애플리케이션 모니터링 서비스 가이드

WhaTap Support

Version 1.1.0

Table of Contents

애플리케이션 모니터링 서비스 가이드

제목 : 애플리케이션 모니터링 서비스 가이드
작성자 : WhaTap Support
이메일 : support@whatap.io
날짜 : 2020-04-16
버전 : 1.1.0

설명 : 본 문서는 WhaTap 애플리케이션 모니터링 서비스 사용법에 대해 설명합니다.

Chapter 1. 시작하기

와탭은 모니터링 서비스입니다. Public 클라우드에서 와탭 모니터링이 서비스되고 있습니다.
사용자는 와탭홈페이지 에 회원 가입후 프로젝트를 생성하고 에이전트를 설치하면 즉시 서비스를 사용할 수 있습니다.

1.1. 지원 환경

와탭 모니터링 UI는 웹브라우저와 모바일 앱을 통해 사용합니다.

지원 브라우저 범위는 다음과 같습니다.

Table 1. 지원 브라우저

브라우저	권장여부	지원버전
Google Chrome	O	58 이상
Mozilla FireFox	O	52 이상
Edge	X	38.14393 이상
Safari	X	Untested

브라우저 호환성과 성능을 이유로 Chrome, Firefox 최신버전 사용을 권장 합니다.
UI는 HTML5 표준기술로 구현되어 Internet Explorer 지원하지 않습니다.

모바일 앱은 앱스토어에서 다운로드 받을 수 있습니다.

- : https://play.google.com/store/apps/details?id=io.whatap.app.WhaTap
- : https://itunes.apple.com/kr/app/whatap/id1252504621?mt=8

1.2. 회원 가입

- http://www.whatap.io 에 접속 후 무료로 시작하기 버튼을 클릭합니다.
- 회사명, 이름, 이메일계정, 비밀번호, 전화번호를 입력하여 회원가입을 진행합니다.

가입정보의 Email, 전화번호로 장애알람을 발송합니다.

[create account] | https://img.whatap.io/media/user_guide_application/intro/create_account.png

1.2.1. 와탭 테넌트 이해

와탭은 멀티테넌시(Multitenancy) 아키텍처 입니다.

와탭 서비스에서 사용자는 상하 구분이 없으며 프로젝트를 만든 순간부터 이 프로젝트의 소유권(최고 권한)을 가집니다. 최고 권한 사용자는 다른 사용자에게 프로젝트를 관리 할 수 있는 권한(Admin)을 부여 하거나 모니터링만 할 수 권한(User)을 부여 할 수 있습니다.

이런 방식을 테넌트 권한 관리 방식이라 합니다.

다수의 프로젝트와 사용자가 복잡한 방식으로 시스템을 사용할 때 한사람이 모든 권한을 관리할 수 없어 업무단위로 프로젝트를 만들고 통제하는 방식이 효과적입니다.

그래서 와탭은 소유권이 있는 프로젝트를 관리하는 멀티테넌시를 제공합니다.

[tenant] | https://img.whatap.io/media/user_guide_application/intro/tenant.png

Figure 1. 와탭의 권한관리

1.2.2. 사용자 종류와 권한

와탭 모니터링 서비스의 사용자 권한 그룹은 **Super Admin(SA) / Admin / User** 로 구분되며 각 그룹별 권한은 다음과 같습니다.

권한그룹	프로젝트 당 계정 수	모니터링	사용자 초대	사용자 권한 변경	사용자 제외	프로젝트 삭제
Super Admin	1	O	O	O	O	O
Admin	제한 없음	O	O	O	O	X
User	제한 없음	O	X	X	X	X

Super Admin (Owner)

프로젝트 당 1개 계정, 사용자 초대, 사용자 권한 변경, 사용자 제외, 프로젝트 삭제

```
프로젝트를 생성한 계정에 부여되며 프로젝트 삭제를 포함한 모든 권한을 가지게 됩니다.
프로젝트에 다른 사용자를 초대하거나 권한을 부여하고 프로젝트에서 사용자를 제외 할 수 있습니다.
 - 사용자 초대는 Admin, User 권한으로 초대 할 수 있습니다.
 - 권한 변경 시 Super Admin 권한으로의 변경은 제한되어 있습니다.
 - Super Admin 이외 사용자를 프로젝트에서 제외 할 수 있습니다.
 - 다른 사용자에게 Super Admin 권한을 위임할 수 있습니다.
```

Admin

계정 수 제한 없음, 사용자 초대, 사용자 권한 변경, 사용자 제외

```
프로젝트에 다른 사용자를 초대하거나 권한을 부여하고 프로젝트에서 사용자를 제외 할 수 있습니다.
 - 사용자를 초대하고 Admin 또는 User 권한을 부여 할 수 있습니다.
 - Admin 에서 Super Admin 권한으로의 변경은 제한되어 있습니다.
 - Super Admin 계정과 본인 계정을 제외한 사용자를 제외 할 수 있습니다.
```

User

계정 수 제한 없음

```
모니터링 전용의 계정으로, 프로젝트 관리 및 사용자 관리 권한이 부여 되지 않습니다.
```

1.3. 프로젝트 생성

모니터링 에이전트 등록을 위해 프로젝트를 생성합니다.

프로젝트 생성 버튼을 선택하면 아래와 같이 프로젝트 생성 창이 나타납니다.

[Screenshot 2020 12 04 Cloud Monitoring] | https://img.whatap.io/media/images/Screenshot_2020-12-04_Cloud_Monitoring.png

Figure 2. 회원 가입후 첫 화면

Java 아이콘을 선택한 뒤, 프로젝트명과 데이터 서버 지역(Region), 그룹을 선택한 뒤 프로젝트를 생성합니다

[create project] | https://img.whatap.io/media/user_guide_application/intro/create_project.png

Figure 3. 프로젝트 생성

이후, 생성된 프로젝트를 선택하여 화면 안내를 확인합니다.

1.3.1. 멀티리전의 이해

와탭은 데이터 수집 리전을 여러곳에 두고 서비스를 합니다. 하나의 수집리전은 다수의 수집서버가 ScaleOut 되는 구성입니다.

모니터링 할 시스템 위치에 따라 가까운 곳의 리전을 선택할 수있습니다.

[region] | https://img.whatap.io/media/user_guide_application/intro/region.png

Figure 4. 멀티 리전

2020년 4월 기준 "Seoul", "Tokyo", "Singapore", "Mumbai", "California", "Frankfurt" 6개 지역에서 서비스하고 있습니다.
서울 지역은 Public 으로 AWS, Azure 리전과 Private인 NHN Toast, NaverCloud 리전에서 서비스 하고 있습니다.

[global] | https://img.whatap.io/media/user_guide_application/intro/global.png

Figure 5. 와탭 리전

1.4. 프로젝트 관리

1.4.1. 프로젝트 그룹

와탭은 프로젝트 단위로 테넌트를 관리합니다. 하나의 사용자는 여러개의 프로젝트를 소유할 수 있고 여러개 프로젝트를 관리하기 위해 그룹 개념을 제공하고 있습니다.

하나의 프로젝트는 두개 이상의 그룹에 속할 수 없습니다.
프로젝트는 그룹에 속하지 않을 수 있습니다.

[Screenshot 2020 12 15 Cloud Monitoring] | https://img.whatap.io/media/images/Screenshot_2020-12-15_Cloud_Monitoring.png

1.5. 유료 전환

체험 기간 종료후 서비스를 계속 이용하고자 한다면 결제 정보를 등록하고 유료 전환 해야 합니다.

결제는 카드 정기결제, 세금계산서 발행 두가지 중 선택할 수 있습니다.

1.5.1. 결제정보 등록

신용카드정보 등록

사전 준비사항

- 결제 등록할 신용카드
- 그룹 소유권자 또는 프로젝트 소유권자 계정으로 로그인

그룹 소유권자 또는 프로젝트 소유권자 계정으로 로그인 후 왼쪽 메뉴 하단의 이용 내역을 선택 합니다.

[이용내역] | https://img.whatap.io/media/images/usage_history.png

Figure 6. 좌측 하단 이용내역

이용 내역 아래 결제 정보 메뉴를 선택합니다.

그림과 같이 신용카드(나이스페이)를 선택후 등록할 카드 정보를 입력합니다.

- 개인 카드인 경우 개인 선택, 주민등록번호 입력
- 기명 법인카드를 포함한 법인 카드인 경우 법인 선택, 사업자등록번호 입력

결제 계정정보를 입력합니다.

- 청구서를 수신할 회사내 대표 담당자 정보를 입력합니다.
- 함께 청구서를 수신할 담당자가 있다면 청구서 참조 수신메일에 추가 합니다.

[카드등록] | https://img.whatap.io/media/images/billing_card.png

Figure 7. 카드정보 등록

전월 사용량 청구서는 익월 5일 이내 발송되며, 결제는 25일에 진행됩니다.

'카드 등록' 버튼을 클릭할 때 입력정보가 불충분하다는 메시지가 나오는 경우 이름, 이메일, 전화번호가 모두 입력되어있는지 확인하세요.

세금계산서 발행 정보 등록

사전 준비사항

- 사업자등록증
- 그룹 소유권자 또는 프로젝트 소유권자 계정으로 로그인

그룹 소유권자 또는 프로젝트 소유권자 계정으로 로그인 후 왼쪽 메뉴 하단의 이용 내역을 선택 합니다.

[이용내역] | https://img.whatap.io/media/images/usage_history.png

Figure 8. 좌측 하단 이용내역

이용 내역 아래 결제 정보 메뉴를 선택합니다.

계산서발행을 선택후 정발행/역발행 중 하나를 선택합니다. 정발행인 경우 회사 정보와 계산서 발행일을 선택 합니다.

결제 계정정보를 입력합니다.

- 청구서를 수신할 회사내 대표 담당자 정보를 입력합니다.
- 함께 청구서를 수신할 담당자가 있다면 청구서 참조 수신메일에 추가 합니다.

세금계산서 발행일을 선택합니다.

- 25일: 사용월 25일
- 말일: 사용월 마지막 날짜
- 청구서발행 당일: 사용월 기준 익월 초 영업일 3일이내

[세금계산서 정보] | https://img.whatap.io/media/images/tax_bills.png

Figure 9. 세금계산서 정보 등록

전월 사용량 청구서는 익월 5일 이내 발송되며, 선택하신 계산서 발행일로 작성됩니다. 결제는 25일 까지 해 주시면 됩니다.

프로모션 코드 등록

사전 준비 사항

- SA 계정으로 로그인
- 결제정보등록

1. 사이트에 SA 계정으로 로그인 하신 후 왼쪽 메뉴 하단 이용 내역을 선택 합니다.

[file NVdK0UzSsH] | https://img.whatap.io/media/images/file-NVdK0UzSsH.png

2. 이용 내역 아래 결제 정보 메뉴를 선택합니다. 결제 정보를 우선 입력한 후 하단 프로모션 정보에 프로모션 코드값을 입력하고 추가 버튼을 클릭합니다.

[Untitled] | https://img.whatap.io/media/images/Untitled.png

1.5.2. 프로젝트 유료전환

결제정보가 등록되었다면 프로젝트를 유료전환 할 수 있습니다.

이용내역 - 프로젝트 유료전환 메뉴로 이동해 상태를 변경합니다.

유상 전환된 프로젝트는 상태가 "유료 전환" 으로 변경되며 유상전환일이 등록됩니다.

프로젝트명 우측 '구매 이력' 버튼을 클릭하면 현재까지 구매 내역을 확인하실 수 있습니다.

[유료전환 메뉴] | https://img.whatap.io/media/images/paid_conversion.png

Figure 10. 유료전환 메뉴

월간 예상비용은 할인/프로모션 내역이 적용되지 않은 예상비용입니다.
프로모션 고객은 "청구서 미리보기" 메뉴 에서 예상비용 확인 가능합니다.

1.5.3. 청구 및 지불현황

납부이력, 미납금 및 청구 내역을 확인하기 위해서는 이용내역 - 청구 및 지불현황 메뉴로 이동합니다.

[청구 및 지불현황] | https://img.whatap.io/media/images/bill_letter.png

Figure 11. 청구 및 지불현황

"청구서 상세" 버튼을 클릭하면 상세 내역을 확인 할 수 있습니다.
와탭 서비스는 시간단위로 과금됩니다. 매월 기준시간은 해당월의 일수 * 24시간으로 계산됩니다.

```
월 기준 시간
 28일 : 28 * 24 = 672시간
 29일 : 29 * 24 = 696시간
 30일 : 30 * 24 = 720시간
 31일 : 31 * 24 = 744시간
```

[청구서 상세] | https://img.whatap.io/media/images/bill_letter_detail.png

Figure 12. 청구서 상세

1.5.4. 청구서 미리보기

청구서 미리보기에서는 당월 현재까지 할인/프로모션이 적용된 사용요금을 확인 할 수 있습니다.

[청구서 미리보기] | https://img.whatap.io/media/images/bill_preview.png

Figure 13. 청구서 미리보기

1.5.5. 사용량

월별 사용량 정보를 시간단위로 상세 제공 합니다.

[사용량] | https://img.whatap.io/media/images/usage.png

Figure 14. 사용량

Chapter 2. 실시간 모니터링

에이전트에서 수집된 정보는 대시보드에서 즉시 확인 할 수 있습니다.
이를 통해 즉각적인 장애 인지와 잠재적인 문제 요소를 빠르게 식별 할 수 있습니다.

2.1. 대시보드를 통한 시스템 전체 현황 파악

수집된 정보는 대시보드로 실시간 확인할 수 있습니다.
CPU, Memory를 제외한 일반적인 차트는 안정적인 데이터는 파란색 계열로 표현되고, 문제로 식별되는 요소들은 붉은색 계열로 표시되므로 사용자가 현황을 쉽게 인지 할 수 있습니다.

[780] | https://img.whatap.io/media/user_guide_application/dashboard/780.png

Figure 15. APM 대쉬보드(메인보드)

대시보드 최상단에서는 프로젝트와 사용자 계정에 대한 필수 메뉴를 배치 했습니다.

[dashboard top] | https://img.whatap.io/media/user_guide_application/dashboard/dashboard_top.png

메뉴 명

왼쪽 끝에는 현재 화면의 메뉴명이 나타납니다.

알람

우측 종 모양 아이콘을 선택하면 최근 발생한 이벤트를 확인 할 수 있습니다.

LiveChat

사용중 문의 사항이 있으면 언제든 말풍선을 클릭해 주세요.

