

핀포인트는 어떻게 observability 를 강화했는가

구태진
Observability Platform

NAVER

CONTENTS

DEVIEW
2019

1. 소개
2. 실시간 observability 확보하기
3. 효율적으로 observability 정보 저장하기
4. 다양한 컴포넌트에 대한 observability 확보하기
5. What's next

1. 소개

What is

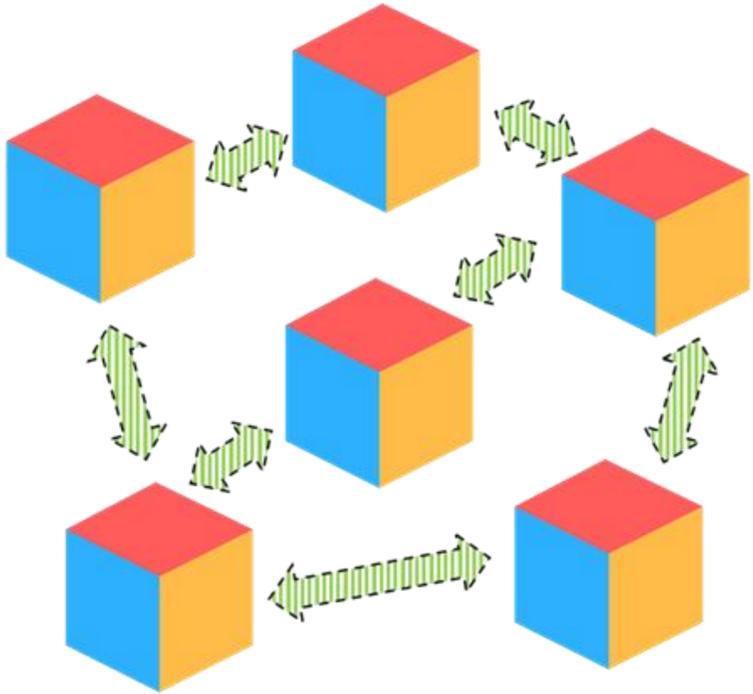
Observability?



