

Performance, Load Testing

목표 Active User 산정

인증 요청		비밀번호 재설정 이메일 발송 요청		사용자 등록 요청	
2017-08-14 15:19:44	32	2014-02-06 14:25:03	10	2017-01-10 15:11:56	129
2019-04-17 11:42:24	31	2019-07-05 11:22:03	9	2019-10-30 14:30:12	90
2019-04-17 11:42:26	31	2013-11-22 16:52:45	8	2013-05-13 18:37:09	88
2019-04-17 11:42:25	30	2019-10-15 11:21:28	8	2014-01-22 01:47:10	78
2013-11-15 08:46:25	24	2019-10-15 11:21:29	7	2013-06-28 13:38:10	62
2016-06-13 15:49:03	23	2014-02-14 16:25:36	7	2019-09-17 08:17:25	55
2016-06-13 15:48:48	23	2019-10-15 17:41:35	7	2019-10-22 13:51:26	47
2016-06-13 15:49:05	21	2019-10-30 14:30:19	7	2013-07-25 12:02:45	42
2016-06-13 15:48:43	20	2013-12-12 10:24:22	7	2018-06-04 14:37:05	40
2016-06-13 15:49:00	20	2019-08-01 08:15:19	7	2019-08-28 08:16:57	36
2017-08-16 11:38:34	20	2019-10-30 14:30:17	7	2015-07-30 19:32:03	32
2016-06-13 15:48:58	19	2019-10-30 14:30:23	7	2017-01-20 11:43:16	28
2016-06-13 15:48:56	18	2019-10-30 14:30:22	7	2013-07-09 12:34:15	27
2016-06-13 15:48:47	17	2013-12-18 21:26:43	6	2013-05-13 19:11:41	27
2016-06-13 15:49:02	17	2019-10-30 14:30:24	6	2019-08-14 08:23:15	26
2017-08-14 15:19:34	17	2014-02-14 16:37:50	6	2019-11-07 15:33:32	25
2020-01-30 10:35:53	16	2014-02-06 12:10:26	6	2019-08-21 09:02:34	25
2016-06-13 15:48:44	16	2014-02-06 14:07:31	6	2015-03-02 11:12:32	25
2016-06-13 15:48:51	16	2019-09-17 08:17:27	6	2018-02-27 19:12:04	24
2016-06-13 15:48:42	15	2019-03-05 14:55:50	6	2016-09-01 13:41:42	24
2019-02-20 16:00:29	15	2019-10-15 17:41:37	6	2017-03-30 16:59:29	22
2014-07-29 13:37:27	14	2013-11-18 11:45:22	6	2020-02-26 14:24:54	22
2017-05-31 14:40:36	14	2013-08-28 15:01:53	6	2019-08-01 08:15:18	21
2016-06-13 15:48:55	14	2019-10-22 13:51:29	6	2019-11-26 16:13:14	21
2020-01-21 16:04:05	14	2013-10-22 19:48:49	6	2019-09-02 09:10:18	21
2019-02-20 16:00:28	14	2019-10-30 14:30:13	6	2015-03-02 11:14:38	20
2016-06-13 15:49:04	14	2019-08-14 08:23:16	6	2016-06-15 12:04:57	20
2019-02-20 16:00:27	14	2013-08-09 15:31:03	6	2019-10-15 11:21:27	20
2019-04-17 11:42:27	14	2014-02-05 10:44:06	6	2019-07-24 11:23:34	20
2016-06-13 15:48:57	13	2013-09-09 14:08:58	6	2014-03-03 14:13:15	20
2019-02-20 16:00:30	13	2019-10-30 14:30:15	6	2015-03-02 11:23:40	20
2016-06-13 15:49:06	13	2019-10-30 14:30:25	6	2018-05-23 13:27:28	20
2020-01-30 10:35:48	13	2019-10-30 14:30:26	6	2016-12-09 09:57:12	20

〈현행 시스템에 쌓인 로그를 통해 시간 초 별 동시 요청 개수를 집계하여 내림차순 정렬한 결과〉

위 데이터 중 "사용자 등록 요청"의 상위 집계 결과의 대부분 경우가 엑셀 업로드를 통한 사용자 일괄 등록에 해당하여 동시 요청 계산에서 제외.

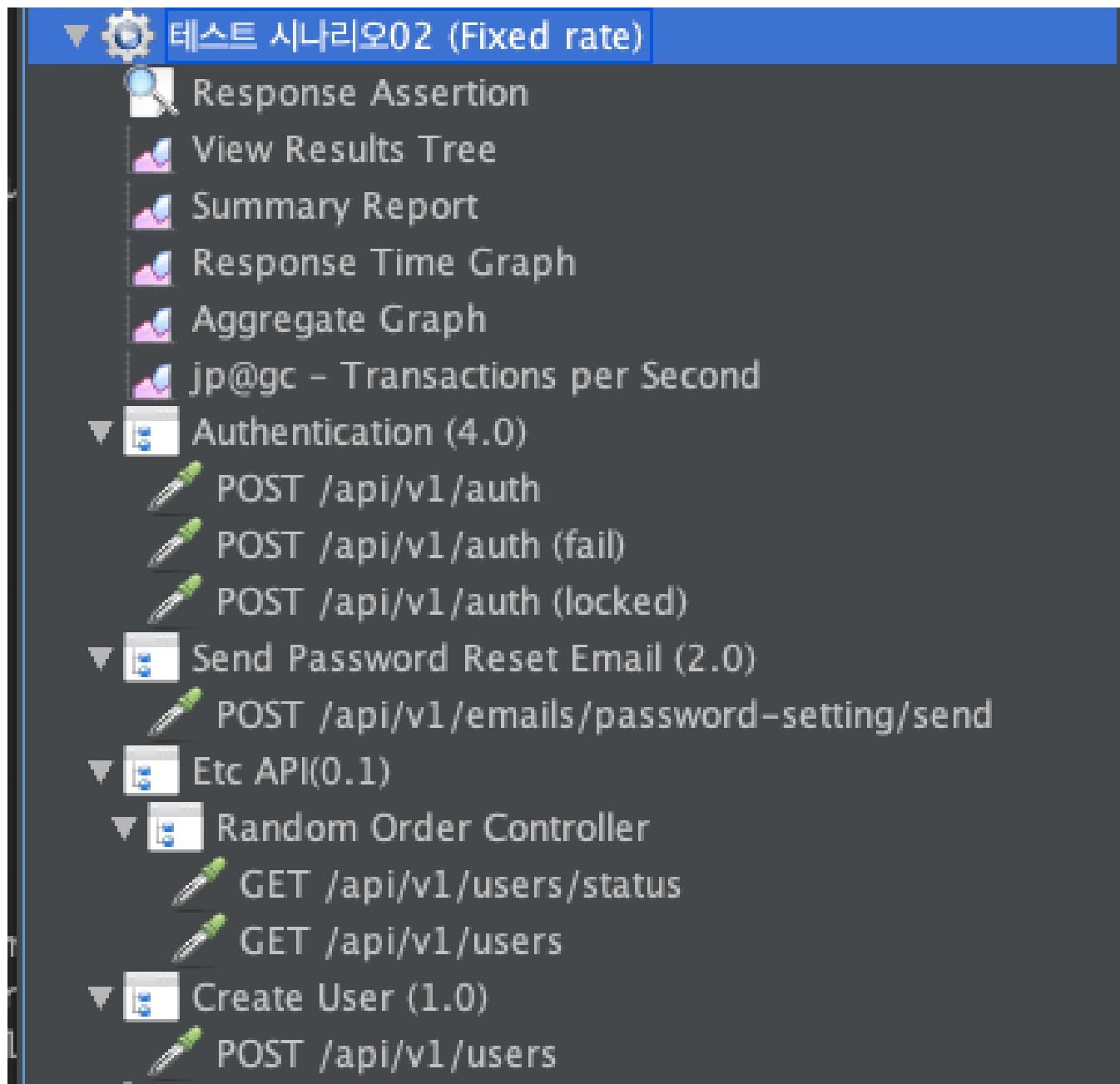
단위 시간 당 "인증 요청"의 최대 횟수인 32와 "비밀번호 재설정 이메일 발송 요청"의 최대 횟수인 10을 합한 후 기타 API의 동시 요청을 20으로 추측하여

시스템에서 목표로 하는 Active User의 수는 60명으로 한다.

테스트 시나리오

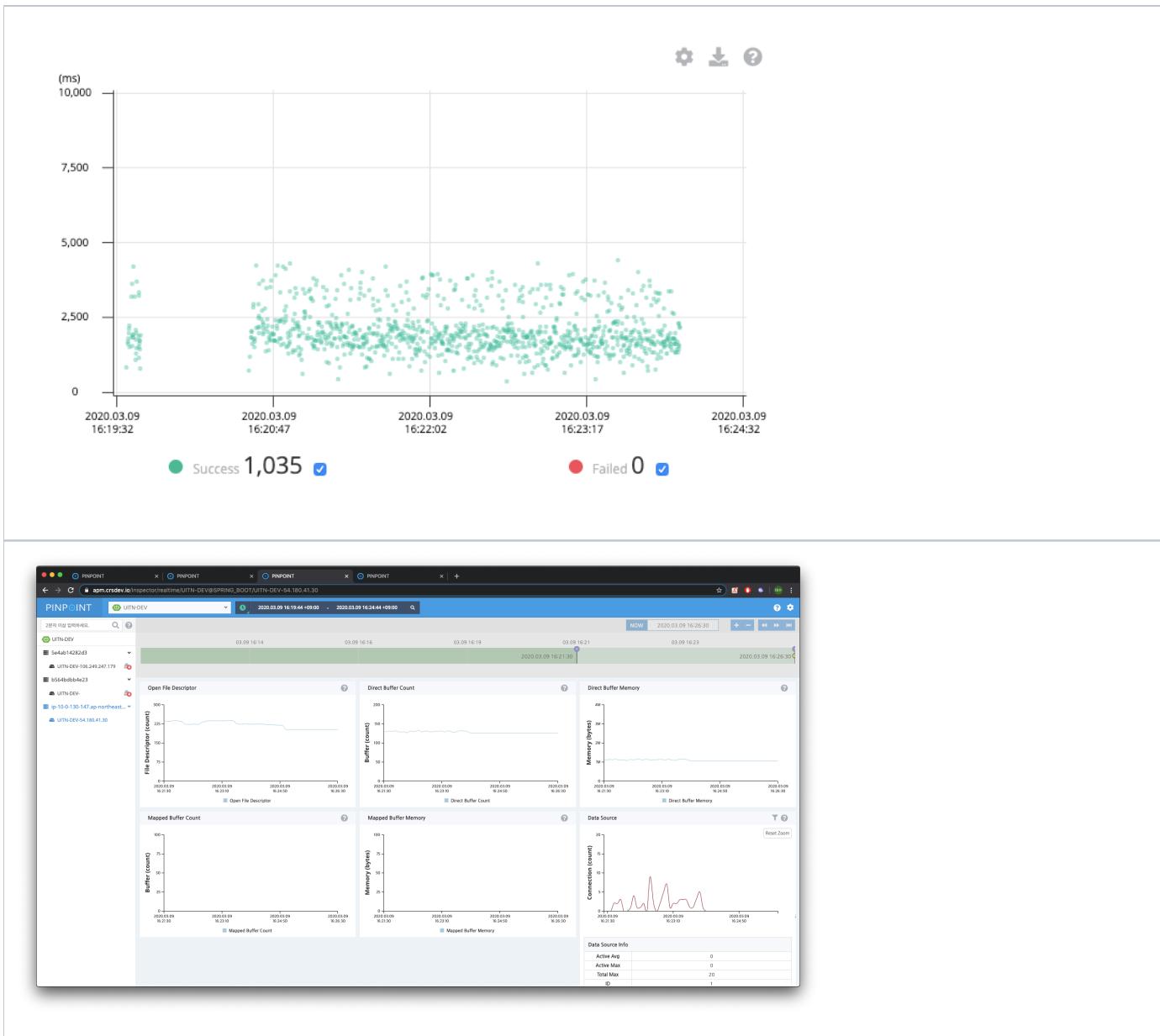
테스트 시나리오는 요청 횟수의 비율을 유지하여 아래와 같이 구성한다.