계정 정보

계정 정보를 관리하거나 언어를 변경할 수 있습니다. 소유자인 경우 프로젝트를 유상ㅈ전환 할 수 있습니다.

핀 모양 버튼을 사용해 에이전트를 필터링 할 수 있습니다.

[agent select] | https://img.whatap.io/media/user_guide_application/dashboard/agent_select.png

Figure 16. 에이전트 선택

2.1.1. 에이전트 요약

등록된 에이전트를 요약하고 있습니다. 에이전트의 실행 상태와 수를 나타냅니다.

[agent summary] | https://img.whatap.io/media/user_guide_application/dashboard/agent_summary.png

2.1.2. 액티브 트랜잭션

액티브 트랜잭션은 Arc Equalizer차트입니다. 모니터링 대상 애플리케이션에 실행 중 트랜잭션을 시각화하여 아크이퀄라이저로 보여줍니다.
인스턴스(에이전트) 수만큼 아크가 분할됩니다.

5초마다 현재 서버에서 처리 중인 요청의 수를 표현해서 해당 요청이 각각 어느 정도의 시간동안 처리 중인지 알 수 있습니다. 5초 간격의 시간에 감지된 요청들은 위험 여부를 파악할 수 있도록 다음과 같이 색으로 분류됩니다.

수행 시간에 따른 이퀄라이저 색 변화

- 0초 ~ 3초: 파란색
- 3초 ~ 8초: 주황색
- 8초 이상: 빨간색

파란색이 많이 표현되는 상황은 문제 되지 않지만 이중 일부가 보라색이나 빨간색으로 바뀌어 표현되지 않는지 추이를 지켜봐야 합니다.

장애 상황에는 액티브 트랜잭션 수가 증가하고 빨간색 비율이 높아지는 반면 매우 응답이 빠른 시스템의 경우 처리되는 트랜잭션 수(TPS)는 높아도 액티브 트랜잭션 수는 낮을 수 있습니다.

[act tx] | https://img.whatap.io/media/user_guide_application/dashboard/act_tx.png

Figure 17. 액티브 트랜잭션(아크 이퀄라이저)

```
현재 진행중인 트랜잭션 수를 표시합니다.
- 좌측 상단에는 진행중인 트랜잭션 수가 가장 많은 에이전트 이름이 표시됩니다.
- 우측 상단의 아이콘을 통해 뷰를 전환하거나 상세보기 메뉴로 이동 할 수 있습니다.
- 가운데 숫자는 선택된 에이전트의 액티브 트랜잭션 수 합계입니다.
- 액티브 트랜잭션 둘레에 두 개의 바 는 처리량(TPS)에 따라 3단계 속도로 차트 주변을 회전합니다.
```

2.1.3. 응답 시간 분포도 (히트맵)

히트맵 차트는 종료된 트랜잭션 응답시간을 분포도로 표현합니다. 가로축은 트랜잭션 종료 시간을 나타내며 세로축은 수행 시간을 나타내므로 수행 시간이 긴 트랜잭션은 분포도 상단에 위치하게 됩니다.

히트맵 내 영역을 드래그 하면 세부 트랜잭션 정보를 확인 할 수 있습니다.

오류가 발생하지 않았고 수행시간 500ms이하 트랜잭션의 세부 정보는 url당 5분에 1건만 수집됩니다.
TPS나 평균응답시간 같은 통계정보에는 영향을 미치지 않으며, 에이전트 설정을 통해 정책 조정 가능 합니다.

히트맵 점 색상의 진하기는 밀도를 표현 합니다.

- 정상TR : 하늘색 → 파란색 → 남색
- 에러TR : 노란색 → 주황색 → 빨강색

[hitmap] | https://img.whatap.io/media/user_guide_application/dashboard/hitmap.png

Figure 18. 히트맵

```
히트맵 차트는 편의 기능들을 제공합니다.
- 상단의 숫자는 차트 내 트랜잭션 건 수 / 에러 건 수 를 의미합니다.
- 수행 시간을 나타내는 Y축은 자동으로 조정할 수 있습니다.
- 에러 트랜잭션만 필터 할 수 있습닏다.
- 차트는 상하 5초 ~ 80초 까지 확대/축소 할 수 있습니다.
```

2.1.4. 액티브 스테이터스

액티브 트랜잭션을 수행 구간별로 분류하여 이퀄라이저로 보여줍니다. 프로젝트내의 모든 에이전트는 5초마다 액티브트랜잭션 수를 수집함과 동시에 진행 상태를 수집합니다.

[Screenshot 2020 12 15 Panel Site 1807 Application Monitoring] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_Panel_Site_1807_-_Application_Monitoring.png

Figure 19. 액티브 스테이터스

액티브 스테이터스 상태는 5가지로 구분됩니다.

METHOD

아래 4가지 상태가 아닌 경우

SQL

Sql을 수행중인 상태

HTTPC

Http 외부 호출 중인 상태

DBC

Db Pool 에서 getConnection을 하고 있는 상태

SOCKET

Socket의 connect()를 호출하고 있는 상태

이퀄라이저 색깔의 의미는 중요도를 나타내고 있습니다.
DBC와 SOCKET 은 한두개가 지속적으로 나타난 것까지 문제로 인식되어야합니다.
반면 SQL과 HTTPC는 갯수가 상대적으로 많은 경우 분석이 필요하고 METHOD는 일반적인 상황이므로 파란색입니다.

2.1.5. 평균 응답시간

모니터링 대상 서버가 평균 응답시간이 얼마인지를 실시간 모니터링 할수있습니다.

[response time1] | https://img.whatap.io/media/images/response_time1.png

Figure 20. 응답시간 전체

개별 응답시간을 모니터링 하고자 할때는 차트 상단의 전환 버튼을 클릭합니다. 다만 프로젝트에 애플리케이션 수가 너무 많으면(수십개 이상) 효과적이지 않습니다.

[response time] | https://img.whatap.io/media/images/response_time.png

Figure 21. 응답시간 개별

차트 상단의 버튼을 클릭하면 서버당 데이터를 보여주는 차트를 확인할 수 있습니다. 해당 차트는 평균,최대응답시간, 이름을 기준으로 정렬하여 볼 수 있습니다.

[10.52.14] | https://img.whatap.io/media/images/10.52.14.png

Figure 22. 응답시간 상세보기

2.1.6. TPS

TPS(Tansaction Per Second)는 초당 처리된 트랜잭션 건수를 의미하며 서비스 성능지표 중 기준이 됩니다.

와탭은 프로젝트 전체 TPS를 실시간으로 보여줍니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring.png

Figure 23. TPS 전체

개별 TPS를 실시간 모니터링 해야 할 경우에는 차트 상단의 전환 버튼을 클릭합니다. 다만 프로젝트에 애플리케이션 수가 너무 많으면(수십개 이상) 효과적이지 않습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring1] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring1.png

Figure 24. TPS 개별

차트 상단의 상세보기 버튼을 클릭함으로 상세한 최근 TPS현황을 조회할 수있습니다.

[tps] | https://img.whatap.io/media/images/tps.png

Figure 25. TPS 상세보기

2.1.7. 시스템 CPU

시스템 CPU 사용량 입니다. 실시간 정보이기에 CPU 사용량 변화 추이를 즉시 파악할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring2] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring2.png

Figure 26. 시스템 CPU

2.1.8. Heap(Process) Memory

각 서버당 사용 가능한 최대/현재 메모리를 보여줌으로써 위험수치에 있는 서버를 알 수 있으며 시간에 따른 메모리 사용량 변화를 실시간으로 볼 수 있습니다.

Java 와 Node.js 는 힙 메모리 총량과 사용량을 보여줍니다.
PHP, Python 과 .Net 은 프로세스 메모리 사용량을 보여줍니다.

메모리 라인 차트는 보통 계속해서 물결 치며 이는 애플리케이션 서버가 요청을 처리 하기 위해 메모리를 사용할 때 증가하며 GC를 통해서 메모리를 정리할 경우에 감소합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring3] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring3.png

Figure 27. 힙메모리

2.1.9. 사용자

일반적으로 접속 사용자라 함은 현재 네트웍으로 접속하여 연결된 사용자를 의미합니다.
하지만 웹 시스템은 비연결 네트웍을 사용하기 때문에 접속 되어있다는 개념 보다는 최근 요청을 보낸적이 있는 사용자가 측정대상이 될 수있습니다.

따라서, 최근이란 얼마의 시간을 의미하는지가 중요한데 와탭에서는 5분을 기준으로 하고 이를 **실시간 사용자(Realtime User)** 라고 부릅니다.

와탭의 실시간 사용자 수는 "최근 5분동안 트랜잭션을 유발 한 적이 있는 사용자의 수" 를 의미합니다.
실시간(동시) 사용자는 같은 시간대에 시스템을 사용하고 있는 사용자를 의미하게 되므로 산정 방식 또는 측정 방식에 따라 다양합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring4] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring4.png

Figure 28. 실시간 사용자

[user] | https://img.whatap.io/media/images/user.png

Figure 29. 실시간 사용자 상세 보기

사용자 수는 단순히 합산되는 것이 아니라 HyperLogLog [1] 로 산출 됩니다.

2.2. 트랜잭션 상세 분석

진행중인 트랜잭션 즉 액티브 트랜잭션이나 종료된 트랜잭션은 보다 상세하게 어느 구간에서 지연되고 있는지를 분석 할 수있습니다.

2.2.1. 액티브 트랜잭션 상세 보기

액티브 트랜잭션의 옆에 화살표를 클릭하면 세부 상태를 확인할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring6] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring6.png

좌측에는 애플리케이션 서버의 리스트와 진행중인 트랜잭션의 총 개수를 표현하며 해당 서버를 선택하게 되면 우측에 세부정보가 표현됩니다.

화면에 표시되는 요약 정보의 표시 항목은 다음과 같습니다.

시작 시간

트랜잭션의 시작 시간

Transaction URL

트랜잭션 이름

에이전트

에이전트 이름

경과시간

트랜잭션 시작후 지금까지 경과 시간 (ms)

SQL / HTTPC

SQL 또는 HTTPCall을 수행하면서 사용한 시간

IP

해당 트랜잭션을 수행한 IP

사용자 토큰

해당 트랜잭션을 수행한 사용자의 토큰

CPU

CPU 사용시간 (ms)

SQL 건수

SQL 수행 건수

HTTC 건수

HTTPCall 건수 s

액티브 트랜잭션 리스트에서 해당 트랜잭션을 클릭하면 우측에 세부사항이 표시됩니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring7] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring7.png

표시되는 세부 정보의 표시 항목은 다음과 같습니다.

Application

애플리케이션 서버 명(에이전트 단위의 식별자)

Start Time

트랜잭션의 시작 시간

Elapsed

경과 시간

Memory Alloc

메모리 할당량 (byte)

CPU Time

CPU 사용시간 (ms)

SqlCount

SQL 수행 건수

SqlTime

SQL 수행 시간 (ms)

HttpcCount

외부 호출 건수

HttpcTime

외부 호출 시간 (ms)

ClientIp

클라이언트 IP

화면의 각 버튼별 기능은 다음과 같습니다.

[새로고침]

액티브 트랜잭션 목록을 갱신합니다.

[쓰레드 중지]

진행중인 쓰레드를 중지 할 수있습니다. Thread.stop() 메소드를 호출 합니다.

쓰레드 스톱을 호출하려면 암호가 필요합니다.
암호는 에이전트가 설치된 디렉토리 paramkey.txt 파일 내에 있습니다.

2.3. 히트맵 상세 분석

실시간 히트맵을 드래그 하거나 [→]를 클릭하면 아래와 같은 상세 분석 화면이 열립니다.

차트에서 에러 트랜잭션이 포함된 점은 노란색이나 붉은 색으로 나타나고 정상 트랜잭션만 포함된 경우에는 파란색으로 나타납니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring11] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring11.png

Figure 30. 히트맵 상세 분석

분포도 화면에서 선택된 영역이 표시되고 그 영역에 포함되는 트랜잭션 수와 에러 수가 오른쪽 화면에 나타납니다. 리스트 중에서 애플리케이션을 선택하면 그 애플리케이션의 트랜잭션 목록이 하단에 나타납니다. 트랜잭션 하나를 클릭하면 프로파일 상세 내용을 분석 할 수있습니다.[다음장에서 설명]

차트 아래 부분의 [에러]을 클릭하면 아래 처럼 에러가 발생한 트랜잭션만을 조회 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring12] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring12.png

Figure 31. 히트맵 에러

Transaction List는 세부 정보를 가지고 있습니다. List 에서 원하는 Transaction을 선택하게 되면 해당 트랜잭션의 프로파일을 보여줍니다.

2.3.1. 사전 정의된 에러

아래 경우는 트랜잭션 수행과정에 에러가 없는 경우라도 히트맵에서 에러로 표기됩니다.

Table 2. WhaTap Error

에러	발생기준	기준시간	비고
SLOW_HTTPC	HTTPC 실행시간	10,000 ms (기본값)	profile_error_httpc_time_max
TOO_MANY_RECORDS	SQL Fetch	10,000 건 (기본값)	profile_error_jdbc_fetch_max
SLOW_SQL	SQL 실행시간	30,000 ms (기본값)	profile_error_sql_time_max

예를 들어, SLOW_HTTPC 의 기준을 기본값 10초에서 30초로 변경하고자 하는 경우 에이전트 옵션에 다음과 같이 추가합니다.

```
profile_error_httpc_time_max=30000
```

에러로 표기하지 않을 경우 0으로 설정 합니다.

```
profile_error_httpc_time_max=0
```

상세 내역은 에이전트 옵션을 참고 할 수 있습니다.