Server



Application



Distributed systems

Monitoring

Observability



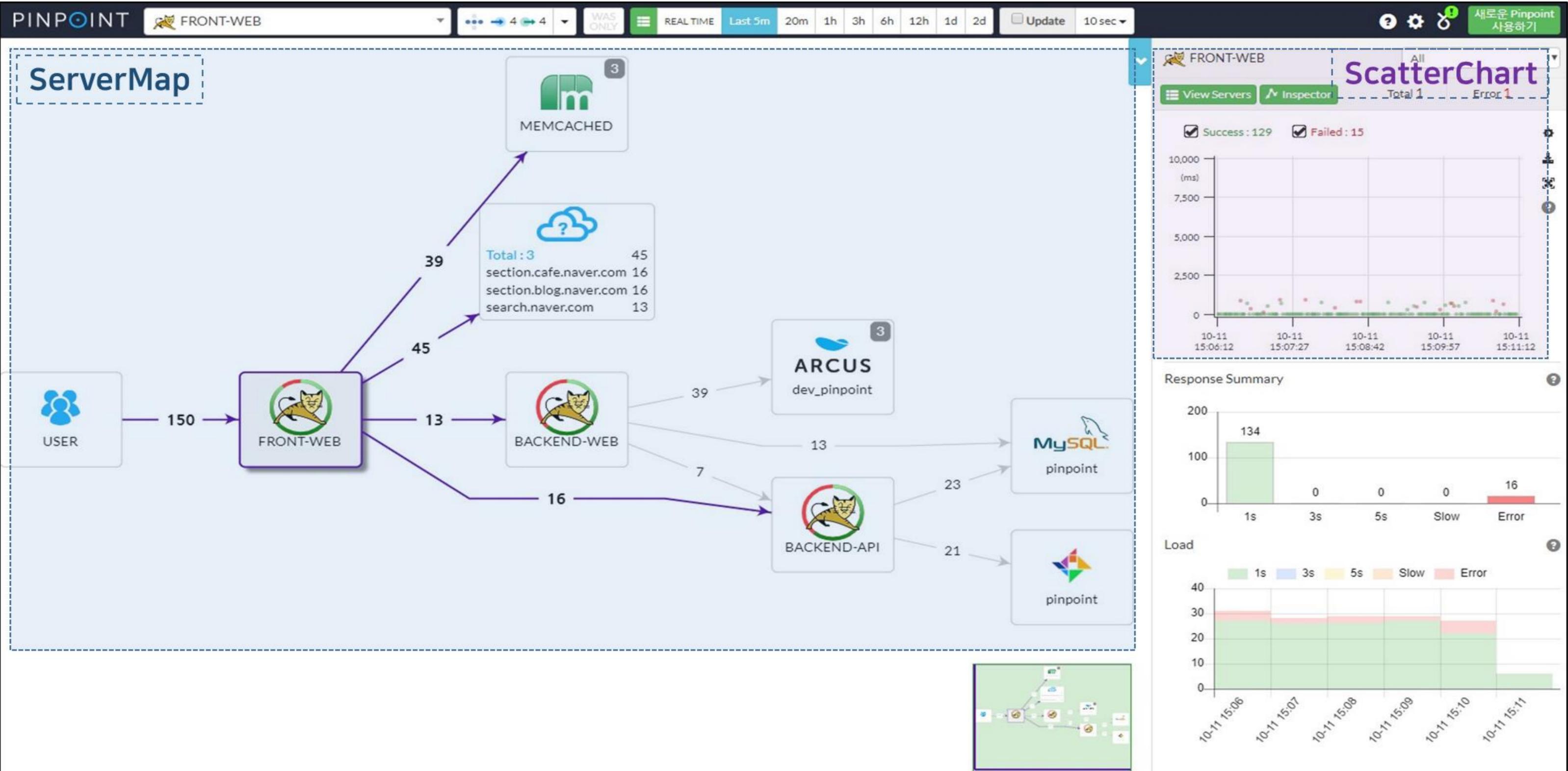
Pinpoint is APM tool for

large-scale distributed systems

JAVA_OPTS = "\$JAVA_OPTS

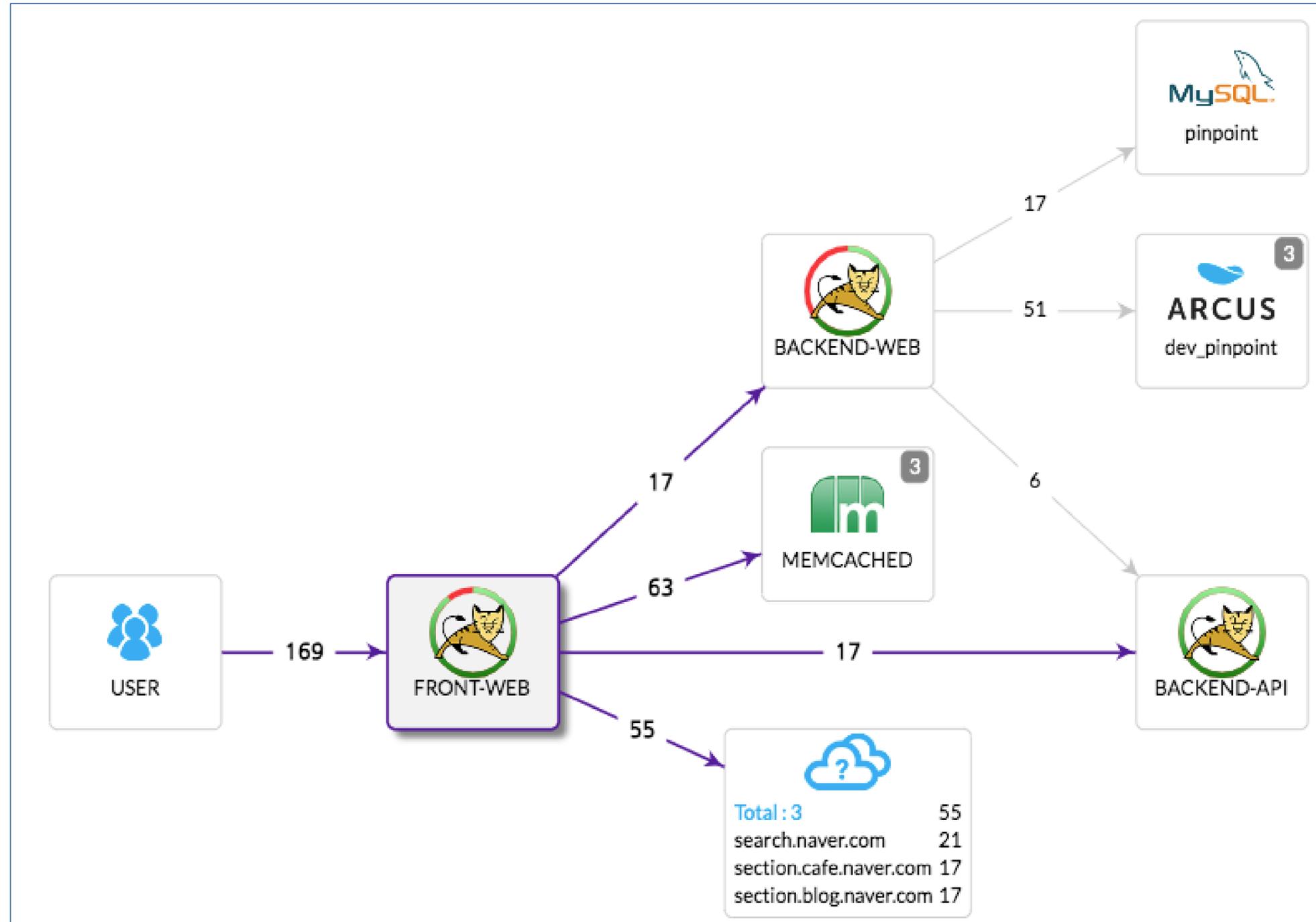
-javaagent:\$PINPOINT_HOME/pinpoint-bootstrap-1.9.0.jar