API	항목	비율
사용자 인증	<ul style="list-style-type: none"> 사용자 인증 성공 사용자 인증 실패 (비밀번호 오류) 사용자 인증 실패 (잠김) 	40%
비밀번호 재설정 메일 발송	<ul style="list-style-type: none"> 비밀번호 재설정 메일 발송 	20%
사용자 생성	<ul style="list-style-type: none"> 사용자 생성 (Sequence를 통해 항상 성공케이스로) 	10%
기타	<ul style="list-style-type: none"> 사용자 상태 조회 사용자 조회 	10%



Active User 10

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec
1차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3.medium



테스트 결과

executeRootHandler(HttpHandler handler, HttpServerExchange exchange)	15:16:5 / 116	0	4,207	
` doPost(HttpServletRequest request, HttpServletResponse response)	15:16:57 117	1	4,206	
initBinder(WebDataBinder binder)	15:16:59 217	2,100	0	
authentication(HttpServletRequest httpServletRequest, Authentication)	15:16:59 217	0	2,105	
getConnection()	15:16:59 217	0	1	
` executeUpdate(String sql)	15:16:59 218	0	23	
` SQL	set autocommit=0			
` prepareStatement(String sql)	15:16:59 251	10	0	
` SQL	select user0_.user_key as user_key1_16_0_, userl...			
` executeQuery()	15:16:59 251	0	3	
` SQL	select user0_.user_key as user_key1_16_0_, userl...			
` SQL-BindValue	taesu@crscube.co.kr, taesu@crscube.co.kr			
` execute(String sql)	15:16:59 257	3	6	
` SQL	COMMIT			
` executeUpdate(String sql)	15:16:59 263	0	53	

Handler method 진입 전에 gap 타임이 매우 길게 잡혀있으며 전체 요청 처리 시간 (4207ms)의 반 이상을 차지함(2100ms)

Handler method 이전에 실행되는 Interceptor의 특정 로직에서 Thread가 긴 작업에 의해 Block이 될 가능성이 클 것으로 예측.

```

@Override
public boolean preHandle(HttpServletRequest req, HttpServletResponse re, Object handler) throws Exception {
    final String authorization = req.getHeader("Authorization");
    if (Strings.isBlank(authorization) || !authorization.toLowerCase().startsWith("basic")) {
        req.setAttribute("exception", new InvalidClientException("Client authentication required"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    String base64Credentials = authorization.substring("Basic".length()).trim();
    byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);
    String credentials = new String(credDecoded, StandardCharsets.UTF_8);
    final String[] values = credentials.split(":", 2);

    Client client;
    try {
        client = this.clientRetrieveService.retrieveById(values[0]);
        if (!this.passwordEncoder.matches(values[1], client.getSecret())) {
            throw ResourceNotFoundException.from(values[0]);
        }
    } catch (ResourceNotFoundException e) {
        req.setAttribute("exception", new InvalidClientException(values[0] + " is Invalid client"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    return true;
}

public class ClientRetrieveService {
    private final ClientRepository clientRepository;

    @Cacheable(value="client")
    public Client retrieveById(String clientId) {
        return this.clientRepository.findById(clientId).orElseThrow(
            () -> ResourceNotFoundException.from(clientId)
        );
    }
}

```

Select의 횟수를 줄이고자 ClientRetrieveService의 retrieveById 메소드에 캐시가 적용되어있었으며 Cache의 내부 구현이 ConcurrentHashMap임을 확인하여 병목 현상의 지점이 될 것으로 예상하여 Cache 어노테이션을 제거 후 다음 테스트 수행.

차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
2차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3. medium	Client 조회 로직에 Cache 제거

```

@Override
public boolean preHandle(HttpServletRequest req, HttpServletResponse re, Object handler) throws Exception {
    final String authorization = req.getHeader("Authorization");
    if (Strings.isBlank(authorization) || !authorization.toLowerCase().startsWith("basic")) {
        req.setAttribute("exception", new InvalidClientException("Client authentication required"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    String base64Credentials = authorization.substring("Basic".length()).trim();
    byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);
    String credentials = new String(credDecoded, StandardCharsets.UTF_8);
    final String[] values = credentials.split(":", 2);

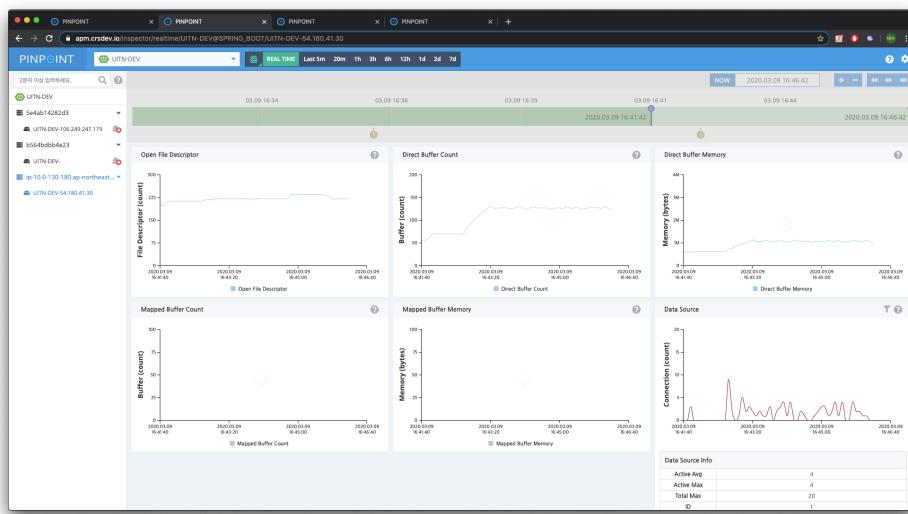
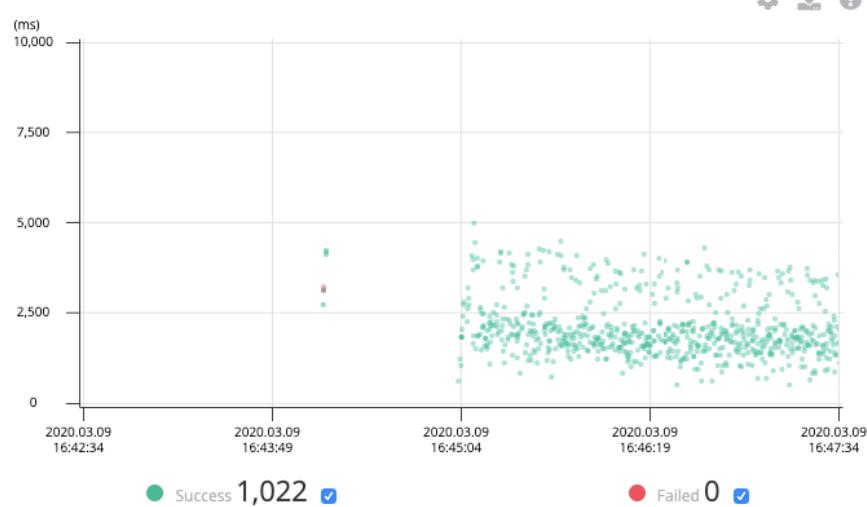
    Client client;
    try {
        client = this.clientRetrieveService.retrieveById(values[0]);
        if (!this.passwordEncoder.matches(values[1], client.getSecret())) {
            throw ResourceNotFoundException.from(values[0]);
        }
    } catch (ResourceNotFoundException e) {
        req.setAttribute("exception", new InvalidClientException(values[0] + " is Invalid client"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    return true;
}

public class ClientRetrieveService {
    private final ClientRepository clientRepository;

    public Client retrieveById(String clientId) {
        return this.clientRepository.findById(clientId).orElseThrow(
            () -> ResourceNotFoundException.from(clientId)
        );
    }
}

```



executeRootHandler(HttpServletRequest handler, HttpServerExchange exchange)	16:45:21 819	0	4,011	
` doPost(HttpServletRequest request, HttpServletResponse response)	16:45:21 819	0	4,010	
getConnec` doPost(HttpServletRequest request, HttpServletResponse response)	16:45:21 820	1	0	
` prepareStatement(String sql)	16:45:21 820	0	0	
SQL select client0_client_key as client_k1_0_, client0_...				
` executeQuery()	16:45:21 820	0	2	
SQL select client0_client_key as client_k1_0_, client0_...				
SQL-BindValue SAFETYAPP_API_STG				
initBinder(WebDataBinder binder)	16:45:23 568	1,746	0	
` authentication(HttpServletRequest httpServletRequest, Authenti...)	16:45:23 568	0	2,260	
getConnection()	16:45:23 622	54	17	
` executeUpdate(String sql)	16:45:23 639	0	1	
SQL set autocommit=0				
` prepareStatement(String sql)	16:45:23 650	10	0	

Cache 어노테이션을 제거하였기 때문에 Handler method 실행 전에 Client 관련 조회하는 쿼리가 발생 된 것을 확인.

여전히 1746~2000ms 범위의 gap time이 존재함을 확인.

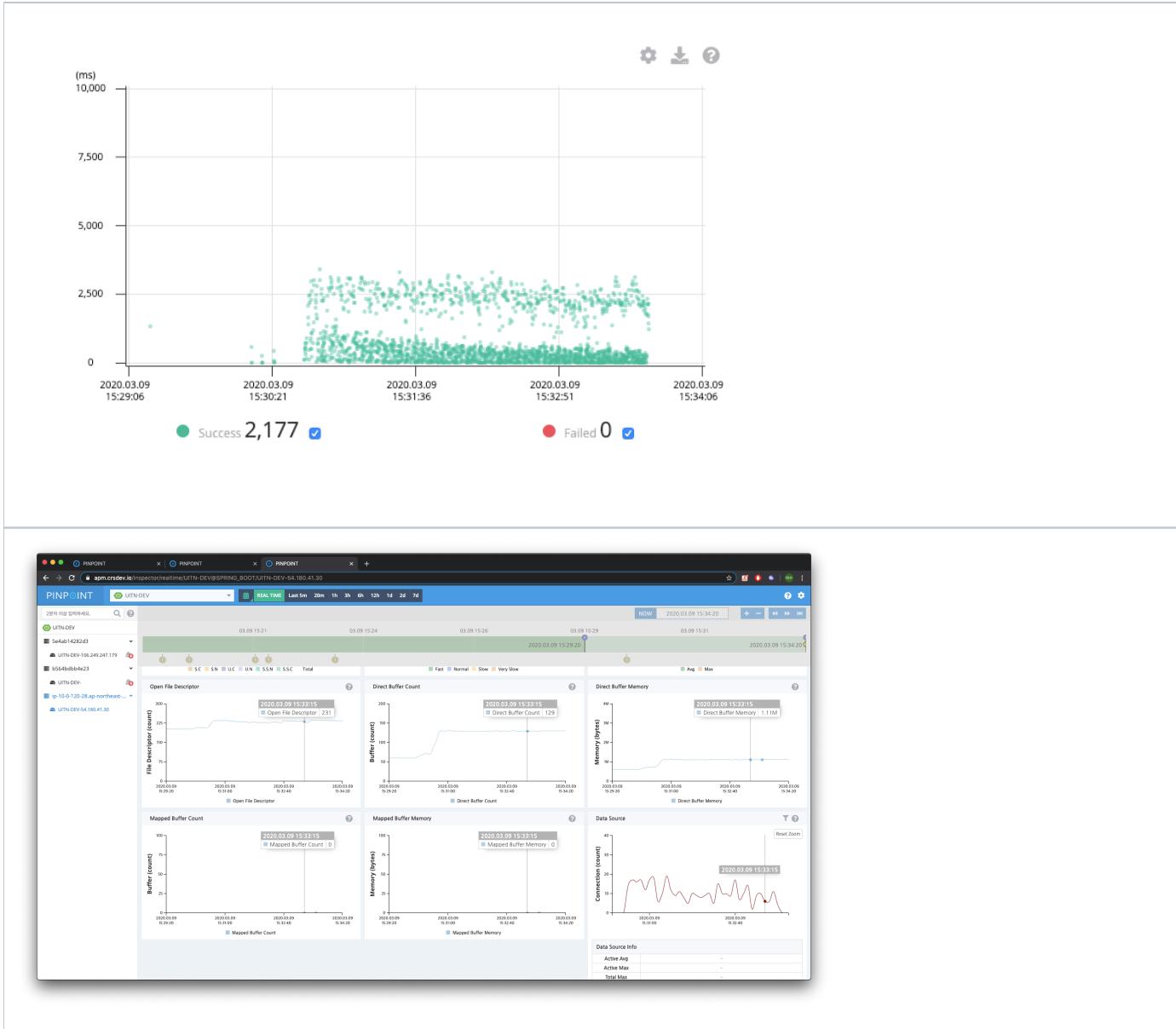
→ Cache가 원인이 아님을 확인 후 Interceptor 실행 이전의 문제인지 확인하기 위해 Client validation 로직을 제거후 다음 테스트 수행

차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
3차	2048	1024	-Xms256m - Xmx768m	20	MariaDB t3. medium	Client 조회 및 검증로직 제거

```

@Override
public boolean preHandle(HttpServletRequest req, HttpServletResponse re, Object handler) throws Exception
{
    return true;
}

```



Servlet Process	/api/v1/auth	15:33:24 146	0	2,627
http.status.code	200			
REMOTE_ADDRESS	10.0.60.157			
executeRootHandler(HttpHandler handler, HttpServerExchange exchange)		15:33:24 147	1	2,626
doPost(HttpServletRequest request, HttpServletResponse response)		15:33:24 147	0	2,625
initBinder(WebDataBinder binder)		15:33:24 148	1	0
authentication(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse)	initBinder(WebDataBinder binder)	15:33:24 148	0	2,619
getConnection()		15:33:24 148	0	1
executeUpdate(String sql)		15:33:24 149	0	5
SQL	set autocommit=0			
prepareStatement(String sql)		15:33:24 155	1	1
SQL	select user0_user_key as user_key1_16_0_, userl...			
executeQuery()		15:33:24 156	0	1
SQL	select user0_user_key as user_key1_16_0_, userl...			
SQL BindValue				

Handler method 이전에 존재하던 gap time이 사라지거나 1ms 정도로 크게 줄어듬.