2.4. 트랜잭션 프로파일 상세 분석

와탭은 트랜잭션의 성능을 분석하기 위해 이름과 클라이언트 정보등의 속성들과 트랜잭션의 처리 성능 그리고 각 구간별 상세 수행이력을 수집하고 보여줍니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring13] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring13.png

Figure 32. 프로파일 상세 분석

이 중 상단에 위치한 트랜잭션 속성과 성능 정보에 대한 설명은 다음과 같습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring14] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring14.png

트랜잭션 속성 및 성능

프로젝트 코드	와탭 모니터링 프로젝트 코드
Oid	애플리케이션(에이전트) 오브젝트 아이디
Oname	애플리케이션 오브젝트 이름
Type	애플리케이션 타입 - JAva, Node.Js, PHP, Python등
Tid	트랜잭션 아이디
UrlHash	URL의 hash 값
UserToken	사용자 토큰 - 이값은 와탭이 부여한 고유한 사용자 아이디
Method	HTTP 메소드 - GET,POST,PUT,HEAD등
StartTime	트랜잭션 시작시간
EndTime	트랜잭션 종료 시간
Elapsed	트랜잭션 수행 시간
CpuTime	CPU 점유시간
MemAlloc	트랜잭션이 사용한 메모리 량
Ipaddr	클라이언트 IP
Country	클라이언트 국가 정보
City	클라이언트 도시 정보
SqlCount	SQL 수행 건수
SqlTime	SQL수행 시간
FetchCount	SQL에서 데이터를 조회한 레코드 건수
FetchTime	레코드를 조회하는데 걸린 시간(중첩이 발생하거나, 타 로직이 포함될 수도 있음)
HttpcCount	외부 Http Call 건수
HttpcTime	외부 Http Call시간

ClientOs	클라이언트 OS
ClientOsPro	클라이언트 OS 회사
ClientType	클라이언트 타입 (예 브라우저)
ClientName	클라이언트 브라우저 이름
UserAgent	웹 클라이언트 정보, 이것을 파싱하여 ClientOs, ClientOsPro등의 정보를 추출함
MTID	멀티서버 트랜잭션 아이디
Depth	멀티서버 트랜잭션이 몇번째 티어인지 표시
Caller TID	나를 호출한 트랜잭션의 Tid

1. [Show Stats]를 클릭하면 해당 URL의 통계 데이터를 조회 할 수있습니다
2. [MTID]를 클릭하면 연관트랜잭션 [2]을 추가 조회 분석 할 수 있습니다.
3. 에이전트 애플리케이션 타입이나 설정에 따라 표시되지 않는 항목이 있을 수 있습니다. 에이전트 메뉴얼을 참고 바랍니다.

트랜잭션 다이어그은 트랜잭션 수행 상세 이력입니다. 수행 상세 이력을 보여주기 위해 나타나는 각 칼럼 정보는 다음과 같습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring15] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring15.png

트랜잭션 프로파일 정보

번호	스텝의 발생 순서
누적시간	해당 스텝까지의 누적 수행 시간
스텝 시간	해당 스텝에서의 응답시간
CPU Time	해당 스텝까지 누적 CPU 사용 시간
메모리	해당 스텝까지 누적 메모리 사용량
단계	해당 스텝의 세부 수행 내역

이 중 단계 는 다음과 같이 구분됩니다.

단계

HTTP-HEADERS	수집된 HTTP Header 정보를 보여줍니다.
HTTP-PARAMETERS	수집된 HTTP Parameter 정보를 보여줍니다.
DB연결	DB 연결 URL과 연결 시간을 보여줍니다.
SQL	JDBC SQL에 대한 SQL문, 응답시간 에러를 보여줍니다.
HTTP Call	외부 http서비스 URL과 응답시간, 에러를 보여줍니다.
Message	메세지 스텝을 비정형적 모든 구간에 대한 이력을 표시합니다. file오픈, Http Header 출력, 파라미터 출력 혹은 사용자도 임의의 위치를 지정하는데 사용 할 수있습니다.
SOCKET	Socket 오픈을 표현합니다. 응답시간은 추적하지 않습니다.
METHOD	에이전트에 추적이 설정된 메소드 이름과 응답시간을 보여줍니다.
ACTIVE STATCK	액티브 스택이 수집되었음 나타냅니다. 클릭해서 엑티브 스택을 확인 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring16] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring16.png

Figure 33. 액티브스택 스텝 상세보기

트랜잭션 프로파일은 트리뷰 형태로도 볼수있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring17] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring17.png

Figure 34. 프로파일 TREE View 분석

메소드 프로파일 설정을 많이 한 경우 리스트뷰에서 상태 분석이 잘 안될수 있습니다. 반대로 메소드 프로파일 설정을 거의 하지 않은 상태에서는 리스트 뷰가 분석하기에 유리합니다.

2.4.1. 트랜잭션 파라미터 조회

SQL 변수, HTTP 쿼리를 조회하기 위해서는 에이전트 설정을 추가합니다.

whatap.conf

```
profile_sql_param_enabled=true #SQL 파라미터 조회 옵션 ①
profile_http_parameter_enabled=true #HTTP 파라미터 조회 옵션 ②
```

① 옵션이 적용되면 SQL 파라메타를 암호화하여 수집합니다.

② 옵션이 적용되면 HTTP 쿼리 파라메타를 암호화하여 수집합니다.

설정 적용 후 프로파일 스텝 내 SQL문을 클릭하면 파라미터를 조회 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring18] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring18.png

파라미터 조회에 필요한 키는 에이전트 설치된 서버 $WHATAP_HOME/paramkey.txt 내에 6자리 문자열입니다. 필요시 다른 문자열로 변경 가능합니다.

paramkey.txt

```
whatap@vmwas01:/apps/whatap$ cat paramkey.txt
A1B2C3
```

paramkey.txt 내 키는 SQL 변수 조회, HTTP 쿼리 조회, Thread 중지에 필요합니다.

2.5. 트랜잭션 연계 추적

2.5.1. 멀티서버 트랜잭션

서비스(업무) 별로 분리 되어 있는 애플리케이션 서비스들간의 호출을 추적할 수 있습니다. 와탭에 등록되어 있는 프로젝트라면 프로젝트에 등록된 애플리케이션 서비스들간의 호출에 대한 추적을 말합니다

멀티서버 트랜잭션 추적을 위해서는 에이전트 설정파일에 다음 옵션을 설정합니다.

whatap.conf

```
mtrace_enabled=true ①
```

① 기본값 10% 비율로 멀티서버 트랜잭션 추적을 합니다.

멀티서버 트랜잭션 추적이 적용된 경우 히트맵 트랜잭션을 드래그 하면 **M** 아이콘을 볼 수 있습니다.

[mtrace M] | https://img.whatap.io/media/user_guide_application/dashboard/mtrace_M.png

멀티서버 트랜잭션이 추적되는 경우라면 프로파일 정보 상단에 **MTID** 값이 나타납니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring10] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring10.png

- 트랜잭션에서 외부 호출을 하는 경우에도 동일한 MTID가 생성됩니다.
- 업무별로 프로젝트가 분리 되어 있더라도 처음 발급한 MTID를 통해 애플리케이션간의 모든 트랜잭션을 확인 할 수 있습니다.

MTID를 클릭하면 동일한 MTID를 갖는 트랜잭션 서비스들의 개별 수행시간을 확인 할 수 있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring11] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring11.png

테이블 뷰에서는 각 티어별 트랜잭션 프로파일 정보를 확인 할 수 있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring12] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring12.png

같은 프로젝트에 속한 애플리케이션 서비스들간의 호출관계는 프로파일 내에서 데이터를 바로 확인 할 수 있습니다.

- Caller: 서비스를 호출한 트랜잭션 (호출자)
- Callee: 서비스를 호출 당한 트랜잭션 (피호출자)

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring13] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring13.png

[1] 매우 적은 메모리로 집합의 원소 개수를 추정하는 확률적 자료구조
[2] 멀티 서버 트랜잭션 연계 추적은 고급기능 쳅터에서 별도 설명되어있습니다.

Chapter 3. 토폴로지

모니터링 대상 서버로부터 실시간으로 수집되고 있는 정보를 통해 애플리케이션 서버 간의 연관관계를 용이하게 파악할 수 있습니다.

3.1. 토폴로지란

컴퓨터 네트워크 요소들을 물리적으로 연결해 놓은 방식을 의미합니다. (위키백과) 노드와 링크로 구성되며 어떠한 노드도 하나 이상의 링크를 가지는 연결 구조로 정의됩니다.

[topology definition] | https://img.whatap.io/media/user_guide_application/topology/topology_definition.png

Figure 35. 토폴로지 정의

토폴로지 정의가 설명하는 토폴로지의 범주에는 링크 정보를 가지지 않은 노드를 포함하지 않고 있으나 모니터링 관점에서의 토폴로지에서는 고립된 노드의 정보도 필요로하게 됩니다.

3.2. 토폴로지 분류

와탭이 제공하는 토폴로지의 종류는 총 5가지 형태로 분류됩니다.

- 애플리케이션 토폴로지 – 프로젝트 범위에서 포함된 모든 애플리케이션의 관계 정보를 표현합니다.
- 그룹 토폴로지 – 프로젝트 범위에서 애플리케이션을 그룹 단위의 관계 정보를 표현합니다.
- 통합 토폴로지 – 다중 프로젝트에 걸쳐 애플리케이션 그룹 단위의 관계 정보를 표현합니다.
- 인스턴스 토폴로지 – 단일 애플리케이션의 관계 정보를 표현합니다.
- Netstat 토폴로지 – 애플리케이션의 리스닝 정보, 아웃바운드 호출 대상 정보를 표현합니다.

3.3. 토폴로지 표현 정보

토폴로지에서 표현하는 정보는 노드와 링크 및 부가정보로 구분되며, 각각이 포함하는 정보는 다음과 같습니다.

- 노드 – 프로젝트, 애플리케이션 그룹, 애플리케이션, 데이터베이스, HTTP 호출대상 외부 호스트, 외부 모듈(애플리케이션 호출자)을 표현합니다.
  - 노드로의 호출 정보가 존재하는 경우, 노드 외곽에 에러 비중을 표현합니다.
- 링크 – 애플리케이션 혹은 애플리케이션 그룹 간의 호출 정보, 데이터베이스 호출 정보, 외부 호스트 호출 정보, 외부 모듈로부터의 호출 정보를 표현합니다.
  - 평균 응답시간 – 5초간의 노드 간 트랜잭션 또는 호출 내역의 응답시간 합계를 총 건수로 나눈 정보
  - 건수 – 5초간의 노드 간 트랜잭션 또는 호출 내역의 총 건수 (에러 건수를 포함)
  - 에러 – 5초간의 노드 간 트랜잭션 또는 호출 내역 중 총 에러 건수
  - 액티브TX – 노드 간 트랜잭션 또는 호출 중 현재 진행 상태의 건수
  - 패치 시간 – 5초간의 DB 호출 내역의 레코드 취득 시간 총합
  - 패치 건수 – 5초간의 DB 호출 내역의 레코드 취득 건수 총합

[topology error information] | https://img.whatap.io/media/user_guide_application/topology/topology_error_info.png

Figure 36. 토폴로지 노드 에러 정보

3.4. 토폴로지 제어

토폴로지 화면의 기본 동작은 사용자 편의성을 위하여 다음 기능을 포함합니다.

- 자동 갱신 – 디폴트 30초 주기의 자동 갱신을 통해 토폴로지의 변경 사항이 자동으로 업데이트 됩니다.
  - 사용자가 원하는 경우 갱신 주기를 5초 ~ 5분까지 지정할 수 있으며, 새로고침 버튼을 통해 최신 정보로 토폴로지를 업데이트 할 수 있습니다.

Netstat 토폴로지의 경우, 에이전트 부하로 인해 자동 갱신을 지원하지 않습니다.

- 줌 인/줌아웃 – 마우스 스크롤을 통해 토폴로지를 줌인 또는 줌아웃 하여 노드 수가 많은 경우, 원하는 사이즈로 토폴로지를 표현할 수 있으며, 줌인/줌아웃 레벨은 내부 설정에 유지됩니다.
- 노드 드래그 – 노드의 표시 위치를 사용자가 지정하고자 하는 경우, 드래그 하여 위치를 고정할 수 있습니다.
- 프로젝트 노드 클릭
  - 애플리케이션 토폴로지의 경우, 해당 프로젝트의 애플리케이션 토폴로지로 이동합니다.
  - 그룹 토폴로지의 경우, 해당 프로젝트의 토폴로지 정보를 취득하여, 현재 토포로지에 표현합니다.
- 하이라이트 필터링 – 노드 및 노드와 인접한 링크를 필터링을 통해 하이라이트 할 수 있습니다.

[control topology] | https://img.whatap.io/media/user_guide_application/topology/control_topology.png

Figure 37. 애플리케이션 토폴로지 제어

3.5. 토폴로지 표현 형태

3.5.1. 애플리케이션 토폴로지

프로젝트 내의 애플리케이션 간 호출 연관 정보를 표현합니다.

[application topology] | https://img.whatap.io/media/user_guide_application/topology/application_topology.png

Figure 38. 애플리케이션 토폴로지

3.5.2. 그룹 토폴로지

프로젝트 내의 애플리케이션 그룹 간 호출 연관 정보를 표현합니다.

데이터베이스 호출 내역을 포함한 토폴로지의 경우, 하기와 같이 데이터베이스 노드를 포함합니다.

[group topology] | https://img.whatap.io/media/user_guide_application/topology/group_topology_1.png

Figure 39. 그룹 토폴로지 (DB 노드 포함)

타 프로젝트로 부터의 호출 내역이 존재하는 경우, 타 프로젝트의 그룹 노드를 포함합니다.

[group topology 2] | https://img.whatap.io/media/user_guide_application/topology/group_topology_2.png

Figure 40. 그룹 토폴로지 (타 프로젝트 그룹 노드 포함)

타 프로젝트 그룹 노드를 클릭할 경우, 타 프로젝트의 토폴로지 정보를 함께 표시합니다.

[group topology 3] | https://img.whatap.io/media/user_guide_application/topology/group_topology_with_other_project.png

Figure 41. 그룹 토폴로지 (타 프로젝트 그룹 클릭)

3.5.3. 통합 토폴로지

사용자가 복수 프로젝트에 대한 권한을 보유하고 있다면, 프로젝트 단위 토폴로지로 전체 상황을 조망하기 어려운 제약이 존재합니다.

이와 같은 경우, 복수의 프로젝트를 일괄 선택하여 단일 토폴로지로 보기 위한 요구사항이 존재하게 되므로 이를 통합 토폴로지로 표현합니다.

통합 토폴로지의 진입 경로 – 프로젝트 미선택 상태에서 노출되는 통합 애플리케이션 토폴로지 메뉴를 클릭합니다.

[integration topology menu] | https://img.whatap.io/media/user_guide_application/topology/enter_integration_topology.png

Figure 42. 통합 토폴로지 메뉴

상단의 프로젝트 선택 메뉴를 통해 토폴로지에 표현할 프로젝트를 지정합니다.

