DB 연결 지연과 커넥션 풀

이 문서에서는 대표적인 애플리케이션 성능 장애 유형 중 하나인 **DB 연결 지연**(DBC 지연)과 **Connection Pool**의 상관관계를 모니터링 관점에서 소개하려 합니다.

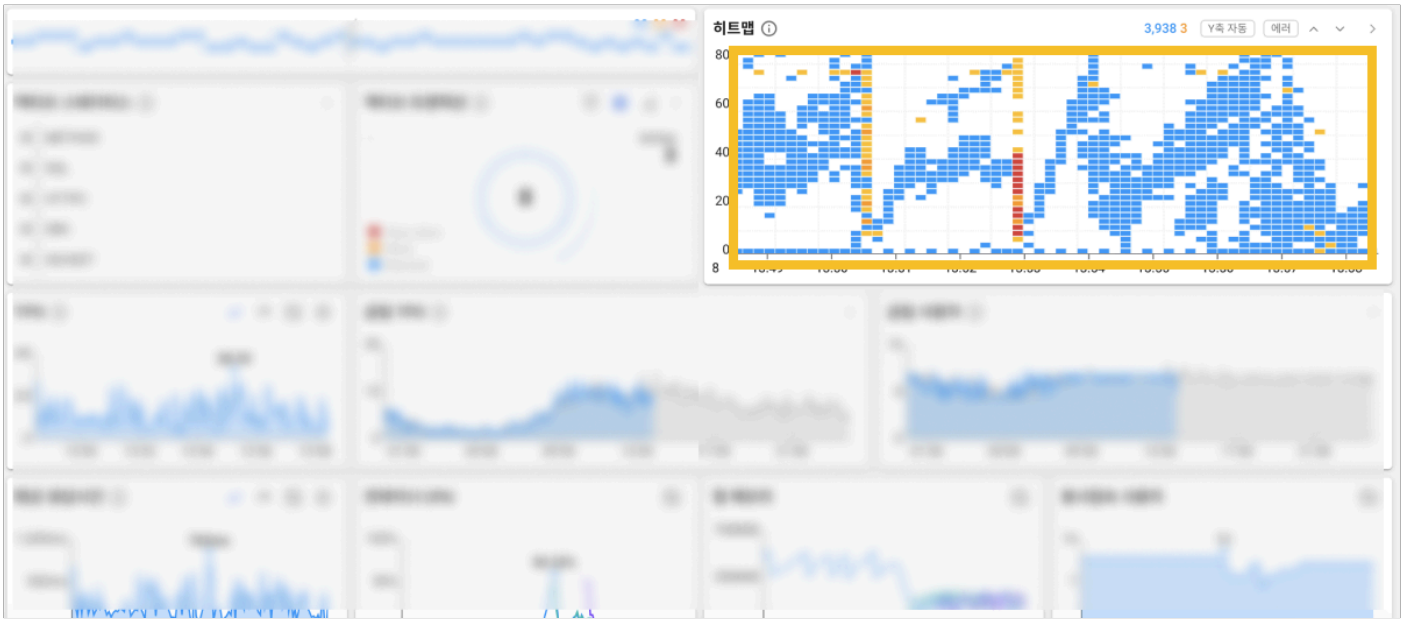
먼저 대시보드의 히트맵 위젯을 통해 1차적으로 장애 상황을 인지했다고 가정 후 분석 메뉴 하위의 히트맵 메뉴로 이동해 시간을 특정하는 방식으로 장애 상황 전후를 살펴보겠습니다. 이를 토대로 트레이스 정보 창과 메트릭스 차트 등을 통해 장애 원인을 추론하는 과정을 안내하겠습니다.

