | FIN\_WAIT 상태에 있는 프로세스 |
| NET | TIME\_WAIT | cnt | TIME\_WAIT 상태에 있는 프로세스 |
| CPU | User | % | User가 사용하고 있는 CPU의 사용량 |
| Sys/User CPU | % | User와 System이 사용하고 있는 CPU의 비율 |

| **Request Process** | | |
| --- | --- | --- |
| 항목 | 단위 | 설명 |
| Bytes Received | bytes | 송신 받은 bytes |
| Bytes Sent | bytes | 수신한 bytes |
| Error Count | cnt | 발생한 에러의 수 |
| Processing Time | sec | Process 처리 시간 |
| Request Count | cnt | 요청 횟수 |

| **Tomcat** | | |
| --- | --- | --- |
| 항목 | 단위 | 설명 |
| Active Service | cnt | 현재 처리 중인 서비스의 수 |
| Elapsed 90% | ms | 상위 90% 서비스의 평균 응답 시간 |
| Elapsed Time | ms | 평균 응답 시간 |
| Error Rate | % | 에러가 발생한 서비스 비율 |
| GC Count | cnt | Garbage Collection 횟수 |
| GC Time | ms | GC 수행에 소요된 시간 |
| Heap Total | MB | Heap Memory 크기 |
| Heap Used | MB | 사용 된 Heap Memory |
| Perm % | % | Permanent Generation 사용 비율 |
| Perm Used | MB | Permanent Generation 사용량 |
| Process CPU | % | JVM 프로세스 CPU 사용률 |
| Queuing Time | ms | Java 프로그램의 front단의 reverse proxy나 gateway 서버로부터 java 프로그램으로 요청이 들어오기 까지의 시간 |
| Recent User | cnt | 최근 5분동안 방문한 사용자(unique) 수 |
| SQL Time by Service | ms | 서비스에 소요된 sql 시간 |
| Service count | cnt/min | 분당 처리 중인 서비스의 수 |
| TPS | tps | 초당 트랜잭션 처리 수 |
| Heap Memory | MB | 전체 Heap Memory 크기 |
| Active Service EQ |  | 모니터링 상태에서 실시간 동시 서비스 요청 건수 표시 |
| Active Service EQ Vertical |  | Active Service EQ를 세로로 표시 |
| Active Service List |  | 실시간 동시 서비스 목록 |
| Active Speed |  | WAS 타입별로 부하량 표현  장애 여부 표시(장애 발생 시 빨간색 표시) |
| 24H Service Count | cnt/min | 일별 총 서비스 수, 시간 별로 표시 |
| Service Group |  | WAS의 Context 별로 처리량(Throughput) 또는 응답시간(Elapsed)을 비교할 수 있도록 Stacked Chart 표시 |
| Today visitor |  | 금일 방문자 확인 |
| Summary |  | 특정 시간에 요청된 항목의 세부 사항 표시  (service, SQL, API Call, IP, User-Agent…) |
| File Descriptor |  | 현재 열려 있는 파일 |