[select projects] | https://img.whatap.io/media/user_guide_application/topology/select_projects.png

Figure 43. 프로젝트 선택

프로젝트 선택 후, 적용 버튼을 클릭하면 복수 프로젝트의 토폴로지를 표시합니다.

토폴로지 화면의 중앙을 기준으로 프로젝트 단위의 클러스터를 형성하여 각 프로젝트에 포함된 그룹을 인접 위치에 군집시켜 표시합니다.

```
대규모 환경의 경우 멀티 프로젝트 구성을 통해 이를 수용하게 되는데,
이를 토폴로지로 표현할 경우 개별 노드를 화면 전체에 균등 분포시키기 보다는 프로젝트 단위로 모아서 배치하는 것이
전체 규모를 파악하는데 용이하므로 군집 단위를 지정하여 화면에 자동 배치합니다.
```

```
프로젝트, 데이터베이스, 외부 호스트, 애플리케이션 및 그룹을 호출하는 외부 모듈을 군집 단위로 합니다.
```

인접한 노드 간의 간격을 조정하고자 하는 경우, 상단의 노드간격 관련 확대/축소 버튼을 활용하여 노드 간격을 조정할 수 있습니다.

클러스터 구성은 프로젝트, 외부 모듈, 외부 호출, DB 호출 단위로 구성합니다.

[integration topology] | https://img.whatap.io/media/user_guide_application/topology/integration_topology.png

Figure 44. 통합 토폴로지

확대 버튼을 통해 노드 간의 간격을 확장 시 토폴로지가 변경된 화면은 다음과 같습니다.

[expand node distance] | https://img.whatap.io/media/user_guide_application/topology/integration_topology_2.png

Figure 45. 노드 간격 확대/축소

3.5.4. 인스턴스 토폴로지

인스턴스 토폴로지는 애플리케이션 토폴로지와 동일한 데이터를 표현하는데, 애플리케이션 토폴로지가 프로젝트에 포함된 전체 애플리케이션을 대상으로 하는 한 편, 인스턴스 토폴로지는 단일 애플리케이션을 대상으로 연관성을 지닌 애플리케이션, 외부 모듈, DB 및 외부 호출 노드와의 연관성을 표현합니다.

진입 경로 – 사이트맵에서 대시보드 > 토폴로지 > 인스턴스 토폴로지를 클릭합니다.

[instance topology] | https://img.whatap.io/media/user_guide_application/topology/instance_topology.png

Figure 46. 인스턴스 토폴로지

애플리케이션 토폴로지에 표현되는 정보가 너무 많아 애플리케이션 단위로 보고자 하는 니즈가 존재하는 경우, 인스턴스 토폴로지를 활용할 수 있습니다.

애플리케이션 토폴로지의 애플리케이션 노드를 더블 클릭하는 경우, 인스턴스 토폴로지 화면으로 전환하여 더블클릭한 애플리케이션 중심의 토폴로지를 표시합니다.

3.5.5. Netstat 토폴로지

프로젝트 내의 애플리케이션과 애플리케이션의 리스닝 정보, 아웃바운드 호출 연관 정보를 표현합니다.

[netstat topology] | https://img.whatap.io/media/user_guide_application/topology/netstat_topology.png

Figure 47. Netstat 토폴로지

애플리케이션과 애플리케이션의 리스닝 정보와 아웃바운드 호출 정보를 노드로 표현하며,
리스닝 포트와의 관계는 직선으로, 아웃바운드 호출 정보와의 관계는 곡선으로 표현합니다.

진입 경로 – 사이트맵에서 대시보드 > 토폴로지 > Netstat 토폴로지를 클릭합니다.

Netstat 토폴로지를 취득하는 과정은 에이전트의 부하를 유발하므로 자동 갱신 기능을 제공하지 않습니다. 명시적으로 데이터 조회하기 버튼을 통해 사용자의 요청이 있는 경우 데이터를 조회합니다.

3.6. 토폴로지 부가 기능

3.6.1. 하이라이트 필터

대량 노드 간의 복잡한 연관관계를 토폴로지로 확인하기 어려운 점을 보완하기 위하여, 특정 노드를 선택하여 이와 연관관계를 가진 노드를 강조하여 보여주기 위한 기능을 제공합니다.

특정 노드에 마우스 오버 시에 해당 노드 및 연관 노드와 링크 정보를 하이라이트 하여 표시하는 기능에 더해, 하이라이트 상태를 유지할 수 있도록 키워드 입력을 통해 원하는 노드의 선택을 유지시킬 수 있습니다. (토폴로지 갱신 시에도 하이라이트 상태가 유지됩니다.)

[highlight filter] | https://img.whatap.io/media/user_guide_application/topology/highlight_filter.png

Figure 48. 하이라이트 필터

3.6.2. 하이라이트 필터 서포트

하이라이트 필터 기능을 통해 키워드를 입력하는 방식으로 하이라이트 상태를 유지할 수도 있으나, 화면 우측 하단의 노드 선택 뷰에 표시되는 말단의 노드를 클릭하여 하이라이트 필터를 적용할 수 있습니다.

본 차트에는 프로젝트, DB 호출, 외부 호출 노드를 그룹핑하여 다이어그램으로 제공하고, 본 차트의 말단 노드를 클릭하면 해당 노드의 명칭이 하이라이트 필터에 적용됩니다.

[node selection view] | https://img.whatap.io/media/user_guide_application/topology/node_selection_support.png

Figure 49. 노드 선택 뷰

노드 선택 뷰는 다음의 기능을 제공합니다.

- 줌인/줌아웃

- 드래그
- 말단 노드 클릭 시, 하이라이트 필터에 적용
- 상위 노드 클릭 시, 말단 노드 비표시

노드 선택 뷰를 활용하지 않는 경우, 노드 선택 뷰 외곽의 점선을 더블 클릭하여 최소화 할 수 있으며, 노드 선택 뷰를 다시 표시하고자 하는 경우, 최소화 된 영역을 더블클릭하여 복원할 수 있습니다.

[minimized node selection view] | https://img.whatap.io/media/user_guide_application/topology/node_selection_support_min.png

Figure 50. 노드 선택 뷰 최소화

[node tree folded] | https://img.whatap.io/media/user_guide_application/topology/node_selection_folded.png

Figure 51. DB 호출만 클릭하여 접은 상태의 노드 트리

3.6.3. 가변 폰트

차트에 표현되는 정보가 많아질수록 줌아웃 하여 차트를 축소해서 보게 될 가능성이 높습니다. 차트를 축소하게 되면 노드의 위치는 고정하여 노드 식별은 가능할 수 있으나, 글자의 사이즈가 상대적으로 작아져 호출 정보를 식별하기 어려울 수 있습니다. 토폴로지 차트에서는 폰트 사이즈를 노드간 링크의 거리에 비례하여 표시합니다.

[dynamic font size] | https://img.whatap.io/media/user_guide_application/topology/dynamic_font_size.png

Figure 52. 가변 폰트

3.6.4. 컨텍스트 메뉴

토폴로지의 노드에서 마우스 우측 버튼을 클릭하시면 컨텍스트 메뉴가 노출됩니다. 컨텍스트 메뉴에는 애플리케이션 모니터링 대시보드로 이동하기 위한 링크 등의 메뉴를 제공합니다.

컨테이너 맵, 노드 자원, 컨테이너 자원 메뉴는 컨테이너 모니터링에만 제공됩니다.

[context menu] | https://img.whatap.io/media/user_guide_application/topology/context_menu.png

Figure 53. 컨텍스트 메뉴

3.6.5. 노드 위치 저장 및 복원

토폴로지 뷰를 유지하고 있는 경우, 차트에 한 번 표시된 노드의 위치는 고정되어 변경되지 않습니다.

단, 화면에 재진입 하는 경우, 차트 엔진이 제공하는 노드 배치 방식에 따라 뷰 영역에 위치시키게 되므로, 위치를 고정하여 사용하고자 하는 사용자의 경우 본 기능을 활용할 수 있습니다.

위치 특정을 위해 필요한 정보만을 웹 브라우저의 로컬 스토리지에 저장 및 복원하는 방식을 활용합니다.

- 위치 저장 – 화면에 표시되어 있는 노드의 위치 정보를 저장합니다.
- 위치 저장(병합) – 화면에 표시되어 있는 노드의 위치 정보에 기존에 저장한 정보를 병합하여 저장합니다.
- 위치 복원 – 라디오 버튼을 체크할 때 현재 표시되어 있는 노드의 위치를 저장된 위치로 이동시킵니다. 리다오 버튼의 체크 상태를 유지할 경우, 차트에 신규 노드가 추가될 때 저장된 정보로 노드의 위치를 이동시킵니다.

[save position] | https://img.whatap.io/media/user_guide_application/topology/save_position.png

Figure 54. 노드 위치 저장 및 복원

3.6.6. 별칭 부여 및 제거

화면 우측 하단의 노드 선택 뷰를 통해 노드의 별칭을 적용할 수 있습니다.

노드 선택 뷰의 노드에서 마우스 우측 버튼을 클릭하면 별칭 입력 창이 표시됩니다. 별칭을 입력 후 엔터를 입력하시면 별칭 정보가 저장됩니다.

차트의 데이터가 새로 조회되는 시점에 저장된 별칭이 적용됩니다.

노드 별칭 정보를 웹 브라우저의 로컬 스토리지에 저장하는 방식을 활용합니다.

[set alias] | https://img.whatap.io/media/user_guide_application/topology/set_alias.png

Figure 55. 별칭 지정

저장된 별칭을 제거하고자 할 경우, 화면 상단의 별칭 초기화 버튼을 클릭하여 저장된 별칭 정보를 삭제할 수 있습니다.

차트 데이터가 새로 조회되는 시점에 별칭 정보가 삭제되어 반영됩니다.

[reset alias] | https://img.whatap.io/media/user_guide_application/topology/reset_alias.png

Figure 56. 별칭 제거

3.6.7. 인바운드/아웃바운드 필터

인바운드/아웃바운드 토글 버튼을 통해 토폴로지 상에 표현된 외부 모듈 및 외부 호출 노드의 표시 여부를 변경할 수 있습니다.

[inbound/outbound filter] | https://img.whatap.io/media/user_guide_application/topology/inbound_outbound_filter.png

Figure 57. 인바운드/아웃바운드 표시 토글

[inbound/outbound displayed] | https://img.whatap.io/media/user_guide_application/topology/inbound_outbound_displayed.png

Figure 58. 인바운드/아웃바운드 표시 상태

[inbound/outbound hidden] | https://img.whatap.io/media/user_guide_application/topology/inbound_outbound_hidden.png

Figure 59. 인바운드/아웃바운드 비표시 상태

3.6.8. 임계치 기반 강조

노드 간 호출의 응답시간 및 에러 건수 임계치 설정 시 임계치를 초과한 노드 및 호출 관계를 강조하여 표시합니다.

[threshold emphasis] | https://img.whatap.io/media/user_guide_application/topology/threshold_emphasis.png

Figure 60. 임계치 설정

[threshold emphasized] | https://img.whatap.io/media/user_guide_application/topology/threshold_emphasized.png

Figure 61. 임계치 초과 정보 강조

3.7. 토폴로지 적용 방법

3.7.1. Java

- java 에이전트 1.7.1 이상
- 에이전트 옵션 – 토폴로지 표현을 위한 정보 수집 옵션

- tx_caller_meter_enabled – 애플리케이션 및 그룹 간 호출 정보를 수집합니다.
- httpc_host_meter_enabled – HTTP 외부 호출 정보를 수집합니다.
- sql_dbc_meter_enabled – 데이터베이스 호출 정보를 수집합니다.
- actx_meter_enabled – 액티브 트랜잭션 정보를 수집합니다.

- 그룹 및 통합 토폴로지 사용을 위해서는 JVM Option을 통해 하기 설정을 추가해야 합니다.

```
-Dwhatap.okind={그룹 식별자}
```

3.7.2. Node.js

- node.js 에이전트 0.4.1 이상
- 에이전트 옵션 – 토폴로지 표현을 위한 정보 수집 옵션
  - tx_caller_meter_enabled – 애플리케이션 및 그룹 간 호출 정보를 수집합니다.
  - httpc_host_meter_enabled – HTTP 외부 호출 정보를 수집합니다.
  - sql_dbc_meter_enabled – 데이터베이스 호출 정보를 수집합니다.
  - actx_meter_enabled – 액티브 트랜잭션 정보를 수집합니다.
- 그룹 및 통합 토폴로지 사용을 위해서는 node.js 환경 설정을 통해 하기 설정을 추가해야 합니다. (프로젝트 에이전트 설치 페이지 참조)

```
process.env.WHATAP_OKIND = '{그룹 식별자}'
```

Chapter 4. 성능 분석

성능 카운트와 트랜잭션 프로파일 분석을 통해서 성능 병목이나 장애 원인을 찾아냅니다.

특히 와탭의 스택 분석을 이해하면 다른 도구로는 발견하지 못한 어려운 문제들을 분석해 낼 수있습니다.

4.1. Cube

Cube는 5분 간의 수집 정보 통계를 활용한 애플리케이션 상세 분석 정보를 제공합니다. Cube는 1일을 5분 구간으로 분할하여 288개의 Cube 단위 통계로 실시간 사용자, 국가별 접속, 히트맵, 트랜잭션, TOP 트랜잭션, TPS, 응답 시간, 자원 사용 정보를 제공합니다.

4.1.1. Cube 란

와탭은 5분단위로 만든 성능 통계를 큐브라고 부릅니다. 큐브 분석은 큐브에 저장된 5분단위 성능 데이터를 활용한 분석 기능입니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring19] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring19.png

4.1.2. 큐브 셀렉트 패널

큐브는 5분단위로 저장되어있기 때문에 특정시간을 선택해야합니다. 특정시간을 선택하기 위해 기준 데이터를 패널에 보여주고 특정시간을 선택할 수 있도록 돕습니다.

[cube] | https://img.whatap.io/media/images/cube.png

Figure 62. 큐브 셀렉트 패널

큐브 셀렉트 패널의 데이터 종류에는 **[TPS] [실시간 사용자] [응답시간] [액티브트랜잭션]** 이 있습니다.

선택된 지표로 낮은 수치를 가진 큐브는 하늘색, 높은 수치를 가진 큐브는 남색으로 표현됩니다.

4.1.3. 애플리케이션 셀렉트

큐브에서 특정 애플리케이션의 데이터를 선택할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring21] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring21.png

Figure 63. 애플리케이션 셀렉트

4.1.4. 큐브 데이터

히트맵 큐브시간 동안 트랜잭션 분포를 보여줍니다. 프로파일 상세 분석이 가능합니다.
[차트 우측 상단 버튼]을 클릭하면 큐브 시간동안으 탑스택과 유니크 스택을 조회할 수있습니다.