미리 알아보기

애플리케이션 대시보드 → 히트맵 트랜잭션 → 트레이스 정보 → 메트릭스 차트

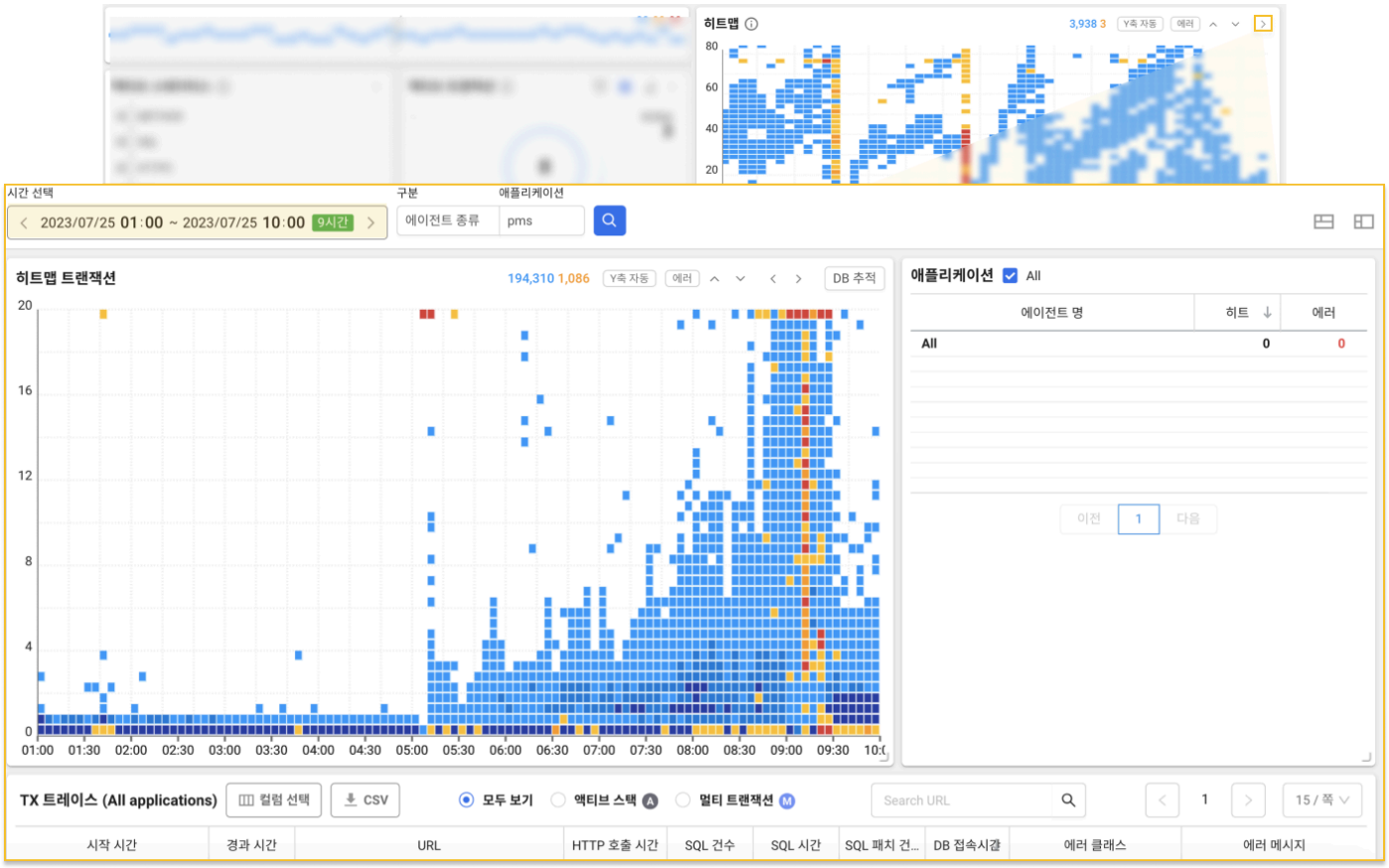
- 문제 현상을 인지했다면 과거 기록을 조회해 장애 현상 시점 전후를 조회합니다. 히트맵 트랜잭션 차트에서 트랜잭션 과부화 패턴 확인 후 TX 트레이스 목록을 통해 트랜잭션 지연 원인을 DB 연결 문제로 예상할 수 있습니다.
- 구간별 조회 후 트레이스 분석을 통해 상세 수행 이력을 스텝별로 살펴봅니다. 경과 시간을 가장 많이 소요하고 있는 것이 DBC 스텝임을 확인할 수 있습니다.
- 메트릭스 차트를 통해 다양한 지표를 함께 살펴봅니다. DB Pool 개수 차트와 DB Active 개수 차트를 비교해 DB Connection Pool의 부족이 원인임을 특정할 수 있습니다.
- 사용자 운영 환경에 따라 Connection Pool의 크기를 점차적으로 조절하거나 누수를 막기 위해 최적화 설정을 하는 방법을 활용할 수 있습니다.

