

히트맵 트랜잭션 살펴보기

이 문서에서는 [와탭 모니터링 서비스](#)의 기능 중 [히트맵 트랜잭션](#) 메뉴를 활용해 웹 애플리케이션 서버(Web Application Server)의 문제를 파악하고 장애를 대응하는 방법을 소개합니다.

애플리케이션 모니터링 업무 담당자는 맡은 업무 분야에 따라 모니터링하고 싶은 요소가 다릅니다. 서버 담당자는 CPU, 메모리, 디스크, 네트워크 등의 리소스를 모니터링하며 자원을 어느 정도 사용하고 있는지 확인하길 원합니다. DB 담당자는 DB 쿼리 성능 향상을 위한 지표를 보길 원합니다.

반면에 전통적인 모니터링 방법에서 서비스(AP) 담당자의 경우 보통 힙(Heap) 사용량과 CPU 사용량을 확인할 뿐이었습니다. 하지만 이러한 방법으로는 서비스(AP)의 문제가 무엇인지 알 수가 없습니다. 서비스(AP)에서 가장 중요한 것은 사용자의 요청에 제대로 응답하는지, 얼마나 빨리, 오류 없이 응답하는지입니다. 이를 파악하기 위해 사용자의 요청이 제대로 수행되었는지를 확인하는 과정이 필요합니다. 사용자의 요청, 한건 한건을 Request라고 하고, 이 Request를 서버에서 처리하고 사용자에게 응답하는 과정을 **트랜잭션(Transaction)**이라고 정의합니다.

와탭 모니터링 서비스에서는 트랜잭션을 '진행 중인 트랜잭션'과 '종료된 트랜잭션'으로 구분해 서비스(AP)의 장애 현황을 파악할 수 있습니다. [애플리케이션 대시보드](#) 메뉴의 [히트맵](#) 위젯에서는 '종료된 개별 트랜잭션'을 분포도 형태의 차트로 확인할 수 있습니다. 히트맵 차트는 5초 단위로 종료된 트랜잭션 정보를 수집해 점 단위로 표현한 위젯입니다. 이 차트를 통해 우리는 1초 걸려야 하는 트랜잭션이 예상과 달리 2초 정도 소요되는 경우, 즉 응답시간이 2배 이상 소요되는 트랜잭션을 찾아 장애 원인을 분석할 수 있습니다.

패턴 분석

분포도 차트의 형태에 따라 어떤 장애가 발생한 것인지 확인하는 방법에 대해 알아보겠습니다.

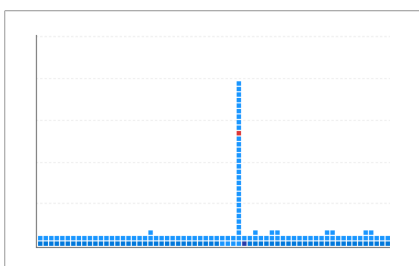
[히트맵 트랜잭션](#)은 시간의 흐름에 따라 사용자의 요청에 대한 응답시간을 분포도 형태로 표현한 차트입니다. [애플리케이션 대시보드](#)에서 히트맵 위젯의 오른쪽 위에 > 을 선택하면 [분석](#) > [히트맵](#) 메뉴로 진입할 수 있습니다.



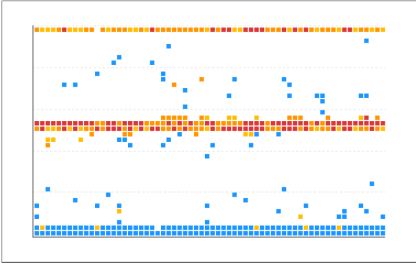
가로축은 트랜잭션의 종료시간, 세로축은 응답시간입니다. 트랜잭션을 종료한 시간마다 사용자의 요청(Request)에 대한 응답(Response) 시간을 차트 위에 사각형으로 표현합니다. 이를 통해 사용자의 요청에 정상적으로 응답했는지 파악할 수 있습니다. 히트맵 트랜잭션 차트에 사각형 상자의 색상은 다음과 같은 의미를 가지고 있습니다.

- **파랑색**: 정상 트랜잭션
- **주황색, 빨간색**: 에러가 발생했거나 응답 요청이 거부된 상태이며 에러 빈도에 따라 빨간색에 가까운 색상으로 표현됩니다.

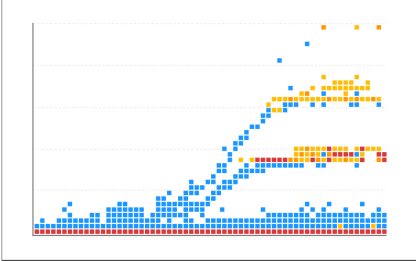
이 차트에서 가장 중요하게 살펴봐야 할 부분은 트랜잭션 상자가 가로 또는 세로로 줄지어 있게 되는 상황입니다. 다음 히트맵 패턴을 참조하세요.