Interceptor에서 수행하는 Client validation 로직에 병목현상이 있음을 확인.

perHandle 메소드엔 Base64 디코딩, BCrypt로 암호화 된 Client Secret 검증 로직이 존재하며 BCrypt를 통한 Client Secret 검증 로직을 제거하여 다음 테스트 수행

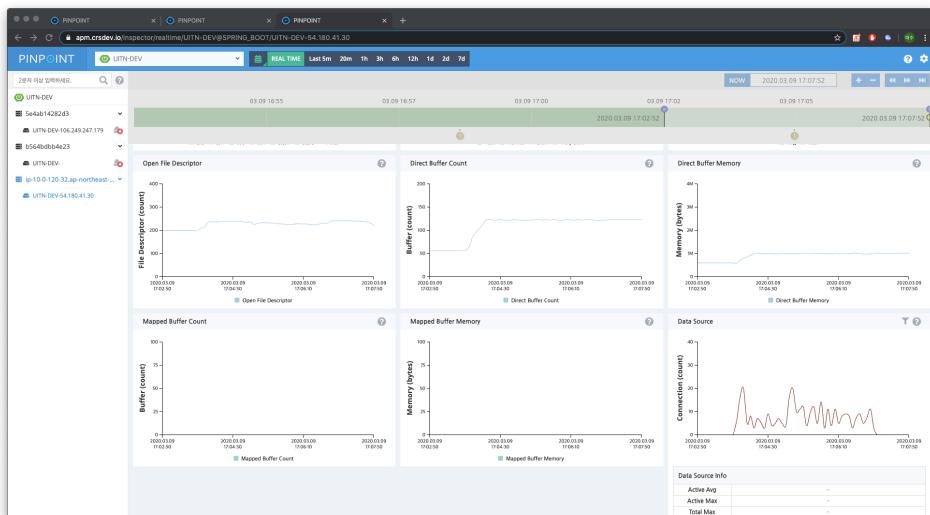
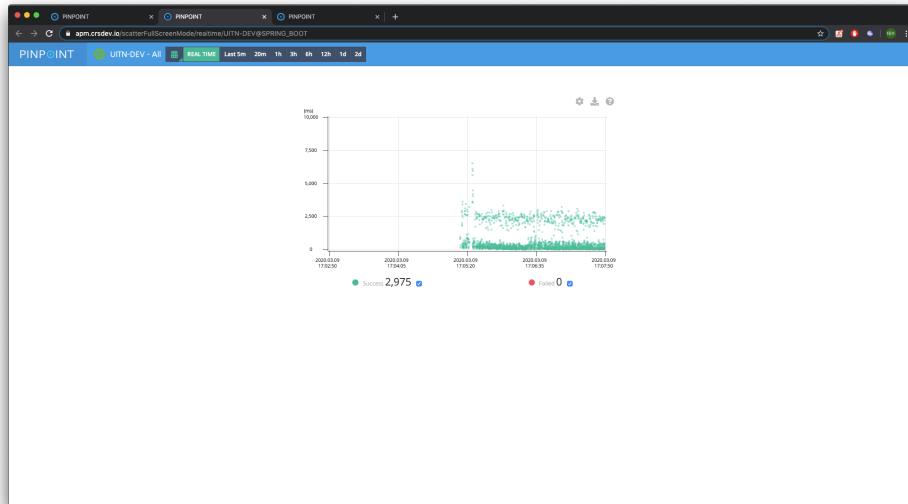
차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
4차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3. medium	BCrypt로 암호화 된 Client Secret 검증 로직 제거

```
@Override
public boolean preHandle(HttpServletRequest req, HttpServletResponse re, Object handler) throws Exception {
    final String authorization = req.getHeader("Authorization");
    if (Strings.isBlank(authorization) || !authorization.toLowerCase().startsWith("basic")) {
        req.setAttribute("exception", new InvalidClientException("Client authentication required"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    String base64Credentials = authorization.substring("Basic".length()).trim();
    byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);
    String credentials = new String(credDecoded, StandardCharsets.UTF_8);
    final String[] values = credentials.split(":", 2);

    Client client;
    try {
        client = this.clientRetrieveService.retrieveById(values[0]);
        if (!this.passwordEncoder.matches(values[1], client.getSecret())) {
            throw ResourceNotFoundException.from(values[0]);
        }
    } catch (ResourceNotFoundException e) {
        req.setAttribute("exception", new InvalidClientException(values[0] + " is Invalid client"));
        req.getRequestDispatcher("/errors/invalidclient").forward(req, re);
        return false;
    }

    return true;
}
```



executeRootHandler(HttpServletRequest handler, HttpServerExchange exchange)	17:06:56 939	0	2,456	<div style="width: 100%; background-color: #007bff; height: 10px;"></div>	3 Connectors
doPost(HttpServletRequest request, HttpServletResponse response)	17:06:56 941	2	2,453	<div style="width: 99%; background-color: #007bff; height: 10px;"></div>	38 FrameworkServlet
getConnection()	17:06:56 974	33	0	<div style="width: 0%; background-color: #007bff; height: 10px;"></div>	0 HikariDataSource
prepareStatement(String sql)	17:06:56 974	0	0	<div style="width: 0%; background-color: #007bff; height: 10px;"></div>	0 MariaDbConnection
SQL	select client0_.client_key as client_k1_0_, client0_...				
executeQuery()	17:06:56 974	0	1	<div style="width: 100%; background-color: #007bff; height: 10px;"></div>	1 ClientSidePreparedStatement...
SQL	select client0_.client_key as client_k1_0_, client0_...				
SQL-BindValue	SAFETYAPP_API_STG				
initBinder(WebDataBinder binder)	17:06:56 975	0	0	<div style="width: 0%; background-color: #007bff; height: 10px;"></div>	0 AuthenticationController
authentication(HttpServletRequest httpServletRequest, Authentication...)	17:06:56 976	1	2,414	<div style="width: 99%; background-color: #007bff; height: 10px;"></div>	2,397 AuthenticationController
getConnection()	17:06:56 976	0	0	<div style="width: 0%; background-color: #007bff; height: 10px;"></div>	0 HikariDataSource
executeUpdate(String sql)	17:06:56 976	0	1	<div style="width: 100%; background-color: #007bff; height: 10px;"></div>	1 MariaDbStatement

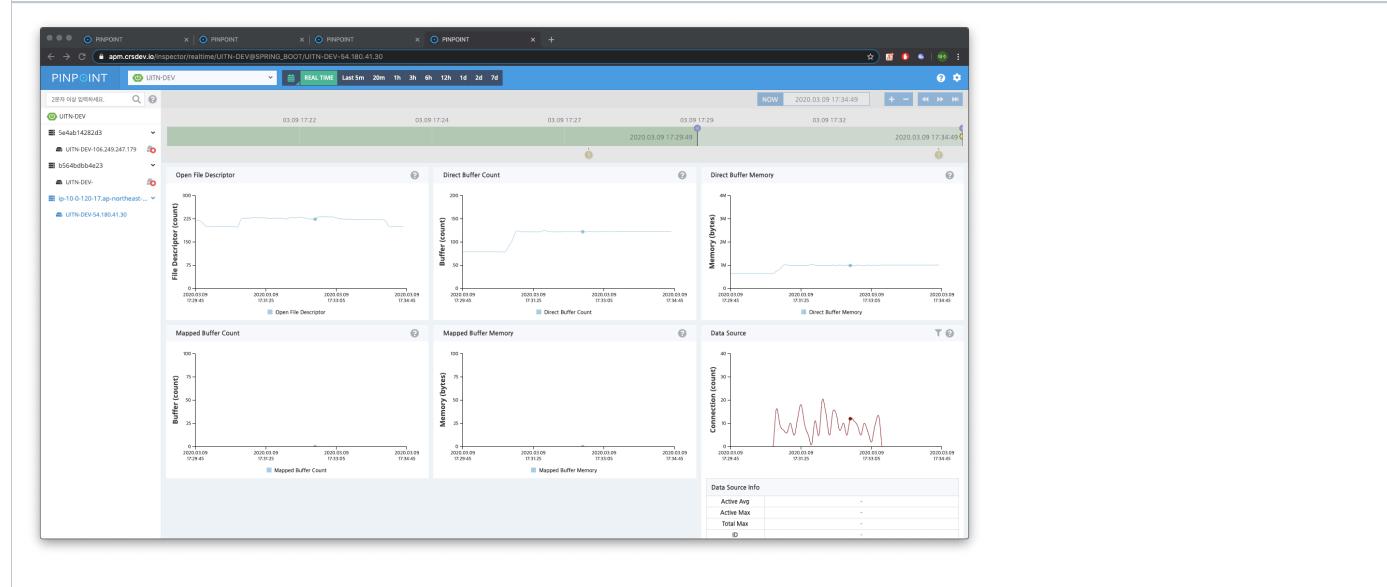
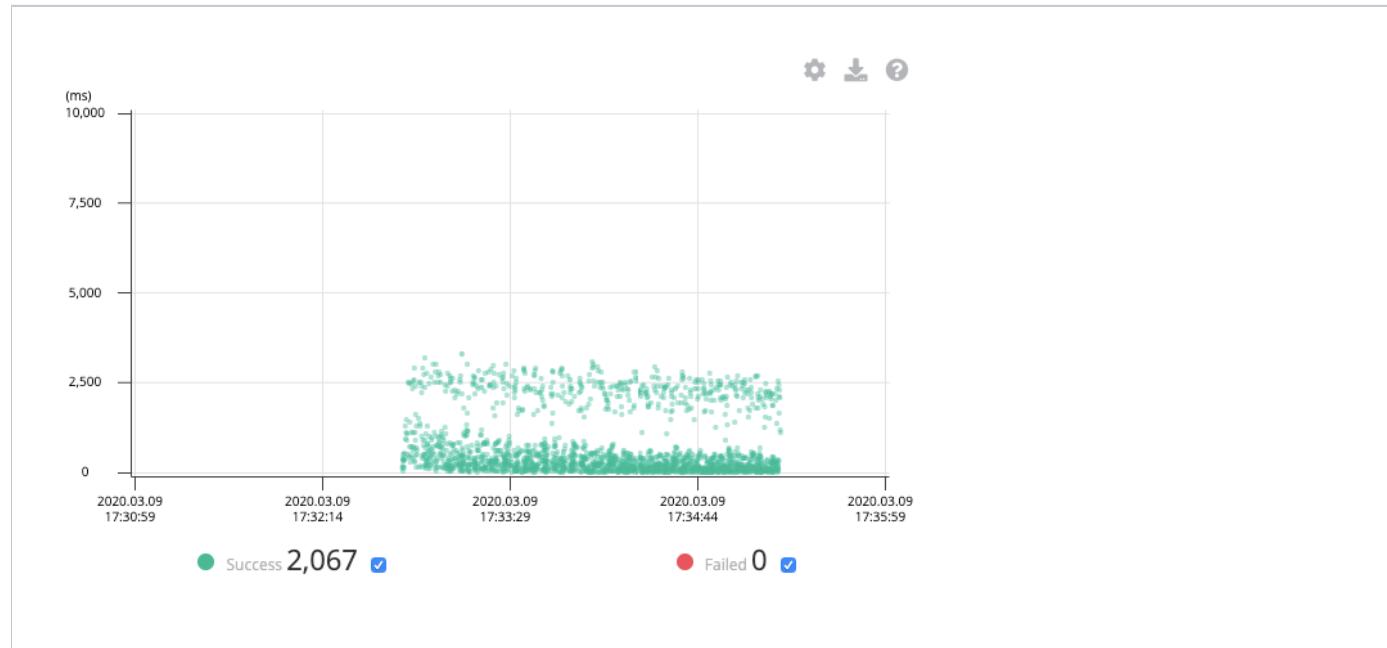
마찬가지로 Handler method 수행 이전에 존재하던 gap time이 0~2ms 정도로 줄어들음.

결과적으로 BCrypt를 통해 Client Secret을 매칭하는 작업이 동시성에 불리함이 있음을 확인.

기존 Client Entity를 조회하는 부분에 대해서만 캐시를 적용하였던 부분을

Base64 디코딩, Client Entity 조회, Client secret 검증 과정 전체에 대해 캐시를 적용하도록 수정 후 다음 테스트 진행.