히트맵 이상 패턴 감지



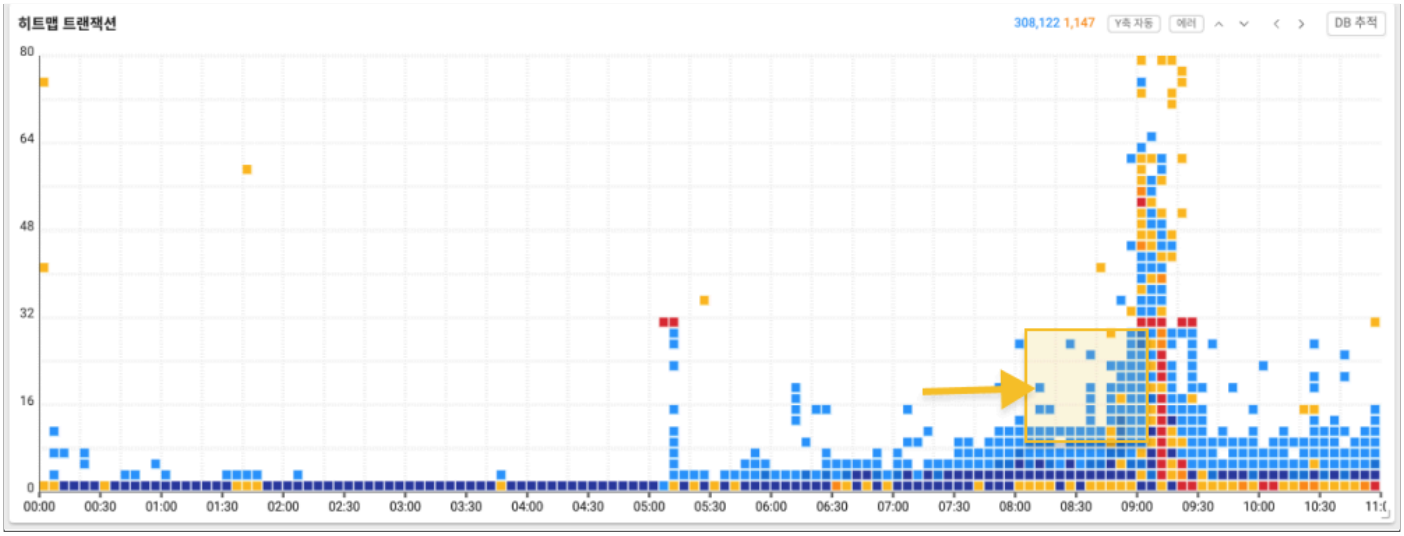
장애 현상을 인지하고 일차적인 분석을 통해 장애 원인을 추적하는 과정은 신속함이 핵심입니다. 와탭은 직관적인 뷰를 제공할 뿐만 아니라 근본 원인 추적을 위한 추가 정보를 확보할 수 있도록 과거 기록 조회 기능을 제공합니다. 애플리케이션 대시보드의 히트맵 위젯에서 예시처럼 급격한 이상 징후를 감지했다면 다음 순서는 무엇일까요?