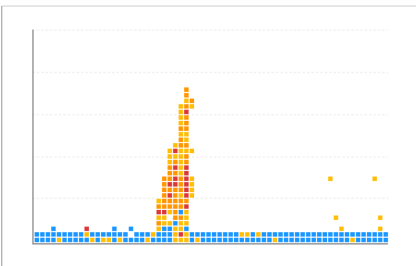
세로줄이 일시적으로 나타나는 패턴입니다. 세로줄이 발생하는 경우는 각 트랜잭션의 응답시간은 다르지만 종료시간은 같다는 것을 의미합니다. 트랜잭션 처리 중 일시적으로 락(Lock)이 발생하면 트랜잭션 처리를 대기합니다. 락(Lock)이 해소되면서 대기 중이던 트랜잭션은 비슷한 시간대에 한 번에 종료합니다. 이러한 현상으로 세로줄이 만들어집니다.



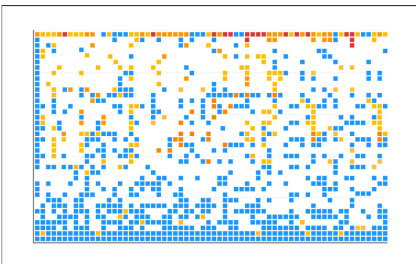
가로줄이 나타나는 패턴입니다. 10초 타임 아웃 조건에서 해당 자원이 부족하면 많은 트랜잭션은 10초 대기 후 타임 아웃 에러가 발생할 것입니다. 이때 히트맵 10초 부근에 가로줄이 발생합니다. 타임 아웃 이후 재시도하는 로직이 있다면 위 그림처럼 가로줄이 10초 단위로 반복합니다.



파도 치는 것처럼 보이는 플라잉 패턴은 특정 리소스나 로그와 같은 공통 자원 부족 현상으로 간격을 두고 나타나는 패턴입니다.

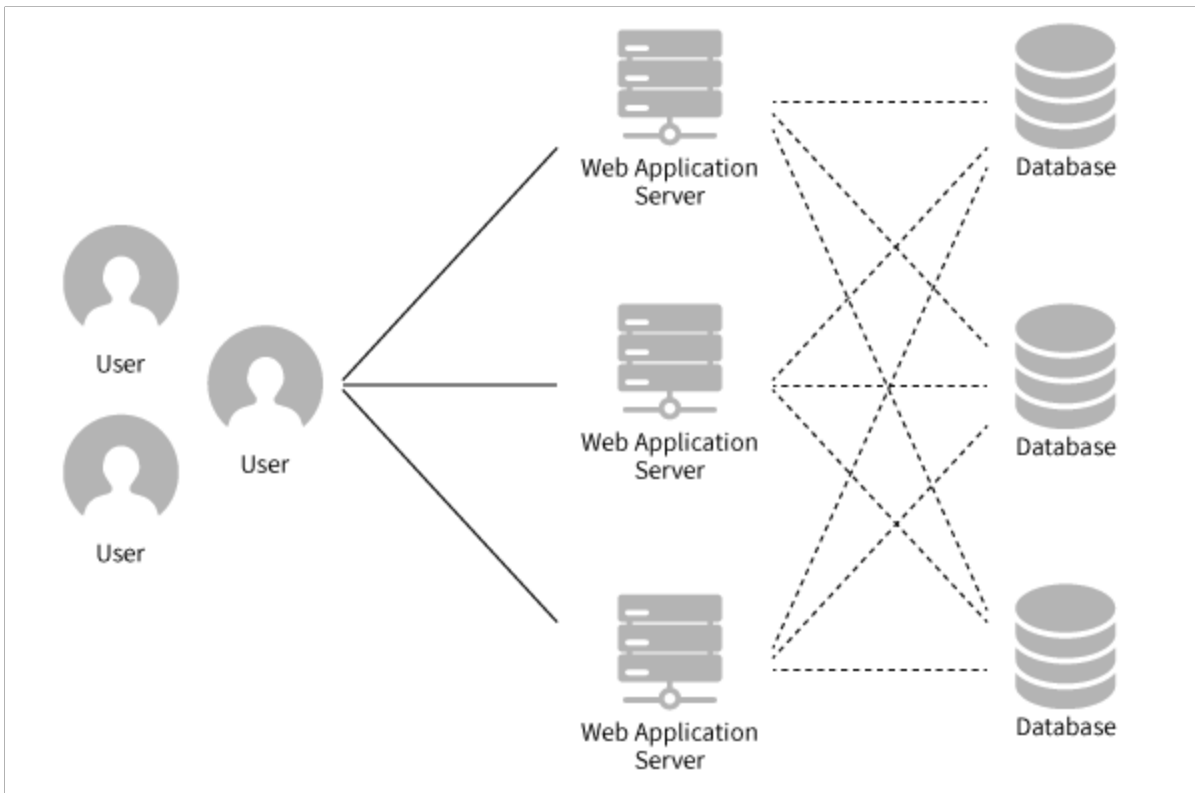


과부하 패턴은 전체 또는 일부 응답에 일시적으로 문제가 발생하는 트랜잭션이 한 번에 밀집할 때 나타나는 패턴입니다.



폭주 패턴은 과도한 트랜잭션 요청이나 부하가 발생할 때 응답시간이 전체적으로 증가하는 패턴입니다.

패턴이 발생했을 때 패턴을 발생시키는 요인이 서버의 내부인지 외부인지 파악하는 것이 중요합니다. 다음 그림과 같은 구조로 시스템을 설계했다고 가정하겠습니다.