차수	MIB	CPU (unit)	JavaOpts	Connection pool size	RDS Spec	비고
5차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3. medium	Base64 Decoding, Client 조회, BCrypt 인증로직 캐싱



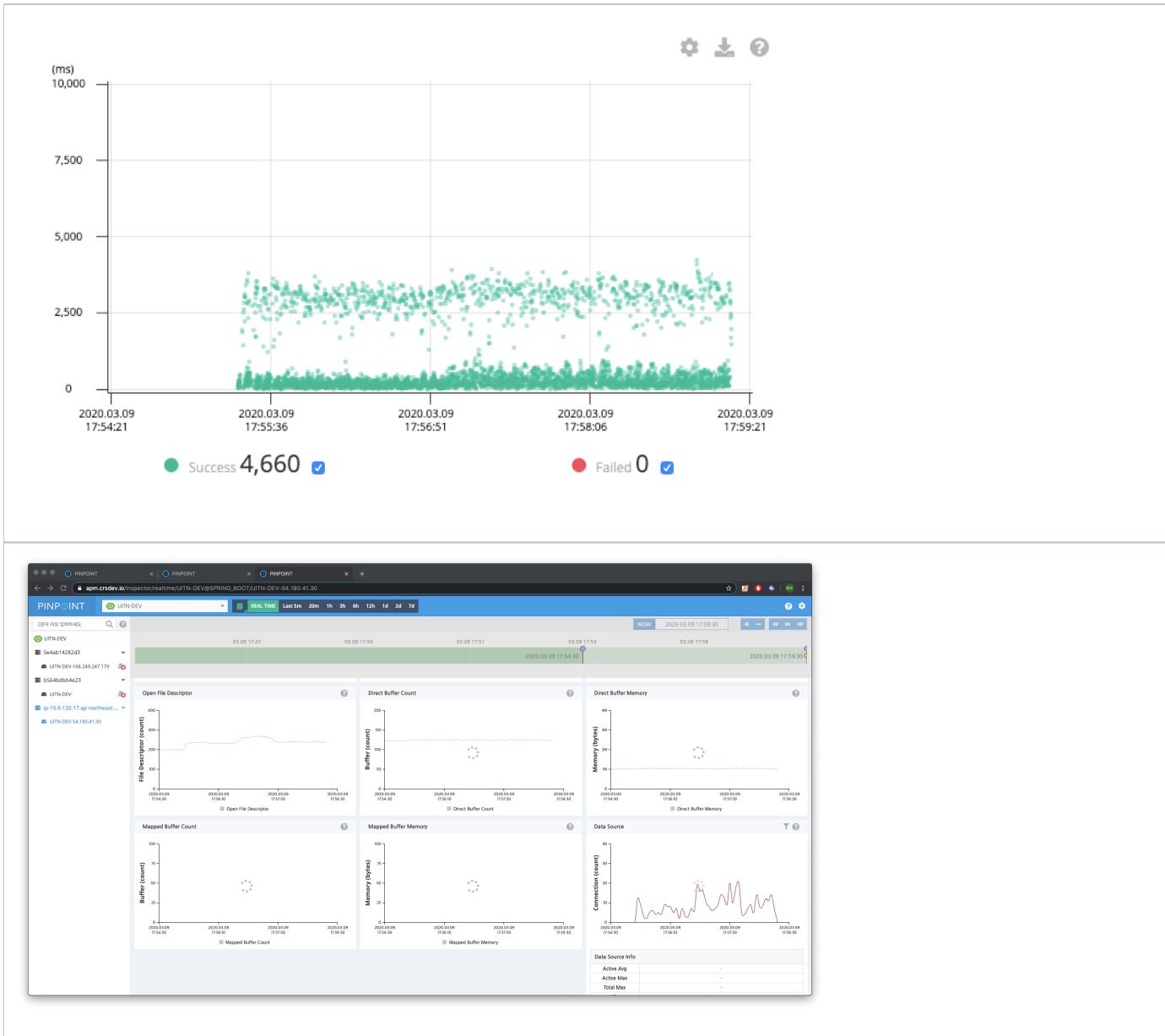
⌚ http.status.code	200				
📍 REMOTE_ADDRESS	10.0.50.250				
↳ executeRootHandler(HttpHandler handler, HttpServerExchange exchange)		17:32:23 197	0	2,433	
↳ doPost(HttpServletRequest request, HttpServletResponse response)		17:32:23 198	1	2,432	
initBinder(WebDataBinder binder)		17:32:23 200	2	0	
authentication(HttpServletRequest httpServletRequest, Authentication...)		17:32:23 200	0	2,418	
getConnection()		17:32:23 200	0	0	
↳ executeUpdate(String sql)		17:32:23 200	0	1	
SQL	set autocommit=0				
↳ preparedStatement(String sql)		17:32:23 203	2	0	
SQL	select user0_user_key as user_key1_16_0_, userl...				
↳ executeQuery()		17:32:23 203	0	1	
SQL	select user0_user_key as user_key1_16_0_, userl...				
SQL-BindValue	taesu@crscube.co.kr, taesu@crscube.co.kr				
↳ executeUpdate(String sql)		17:32:23 205	1	0	

Handler method 수행 이전에 존재하던 gap time이 0~2ms 정도로 유지되었으며 전체적인 응답시간이 200ms ~ 2000ms 대역으로 줄어들음.

각 케이스에서 CPU 사용률에 크게 차이가 없는 것으로 보아 관련 작업에서 발생하는 병목 현상의 주 원인은 I/O Blocking으로 추측할 수 있었으며 <https://stackoverflow.com/questions/36471723/bcrypt-performance-deterioration> 링크에서 BCrypt.hasPw 등의 메소드 호출 시 내부적으로 사용하는 /dev/random에서 지연이 있을 수 있음을 확인 (정확한 원인은 조금 더 파악 필요.....).

Active User 20

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec	비고
6차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3.medium	Active User: 20



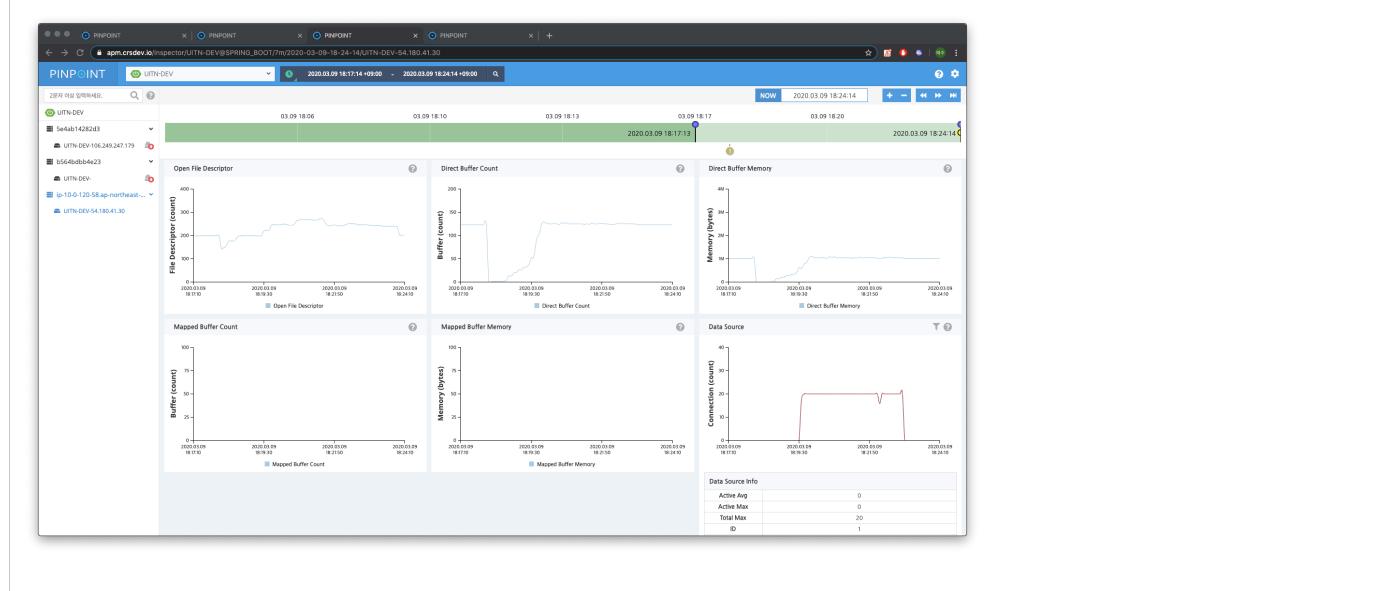
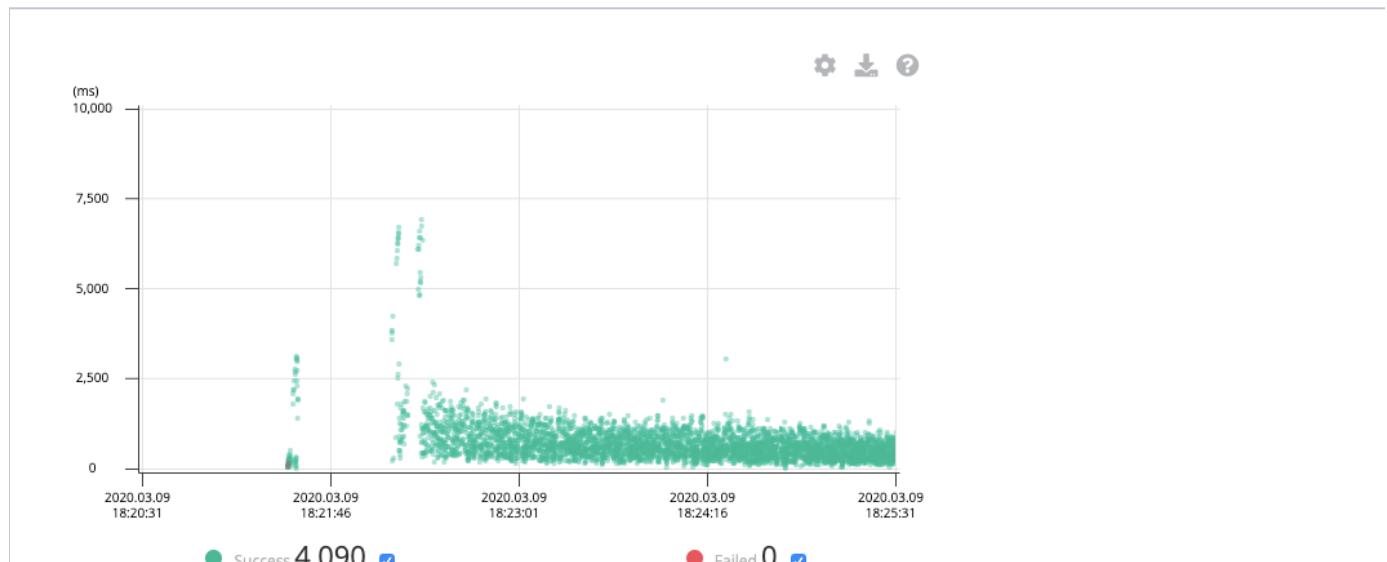
✓ prepareStatement(String sql)	17:59:03 227	16	0	0
SQL	select user0_user_key as user_key1_16_0_, userl...			
✓ executeQuery()	17:59:03 237	10	18	18
SQL	select user0_user_key as user_key1_16_0_, userl...			
SQL-BindValue	taesu@crscube.co.kr, taesu@crscube.co.kr			
✓ executeUpdate(SQL-BindValue)	17:59:03 256	1	1	1
SQL	set autocommit=1			
getConnection()	17:59:06 510	3,253	0	0
✓ executeUpdate(String sql)	17:59:06 510	0	9	9
SQL	set autocommit=0			
✓ prepareStatement(String sql)	17:59:06 522	3	0	0
SQL	select user0_user_key as user_key1_16_0_, userl...			
✓ executeQuery()	17:59:06 528	6	2	2
SQL	select user0_user_key as user_key1_16_0_, userl...			

사용자 정보 조회 후 getConnection() 메소드의 gap 타임에 3000ms의 소요시간이 발생

Client의 Secret을 PasswordEncoder로 매칭하는것과 같은 이유로 사용자 비밀번호 검증 시 PasswordEncoder를 통한 match에서 병목현상이 발생하는 것으로 파악

해당 부분에 Cache를 적용하도록 개선 후 다음 테스트 수행

차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
7차	2048	1024	-Xms256m -Xmx768m	20	MariaDB t3. medium	Active User: 20, User Authentication 시 Password Matcher 캐시 (사용자 비밀번호, BCrypt encoded string, 사용자 아이디)