국가 클라이언트 아이피를 기준으로 국가를 맵핑하여 어느 국가에서 트랜잭션이 들어오는지를 보여줍니다. 원의 크기는 트랜잭션 량을 상대적으로 표시합니다.

TPS 초당 처리된 트랜잭션 건수입니다.

응답시간 트랜잭션 평균응답시간입니다.

CPU #5 프로젝트내 애플리케이션(에이전트)를 IP로 정렬하여 사용량이 많은 서버의 CPU를 바차트로 보여줍니다.

사용자 실시간 사용자를 보여줍니다.

상위 TX 호출건수가 많은 트랜잭션 5건을 보여줍니다. 트랜잭션 통계에서는 좀더 상세한 데이터를 확인할수 있습니다.

힙메모리 #5 애플리케이션(인스턴스) 중 힙메모리를 많이 사용한 인스턴스 5개를 보여줍니다.

본 제품은 MaxMind사의 GeoLite2의 데이터를 사용하고 있으며, 이에 따른 라이센스 정책을 준수하고 있습니다.
GeoLite2는 http://www.maxmind.com에서 확인할 수 있습니다.
(This product includes GeoLite2 data created by MaxMind, available from http://www.maxmind.com)

4.2. 히트맵 분석

히트맵은 응답 시간 분포도입니다. 히트맵 트랜잭션 수행 건수 맵을 의미합니다

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring22] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring22.png

Figure 64. 히트맵 조회

히트맵의 진하기는 밀도를 표현합니다.
빨간색이 보인다면 에러 트랜잭션이 여러건 겹쳐있는 경우를 의미합니다.

히트맵은 특정 기간 동안의 응답 시간 분포를 한눈에 파악 할 수 있는 분포 그래프 입니다. 트랜잭션 발생 위치에 따라 느린 트랜잭션을 쉽게 찾아 낼 수 있으며, 색상을 통하여 에러 발생여부에 대해서 또한 빠르게 찾아 낼 수 있습니다.

분석에서 히트맵은 짧게는 10분 길게는 일일로 지정하여 응답 시간 분포에 대해 조회할 수 있으며, 어느 시점에서 응답시간이 느렸는지, 에러가 많이 발생했는지에 대해 파악할 수 있는 지표가 됩니다. 또한 히트맵이 그려진 모양을 통해 문제의 원인을 파악할 수 있습니다. 전체, 애플리케이션 서버 별 히트맵을 볼 수 있어 애플리케이션 별 개선에 사용할 수 있습니다.

짧은 기간 차트

10분, 30분, 시간 단위로 히트맵은 선택한 시작 시간으로부터 각 지정된 시간 단위 동안의 응답시간 분포를 파악 할 수 있습니다. 5초 단위의 히트맵으로, 드래그시 선택한 기간과 응답시간 내에 발생한 트랜잭션을 조회할 수 있습니다. 조회한 트랜잭션은 클릭 시 프로파일 정보를 조회 할 수 있으며 스텝 정보를 볼 수 있습니다. 시간대 별로 응답시간이 느린 트랜잭션이 무엇인지 판단하는 것이 중요하며, 특정 시간에만 발생하는 트랜잭션인지에 대한 여부 또한 확인해 보아야할 사항입니다. 에러가 발생했다면 에러의 대한 원인이 연쇄반응으로 인하여 발생한지에 대한 여부 또한 파악해야 합니다.

일일이상 차트

일일 단위의 히트맵은 선택한 날짜에 대한 응답시간 분포를 파악할 수 있습니다. 5분 단위의 히트맵으로, 시간 별 추이를 파악하는데 사용합니다. 특정 시간대에 발생한 문제를 찾아낼 수 있으며, 다른 날짜의 히트맵과의 비교를 통해, 주기적으로 발생하는 문제가 아닌지 확인할 필요가 있습니다. 3일,7일까지 선택할 수 있습니다.

조회 조건부

조회하고자 하는 날짜와, 시작시간을 선택하여 원하는 과거의 히트맵 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능으로 원하는 데이터를 조회할 수 있습니다.

- 날짜 – 히트맵 조회 시작 날짜 설정
- 시간 – 히트맵 조회 시작 시각 설정(기간을 1일 이하로 선택시)
- 분 – 히트맵 조회 시작 분 설정(기간을 1일 이하로 선택시)
- 카테고리 – 10분/30분/시간/일일/3일/7일 히트맵 선택
- 애플리케이션 – 전체, 개별 애플리케이션 서버 설정

4.3. 히트맵 패턴의 이해

히트맵은 트랜잭션의 종료시간은 X축, 응답시간은 Y축으로한 분포 차트입니다.

정상적인 웹 애플리케이션이라면 수 초 이하 구간에 집중된 분포를 보입니다.

[h0] | https://img.whatap.io/media/user_guide_application/analysis/h0.png

Figure 65. 일반적인 패턴

4.3.1. 세로줄(LOCK 현상) 패턴

트랜잭션 처리중 일시적인 락(Not only DB Lock)이 발생하면 이로 인해 처리를 대기합니다. 그리고 락이 해소되면 처리 대기중 트랜잭션들은 비슷한 시간대에 함께 종료됩니다.

그러면 세로로 줄이 만들어집니다.

[h3] | https://img.whatap.io/media/user_guide_application/analysis/h3.png

Figure 66. 세로줄 패턴

세로줄 패턴으로 락을 감지하는 것은 매우 강력한 개념입니다. 특히 마이크로서비스아키텍처에서는 백엔드 시스템에서 발생하는 LOCK도 동일하게 감지 될수 있습니다.

[h8] | https://img.whatap.io/media/user_guide_application/analysis/h8.png

Figure 67. Front → API → DB

Front 애플리케이션의 응답 패턴 세로줄은 Back-End 시스템이 사용하는 DB에서 락이 발생한 경우라도 감지 됩니다.

4.3.2. 가로줄(타임 아웃) 패턴

10초 타임아웃 조건에서 해당 자원이 부족하면 트랜잭션들은 10초 대기 후 타임 아웃 에러가 발생할 것입니다. 이때 히트맵 10초 부근에 가로줄 나타납니다.

[h2] | https://img.whatap.io/media/user_guide_application/analysis/h2.png

Figure 68. 가로줄 패턴

타임아웃 이후 재시도하는 로직이 있다면 그림처럼 가로라인이 10초 단위로 반복됩니다.

장애 분석 사례

실제 장애 상황의 히트맵입니다.

[h7] | https://img.whatap.io/media/user_guide_application/analysis/h7.png

Figure 69. 장애 사례

(1)구간에서 응답시간이 급증했고 (2)구간의 빨간 라인은 전형적인 가로라인 패턴입니다.
(1)구간 부하 발생으로 ConnectionPool이 소진되고 (2)구간은 ConnectionPool 부족으로 2차 타임아웃 장애가 발생한 상황 입니다.

4.3.3. 패턴 분석 활용

트랜잭션 응답분포에 줄이 보인다는 것은 병목이 있다는 것입니다. 일시적인 락킹이면 세로줄이 그병목이 타임아웃으로 빠지면 가로즐이 만들어집니다.

[h4] | https://img.whatap.io/media/user_guide_application/analysis/h4.png

Figure 70. 장애 발생 사례

문제를 분석할때 라인에 포함된 트랜잭션만을 선택적으로 분석함으로 문제를 빠르게 찾아낼 수 있습니다.

4.4. 머신러닝 기반 응답패턴 분석

머신러닝 기술을 통해 히트맵 패턴을 분석 후 비정상 여부를 자동 감지해 경고를 발행하는 기능 입니다.

[ml pattern] | https://img.whatap.io/media/user_guide_application/analysis/ml_pattern.png

4.4.1. 비정상 패턴

월 수백TB의 성능 데이터로부터 비정상 패턴을 학습하고 학습된 비정상 패턴과 유사한 패턴이 발생하는 경우 이에 대한 알람을 발행 합니다.

[h3] | https://img.whatap.io/media/user_guide_application/analysis/h3.png

Figure 71. 세로줄 패턴

[h2] | https://img.whatap.io/media/user_guide_application/analysis/h2.png

Figure 72. 가로줄 패턴

[h7] | https://img.whatap.io/media/user_guide_application/analysis/h7.png

Figure 73. 복합 패턴

[ai alert] | https://img.whatap.io/media/images/ai_alert.png

Figure 74. 히트맵 알람

4.5. 스택 분석

와탭은 10초(기본값) 간격으로 수집한 Thread Stack을 활용하여 메소드 레벨의 성능 지연 구간을 분석합니다.

Java와 Python 애플리케이션에서 스택 분석 기능을 사용 할 수 있습니다.

[st1] | https://img.whatap.io/media/user_guide_application/stack/st1.png

Figure 75. 액티브 스택

위 스택에서 탑라인은 socketRead0 입니다.

```
java.net.SocketInputStream.socketRead0(Native Method)
```

TOP STACK이란?

스택에서 탑라인은 덤프를 수행할 쓰레드가 해당 메소드를 수행중이라는 것을 의미합니다. 순간적으로 잡혔을 가능성도 있지만 확률적으로 해당모듈의 처리시간의 합의 비율만큼 스택에 나타날 것입니다. 그래서 이 탑라인 메소드의 빈도를 계산함으로써 메소드 레벨의 성능을 판단할 수 있습니다. 이 탑라인의 빈도 통계를 와탭은 TOP STACK이라고 합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring23] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring23.png

Figure 76. 스택의 탑라인

TOP STACK 분석을 통해 도출된 메소드는 어떤 메소드가 호출했는지에 대한 빈도를 분석할 수있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring24] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring24.png

TOP STACK의 계층 분석에서는 원래 액티브 스택을 확인하기 어려웠습니다. 따라서 Active Stack를 조회할 수있도록 동일 스택을 모아서 Unique Stack 이라는 조회 기능을 제공합니다.

4.5.1. 탑 스택

Stack Trace 상의 각 Step 기준으로, Step과 Step간의 호출 비율을 백분율로 분석한 정보를 제공합니다. 최상위 Step의 적체 빈도를 백분율로 산출하여 내림차순으로 정렬한 결과를 제시합니다.

각 Step을 클릭하면 해당 Step을 호출하는 상위 Step을 호출 빈도 기준 백분율로 산출하여 제공합니다.

Top 스택 통계는 충분히 많은 데이터를 가지고 판단하는 것이 좋습니다.
수집한 스택의 개수가 소수(10개 미만)인 경우 통계 의미를 갖기에 부족합니다.

Top 스택은 튜닝 시 인지 하기 힘들었던 부분의 튜닝 포인트를 찾아내는 것에 유용합니다. 가장 빈번하게 나타난 스택은 현재 애플리케이션 서버에서 가장 많은 응답지연을 발생하는 것이라고 판단 할 수 있습니다. 가장 왼쪽의 나타나는 비율은 애플리케이션 서버 성능에 영향을 미치는 정도라고 할 수 있습니다.

안정적인 애플리케이션 서버일지라도 빈번하게 나타난 스택은 성능 저하를 일으킬 수 있는 가능성이 있으므로 해당 클래스는 유심히 보는 것이 좋습니다.

Top 스택 클릭 시 해당 최상위 스택에 대한 호출 빈도를 확인할 수 있습니다. Top 스택의 호출관계는 1대1 관계이므로 depthTop 스택은 가 밑으로 내려갈수록 정보의 정확성이 떨어질 수 있습니다. 하위 depth에 대한 정보는 참고 용도로 사용하며 튜닝을 진행하시기 바랍니다.

애플리케이션 성능 개선을 위해 최상위 Step의 적체 비율이 높은 모듈의 병목 가능성을 검토 해야 합니다. 적체 비율이 높은 모듈의 경우 작은 성능 개선도 애플리케이션 전체에 상당한 개선 효과를 가져올 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring25] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring25.png

```
whatap.util.ThreadUtil.sleep <- jdbc.Control.exec 의 호출 비율은 87.10%
jdbc.Control.exec <- jdbc.FakePreparedStatement.executeQuery 의 호출 비율은 61.18%
```

whatap.util.ThreadUtil.sleep ← jdbc.Control.exec ← jdbc.FakePreparedStatement.executeQuery 의 호출 비율이 87.10% * 61.18% 를 의미하지는 않음.
(jdbc.Control.exec에서 타 모듈의 호출 가능성이 존재하기 때문)

Top Stack을 통해 호출 비율을 판단 할 경우, 각 Step간 호출 비율을 곱하여 전체 호출 관계 비율을 산출해서는 안됩니다. Top Stack의 호출 비율은 Stack Trace 상에 노출된 정보의 Step간 호출 비율의 산출 결과이기 때문에, Step간 호출 비율로 전체 호출 비율을 도출할 경우 왜곡된 결과를 도출하게 됩니다.

탑 스택 통계에서는 일정 기간을 기준으로 시간에 따른 비율 변화와 수집 건 수에 대한 히스토리를 제공합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring27] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring27.png

퍼센트

조회기간 선택된 탑스택의 비율 변화를 나타냅니다.
장애 시점 현황 파악, 개선 전/후 비교에 유용하게 사용됩니다.

건수

수집되는 스택의 수는 액티브 트랜잭션 수에 비례합니다.
특정 구간에서 수집량이 증가했다면 서비스 지연이나 급격한 유입량 증가가 있었음을 알 수 있습니다.

다음과 같은 다이어그램으로도 확인이 가능합니다. image::https://img.whatap.io/media/images/Screenshot_2020-12-15_W_JAVA_DEMO_5490_-_Application_Monitoring26.png[]

Stack Chart

개별 탑 스택의 비율을 차트로 나타냅니다.

4.5.2. 유니크 스택

Stack Trace 전체의 Hash 값 기준의 산출 결과로 전체 Step이 동일한 호출 비율을 백분율로 분석한 정보를 제공합니다. Top Stack이 Step 간의 호출 비율에 대한 정보를 제공하는 것과 달리, Unique Stack은 Stack Trace 전체의 정확한 호출 정보를 기반으로 한 데이터를 제공하므로, 상세 호출 관계를 파악하는 데 유용한 정보를 제공합니다.

적체 비율이 높은 Stack Trace 식별 할 수 있습니다.