차트의 아래 영역은 대부분 응답시간이 빠른 트랜잭션들이기 때문에 패턴을 찾는 것은 큰 의미가 없습니다. 위 영역의 느린 구간에서 만들어진 패턴의 공통점을 찾아보는 것이 필요합니다. 세로줄 패턴이 발생했을 때 서비스(AP)에서 락(Lock)이 발생했다면 하나의 웹 애플리케이션에서만 패턴이 발생할 것입니다. 반대로 외부와 연결된 Database에서 락(Lock)이 발생하면 모든 웹 애플리케이션에 패턴이 발생할 것입니다.

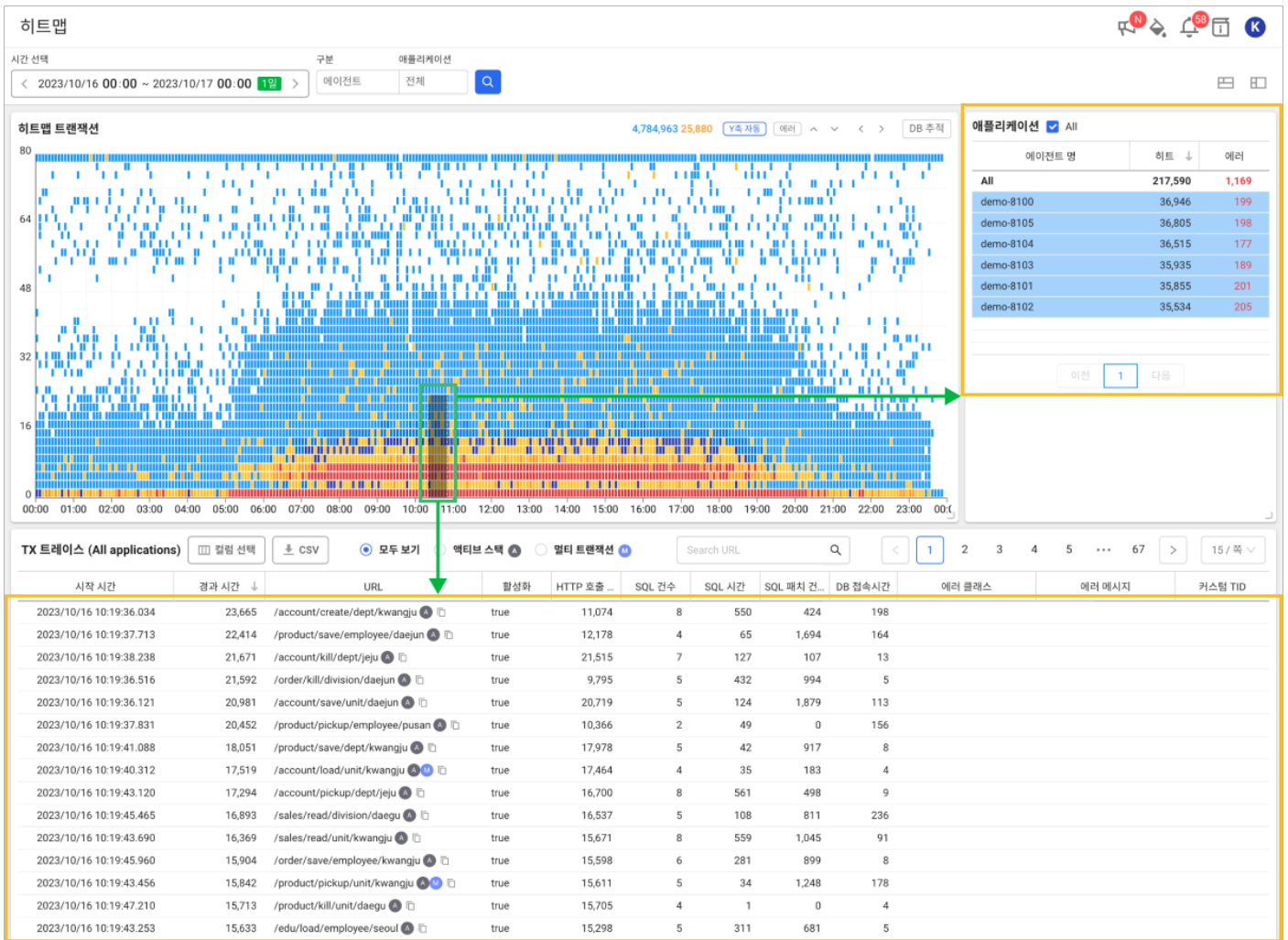


차트 영역을 드래그하면 드래그한 영역의 트랜잭션 정보를 화면 하단에 TX 트레이스 목록으로 불러올 수 있습니다. 오른쪽 애플리케이션 위젯에서는 에이전트를 그룹화해 목록으로 표시합니다.

장애 원인에 대한 공통점을 찾는 것이 목적이기 때문에 첫째 동일한 애플리케이션에서 문제가 발생했는지, 둘째 동일한 URL에서 문제가 발생했는지를 확인하세요. 그외 클라이언트 IP가 동일한지 살펴볼 수도 있습니다.