히트맵 위젯에서 문제 구간을 드래그해 트랜잭션 트레이스 분석을 빠르게 진행할 수도 있고, 또는 분석 메뉴 하위의 히트맵으로 이동해 시간 범위를 보다 넓게 특정하는 방식으로 전후 상황을 함께 살펴볼 수도 있습니다.

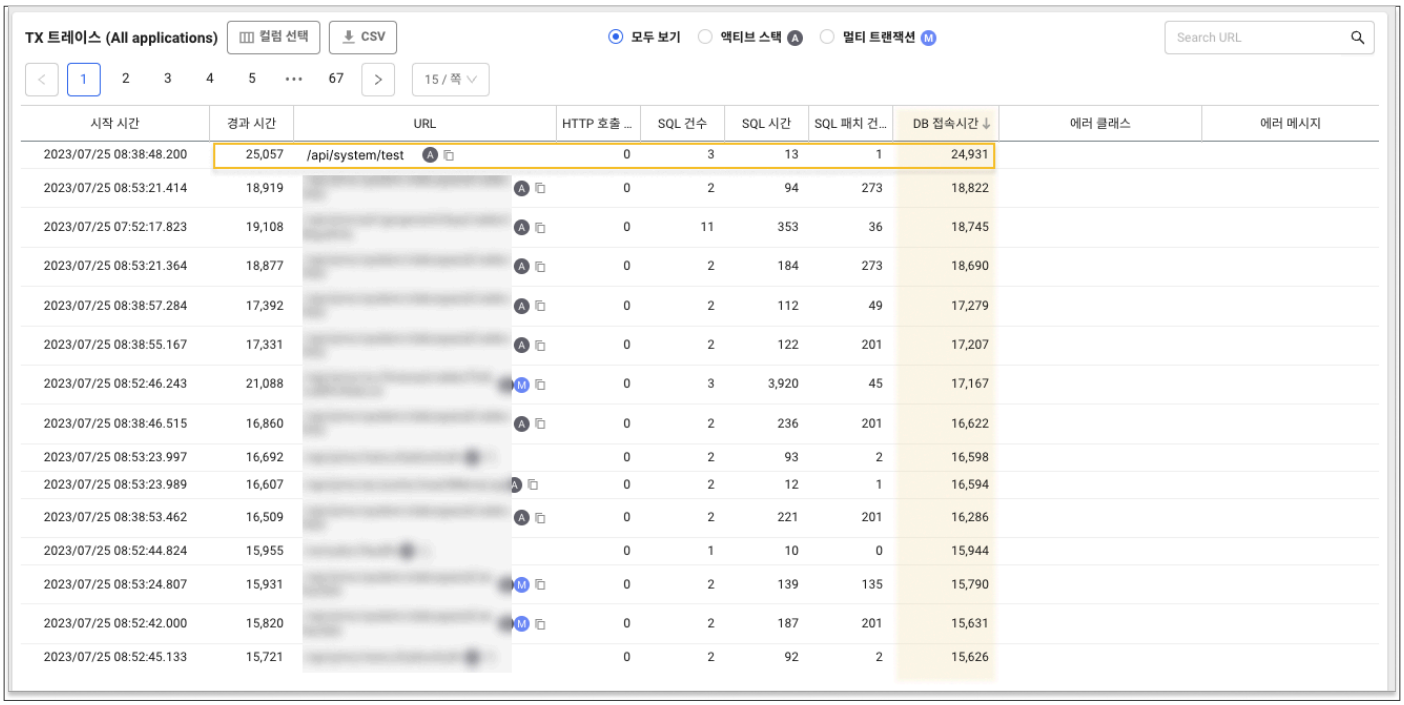


후자의 동선으로 진행해 보겠습니다. 대시보드 히트맵 위젯은 최근 10분간 종료된 트랜잭션 응답 시간 분포도입니다. 즉 시간 범위를 특정해 최근 10분 이상의 과거 내역을 확인하려면 분석 메뉴 하위의 히트맵 메뉴로 이동해야 합니다. 위젯 우측 상단의 › 화살표 아이콘을 선택해 예시와 같이 이동할 수 있습니다.