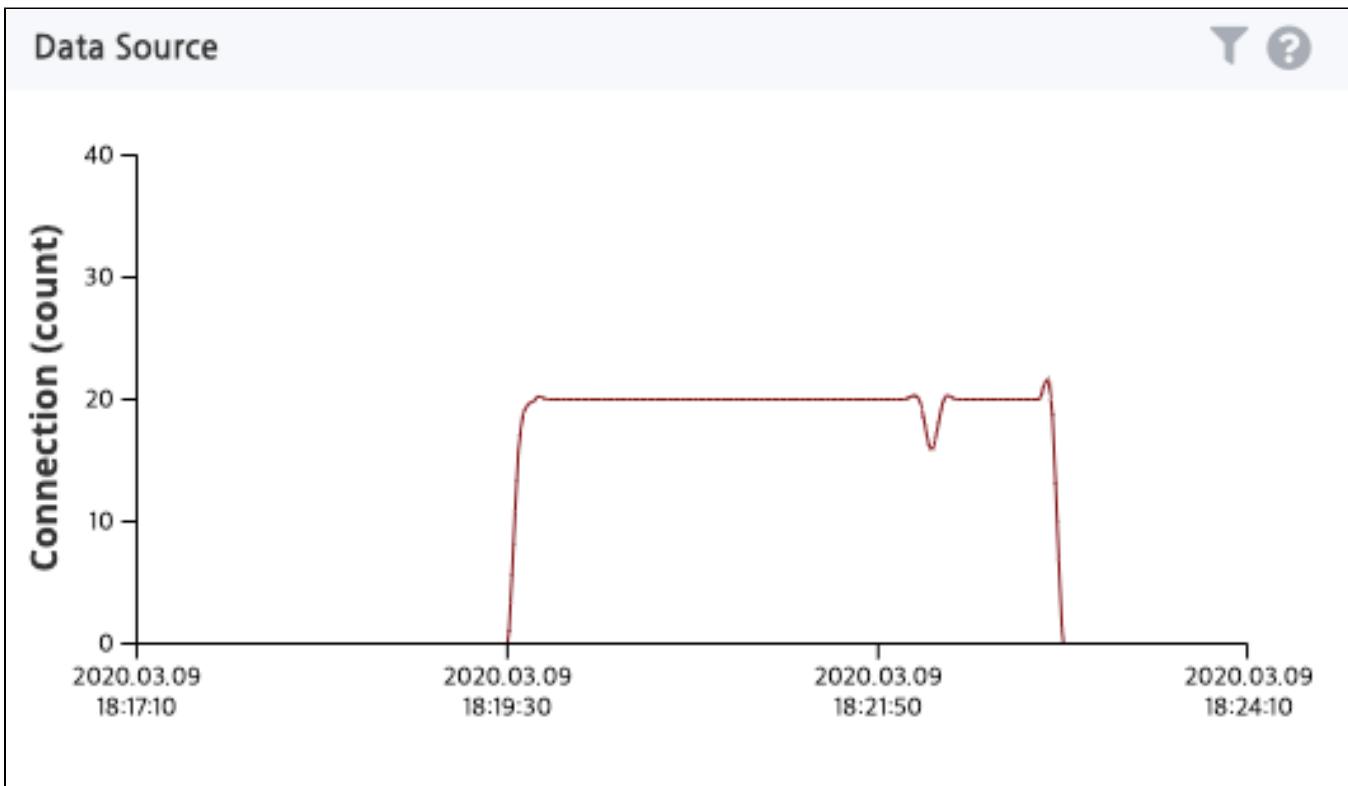


prepareStatement(String sql)	18:21:22 698	68	0
SQL select user0_.user_key as user_key1_16_0_, userl...			
executeQuery()	18:21:22 698	0	2 ↴
SQL SQL-BindValue taesu@crscube.co.kr, taesu@crscube.co.kr			
executeUpdate(String sql)	18:21:22 722	22	0
SQL set autocommit=1			
getConnection()	18:21:22 739	17	360 🔍
executeUpdate(String sql)	18:21:23 099	0	0
SQL set autocommit=0			
prepareStatement(String sql)	18:21:23 105	6	1 ↴
SQL select user0_.user_key as user_key1_16_0_, userl...			
executeQuery()	18:21:23 106	0	3 ↴
SQL SQL-BindValue taesu@crscube.co.kr, taesu@crscube.co.kr			

getConnection() 메소드의 실행 전에 존재했던 gap 타임이 17~30ms 정도로 줄어들었으며 전반적으로 응답 시간이 크게 줄었음을 확인.

전체 TPS가 최대 40 정도로 상승하였으나 응답 그래프에서 부하테스트 초반 지점에서 warm up이 덜 된 이유인지 크게 응답시간이 늦은 구간이 발생함.

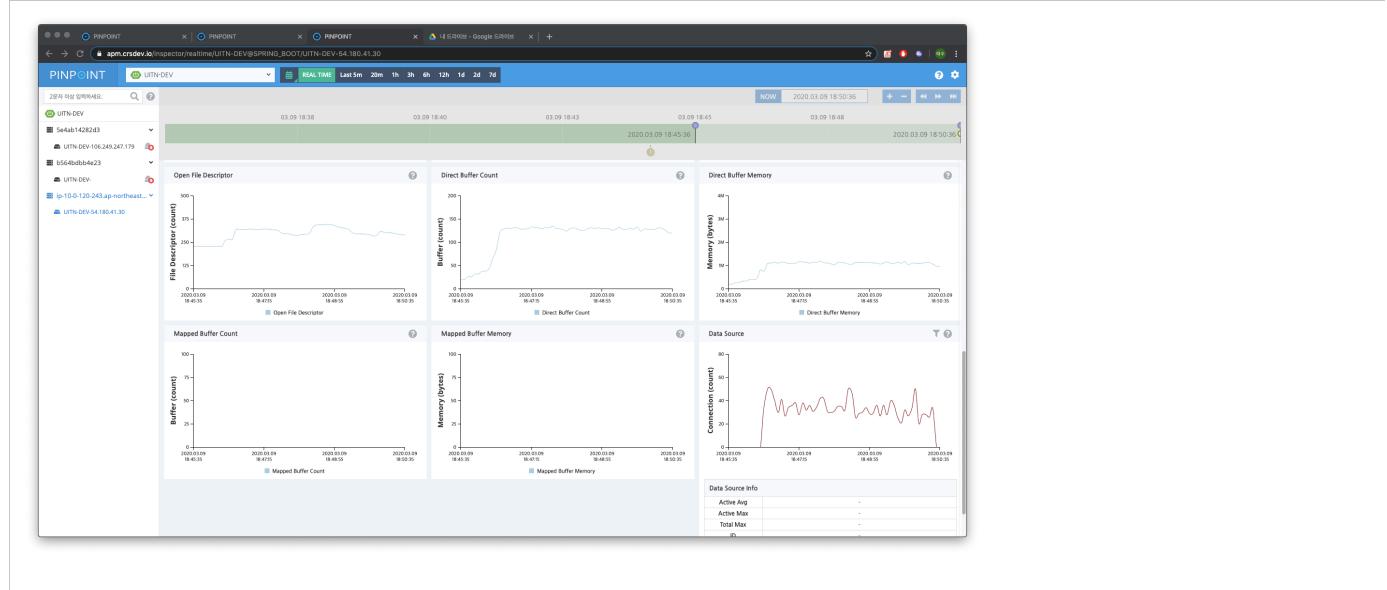
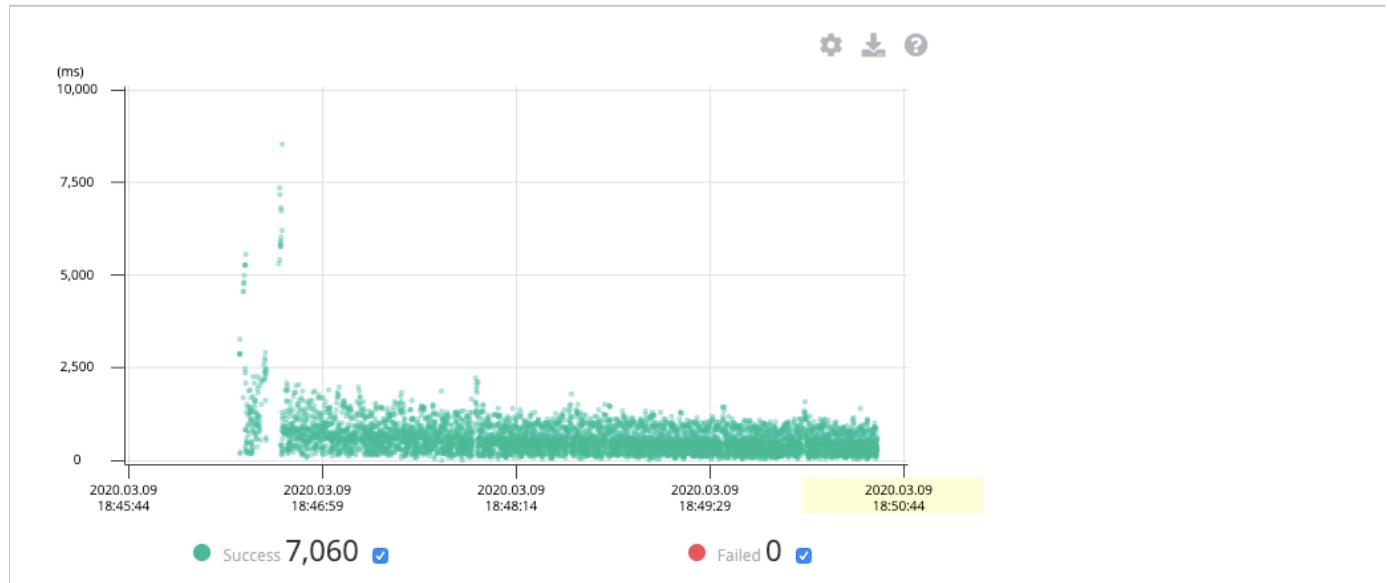
또한 getConnection 메소드에서 커넥션을 가져오는데 걸린 시간이 360ms로 약간 지연된 결과가 보이며 Data Source 그래프의 connection count가 max인 20을 꾸준히 유지하고 있음을 확인



→ Hikari CP의 connection pool size를 조정하여 다음 테스트 수행

Active User 30

차수	MIB	CPU (unit)	Java Opt	Connection pool size	RDS Spec	비고
8차	2048	1024	-Xms256m -Xmx768m	50	MariaDB t3. medium	Connection pool size 50으로 증가



Connection pool 그래프에서 max(50) 아래의 숫자를 유지하고 있음을 확인.

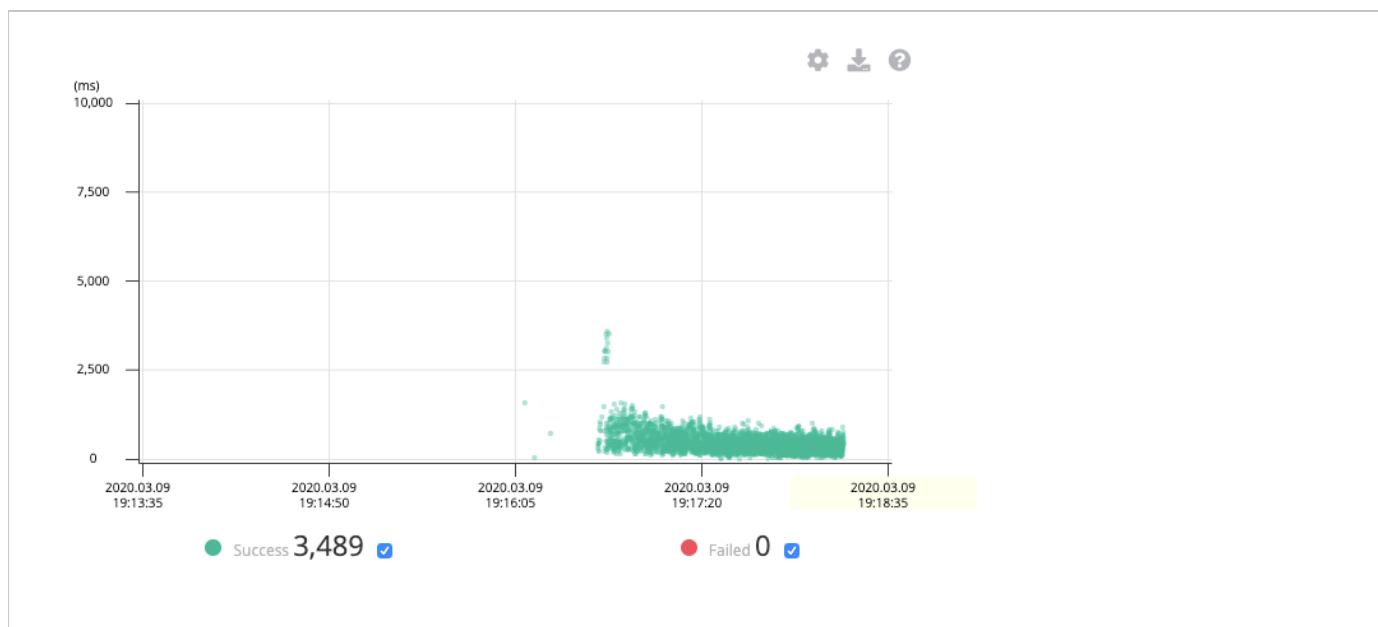
테스트 초기 시간 대에 응답시간이 크게 툰 후 안정화 되는 양상의 그래프가 계속되는 것으로 확인 (범위 5000ms ~ 8500ms)

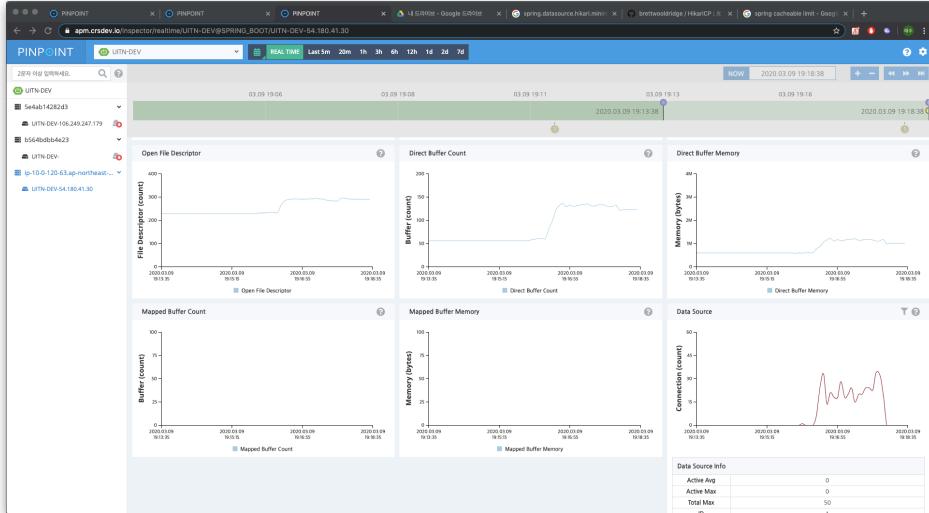
해당 구간에서 전체 API가 고르게 응답시간이 튀는 것으로 결과가 나타났으며 Call tree 상에서 getConnection() 메소드가 주 원인임을 확인.