구간 및 스택 분석

장애를 발생하는 요인이 외부가 아닌 내부의 문제라면 어떻게 확인할 수 있을까요? 와탭은 **액티브 스택**이라는 기술을 이용해 실행 중인 트랜잭션의 스택 정보를 수집합니다. 스택 정보는 10초마다 수집하고, 수집한 데이터는 통계 정보로 확인할 수 있습니다. 여기에는 서비스의 로직을 수행하는 각 메소드, SQL, 외부 호출 정보도 포함합니다.

내부의 문제라면 각 트랜잭션의 수행 이력을 시간별로 확인하고 느려진 구간을 파악한다면 문제를 찾고 해결할 수 있습니다. 외부의 문제라면 SQL, 외부 호출 정보를 확인해보면 됩니다.

예를 들어, 세로줄 패턴에서 락(Lock)을 발생시키는 요인을 찾으려면 같은 패턴 내에서 가장 느렸던 트랜잭션과 가장 빨랐던 트랜잭션 사이의 접점을 확인해보는 것이 방법입니다. **히트맵 트랜잭션** 차트에서 패턴이 발생한 영역을 드래그하세요. **TX 트레이스** 목록으로 드래그한 영역의 트랜잭션 정보를 불러옵니다. **TX 트레이스** 목록을 경과 시간 기준으로 정렬한 다음 가장 오래 걸린 트랜잭션을 선택하세요. 선택한 트랜잭션에 대한 상세 정보가 담긴 **트랜잭션 정보** 창이 나타납니다.

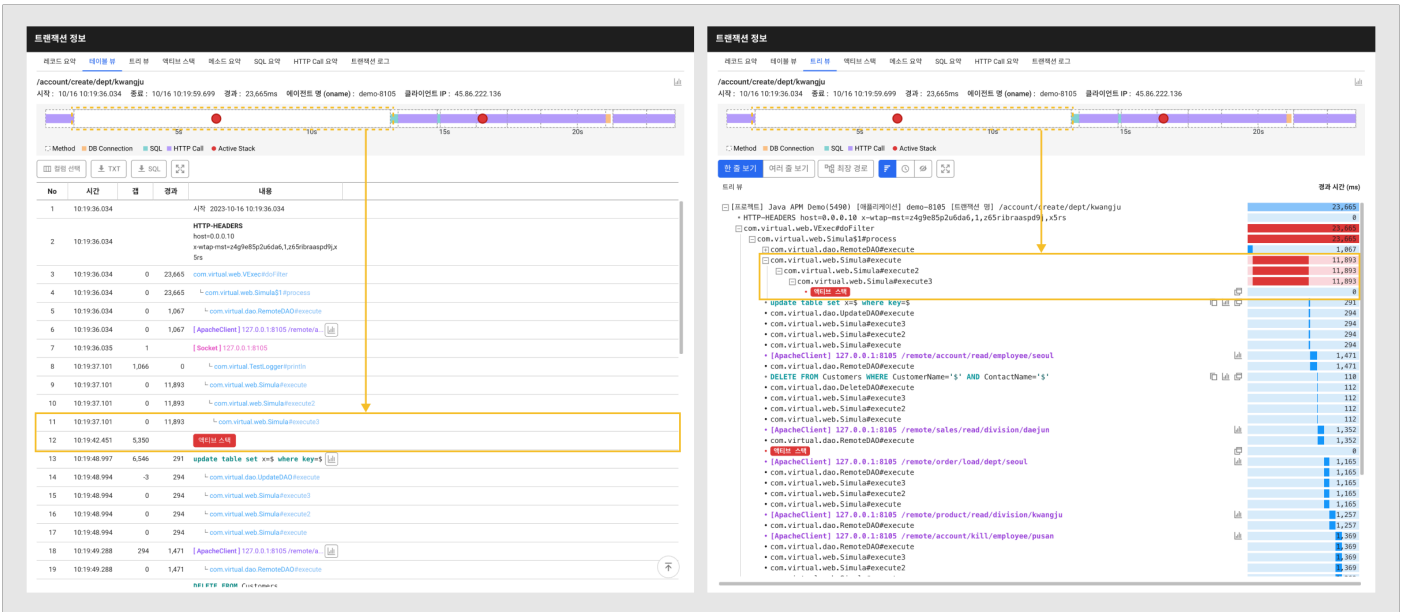
The screenshot shows the 'TX 트레이스 (All applications)' interface. At the top, there are filters for '모두 보기', '액티브 스택', and '일티 트랜잭션'. A search bar is also present. Below the filters is a table of transactions. The first row is highlighted in yellow, and a red circle highlights the '경과 시간' (Elapsed Time) column. An arrow points from this row to a detailed view window titled '트랜잭션 정보' (Transaction Info).