히트맵 트랜잭션 차트 상단의 시간 선택자를 통해 원하는 시간 범위를 지정해 과거 기록을 조회할 수 있습니다. 이상 패턴을 확정하기 위해 문제 구간 전후로 넓게 살펴보니 응답이 지연되고 있는 트랜잭션이 밀집한 과부하 패턴을 확인할 수 있었습니다.



이와 같은 갑작스러운 응답 시간 증가와 트랜잭션 과부하의 원인을 알아보기 위해 히트맵 트랜잭션 차트에서 문제 영역을 구간별로 드래그해 트랜잭션 트레이스 목록(TX 트레이스 목록)을 조회해 보겠습니다.



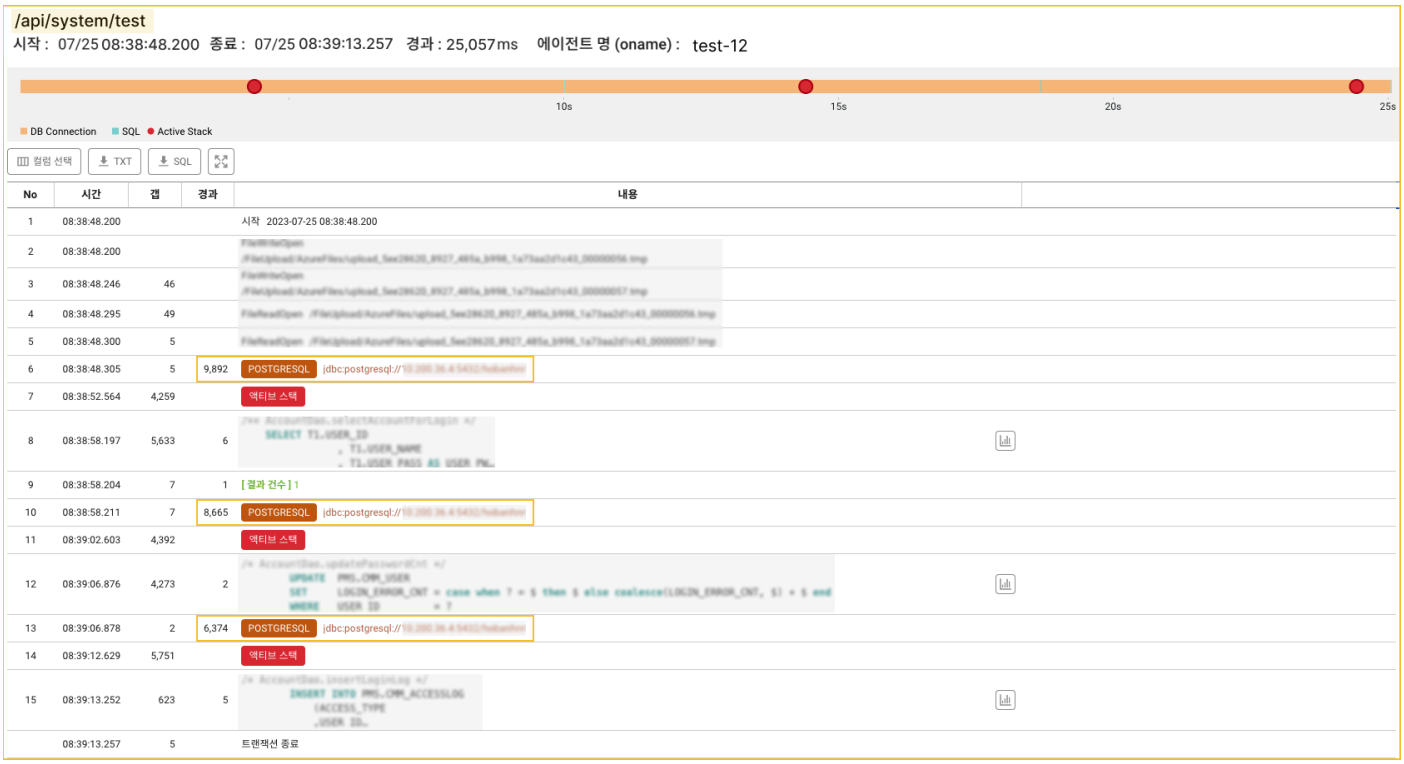
시작 시간	경과 시간	URL	HTTP 호출 ...	SQL 건수	SQL 시간	SQL 패치 건...	DB 접속시간 ↓	에러 클래스	에러 메시지
2023/07/25 08:38:48.200	25,057	/api/system/test	0	3	13	1	24,931		
2023/07/25 08:53:21.414	18,919		0	2	94	273	18,822		
2023/07/25 07:52:17.823	19,108		0	11	353	36	18,745		
2023/07/25 08:53:21.364	18,877		0	2	184	273	18,690		
2023/07/25 08:38:57.284	17,392		0	2	112	49	17,279		
2023/07/25 08:38:55.167	17,331		0	2	122	201	17,207		
2023/07/25 08:52:46.243	21,088		0	3	3,920	45	17,167		
2023/07/25 08:38:46.515	16,860		0	2	236	201	16,622		
2023/07/25 08:53:23.997	16,692		0	2	93	2	16,598		
2023/07/25 08:53:23.989	16,607		0	2	12	1	16,594		
2023/07/25 08:38:53.462	16,509		0	2	221	201	16,286		
2023/07/25 08:52:44.824	15,955		0	1	10	0	15,944		
2023/07/25 08:53:24.807	15,931		0	2	139	135	15,790		
2023/07/25 08:52:42.000	15,820		0	2	187	201	15,631		
2023/07/25 08:52:45.133	15,721		0	2	92	2	15,626		