The screenshot shows the Pinpoint application monitoring interface. At the top, there are five tabs labeled 'PINPOINT' with the current tab being 'PINPOINT'. Below the tabs is a timeline from 18:45 to 18:50. The main area displays a table of API requests with columns: #, StartTime, Path, Res(ms), Exception, Agent, Client IP, and Transaction. A specific row for 'getConnection()' is highlighted in green. Below this table is a detailed 'Call Tree' section for the transaction 'UITN-DEV-54.180.41.30*1583747079928*127'. The call tree shows the execution flow from the client to the database, with various methods like 'sendPasswordSettingMail', 'getPasswordSettingEmailRequest', 'getConnection()', 'prepareStatement', 'executeQuery', 'executeUpdate', 'prepareStatement', and 'executeQuery' being called. The 'getConnection()' method is shown as a significant bottleneck, taking approximately 4,939ms. The 'Call Tree' section also includes a table with columns: Method, Argument, StartTime, Gap(ms), Exec(ms), Exec(%), Self(ms), Class, API, Agent, and Application.

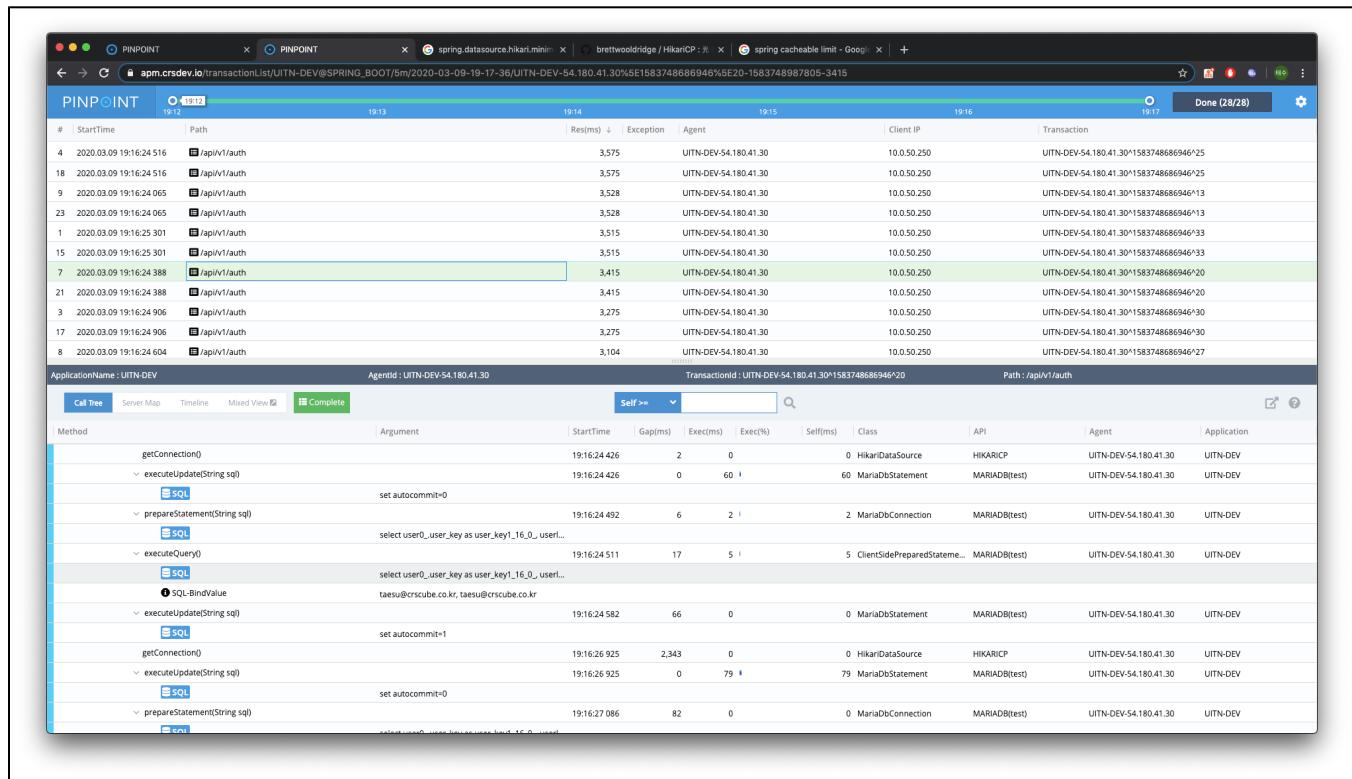
jvm 의 Xms 설정을 768m으로 동일하게 설정한 후 다음 테스트 진행

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec	비고
9차	2048	1024	-Xms768m -Xmx768m	50	MariaDB t3.medium	Xms 옵션 조정





테스트 초기 시간 대에 응답시간이 크게 튀는 문제가 어느정도 해결된 것을 확인 (범위 5000ms ~ 8500ms)에서 (2700ms ~ 3500ms)로 변동



해당 구간 API는 모두 인증과 관련된 API로 비밀번호 매칭 시 Cache hit rate와 관련된 것으로 파악.

최초 인증 시 Cache hit이 되지 않아 비밀번호 검증 로직이 수행되었고 그에 따라 응답시간이 늘었으나 점차 Cache hit이 되어 응답시간이 줄어들 은 것으로 파악.

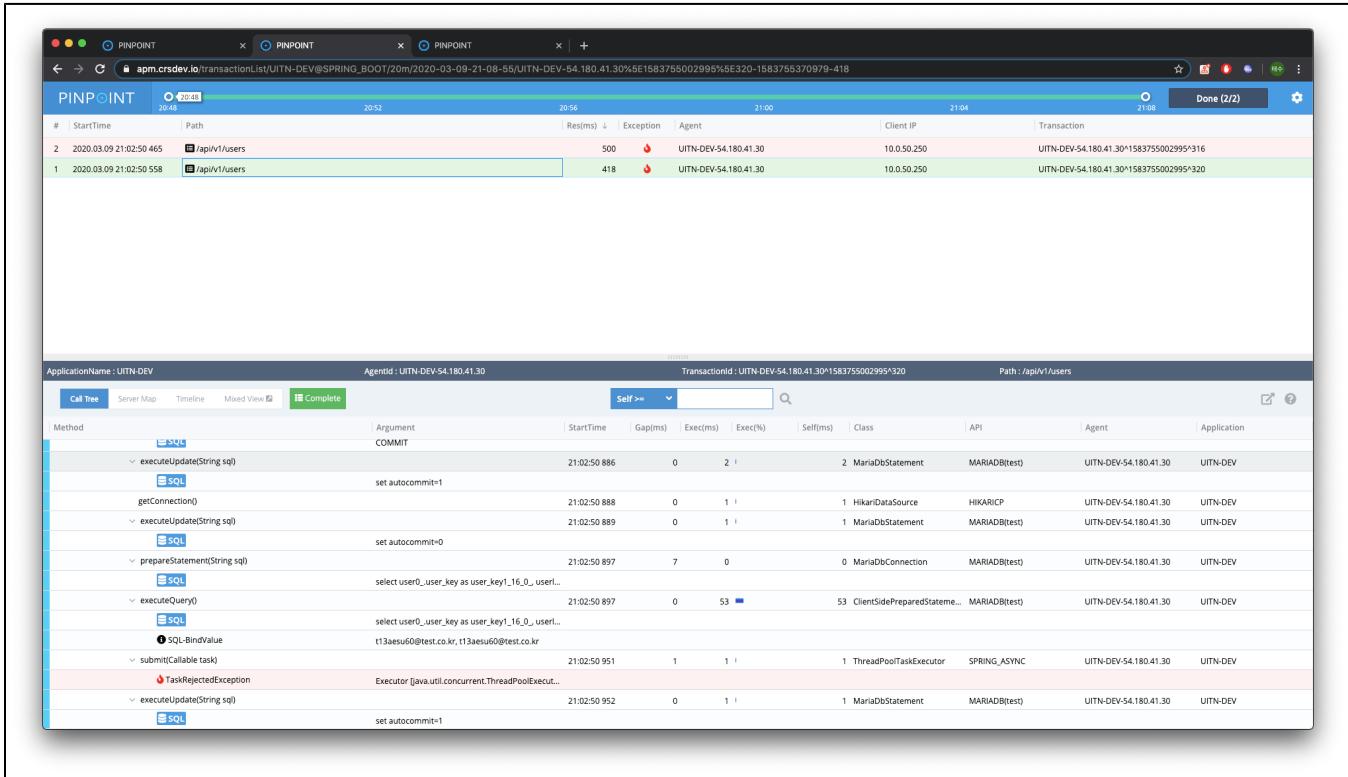
CPU 사용률이 70% 아래를 유지하고 있으며 Connection pool의 사용량도 40개 미만으로 유지되어 수치 조정

Active User 100 (Ramp-Up Period: 5s)

차수	MB	CPU (unit)	Java Opt	Connection pool size	RDS Spec	비고
10 차	2048	1024	-Xms768m -Xmx768m	100	MariaDB t3. medium	Connection pool size 증가 및 Active User 수 증가, Ramp-Up Period 5s 적용



Max TPS: 75, CPU 사용률: 50%, Major GC count: 1회, Connection 100개가 지속적으로 사용됨



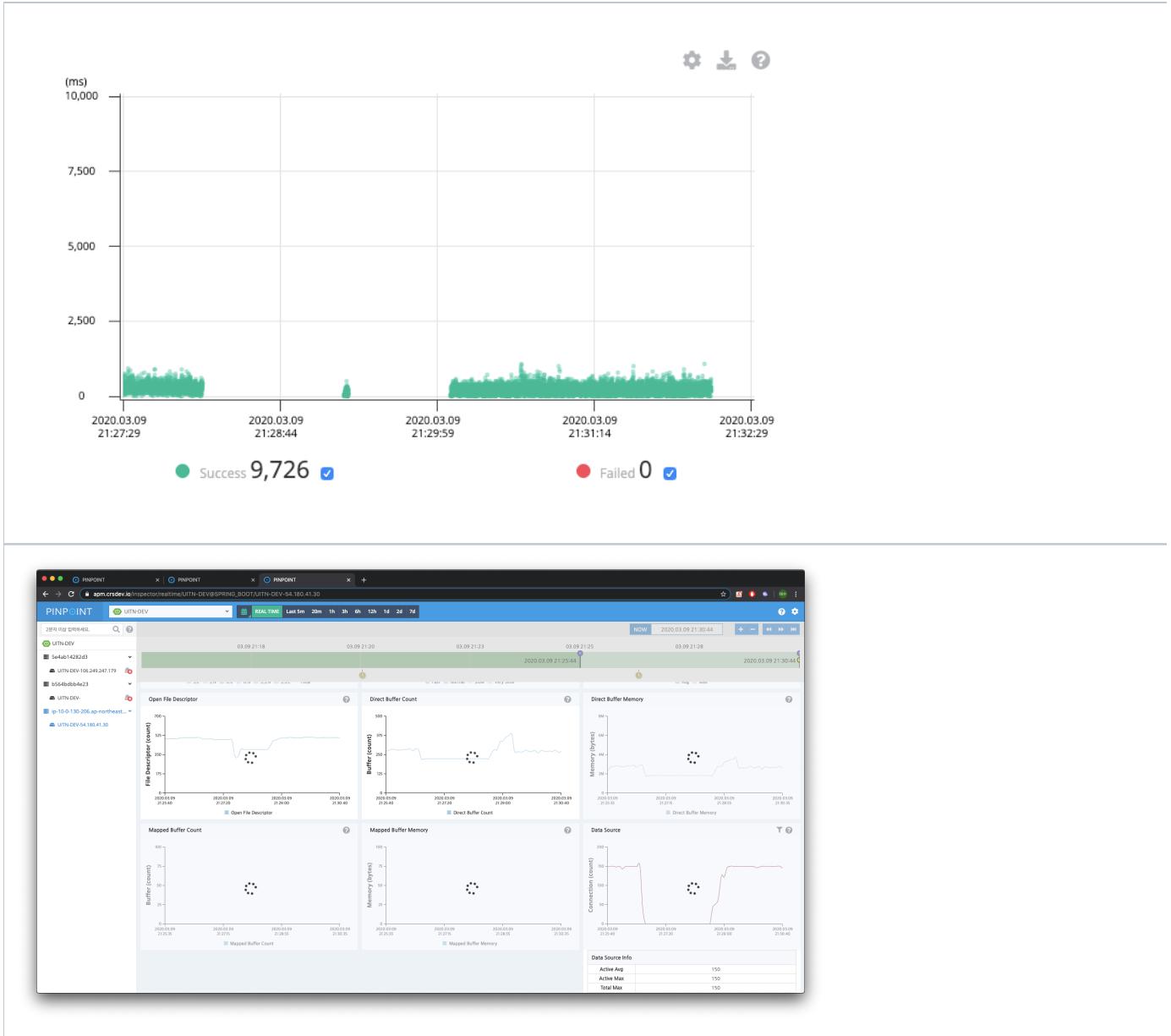
2건의 에러 발생, TaskRejectedException으로 사용자 생성 후 메일을 발송하는 작업을 담당하는 비동기 스레드 풀이 부족한 것으로 판단