| 시작 시간 | 경과 시간 | URL | 활성화 | HTTP 호출 ... | SQL 건수 | SQL 시간 | SQL 패치 건... | DB 접속시간 | 에러 클래스 | 에러 메시지 | 커스텀 TID |
|-------------------------|--------|-------------------------------|------|-------------|--------|--------|-------------|---------|--------|--------|---------|
| 2023/10/16 10:19:36.034 | 23,665 | /account/create/dept/kwangju | true | 11,074 | 8 | 550 | 424 | 198 | | | |
| 2023/10/16 10:19:37.713 | 22,414 | /product/save/employee/daejun | true | 12,178 | 4 | 65 | 1,694 | 164 | | | |
| 2023/10/16 10:19:38.238 | 21,671 | /account/kill/dept/jeju | true | 21,515 | 7 | 127 | 107 | 13 | | | |
| 2023/10/16 10:19:36.516 | 21,592 | /order/kill/division/daejun | true | 9,795 | 5 | 432 | 994 | 5 | | | |

The detailed view window shows the transaction details for the selected transaction. It includes a timeline chart at the top, followed by a table of events. The events table has columns for 'No', '시간' (Time), '값' (Value), '경과' (Elapsed), and '내용' (Content). The events include HTTP headers, database connections, and SQL execution.

| No | 시간 | 값 | 경과 | 내용 |
|----|--------------|-------|--------|---|
| 1 | 10:19:36.034 | | | 시작 2023-10-16 10:19:36.034 |
| 2 | 10:19:36.034 | | | HTTP-HEADERS host=0.0.0.10 x-wap-mstz=42e85p2u6da6,1z65nbraaspdRjx Srs |
| 3 | 10:19:36.034 | 0 | 23,665 | com.virtual.web.VExecFilter |
| 4 | 10:19:36.034 | 0 | 23,665 | com.virtual.web.Simula\$1Process |
| 5 | 10:19:36.034 | 0 | 1,087 | com.virtual.dbc.RemoteDAO\$execute |
| 6 | 10:19:36.034 | 0 | 1,087 | [ApacheClient] 127.0.0.1:8105 /remote/a... |
| 7 | 10:19:36.035 | 1 | | [Socket] 127.0.0.1:8105 |
| 8 | 10:19:37.101 | 1,066 | 0 | com.virtual.TestLogger\$print |
| 9 | 10:19:37.101 | 0 | 11,893 | com.virtual.web.Simula\$execute |
| 10 | 10:19:37.101 | 0 | 11,893 | com.virtual.web.Simula\$execute2 |
| 11 | 10:19:37.101 | 0 | 11,893 | com.virtual.web.Simula\$execute3 |
| 12 | 10:19:42.451 | 5,350 | | 액티브 스택 |
| 13 | 10:19:48.997 | 6,546 | 291 | update table set x=y where key=y |
| 14 | 10:19:48.994 | -3 | 294 | com.virtual.dbc.UpdateDAO\$execute |
| 15 | 10:19:48.994 | 0 | 294 | com.virtual.web.Simula\$execute3 |
| 16 | 10:19:48.994 | 0 | 294 | com.virtual.web.Simula\$execute2 |
| 17 | 10:19:48.994 | 0 | 294 | com.virtual.web.Simula\$execute |
| 18 | 10:19:49.288 | 294 | 1,471 | [ApacheClient] 127.0.0.1:8105 /remote/a... |
| 19 | 10:19:49.288 | 0 | 1,471 | com.virtual.dbc.RemoteDAO\$execute |

테이블 뷰 탭에서 단계별 **시간**과 **경과** 시간을 확인할 수 있습니다. **시간**은 각 단계가 시작 또는 종료한 시간, **경과**는 각 메소드 시작부터 종료까지 총 소요 시간입니다. **값**은 앞서 실행한 메소드 사이의 간격입니다.



테이블 뷰 탭을 통해서 느려진 구간을 찾아볼 수 있습니다. 같은 방식으로 가장 빨랐던 트랜잭션에서도 느려진 구간을 확인하세요. 공통점을 발견할 수 있을 것입니다. 느려진 구간이 서비스(AP)의 로직에 의해 발생한 상황이라면 **액티브 스택**을 확인하세요. **액티브 스택** 버튼을 선택하면 상세 정보가 포함된 창이 나타납니다. **트리뷰** 탭에서는 **액티브 스택**과 앞서 실행된 메소드와의 상관 관계를 확인할 수 있습니다. 타임라인 바에서 해당 구간을 클릭하면 선택한 구간에 대한 정보를 펼쳐서 확인할 수 있습니다.

- ❗ **액티브 스택**은 10초에 한 번씩 스냅샷을 저장하고 있기 때문에 어느 구간에서든 확인할 수 있습니다. 응답시간이 길어질 수 있는 구간을 확인할 확률이 높아지기 때문에 **액티브 스택**을 통해서 구간이 벌어진 포인트를 찾아낼 수 있습니다.
- TX 트레이스** 목록에서 **액티브 스택**을 포함하는 트랜잭션은 **A** 아이콘을 표시합니다.

히트맵 차트에서 트랜잭션을 보는 것은 연관된 트랜잭션을 찾는 것이 목적입니다. 같이 참조해야 할 트랜잭션을 확인해 응답시간이나 종료시간에 영향을 주는 요소를 파악해야 합니다. 애플리케이션에서 모든 메소드를 추적하면 구간 분석이 가능하나 오버헤드를 일으켜 응답시간에 왜곡을 줄 수 있습니다. 그렇기 때문에 선별된 클래스 메소드만 추적해야 합니다. 그 선별 요점은 운영자의 관점에서만 결정하지만 장애를 일으키는 락(Lock)이나 다른 요소들이 추적에서 제외될 수 있습니다. 이 제외될 수 있는 부분을 보조하는 것이 **액티브 스택** 기술입니다.