TX 트레이스 목록을 통해 다수의 트랜잭션에서 경과 시간 대비 DB 접속 시간이 차지하는 비율이 큰 것을 확인할 수 있습니다. 예시로 `/api/system/test` URL의 경우 총 경과 시간 25,057ms 중 DB 연결에만 24,931ms가 소요되었습니다. 즉 일차적으로 DB 연결에 문제가 있다는 것을 추측할 수 있습니다.
다음으로 해당 트랜잭션을 선택해 트레이스 정보 창에서 상세 정보를 살펴보겠습니다.

> - 히트맵의 가로축은 트랜잭션 종료시간을, 세로축은 응답 시간을 의미합니다. 히트맵에서 화살표 아이콘을 선택해 세로축 응답 시간 지연 구간을 80초까지 조회할 수 있습니다.
> - 과부하 패턴 외에 히트맵 패턴 분석에 대한 자세한 내용은 다음 문서를 참조하세요.

트랜잭션 트레이스 분석

트레이스 정보 창에서 트랜잭션 상세 수행 이력을 스텝별로 확인할 수 있습니다. 이와 같은 트랜잭션 트레이싱을 통해 트랜잭션의 성능 저하 또는 오류의 원인을 추적할 수 있습니다. 트레이스 분석 창 조회 시 가장 먼저 살펴봐야 할 정보는 스텝별 경과 시간입니다. 소요 시간을 비교하는 것으로 성능 저하 또는 오류의 원인을 빠르게 특정할 수 있기 때문입니다.