Connection pool size를 150으로 증가 및 AsyncThreadPool의 설정을 조정

asyncPoolMinCapacity: 50	asyncPoolMinCapacity: 50
asyncPoolMaxCapacity: 100	-> asyncPoolMaxCapacity: 150
asyncPoolQueueCapacity: 20	asyncPoolQueueCapacity: 20

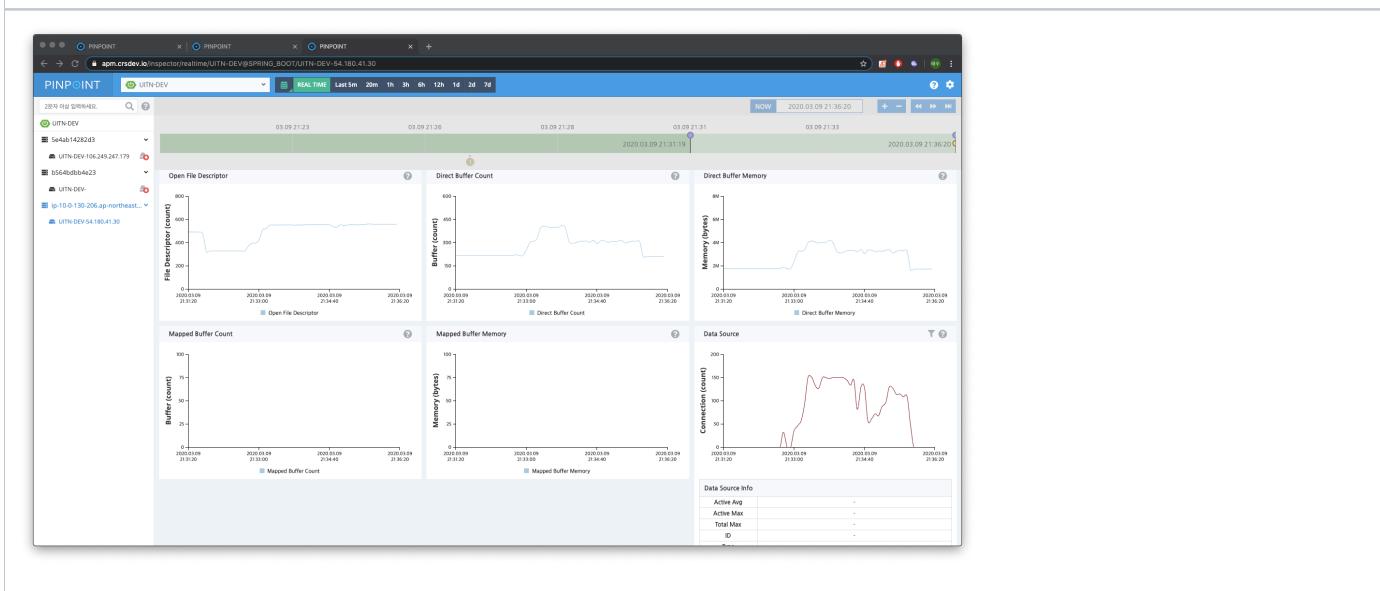
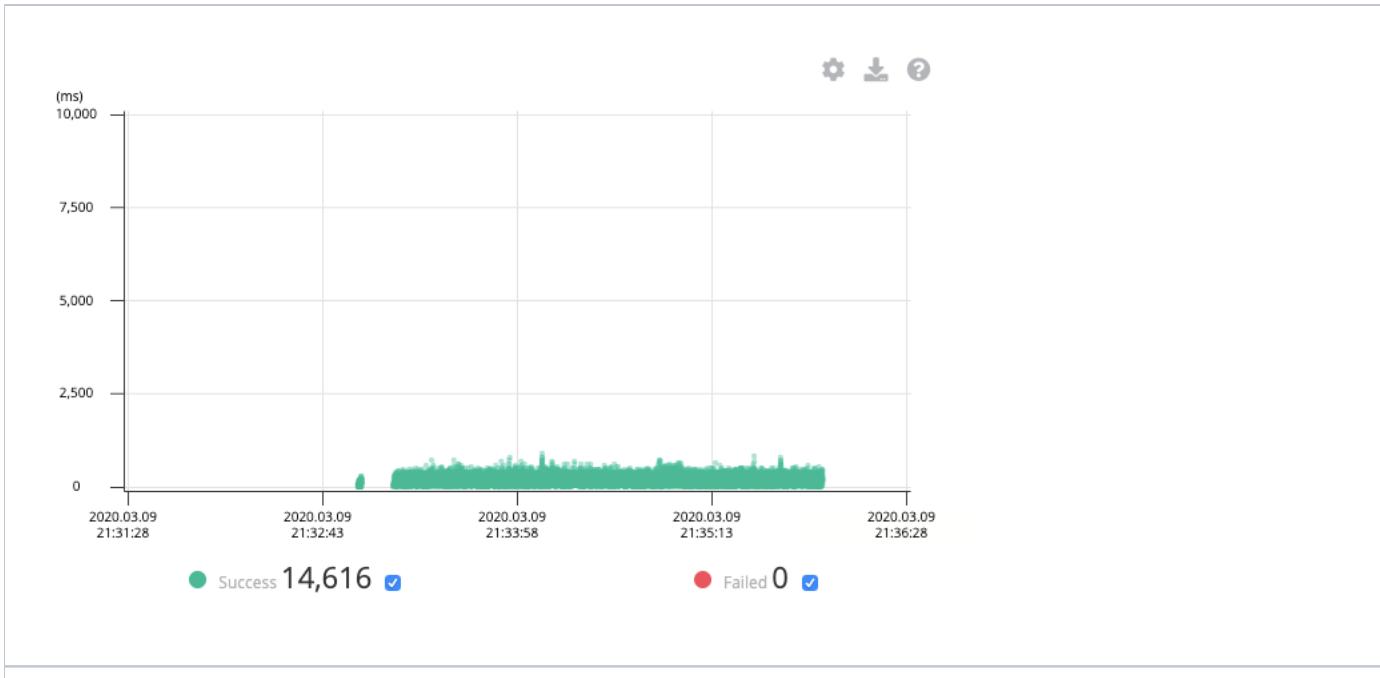
Active User 100 (Ramp-Up Period: 5s)

차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
11차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3. medium	Connection pool size 150으로 증가 및 비동기 스레드 풀 최대 크기 150으로 증가



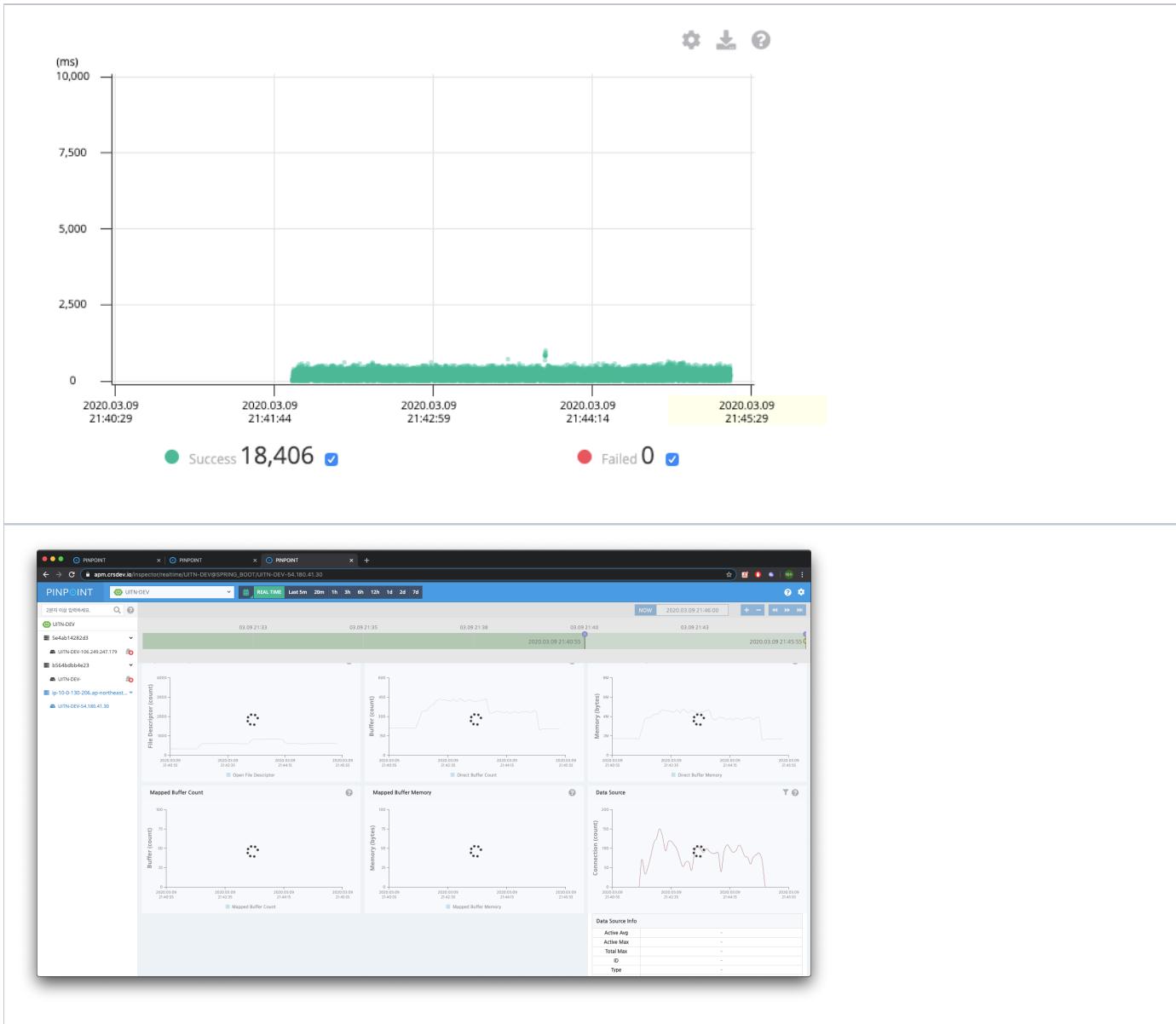
Active User 150 (Ramp-Up Period: 5s)

차수	MIB	CPU(unit)	Java Opt	Connection pool size	RDS Spec	비고
11차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3.medium	Active User 수 증가



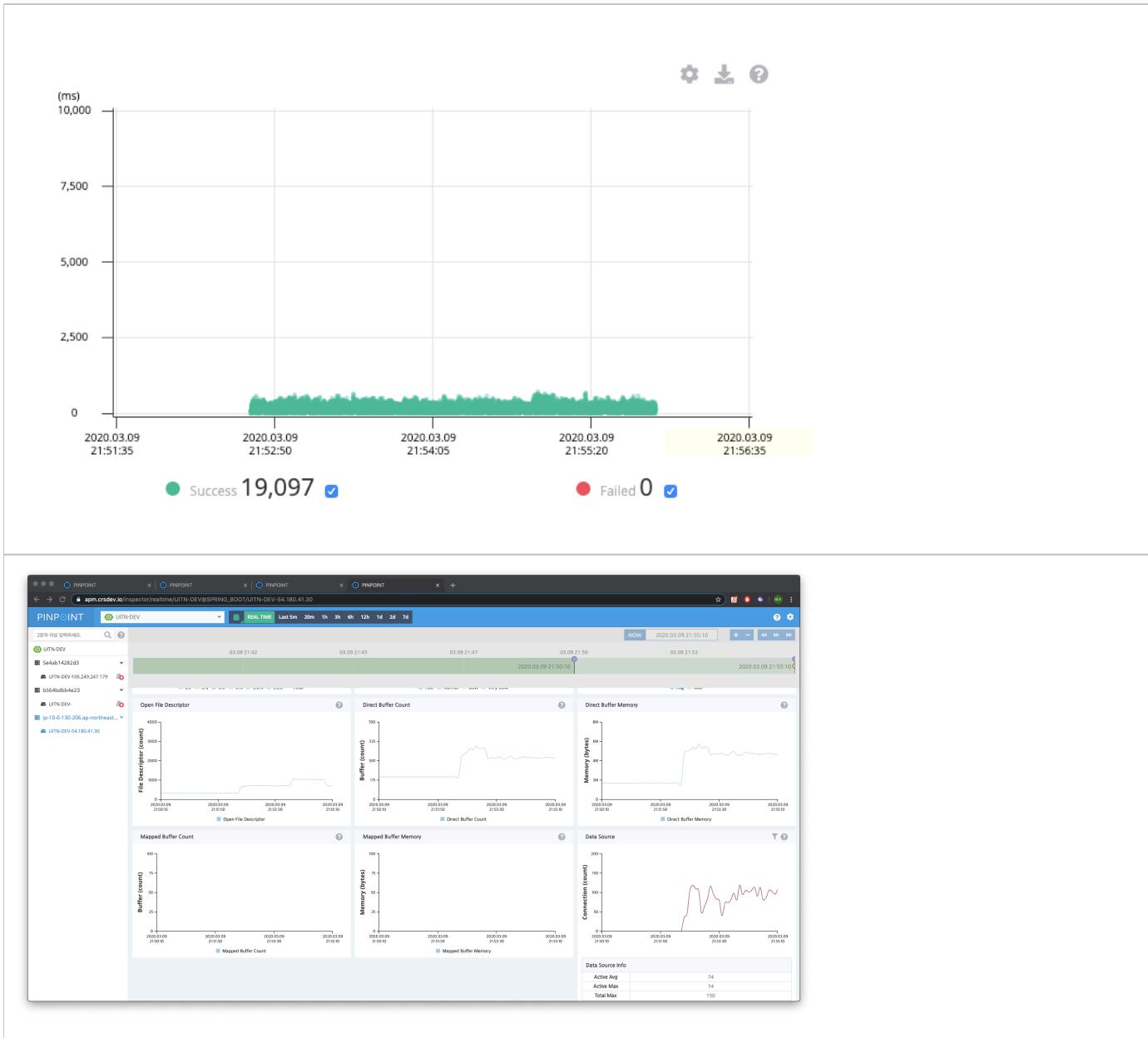
Active User 200 (Ramp-Up Period: 5s)