와탭의 모니터링 서비스는 애플리케이션 서버를 운영하면서 통상적으로 반복되는 문제를 찾아내기 위해 반복적으로 **액티브 스택**을 스냅샷에 남깁니다. 장애를 일으키는 핵심적인 요인을 확률적으로 높게 찾아낼 수 있는 것입니다. 이렇게 수집한 자료를 통계적으로 확인할 수 있는 메뉴가 **분석 > 스택**입니다. 스택에 대한 자세한 내용은 [다음 문서](#)를 참조하세요.

- ① **트랜잭션 정보** 창의 **레코드 요약** 탭을 선택하면 트랜잭션에 대한 기본 정보를 조회할 수 있습니다. 클라이언트 IP, 사용 기기, 운영체제, 국가 정보 등을 제공합니다. 그 외 패치 건수 및 시간, SQL 건수 및 시간 등을 확인해 얼마나 많은 호출을 사용하는지도 파악할 수 있습니다.

| 트랜잭션 정보 | |
|---|--|
| 레코드 요약 | 테이블 뷰 트리 뷰 액티브 스택 메소드 요약 SQL 요약 HTTP Call 요약 트랜잭션 로그 |
| /account/create/dept/kwangju | |
| 에이전트 명 (oname) : demo-8105 | 에이전트 ID (oid) : -857948929 |
| 에이전트 그룹 명 : demo-okind-1 | 에이전트 그룹 ID : -628198688 |
| 에이전트 서버 명 : node-1 | 에이전트 서버 ID : 334634079 |
| 프로젝트 코드 : 5490 | |
| | |
| 트랜잭션 ID : -1267714563417967437 | 멀티 트랜잭션 ID : 0 |
| 시작 시간 : 23/10/16 10:19:36.034 | 종료 시간 : 23/10/16 10:19:59.699 |
| 경과 시간 : 23,665ms | HTTP 메소드 : GET |
| HTTP 호출 시간 : 11,074ms | HTTP 호출 건수 : 9 |
| DB 접속 시간 : 198ms | SQL 시간 : 550ms |
| SQL 건수 : 8 | SQL 패치 시간 : 6ms |
| SQL 패치 건수 : 424 | |
| | |
| CPU 사용 시간 : 6ms | 메모리 할당 사용량 : 889,664byte |
| | |
| 클라이언트 IP : 45.86.222.136 | 도메인 : 0.0.0.10 |
| WClientID : 6269270558407666527 | 국가 : GB |
| Referer : http://www.test.com/account/kill/division/daejun | 클라이언트 타입 : Other |
| 컨테이너 : agent.service | 운영체제 : Solaris |
| 컨테이너 키 : 1395943798 | 상태 : 200 |
| 도시 : Cardiff | 클라이언트 명 : Other |
| 유저 에이전트 : Mozilla/5.0 (X11; U; SunOS sun4u; en-US; rv:1.2.1) Gecko/20021212 | |

- 트랜잭션 정보에 대한 자세한 내용은 [다음 문서](#)를 참조하세요.

호출 관계 분석

메소드 요약

메소드 요약 탭을 선택하세요. SQL과 마찬가지로 메소드 또한 하나의 트랜잭션 안에서 얼마나 많이 호출됐는지 확인해볼 수 있습니다.

레코드 요약 테이블 뷰 트리 뷰 액티브 스택 **메소드 요약** SQL 요약 HTTP Call 요약 트랜잭션 로그

/account/save/unit/daegu 

| No | 클래스 | 메소드 | 파라미터/리턴 | 건수 | 합계 (ms) | 최대 (ms) |
|----|---------------------------|----------|----------------------------|----|---------|---------|
| 1 | com.virtual.dao.Remot... | execute | ()V | 2 | 3,044 | 1,980 |
| 2 | com.virtual.web.VExec | doFilter | (Ljavax/servlet/Servlet... | 1 | 4,056 | 4,056 |
| 3 | com.virtual.web.Simul... | process | ()V | 1 | 4,056 | 4,056 |
| 4 | com.virtual.TestLogger | println | (Ljava/lang/String;)V | 1 | 0 | 0 |
| 5 | com.virtual.web.Simula | execute | ()V | 1 | 952 | 952 |
| 6 | com.virtual.web.Simula | execute2 | ()V | 1 | 952 | 952 |
| 7 | com.virtual.web.Simula | execute3 | ()V | 1 | 952 | 952 |
| 8 | com.virtual.dao.Select... | execute2 | ()V | 1 | 952 | 952 |

이렇게 확인한 정보를 기반으로 반복적으로 실행되는 메소드가 있는지, 동일한 메소드가 반복적으로 호출되는지를 확인하고 개선점을 찾아볼 수 있습니다.

• **SQL 요약**

장애를 일으키는 요인이 내부 로직이 아니라 SQL인 경우를 살펴 보겠습니다. 하나의 트랜잭션 안에서 SQL의 수행을 통계적으로 확인해 보려면 **SQL 요약** 탭을 선택하세요.