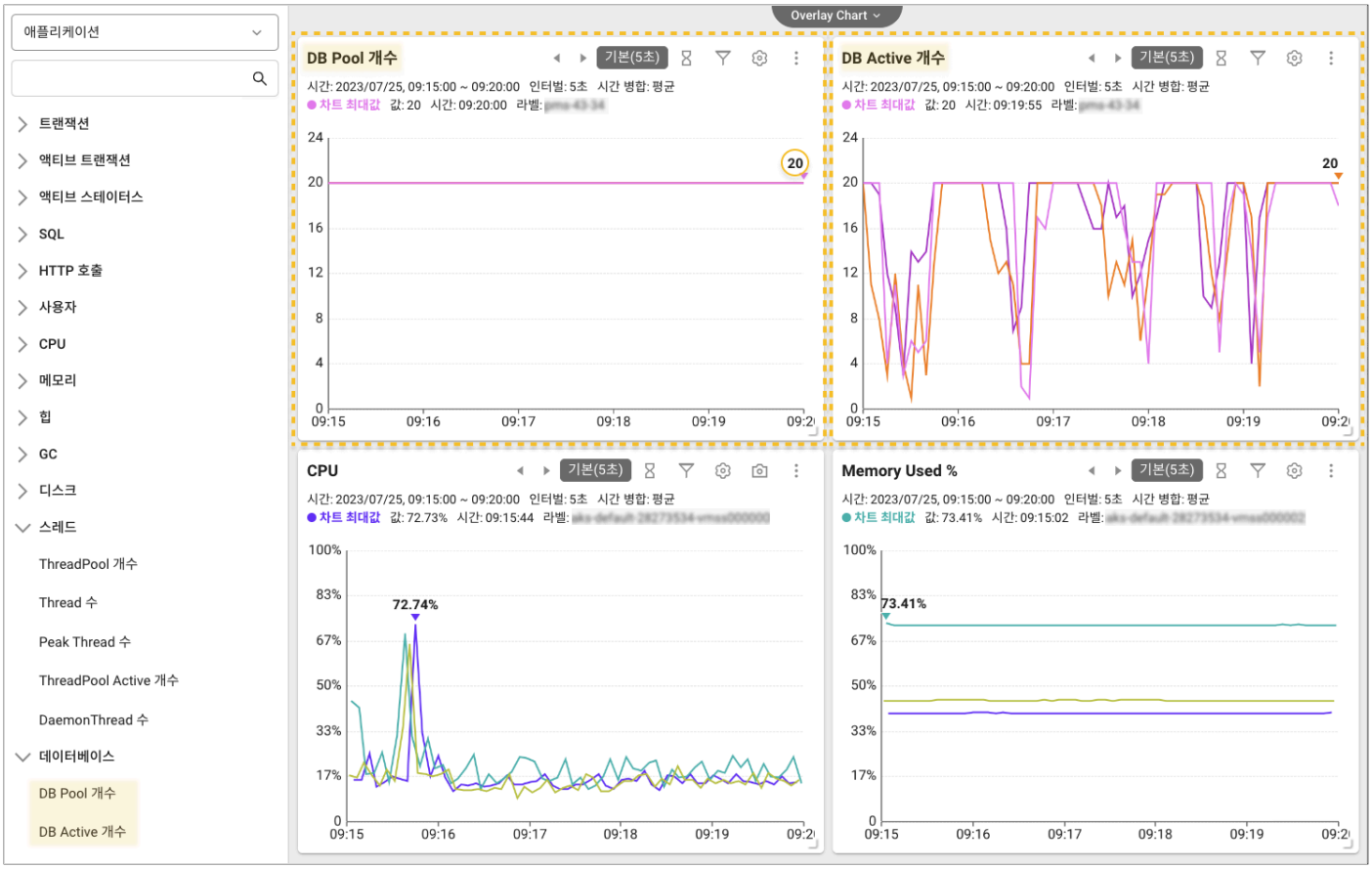
해당 URL의 상세 트레이스를 살펴보겠습니다. `/api/system/test` 수행 중 PostgreSQL DB에 접속해 처리하는 쿼리 수행 시간은 1ms에서 6ms 사이로 비교적 큰 문제가 없습니다. 하지만 해당 데이터베이스 접속 대기 시간이 그에 비해 확연히 오래 걸리고 있으며, 경과 시간을 가장 많이 소요하고 있는 것이 DB Connection 스텝임을 확인할 수 있습니다.