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec	비고
12차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3.medium	Active User 수 증가



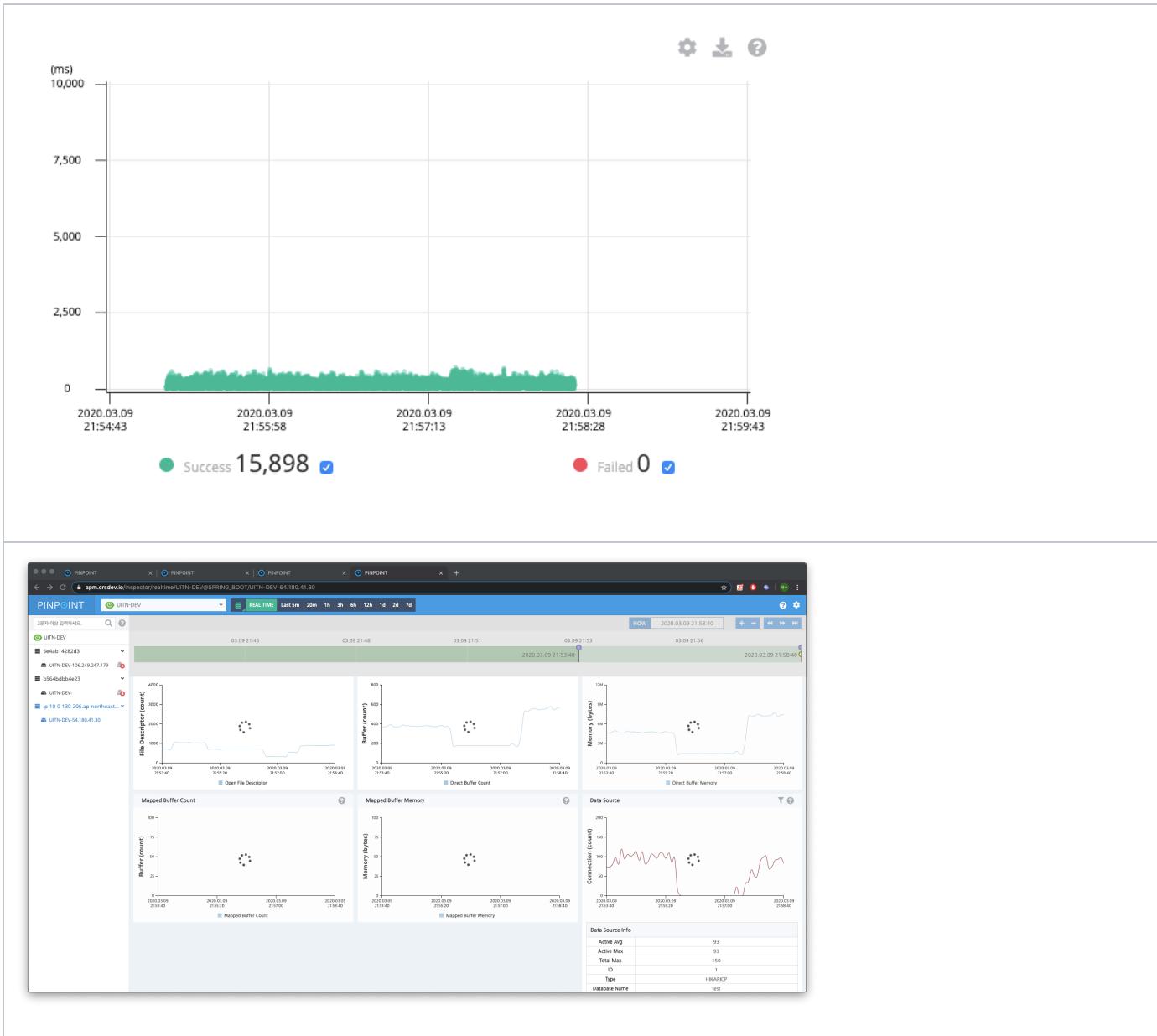
Active User 300 (Ramp-Up Period: 5s)

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec	비고
13차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3.medium	Active User 수 증가



Active User 500 (Ramp-Up Period: 5s)

차수	MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec	비고
14차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3.medium	Active User 수 증가

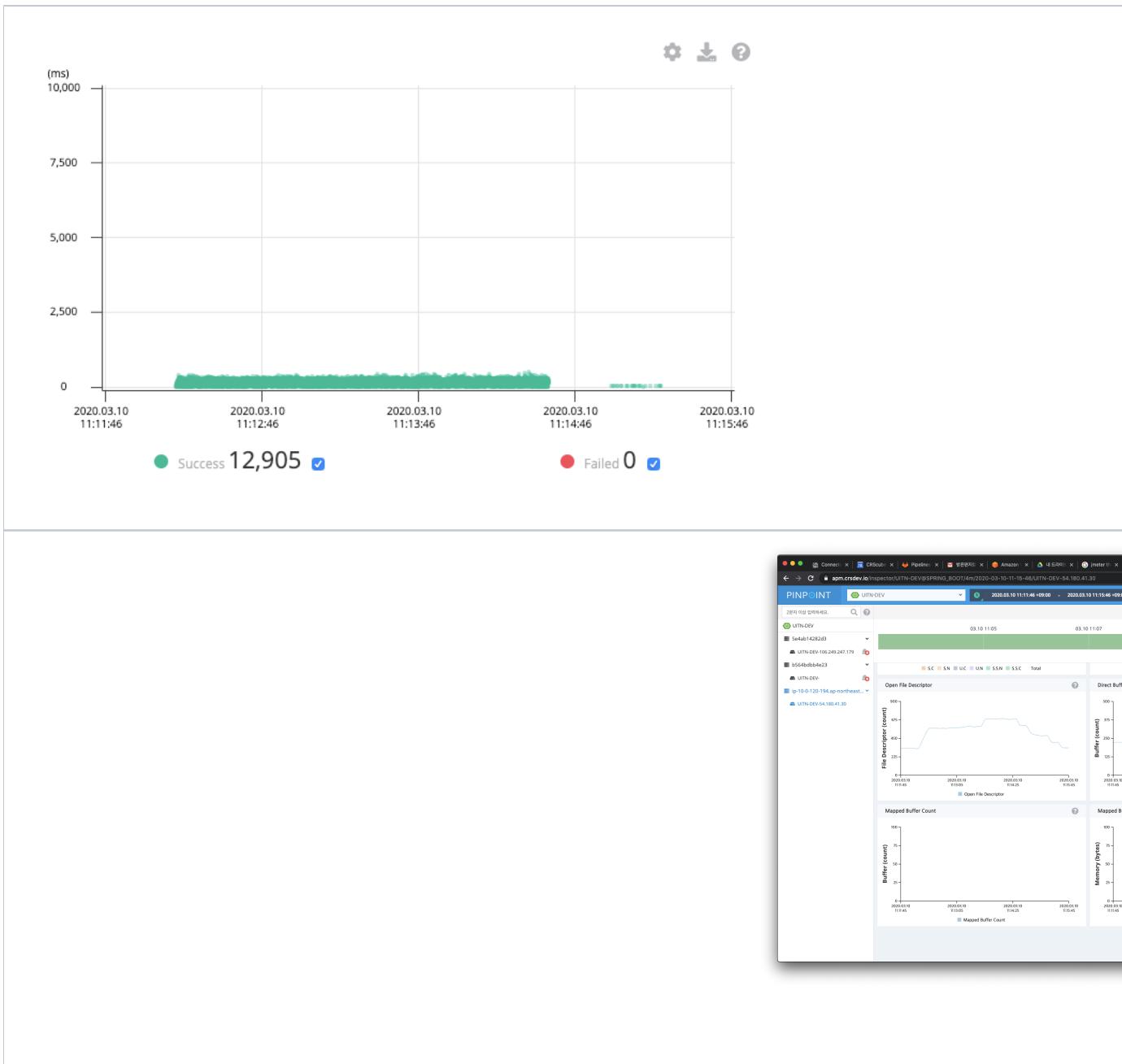


JMeter의 Active User를 증가시켜도 결과가 안정적인 것처럼 보이지만 더 이상 TPS가 오르지 않는 것으로 보아 테스트 하는 PC의 네트워크 대역폭에 한계가 있을 것으로 판단.

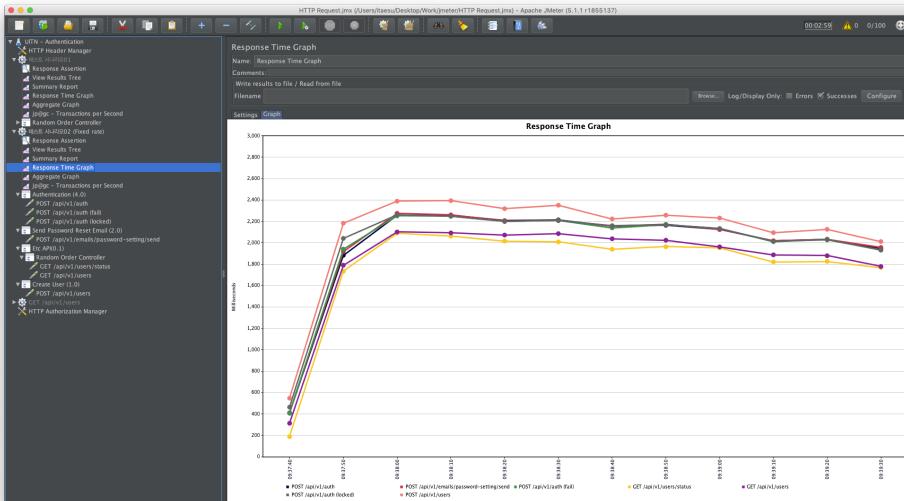
Active User 100에 대해서 서로 다른 PC에서 테스트를 진행하도록 시나리오 수정

Active User 200 (Test PC: 2)

차수	MIB	CPU (unit)	Java Opts	Connection pool size	RDS Spec	비고
15차	2048	1024	-Xms768m -Xmx768m	150	MariaDB t3. medium	Active User 200, Test pc를 두대로 증가



PC1



TPS가 100이상 오르지 않은 것으로 보아 사내에서 내보내는 Network 대역폭에 제한으로 인해 Active User 200인 상황에 대한 재현은 개발 PC로 불가한 것으로 파악

목표 Active User에 대응하기 위한 용량 산정

CPU

1 core(1024 unit)로 진행하였을 때 CPU 사용률이 70% 미만으로 동작하여 별도의 CPU Unit이 더 필요하지 않음.

Memory

현재 컨테이너에 허용 된 메모리는 1024MB이며 테스트 시 Active Thread가 16개가 유지되었고
메모리 부족으로 인해 빈번한 GC가 발생하지 않은 것으로 판단하여 현재 용량 (`-Xms768m -Xmx768m`) 유지

(추후 비동기 작업이 늘어남에 따라서 AsyncExecutorPool의 사이즈를 늘려야 한다면 메모리 확보 필요)

Connection Pool Size

최대 150까지 Connection pool size 늘리며 테스트를 진행했으나 현재 RDS Spec이 t3.medium이므로 별도로 조정하지 않는 이상 최대 296개의 커넥션이 허용 됨.

이중화 구성을 대비하여 Connection pool size를 130으로 지정.

MIB	CPU(unit)	Java Opts	Connection pool size	RDS Spec
2048	1024	<code>-Xms768m -Xmx768m</code>	130	MariaDB t3.medium