레코드 요약 테이블 뷰 트리 뷰 액티브 스택 메소드 요약 **SQL 요약** HTTP Call 요약 트랜잭션 로그

/account/save/unit/daegu 

| No | 데이터베이스 | SQL | 건수 | 합계 (ms) | 최대 (ms) |
|----|----------------------------|------------------------|----|---------|---------|
| 1 | jdbc:mysql://localhost:... | SELECT DISTINCT ena... | 1 | 945 | 945 |

동일한 SQL을 얼마나 반복적으로 호출하는지를 확인하세요. 로직에서 호출하는 횟수를 줄이거나 다른 해결 방법을 찾아 DB의 부담을 줄일 수 있는 방법을 찾아야 합니다. DataBase의 부담을 줄여주는 것은 서버 운영을 위해 매우 중요합니다. 애플리케이션 서버는 Scale-Out이 가능합니다. 리소스를 지속적으로 늘리는 것이 가능합니다. 반면에 DataBase를 분할하는 것은 애플리케이션에 대해 엄청나게 많은 재개발을 수반합니다. 성능 튜닝의 기본은 가급적 애플리케이션에서 DataBase에 부담을 줄여주는 방향으로 진행해야 합니다.

• **HTTP Call 요약**


레코드 요약 테이블 뷰 트리 뷰 액티브 스택 메소드 요약 SQL 요약 **HTTP Call 요약** 트랜잭션 로그

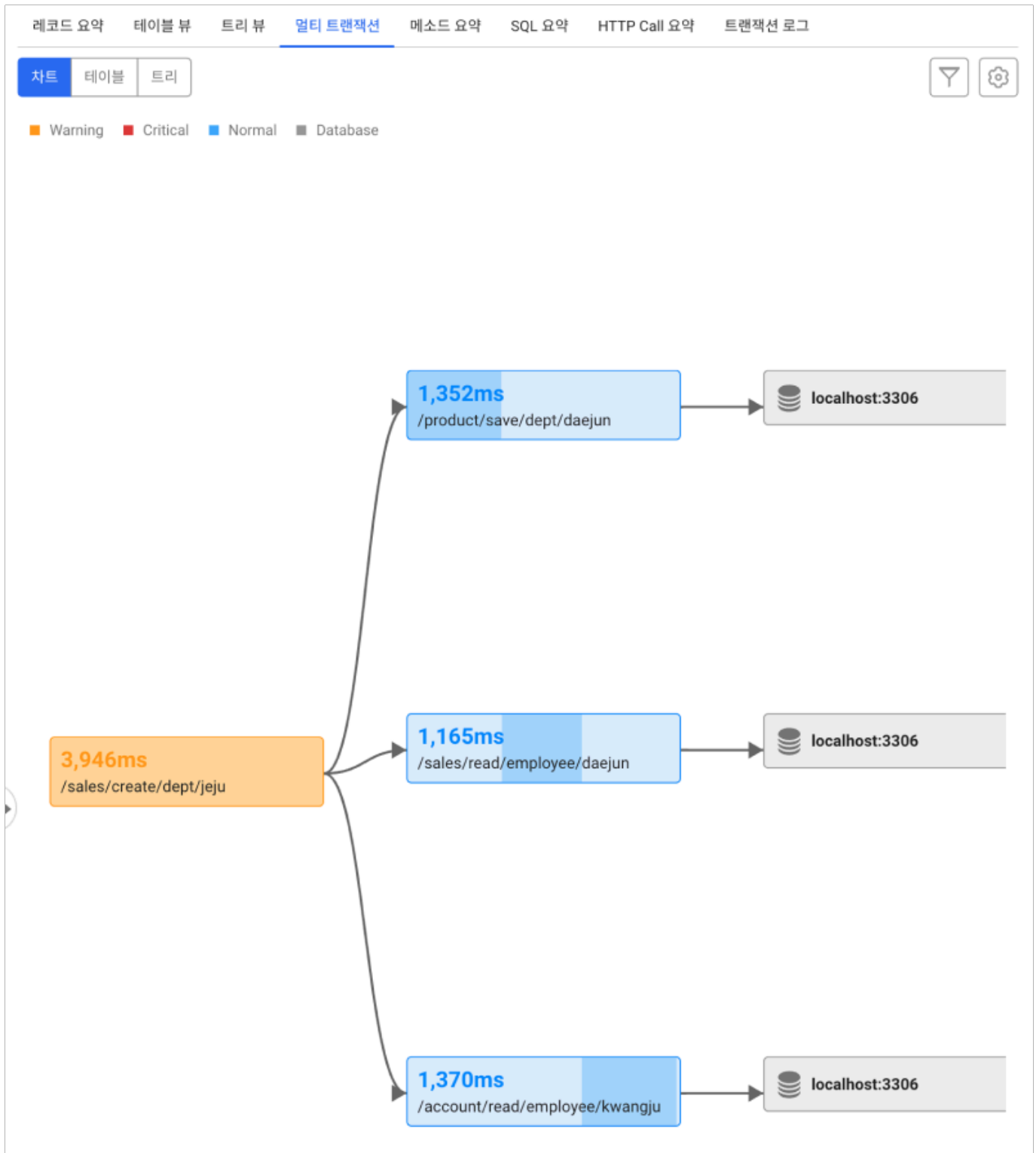
/account/save/unit/daegu 

| No | 원격 서버 | URL | 건수 | 합계 (ms) | 최대 (ms) |
|----|----------------|--------------------------|----|---------|---------|
| 1 | 127.0.0.1:8100 | /remote/order/pickup/... | 1 | 1,980 | 1,980 |
| 2 | 127.0.0.1:8100 | /remote/account/delet... | 1 | 1,064 | 1,064 |

[HTTP 호출 요약](#) 탭을 선택하세요. HTTP 호출의 호출 건수, 합계 시간, 평균 시간을 제공합니다.