다시 말해 이 경우 앞서 예상한 대로 DB 연결 지연으로 인해 트랜잭션 지연이 발생했음을 특정할 수 있습니다. 그렇다면 이런 DB 연결 지연을 일으키는 원인은 무엇일까요?

DB Connection Pool 확인

DB 연결 지연을 일으키는 원인 중 하나는 **Connection Pool의 부족**입니다. Connection Pool의 크기는 동시에 처리 가능한 연결 개수를 의미합니다. 데이터베이스는 요청에 의한 Connection을 생성하고 종료하는 과정에 많은 시간을 소모합니다. Connection Pool은 미리 일정 수의 Connection을 생성해 두고 다른 요청이 있을 때 현재 Connection Pool을 다시 사용할 수 있도록 합니다. 이를 통해 매번 새로운 Connection을 만들 필요가 없어지기 때문에 응답 시간을 줄일 수 있습니다. 문제는 미리 생성해 둔 Connection 개수가 애플리케이션이 필요로 하는 Connection 개수에 비해 부족한 경우로 이때 DB 연결 지연이 발생하게 됩니다.

DB Connection Pool의 상황을 확인하기 위해 분석 메뉴의 메트릭스 차트로 이동합니다. 먼저 메트릭스 차트 좌측 목록에서 데이터베이스를 선택해 DB Pool 개수와 DB Active 개수 차트를 확인하겠습니다.



각 차트에서 확인할 수 있듯이 DB Pool의 최대 연결 개수는 20개로 설정되어 있습니다. DB Active 개수 차트를 살펴보면 20개 Pool에서 지속적인 점유가 이루어지고 있는 것을 확인할 수 있습니다. 미리 설정된 Pool을 초과하는 요청이 들어오면 DB에 남는 Pool이 생길 때까지 대기하게 되고 이에 따라 지연이 발생하게 됩니다. CPU 및 메모리 사용량 등 자원 현황 조회 시 별다른 특이사항을 찾을 수 없었습니다. 즉 해당 프로젝트의 경우 DB Connection Pool의 부족이 DBC 지연을 일으킨 결정적인 원인이라고 볼 수 있습니다.

그렇다면 이처럼 DB Connection Pool이 부족한 경우 어떻게 해야 할까요? 물론 해결 방법은 사용자 환경마다 다를 수 있습니다. 이 문서에서는 그 중 **Connection Pool 크기 조절**과 **Connection 재활용 및 최적화**를 통한 선 조치에 대해 간략하게 안내하겠습니다.

Connection Pool 크기 조절

먼저 Pool의 최대 연결 개수를 조절하는 방법을 활용할 수 있습니다. Pool을 점차 늘려 DB Active 개수가 Pool 최대치 내에서 유지되는 것을 확인해 적절한 Pool 개수를 설정하는 것입니다. 하지만 Pool의 개수를 지나치게 크게 지정한 경우 자원 소비량의 과도한 증가로 오히려 애플리케이션 성능에 악영향을 줄 수 있습니다. 충분한 메모리와 처리 능력을 갖추고 있는지 확인 후 각자의 운영 환경에 알맞게 조절해야 합니다.

Connection 재활용 및 최적화

두 번째로 Connection 재활용 및 최적화를 통한 방법을 활용할 수 있습니다. 예를 들어 애플리케이션이 사용한 Connection을 반환하지 않아 Connection Leak이 발생한 경우라면 아무리 Connection Pool의 개수를 늘린다고 해도 해당 현상을 해결할 수 없습니다. 이 경우 DB Active 개수는 우상향 그래프를 그리게 됩니다. Connection Leak을 막기 위해서는 사용한 Connection을 반드시 반환하도록 관리해야 합니다. Connection 사용 후 명시적으로 반환하도록 설정하거나 재사용하는 방식을 적용해 볼 수 있습니다.