상세 호출 Step 검토를 통한 호출 경로 상에 이상 모듈의 존재 여부를 파악 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring27 Y8LaYJq] | https://img.whatap.io/media/images/Screenshot_2020-

12-15_W_JAVA_DEMO_5490_-_Application_Monitoring27_Y8LaYJq.png

4.5.3. 액티브 스택

액티브 스택 메뉴를 선택하면 수집된 ActiveStack을 차트로 확인할 수있습니다. 차트는 5분 간의 단위 통계 데이터를 Active Transaction의 수를 막대로, TPS를 꺾은선으로 표시합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring28] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring28.png

막대를 클릭하면, 클릭한 시간대의 Active Transaction의 정보가 나오며, 그 정보를 클릭하면 해당 Transaction의 ActiveStack을 볼 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring29] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring29.png

4.5.4. 스택통계 활용 예

스택 - 탑스택 메뉴에서 특정 시간대 내역을 조회한 예 입니다.
프로파일이나 트랜잭션 통계정보로는 알 수 없는 다양한 개선포인트를 찾을 수 있습니다.

스택 - 탑스택 메뉴에서 탑스택 정보를 조회 합니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring31] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring31.png

Figure 77. 스택 - 탑스택

점유율 순서대로 상위 항목 부터 Class/Method 정보를 확인 합니다.

```
최 상위 79.92% 비율을 차지하는 탑스택은 org.apache.logging.log4j.core.layout.TextEncoderHelper.copyDataToDestination()
메소드 입니다. ①
두번째로 높은 8.01% 비율을 차지하는 탑스택은 sun.misc.Unsafe.park() 메소드 입니다. ②
```

히스토리 기능으로 시간대 별 추이를 확인 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring30] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring30.png

Figure 78. 스택 - 탑스택 - 히스토리

```
06:00 시 경 TextEncoderHelper.copyDataToDestination() 비율이 감소한 것이 확인 됩니다.
동 시간대 sun.misc.Unsafe.park는 감소된 TextEncoderHelper.copyDataToDestination() 비율만큼 증가 했습니다.
```

상위 1,2 탑스택은 모두 Logging 과 관련되어 있습니다. 즉, 이 시스템은 전체 업무에서 90% 가까운 수행시간을 Logging 에 소비하고 있는 것이 확인 됩니다.

확인한 점유율을 바탕으로 개선 기대효과를 산정 합니다.

개선 기대효과 산정

사례 중 copyDataToDestination() 79.92%, sun.misc.Unsafe.park() 8.01% 비율은 서비스 수행중 점유된 시간 비율입니다.
즉, copyDataToDestination() 점유율을 반으로 낮춘다면 응답속도 기준으로 44% 가량 성능이 개선된다고 기대 할 수 있습니다.

탑스택 정보를 토대로 개선 방안을 검토합니다.

org.apache.logging.log4j.core.layout.TextEncoderHelper.copyDataToDestination()

```
private static void copyDataToDestination(final ByteBuffer temp, final ByteBufferDestination destination) {
        synchronized (destination) {
            ByteBuffer destinationBuffer = destination.getByteBuffer();
            if (destinationBuffer != temp) { // still need to write to the destination
                temp.flip();
                if (temp.remaining() > destinationBuffer.remaining()) {
                    destinationBuffer = destination.drain(destinationBuffer);
                }
                destinationBuffer.put(temp);
                temp.clear();
            }
        }
    }
```

① TextEncoderHelper 는 성능 문제를 비롯한 여러가지 이유로 Deprecated 된 API 입니다.

② sun.misc.Unsafe.park()은 TextEncoderHelper 내부의 synchronized 와 관련 있습니다.

4.6. 성능 카운터

4.6.1. 성능 추이 분석

분석 | 성능 추이 에서는 지정한 시간에 해당하는 성능에 영향을 끼치는 정보들을 조회할 수 있습니다.

성능 추이에 조회할 수 있는 정보로는 실시간 사용자, 트랜잭션/초(합계), 응답시간, CPU, 힙 메모리, 액티브 TX 등 정보와 많이 처리된 트랜잭션#10, HTTP Call#10, SQL#10 등의 정보를 조회할 수 있습니다. 또한 전체, 개별 애플리케이션 서버 별로 그래프를 확인할 수 있어 성능에 많은 영향을 끼치는 애플리케이션 서버가 무엇인지도 쉽게 알 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring32] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring32.png

Figure 79. 성능 추이

화면에 표시되는 정보는 다음과 같습니다.

Table 3. 성능추이 항목

항목	내용
동시접속 사용자	실시간 방문자 수 추이
TPS	처리된 트랜잭션의 개수 합 추이
응답시간	응답시간의 평균 추이
CPU	CPU 사용량의 평균(%) 추이
힙 메모리	힙 메모리의 사용량 추이
액티브 트랜잭션	실행 상태의 트랜잭션 수 추이
트랜잭션 #10	많이 처리된 트랜잭션 상위 10개
HTTP 호출 #10	HTTP Call 상위 10개
SQL #10	많이 호출된 SQL 상위 10개

Chapter 5. 통계 분석

과거 데이터를 기간별, 애플리케이션 서버별 등 원하는 조건을 설정하여 조회할 수 있습니다. 트랜잭션, 에러, SQL, 원격 HTTP 호출, 클라이언트 IP, 클라이언트 브라우저를 사용하여 과거 데이터를 통한 분석 및 튜닝이 가능합니다. 실시간 모니터링을 진행하지 못한 기간은 통계에서 제공하는 과거 데이터 조회를 통해 장애 및 성능 문제를 찾아낼 수 있습니다. 또한, 실시간 모니터링에서 발견하지 못하거나 프로파일 정보에서 발견하기 힘든 튜닝 포인트를 찾아낼 수 있으며 거시적인 문제의 추이 또한 그래프와 표를 통하여 파악할 수 있습니다.

5.1. 트랜잭션

과거의 해당 기간동안 처리된 트랜잭션 정보를 조회할 수 있습니다. 조회한 정보는 내보내기 버튼을 사용해 CSV로 다운로드 할 수 있습니다.

[Screenshot 2020 12 15 W JAVA DEMO 5490 Application Monitoring33] | https://img.whatap.io/media/images/Screenshot_2020-12-

15_W_JAVA_DEMO_5490_-_Application_Monitoring33.png

Figure 80. 조회 조건

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 서버 전체, 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 정렬된 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 시작일 - 통계 조회 시작 날짜 설정
- 시작 시간 - 통계 조회 시작 시간 설정
- 기간 - 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 구분 - 검색할 애플리케이션 구분 (에이전트, 에이전트 종류, 에이전트 노드)
- 애플리케이션 - 전체 또는 개별 애플리케이션 서버 설정
- 필터 - 필터링 기준 및 값 설정
  - 경과시간 -
  - 에러메세지 -
  - HTTP 호출 건수 -
  - SQL 건수 -
  - SQL 패치 건수 -
  - URL -
  - IP -
  - 도메인 -
  - 유저 에이전트 -
  - referer -
  - 트랜잭션 ID -
  - 멀티 트랜잭션 ID -
  - 사용자 ID -

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring.png

Figure 81. 조회 결과

조회한 정보는 “컬럼 추가”를 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. 트랜잭션 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- 트랜잭션 – 트랜잭션 서비스 URL
- 건수 – 트랜잭션이 수행된 총 횟수 (에러 횟수 포함)
- 에러 – 트랜잭션 에러 카운트
- 평균시간 – 평균 응답시간 (ms)
- 합계시간 – 응답시간의 총 합(ms)
- 최대시간 – 최대 응답시간 (ms)
- 최소시간 – 최소 응답시간 (ms)
- 평균CPU – 평균 CPU 사용 시간 (ms)
- 메모리 평균 – 평균 메모리 사용량 (Byte)
- HTTPC 평균 –
- HTTPC 건수 –
- SQL 건수 –
- SQL 패치 –
- SQL 패치시간
- SQL 시간 –

활용처

```
과거 특정 기간의 트랜잭션 발생 빈도를 확인할 수 있습니다.
응답시간이 느린 트랜잭션을 기간 별, 애플리케이션 서버 별로 찾아낼 수 있습니다.
평균 응답시간을 통하여 튜닝이 필요한 트랜잭션이나 애플리케이션 서버를 찾아낼 수 있습니다.
```

트랜잭션(URL)별 추이 분석

트랜잭션(URL)별 추이를 확인 할 수있습니다.

[tran detail] | https://img.whatap.io/media/user_guide_application/stat/tran_detail.png

Figure 82. 트랜잭션(URL)별 처리량 추이

장애 발생시 장애 시점과 같이 연동되는 URL을 확인하는데 유용합니다.

[tran detail2] | https://img.whatap.io/media/user_guide_application/stat/tran_detail2.png

Figure 83. 에이전트별 트랜잭션 추이 비교

에이전트 비교를 통해서 로드벨런스 상태 등을 확인 할수 있습니다.

전체 처리양을 비교했을때는 밸런스가 맞을 수 있지만 핵심 업무는 한쪽으로 치우칠 여지가 있습니다. 이런경우 주요 URL이 모든 서버에서 밸런스가 맞은 상태로 처리되었는지를 확인 할수 있습니다.

트랜잭션(URL)별 프로파일 분석

처리량 막대그래프에서 특정 시점을 클릭하면 그 시점(5분) 동안의 트랜잭션을 조회하여 프로파일을 확인 할 수있습니다.

[tran p1] | https://img.whatap.io/media/user_guide_application/stat/tran_p1.png

Figure 84. 특정구간 클릭

[tran p2] | https://img.whatap.io/media/user_guide_application/stat/tran_p2.png

Figure 85. 클릭 시점의 URL 프로파일 조회

특정 URL을 필터링 해서 프로파일을 검토하고자 할때 본 기능을 활용합니다.

5.2. 에러

과거의 해당 기간동안 발생한 정보를 조회할 수 있습니다. 한 눈에 해당 기간동안 발생한 에러를 조회할 수 있으며, 조건을 통해 원하는 에러만 찾을 수 있습니다. 모니터링을 하지 못한 상황에서는 가장 먼저 에러를 조회해야 하며, 에러가 발생한 최초 시점부터 찾아가는 것이 좋습니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 필터 기능을 통해 찾고자하는 에러만을 조회할 수 있으며 에러 건수에 대한 내림차순으로 정보가 조회됩니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring3] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring3.png

- 시작일 - 통계 조회 시작 날짜 설정
- 기간 - 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 필터 - 필터링 기준 및 값 설정
  - 클래스 - 에러 클래스에 포함된 문자열 필터링
  - 메세지 - 에러 메세지에 포함된 문자열 필터링
  - 트랜잭션 - 트랜잭션 URL에 포함된 문자열 필터링
- 정렬 순서 - 정렬될 순서 설정
- 애플리케이션 - 전체 또는 개별 애플리케이션 서버 설정

조회한 정보는 컬럼 추가 버튼을 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. 에러 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- 클래스 - 에러 클래스
- 트랜잭션 - 에러가 발생한 트랜잭션 URL
- 메세지 - 에러 메시지
- 시간 - 최초 에러 발생 시간
- 건수 - 에러 건수

에러 통계의 활용

특정 기간의 에러 건수를 조회할 수 있습니다. 모니터링을 하고 있지 않았다면 가장 먼저 수행해야할 작업입니다. 하루 단위의 조회를 통해 에러의 증감을 찾아낼 수 있습니다. 과거 에러 데이터를 통해 새로운 에러 발생 여부를 알 수 있습니다.

에러별 추이 분석

각 에러의 발생위치를 추적하기 위해 스택정보를 수집하고 있습니다. URL별 에러별 5분에 한 건의 스택을 보여줍니다. 또한 에러별 발생빈도를 보여줍니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring4] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring4.png

Figure 86. 에러 상세 분석

해당에러가 어느 시점부터 많아졌는지를 확인할 수 있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring5] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring5.png

Figure 87. 에러 상세 분석

에이전트별 비교를 통해 에러가 한쪽에서만 발생했는지 여러서버에서 발생했는지등을 확인할수 있습니다.

5.3. SQL

과거의 해당 기간동안 실행한 SQL 정보를 조회할 수 있습니다. SQL의 실행이력과 함께 평균 실행시간 및 에러 건수등을 조회할 수 있습니다. SQL 수행시간, 페치 개수와 같은 정보는 SQL 튜닝에 있어 가장 중요하게 봐야할 부분입니다. 예를 들어, 과도한 Fetch 수 는 애플리케이션의 성능에 전체적인 영향을 미칠 수 있으며, 해당 트랜잭션에 응답시간을 느리게 만드는 원인이 됩니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 애플리케이션 – 전체 또는 개별 애플리케이션 서버 설정
- 정렬 순서 – 정렬 기준 설정
  - 개수 – SQL 실행 건수 내림차순
  - 에러 – SQL 에러 건수 내림차순
  - 합계 시간 – SQL 수행시간의 합 내림차순å
  - 최소 시간 – 최소 SQL 수행 시간 오름차순
  - 합계 시간 – 최대 SQL 수행 시간 내림차순
  - 평균 시간 – 평균 SQL 수행 시간 내림차순
  - 패치 개수 – SQL fetch 건수 내림차순
- 필터 – 필터링 기준 및 값 설정
  - SQL – SQL 문에 포함된 문자열 필터링
  - 데이터베이스 – Database connection url에 포함된 문자열 필터링

조회한 정보는 “컬럼 추가”을 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. SQL 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- 데이터베이스 – Database connection url 주소
- SQL – 실행한 SQL 문
- 해시 – SQL 문의 hash 값
- 건수 – SQL 실행 건수
- 에러 – SQL 에러 건수
- 평균 시간 – SQL 평균 수행 시간 (ms)
- 합계 시간 – SQL 총 수행 시간 (ms)
- 최대 시간 – 최대 SQL 수행 시간 (ms)
- 패치 건수 – SQL fetch 건수
- 패치 시간 – SQL fetch 시간
- 서비스 – SQL문을 수행했던 URL중 하나(샘플)

SQL별 추이 분석

와탭은 SQL별로 호출건수를 수집하고 있습니다. 특정 SQL이 시간대별로 어떤 빈도로 수행되었는지를 확인할 수있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring6] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring6.png

Figure 88. SQL 상세 분석

SQL에 대해서 에이전트별 호출빈도를 확인할 수있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring7] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring7.png

Figure 89. 애플리케이션별 SQL 상세 분석

에이전트별 비교를 통해 에러가 한쪽에서만 발생했는지 여러서버에서 발생했는지등을 확인할수 있습니다.