- **멀티 트랜잭션**

모노리틱 환경에서는 단일 APM 구성이었습니다. 이 경우 해당 APM의 트랜잭션만을 확인하면 됩니다. 최근에는 마이크로서비스 아키텍처와 같은 기술 트렌드가 주류를 이루고 있습니다. 단일 APM이 작은 단위로 나뉘어지다 보니 수집해야 하는 트랜잭션도 그만큼 늘어납니다. 이런 경우 나누어진 트랜잭션을 모두 수집하는 것은 부하가 많이 걸리기 때문에 하나의 트랜잭션에 연계된 트랜잭션을 같이 수집하도록 합니다. 이와 같이 다른 APM과 연계된 트랜잭션을 와탭은 **멀티 트랜잭션**으로 정의합니다. [TX 트레이스](#) 목록에서 **멀티 트랜잭션**을 포함하는 트랜잭션은  아이콘을 표시합니다.



멀티 트랜잭션 탭을 선택하세요. 트랜잭션 별 연계 정보를 차트를 통해 확인할 수 있습니다. 테이블, 트리 버튼을 선택해 테이블 형식 또는 트리 구조 형식으로 트랜잭션 간의 호출 관계를 파악할 수 있습니다.

- ① ○ 멀티 트랜잭션 추적에 대한 자세한 내용은 [다음 문서](#)를 참조하세요.
- 트랜잭션에 따라 멀티 트랜잭션 탭을 지원하지 않을 수 있습니다.

트랜잭션 로그 분석

| 레코드 요약 | 테이블 뷰 | 트리 뷰 | 액티브 스택 | 메소드 요약 | SQL 요약 | HTTP Call 요약 | 트랜잭션 로그 |
|--|-----------|----------------------------|---|----------------------------|---------------------------------|----------------------------------|--|
| /account/save/unit/daegu | | | | | | | |
| <input type="text" value="키워드(를) 입력해주세요"/> <input type="button" value="Q"/> <input type="button" value="🔍"/> <input type="button" value="⚙️"/> | | | | | | | |
| ▶ | oname | 타임스탬프 | 로그 | | | | |
| ▶ | demo-8100 | 2023-08-01 11:57:24.293 | @txid 5501194250256405697 | pcode 5490 | oname demo-8100 | onodeName node-0 | http://127.0.0.1:8100/remote/order/pickup/dept/daegu status=200 |
| ▶ | demo-8100 | 2023-08-01 11:57:25.245 | @txid 5501194250256405697 | pcode 5490 | oname demo-8100 | onodeName node-0 | select distinct ename, deptno, sal, job from emp |
| ▶ | demo-8100 | 2023-08-01 11:57:25.279 | @txid 5501194250256405697 | pcode 5490 | oname demo-8100 | onodeName node-0 | select productmodelid, name |
| ▶ | demo-8100 | 2023-08-01 11:57:26.342 | @txid 5501194250256405697 | pcode 5490 | oname demo-8100 | onodeName node-0 | http://127.0.0.1:8100/remote/account/delete/employee/pusan status=20 |
| ▶ | demo-8100 | 2023-08-01 11:57:26.369 | @txid 5501194250256405697 | pcode 5490 | oname demo-8100 | onodeName node-0 | select ename, sal+1000 from emp |

트랜잭션 로그 탭에서는 해당 트랜잭션이 수행되는 동안 남겨진 로그를 통해 추가 개선할 사항을 찾아볼 수도 있습니다.

❗ 트랜잭션 로그에 대한 자세한 내용은 [다음 문서](#)를 참조하세요.

마무리

히트맵 트랜잭션을 통해 패턴을 확인하고 공통점을 파악합니다. 파악된 공통점을 통해서 메소드 구간을 분석합니다. 구간 정보가 부족한 경우 **액티브 스택**을 통해서 추가 정보를 확보하세요. 그 다음 SQL, 메소드, HTTP 호출 정보를 확인해 반복적이고 부담을 주는 로직에 대한 개선점을 찾아 보세요. 와탭 모니터링 서비스의 **히트맵 트랜잭션**을 통해서 모니터링 업무의 효율을 한 단계 끌어 올릴 수 있습니다.

❗ 같이 읽을 거리

- [월간 와탭: 애플리케이션 모니터링 프로파일 활용하기](#)
- [월간 와탭: 개별 트랜잭션 분포도를 보고싶다면?](#)
- [월간 와탭: 소중한 휴가를 지켜줄 와탭 머신러닝](#)
- [와탭 5분 세미나: 히트맵 가로 세로 라인 분석하기](#)