5.4. 원격 HTTP 호출

과거의 해당 기간동안 실행한 외부 Http Call 정보를 조회할 수 있습니다. 외부 Http Call 실행 이력, 응답시간, 에러 정보를 조회할 수 있습니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 애플리케이션 – 전체 또는 개별 애플리케이션 서버 설정
- 정렬 순서 – 정렬 기준 설정
  - 건수 – Http call 실행 건수 내림차순
  - 에러 – Http call 에러 건수 내림차순
  - 합계 시간 – Http call 응답시간의 합 내림차순
  - 최소 시간 – 최소 Http call 응답시간 오름차순
  - 합계 시간 – 최대 Http call 응답시간 내림차순
  - 평균 시간 – 평균 Http call 응답시간 내림차순
- Filter – 필터링 기준 및 값 설정
  - URL – Http call URL 주소에 포함된 문자열 필터링
  - 호스트 – Http call을 보낸 Host의 IP에 포함된 문자열 필터링

조회한 정보는 “컬럼 추가”를 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다.

- URL – Http call 을 요청한 url 주소
- 호스트 – Http call 을 요청한 Host 주소
- 포트 – Http call 을 요청한 포트
- 건수 – Http call 실행 건수
- 에러 – Http call 에러 건수
- 평균 시간 – Http call 평균 응답시간 (ms)
- 합계 시간 – Http call 총 응답시간 (ms)
- 최소 시간 – 최소 Http call 응답시간 (ms)
- 합계 시간 – 최대 Http call 응답시간 (ms)

외부 HTTP Call별 추이 분석

와탭은 Http Call별로 호출건수를 수집하고 있습니다. 특정 Http Call이 시간대별로 어떤 빈도로 수행되었는지를 확인할 수있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring8] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring8.png

Figure 90. Http Call 상세 분석

막대 차트에서 빨간 색은 에러가 있었던 횟수를 처리 건수 대비 비율로 나타냅니다. 해당 호출이 어떤 시간대에 에러가 많이 났는지를 확인할 수있습니다.

개별 Http Call에 대해서 에이전트별 호출빈도를 확인할 수있습니다.

[Screenshot 2020 12 17 W JAVA DEMO 5490 Application Monitoring9] | https://img.whatap.io/media/images/Screenshot_2020-12-

17_W_JAVA_DEMO_5490_-_Application_Monitoring9.png

Figure 91. 애플리케이션별 Http Call 상세 분석

에이전트별 비교를 통해 에러가 한쪽에서만 발생했는지 여러서버에서 발생했는지등을 확인할수 있습니다.

5.5. 클라이언트 IP

과거의 해당 기간동안 접속한 사용자의 IP 통계 정보를 조회할 수 있습니다. IP 정보를 통해 사용자의 국가 및 도시를 확인 할 수 있으며, 이 정보를 통해 서비스 지역이 아닌 다른 지역에서 잘못된 접근이 발생하는지에 대한 여부를 판단할 수 있습니다. 또한, 방황벽 룰에 따른 특정 IP가 아닌 다른 IP에서 요청이 발생 했을 시 해킹의 위험이 있었는지에 대한 여부를 보실 수 있습니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 애플리케이션 – 전체 또는 개별 애플리케이션 서버 설정
- Filter – 필터링 기준 및 값 설정
  - IP – 사용자 IP 주소에 포함된 문자열 필터링

조회한 정보는“컬럼 추가을 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. 클라이언트 IP 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- IP – 클라이언트 IP 주소
- 국가 – IP 기반의 요청 국가 정보
- 도시 – IP 기반의 요청 도시 정보
- 건수 – 요청 횟수

5.6. UserAgent 통계

Http User Agent 정보를 통계화합니다. User Agent 정보는 브라우저/OS 통계를 위한 기본정보이기도 합니다 그런데 일부 폰이나 브라우저는 일반적인지 않은 UserAgent 정보를 보내기 때문에 원본 텍스트를 기반으로통계를 보여줍니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 애플리케이션 – 전체 또는 개별 애플리케이션 서버 설정
- 필터 – 필터링 기준 및 값 설정
  - 유저에이전트 – 유저에이전트에 포함된 문자열 필터링

조회한 정보는 “컬럼 추가”를 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. 클라이언트 IP 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- 유저에이전트 – 유저에이전트 텍스트

- OS – 사용자 OS 이름
- 브라우저 – 사용자 접속 브라우저 이름
- 건수 – 요청 횟수

[ua 2] | https://img.whatap.io/media/user_guide_application/stat/ua_2.png

Figure 92. 유저에이전트별 시계열 추이

5.7. 클라이언트 브라우저 통계

과거의 해당 기간동안 접속한 사용자의 OS 및 브라우저 정보를 조회할 수 있습니다. 모바일 서비스의 경우 Android, iOS, Window 등 어떤 Device를 많이 사용하는지 판별할 수 있는 정보로 활용할 수 있습니다. 브라우저에 대한 정보 또한 관심있게 살펴보아야 할 우선 순위를 알 수 있습니다. 클라이언트 브라우저 통계는 BilevelChart 로 OS별 비중과 각각의 OS별 브라우저 사용 비중을 쉽게 파악할 수 있습니다.

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 전체 및 개별 통계를 선택적으로 조회할 수 있습니다. 정렬 기준을 선택하여 데이터를 조회할 수 있으며, 필터 기능을 통해 원하는 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)

조회한 정보는 “컬럼 추가”를 통해 선택적으로 원하는 값을 추가/삭제 하여 볼 수 있습니다. 클라이언트 IP 통계에서는 다음과 같은 정보를 조회할 수 있습니다.

- OS – 사용자 OS 이름
- 브라우저 – 사용자 접속 브라우저 이름
- 건수 – 요청 횟수

[ua 1] | https://img.whatap.io/media/user_guide_application/stat/ua_1.png

Figure 93. OS/브라우저별 접속현황

5.8. 이벤트 기록

과거의 해당 기간동안 발생한 여러 수준의 이벤트 정보를 조회할 수 있습니다. 치명적, 경고, 정보로 수준을 나누어서 이벤트를 조회할 수 있으며, 날짜 별, 애플리케이션 별로도 조회가 가능합니다.

[1050] | https://img.whatap.io/media/user_guide_application/stat/1050.png

조회하고자 하는 날짜와, 시작시간, 기간을 선택하여 원하는 과거의 데이터를 조회할 수 있습니다. 애플리케이션 서버 전체, 개별 통계를 선택적으로 조회할 수 있습니다. 수준을 선택해 원하는 수준의 데이터를 선택적으로 데이터를 조회할 수 있습니다.

- 날짜 – 통계 조회 시작 날짜 설정
- 시작 시간 – 통계 조회 시작 시간 설정
- 기간 – 통계 조회 기간 설정 (5분, 1시간, 3시간, 6시간, 12시간, 1일)
- 애플리케이션 – 전체 또는 개별 애플리케이션 서버 설정
- 수준 – 필터링할 이벤트의 수준 설정
  - 치명적 – 서버 구동에 위험을 끼치는 정도
  - 경고 – 서버 구동에 조금 장애가 되는 정도

◦ 정보 – 그 외 여러 정보

[event 1] | https://img.whatap.io/media/user_guide_application/stat/event_1.png

Figure 94. 경고별 상세 빈도

개별 경고가 언제 발생했는지 추이도 확인이 가능합니다.

Chapter 6. 보고서

대기업, 공공기관 및 IT서비스기업에서 사용하는 보고서 양식에 맞추서 데이터를 확인 할 수 있습니다.
보고서 기능을 활용하면 분석 메뉴를 이동하며 워드/엑셀로 데이터를 재가공 하는 일, 반복적인 점검 업무를 자동화 할 수 있습니다.

[보고서] | https://img.whatap.io/media/images/report_app.png

보고서 종류는 상시 업데이트 됩니다.
원하는 양식의 보고서가 있다면 support@whatap.io 로 문의 주세요.

6.1. 보고서 다운로드

보고서를 html 형식으로 다운로드 합니다.

6.2. 보고서 인쇄

보고서를 즉시 인쇄 합니다.

6.3. 보고서 메일 발송 예약

보고서 메일 발송 예약 을 선택하면 예약한 스케줄에 따라 email로 보고서를 수신할 수 있습니다.
출근 직후 수행하던 여러가지 서비스 점검 절차를 보고서 메일 확인으로 대체 할 수 있습니다.

[보고서 메일 발송 예약] | https://img.whatap.io/media/images/report_mail.png

Chapter 7. 에이전트 제어및 상태

와탭 APM 모니터링 서비스는 서버 메뉴를 통해 애플리케이션(서버)의 프로파일 정보를 제공하여, 관리를 위한 다양한 정보를 제공합니다.

7.1. 에이전트 목록

대시보드 왼쪽 서버 메뉴를 선택하면 에이전트로 부터 전송된 애플리케이션 서버 목록이 표시됩니다.

[ag 1] | https://img.whatap.io/media/user_guide_application/agent/ag_1.png

아이디

애플리케이션(에이전트) 아이디

애플리케이션

애플리케이션 서버 식별 용도의 명칭

시작 시간

애플리케이션 서버 기동 시각

활성/비활성

에이전트의 최종 활성/비활성 상태 천이 시각

상태

에이전트의 활성/비활성 여부 (active 또는 inactive로 표시)

버전

에이전트 버전

IP

애플리케이션 서버의 네트워크 인터페이스 IP 주소

CPU 코어

물리 서버의 코어 수

정보

더보기 버튼 클릭 시 하위 버튼을 통해 추가 기능 제공

부트 환경

에이전트 기동 옵션 정보

환경

시스템 환경변수 정보

컴포넌트 버전

로딩된 jar 라이브러리의 버전 정보

스레드 목록/덤프

실행 중 쓰레드 현황

힙 히스토그램

Heap 점유 객체 현황

로드된 클래스

로딩된 클래스 현황

오픈 소켓

아웃바운드 소켓 활용 정보

메소드 성능 통계

아웃바운드 소켓 활용 정보

DB 상태

에이전트 옵션 변경

설정

에이전트 옵션 변경

Throttling 설정

에이전트 옵션 변경

시스템 GC/힙덤프

System.gc() 호출

에이전트 로그

에이전트 로그 확인

에이전트 덤프

에이전트 트랜잭션/쓰레드 덤프 확인

7.2. 정보

7.2.1. 부트 환경

애플리케이션이 실행되면서 주요 정보들을 와탭이 수집합니다. 에이전트가 실행되면서 유지보수를 위한 참고 정보를 서버에 전송하여 보관합니다.

에이전트 타임존, IP,PORT등의 정보들이 포함되며 에이전트 초기화된 후 한번 수집됩니다.

7.2.2. 환경

System 환경 변수 특히 자바는 System.env를 조회합니다. 메뉴가 클릭될때 현재 환경정보를 조 회하는 메뉴입니다.

7.2.3. 컴포넌트 버전

애플리케이션에서 로딩된 jar파일들을 보여줍니다.

에이전트는 기동된후 로딩된 클래스들의 jar파일을 검색하여 사용되는 jar파일 목록을 서버에 전송하고 이 정보를 조회합니다.

7.2.4. 쓰레드 목록/덤프

스레드 목록은 실행 중 쓰레드 현황을 표시합니다.

- 조회 결과
  - 스택 – 스택 정
  - 아이디 – 쓰레드 ID
  - 상태 – 쓰레드 상태
  - 이름 – 쓰레드 명
  - CPU – Cpu 총 사용시간(ms)입니다.
  - 차이 – Delta CPU입니다. 화면이 refresh될때 이전값과의 편차를 보여줍니다.

7.2.5. 로딩된 클래스

로드된 클래스는 애플리케이션 서버에 로딩된 클래스 현황을 제시합니다. ClassNotFoundException/NoClassDefFoundError와 같이 클래스 로딩과 관련된 예외 상황 발생 시 문제 유발 클래스가 로딩이 안 된 상태이거나 중복 로딩되었을 경우 본 화면의 정보를 통해 확인 할 수 있습니다.

[1130] | https://img.whatap.io/media/user_guide_application/agent/1130.png

- 데이터 칼럼
  - 이름 – 클래스 Full Path
  - 유형 – 클래스 타입 (C: class / I: Interface)
  - SUPERCLASS – 상위 클래스
  - 인터페이스 – 인터페이스 클래스
  - 자원 – 클래스 파일 경로
- 이미 로딩된 클래스를 재정의 하기 위한 팝업을 표시 : 실행 중인 애플리케이션 서버에 Java Attach 방식으로 와탭을 설치한 경우, 애플리케이션 서버가 제공하는 일부 모듈의 경우 바이트코드 변환이 적용되지 않아 모니터링 기능이 적용되지 않을 수 있습니다. 이러한 경우 클래스 재정의를 통해 바이트코드 변환을 유도 할 수 있습니다. 에이전트의 plugin 기능을 활용하는 경우에도, 애플리케이션 서버가 기동 된 상태에서 적용 할 경우, 플러그인이 적용될 대상 클래스를 재정의 하여 적용 할 수 있습니다.

[1140] | https://img.whatap.io/media/user_guide_application/agent/1140.png

- 클래스 시그니처 확인: 클래스 로딩 현황의 클래스 및 인터페이스를 클릭하면 클래스 시그니처를 확인 할 수 있습니다.

[1150] | https://img.whatap.io/media/user_guide_application/agent/1150.png

7.2.6. 소켓 오픈 집계

오픈 소켓은 애플리케이션 서버 실행 이후 아웃바운드 호출용으로 Open 되었던 Socket의 누적 정보를 제공합니다.

- 데이터 칼럼
  - 키 – 랜덤하게
  - 호스트 – 아웃바운드 소켓 접속 대상 호스트 IP 주소
  - 포트 – 아웃바운드 소켓 접속 대상 포트
  - 횟수 – 애플리케이션 소켓 접속 대상 포트를 open한 횟수
  - 서비스 – 호출 서비스 URL

[1160] | https://img.whatap.io/media/user_guide_application/agent/1160.png

7.2.7. 메소드 성능 상태

실시간으로 호출되는 메소드의 성능을 메모리에 집계하는데 이것을 조회하는 기능입니다. (자바 에이전트만 가능)

- 데이터 칼럼
  - 번호
  - 해쉬
  - 이름
  - 건수
  - 에러
  - 평균 수행 시간
  - 수행 시간의 합

특정 메소드의 호출 성능을 실시간 조회해 볼수있는 기능입니다.

7.2.8. DB상태

DB Connection Pool에 대한 속성정보들을 보여줍니다.

[ag 2] | https://img.whatap.io/media/user_guide_application/agent/ag_2.png

DB 풀 객체가 생성될때마다 에이전트 메모리에 등록되고 검색 버튼 클릭시 상세 정보가 조회됩니다.

7.2.9. 설정

에이전트의 설정을 수정합니다. 화면에서 입력한 설정들은 에이전트의 설정파일(whatap.conf)에 저장됩니다. 화면에서 값을 잃겨하는것과 whatap.conf를 편집하는 것은 동일 하게 동작합니다.

설정에 대한 키와 값은 에이전트 설명서를 참조합니다.

7.2.10. Throttling 설정

쓰로틀링은 클라이언트 동시 처리 갯수를 설정하고 초과되어 들어오는 요청은 reject하는 기능입니다. 이 쓰로들링에 관한 설정을 편리하게 할 수 있도록 분리 제공된 기능입니다.

쓰로틀링에 대한 자세한 설명은 [트랜잭션 쓰로틀링(Throttling)] 항목을 참조합니다.

7.2.11. 시스템 GC/힙 덤프

에이전트 Java VM에 System.gc()를 호출하는 기능입니다. 현재 Heap 사용량과 gc() 호출후 메모리 사용량을 보여줍니다.

또한 Java VM에 대해 힙덤프를 수행할 수있습니다.

힙덤프는 Java 6이후 JVM에서 사용할 수 있습니다.

7.2.12. 에이전트 로그

에이전트에 남겨지는 와탭의 로그를 조회합니다. 목록을 조회하고 선택한 로그파일의 내용을 확인합니다.

[ag 3] | https://img.whatap.io/media/user_guide_application/agent/ag_3.png

로그 리스트에서 로그명(1)을 클릭하면 로그 내용이 오른쪽 창에 나타납니다.

컨트롤 영역(<<,<,>,>>)(2)의 호출을 통해서 로그 내용을 확인 할 수있습니다.

7.2.13. 에이전트 덤프

에이전트는 진행중인 트랜잭션에 대한 정보를 덤프하는 기능이 있습니다. 이렇게 남겨진 덤프 내용을 확인하는 기능입니다.

에이전트 덤프는 설정에 의해 특정 상황이 되면 덤프를 남깁니다.

쓰레드 덤프는 모든 쓰레드를 덤프하지만 에이전트 덤프는 진행중인 트랜잭션 목록과 콜스택을 덤프합니다.

[ag 4] | https://img.whatap.io/media/user_guide_application/agent/ag_4.png

Chapter 8. 경고 알림

모니터링 임계치/조건을 지정하면 Email, SMS, 메신저, App Push 등 다양한 형태로 알림을 받을 수 있습니다.

8.1. 애플리케이션 AI 알림 설정

AI 모듈이 애플리케이션 실행 분포 패턴을 판단해 알람을 보냅니다. 복잡한 조건 설정 없이도 이슈 상황을 빠르고 정확하게 인지 할 수 있습니다.

[AI 알람] | https://img.whatap.io/media/images/ai_alert_6GUrM0Z.png

8.2. 이벤트 설정

임계치, 설정조건에 따른 알림 설정은 이벤트 | 이벤트 설정 메뉴에서 할 수 있습니다.

[list] | https://img.whatap.io/media/user_guide_application/alert/list.png

애플리케이션 비활성 경고

수집서버가 에이전트로부터 모니터링 정보를 수신하지 못하는 경우 애플리케이션의 다운 알림을 수신 할 수 있습니다. 애플리케이션 비활성 경고 > 이후 드롭다운 목록을 선택하여 몇 초간 모니터링 정보가 수집되지 않았을 때 알림을 수신 할 지 선택하고, Enable 체크박스를 체크합니다.

이벤트 알림 설정을 위해 활용 가능한 모니터링 항목과 설정 가능 값은 다음과 같습니다.

CPU

애플리케이션 서버의 CPU 점유율의 warning 임계치(%), fatal 임계치(%) 및 지속 시간(초)

디스크

디스크 사용률의 warning 임계치(%), fatal 임계치(%) 및 지속 시간(초)

메모리

메모리 사용률의 warning 임계치(%), fatal 임계치(%) 및 지속 시간(초)

히트맵 세로 라인

히트맵상의 같은 시간에 2.초 이상의 수직라인 발생시 전체시간 대비 차지하는 warning 비율(%), fatal 비율(%) 및 지속 시간(초)

액티브 트랜잭션

실시간 트랜잭션 수의 상한 또는 하한(부등식으로 지정)과 지속 시간(초)

트랜잭션 에러

실시간 에러 트랜잭션 수의 상한 또는 하한(부등식으로 지정)과 지속 시간(초)

메트릭스

이벤트 알림 설정의 다른 방법으로, 카테고리를 지정하여 알림의 레벨과 메시지를 등록하는 방식으로 선택된 카테고리에 맞는 조건과 일치했을 때 알림이 설정되는 방식입니다.

옵션 설정에 대해

- **반복** : 해당 이벤트 가 몇 초간 반복적으로 일어날 때 알림을 발생시켜야 하는지 시간(초)을 지정합니다.
  예를 들어 설명하면 CPU 70%가 30초 동안 유지될 경우 알림을 받고 싶은 경우 Repeat을 30sec로 설정합니다. 해당 설정 값이 단 한번이라도 발생시 알림을 받고 싶은 경우 Repeat을 데이터 수집주기인 5sec로 설정합니다.
- **무음** : 이벤트 알림 후 다음 이벤트 알림까지의 최소 대기 시간(초)를 지정합니다. 지나치게 빈번한 이벤트 알림을 방지하고 사용자가 이벤트의

심각성을 간과하게 될 가능성을 줄이기 위해서 적절한 대기 시간[Silent]을 설정해야 합니다.

8.3. 이벤트 수신 설정

이벤트 수신 설정은 경고 알림 | 알림 수신 설정 메뉴에서 할 수 있습니다.

[알람 수신 설정] | https://img.whatap.io/media/images/alert_recieve_setting.png

8.3.1. 사용자별 알람 수신 설정

원하는 알람 수단에 체크 하면 이벤트 알람을 받을 수 있습니다.
프로젝트 최고관리자를 제외한 모든 사용자는 자신의 수신 설정만을 변경 할 수 있습니다.

SMS 알람은 유료사용자에 한해 제공됩니다.

수신레벨

경고/위험 수준 모두 또는 위험 수준의 알람만을 수신하도록 설정 할 수 있습니다.

요일 / 시간

알림 수신을 요일별 / 시간별 수신여부를 설정 할 수 있습니다.

8.3.2. 3rd 파티 플러그인

추가하기 를 클릭하면 개인 채널 이외에 slack, 메신저로 알람을 수신할 수 있습니다.

[알람 외부 연동] | https://img.whatap.io/media/images/3rdparty_plugin.png

slack, telegram, teams, jandi 등 원하는 서비스별 화면안내에 따릅니다.
당사의 지원 범위에 포함되지 않는 사내 메신저 등은 표준 webhook, webhook json 을 통해 연동 할 수 있습니다.

8.3.3. 대량 알림 발생 방지

프로젝트에서 알람이 다량발생하면 일정 시간 동안 알람 송신이 중지됩니다.
송신 중지상태는 설정한 시간 이후에 자동 해소됩니다. 또는 차단 상황에서 나타나는 중단 해제 버튼을 클릭해 즉시 해제 할 수 있습니다.

기본값

5분 사이 10회 이상의 이벤트가 발생하면 3시간동안 알람 송신 중지

8.4. 알림 기록

경고 알림 | 알림 기록 메뉴를 통해 발생한 알림 이력을 조회 할 수 있습니다.
알림 이력은 최근 3개월 까지 제공합니다.

[알림 기록] | https://img.whatap.io/media/images/alert_history.png

Chapter 9. Advanced Feature

특정 상황에서 필요한 고급 기능들에 대해서 설명합니다.

9.1. Open API

수집중인 모니터링 정보를 별도로 활용하거나, 서버의 Scale Up/Out의 자동화를 위해 적용하고자 하는 경우 Open API를 통해 해당 정보를 추출할 수 있는 기능을 제공합니다. 관리 > 프로젝트 관리에서 프로젝트 정보 영역의 API Token 값을 Open API 호출 시 HTTP Header 정보로 전송하여 수집된 정보를 획득할 수 있습니다.

Application OPEN API 문서 를 참고 해 주세요.

9.2. 트랜잭션 쓰로틀링(Throttling)

쓰로틀링(처리 제한)기능은 클라이언트 요청을 제어하는 기능입니다.

9.2.1. 안정성 튜닝이란

시스템의 성능 개선을 튜닝이라고 하는데 튜닝에는 크게 처리량 튜닝과 안정성 튜닝이 있습니다. 처리량 튜닝은 최대 처리량을 늘려주는 튜닝이고 과부하 상태에서도 시스템이 다운되지 않도록 유지하는 것을 안정성 튜닝이라 합니다.

안정성 튜닝이 필요한 이유는 DB와 같은 백엔드 시스템에서 병목이 발생하면 애플리케이션 쪽의 자원을 아무리 추가해도 장애를 막을 수가 없습니다.

[thro 1] | https://img.whatap.io/media/user_guide_application/thro/thro_1.png

백엔드 시스템에서 용량 초과시 프론트 서버는 차례로 폭주하게 됩니다.

사용자요청이 시스템 처리 한계를 넘어서면 시스템 전체가 다운되거나 그와 유사한 상황에 빠질 수 있습니다. 부분적인 자원의 스케일 아웃으로는 문제가 해결되지 않습니다.

이런 경우에는 클라이언트(브라우저)에서 들어오는 요청을 제한(혹은 봉쇄) 해야 합니다.

[thro 2] | https://img.whatap.io/media/user_guide_application/thro/thro_2.png

Figure 95. 사용자/처리량 상관그래프

사용자가 늘어나면 오히려 처리량이 떨어질수 있습니다. 과부하 상태에서도 최대 처리능력을 유지해야합니다.

9.2.2. 서비스 쓰로틀링(호출제어)

와탭은 액티브 트랜잭션(동시에 처리중인 트랜잭션) 수가 임계치 기준 초과 시 안내 페이지로 요청을 전달하여 처리를 중단하는 기능을 제공합니다.

[thro 3] | https://img.whatap.io/media/user_guide_application/thro/thro_3.png

Figure 96. 쓰로틀링 처리 개념

Back-End 폭주는 Front 폭주로 전파됩니다. Back-End 폭주시에는 Front Was를 Scale Out 해도 장애를 막을 수 없습니다. Front Was에서 사용자 요청을 제어 해야합니다.

액티브 트랜잭션의 상황에 따라 초과 요청을 Static 안내 페이지로 강제 forward할 수 있습니다. 할인 행사를 대비하여 특정 페이지에만 쓰로틀링을 적용 할 수 있습니다.

- 폭주가 발생하는 사례
  - 급격한 사용자 폭주
  - 응용 프로그램 오류(루핑, 조건 누락 등)
  - 튜닝 되지 않은 SQL 배포, 테이블 인덱스 누락

9.2.3. 쓰로틀링 동작 개요

쓰로틀링의 모든 동작은 에이전트 옵션을 통해서 제어됩니다.

[thro 4] | https://img.whatap.io/media/user_guide_application/thro/thro_4.png

Figure 97. 쓰로틀링 처리 개념

쓰로틀링은 크게 봉쇄와 제한 두가지 개념으로 동작합니다. 봉쇄는 조건이 되는 모든 트랜잭션을 막는 것이고 제한은 동시 처리갯수를 초과하는 트랜잭션을 부분적으로 제한하는 기능입니다.

```
throttle_enabled=true
```

throttle_enabled옵션을 통해 쓰로틀링의 모든 기능에 대한 ON/OFF를 결정합니다.

9.2.4. 트랜잭션 봉쇄

특정 URL 혹은 특정 IP로부터 들어오는 트랜잭션을 봉쇄하는 기능합니다. robot으로 부터 들어오는 요청이나 잘못 배포된 URL 요청을 봉쇄하는데 사용됩니다.

```
throttle_blocking_url = /a/a.do,/a/b.do
throttle_blocking_ip = 10.0.0.1,10.0.0.2
```

블럭킹된 요청에 응답을 보낼 페이지를 지정할 수 있습니다.

```
throttle_blocked_forward=/blocked.html
```

안내 페이지가 지정이 안되면 안내메세지가 클라이언트에 리턴됩니다.

```
throttle_blocked_message={error: 'blocking'}
```

9.2.5. 트랜잭션 제한

동시처리 갯수를 제한하는 기능입니다.

동시 처리 갯수를 지정합니다.

```
throttle_limit=100
```

동시 처리되는 갯수 초과한 요청을 제한하는데는 2가지 정책이 있습니다.

등록된 URL만 제한

throttle_target_urls옵션이 지정되면 동작합니다. throttle_limit를 넘겼더라도 등록된 URL만 제한합니다.

등록된 URL만 통과

동시 처리되는 갯수 초과한 요청이라도 등록된 URL은 통과됩니다.

```
throttle_passing_url
throttle_passing_url_prefix
```

throttle_target_urls이 지정되면 throttle_passing_url은 무시됩니다.

9.2.6. 활용예

사례1

온라인 쇼핑몰에서 세일 이벤트를 했습니다. 특정 상품을 위한 이벤트 페이지를 만들고 행사를 진행했습니다. throttle_target_urls을 이용하여 해당 페이지만을 제한해서 장애없이 이벤트를 잘 마쳤습니다.

사례2

사례1과 비슷하게 온라인 쇼핑몰에서 전체적으로 사용량이 증가했습니다. 그래서 사용량을 제한해야하는데 결제관련 페이지는 통과 시켜야 했습니다. 이때 throttle_passing_url 옵션이 활용되었습니다.

9.2.7. Rejected 트랜잭션의 에러 메세지

트랜잭션이 Reject되면 트랜잭션 에러가 프로파일 정보에 표시됩니다.

[thro 5] | https://img.whatap.io/media/user_guide_application/thro/thro_5.png

Figure 98. Rejected Error

9.2.8. Reject 발생시 경고 처리

Reject 발생시 운영자에게 경고를 보낼수 있습니다.

```
reject_event_enabled=true
reject_event_interval = 30000
```

경고 발생 간격을 지정할 수있습니다. 기본값은 30초입니다.

[thro 6] | https://img.whatap.io/media/user_guide_application/thro/thro_6.png