-Dpinpoint.agentId=\$AGENT_ID -Dpinpoint.applicationName=\$APPLICATION_NAME"



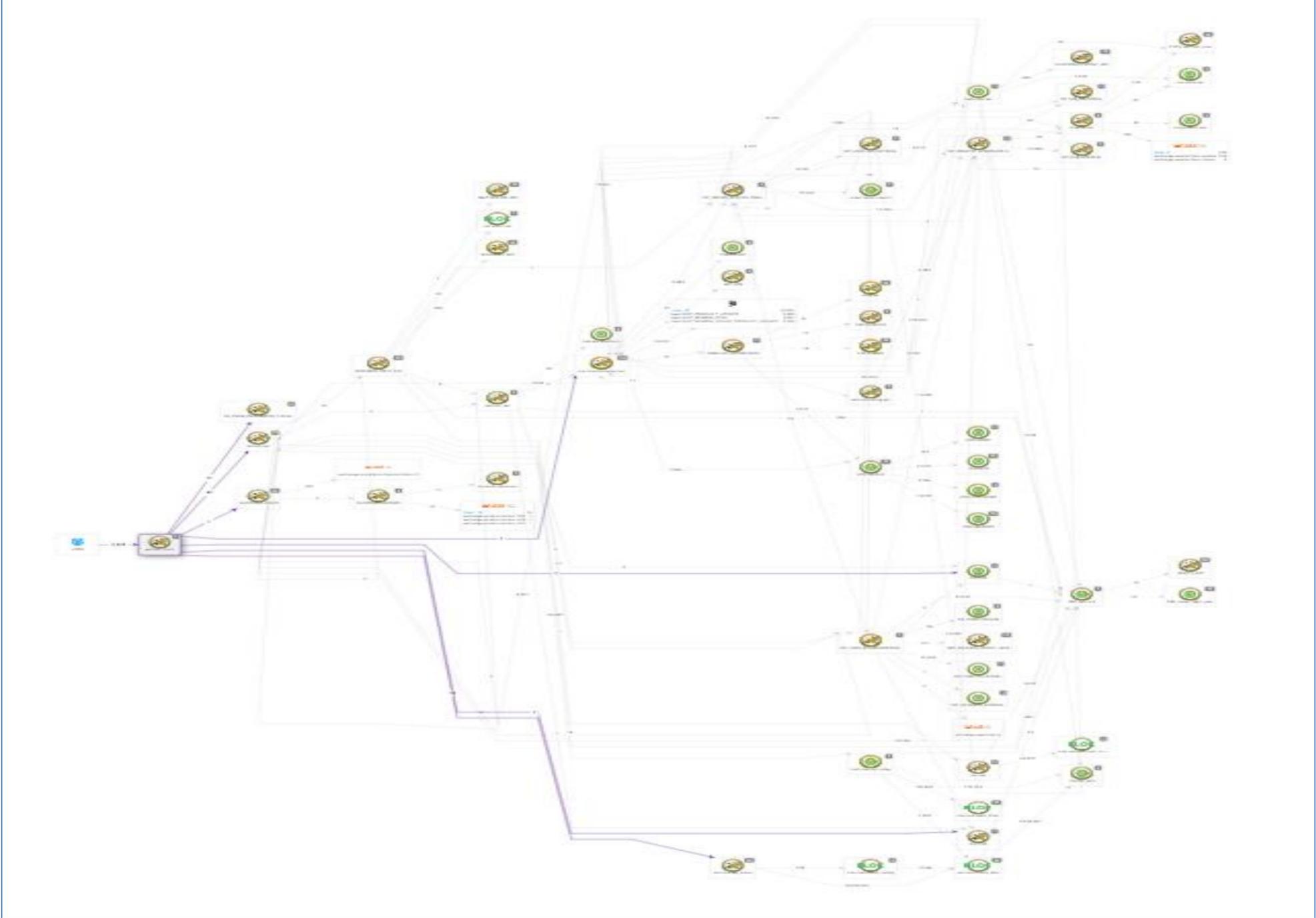
ServerMap

DEVIEW
2019



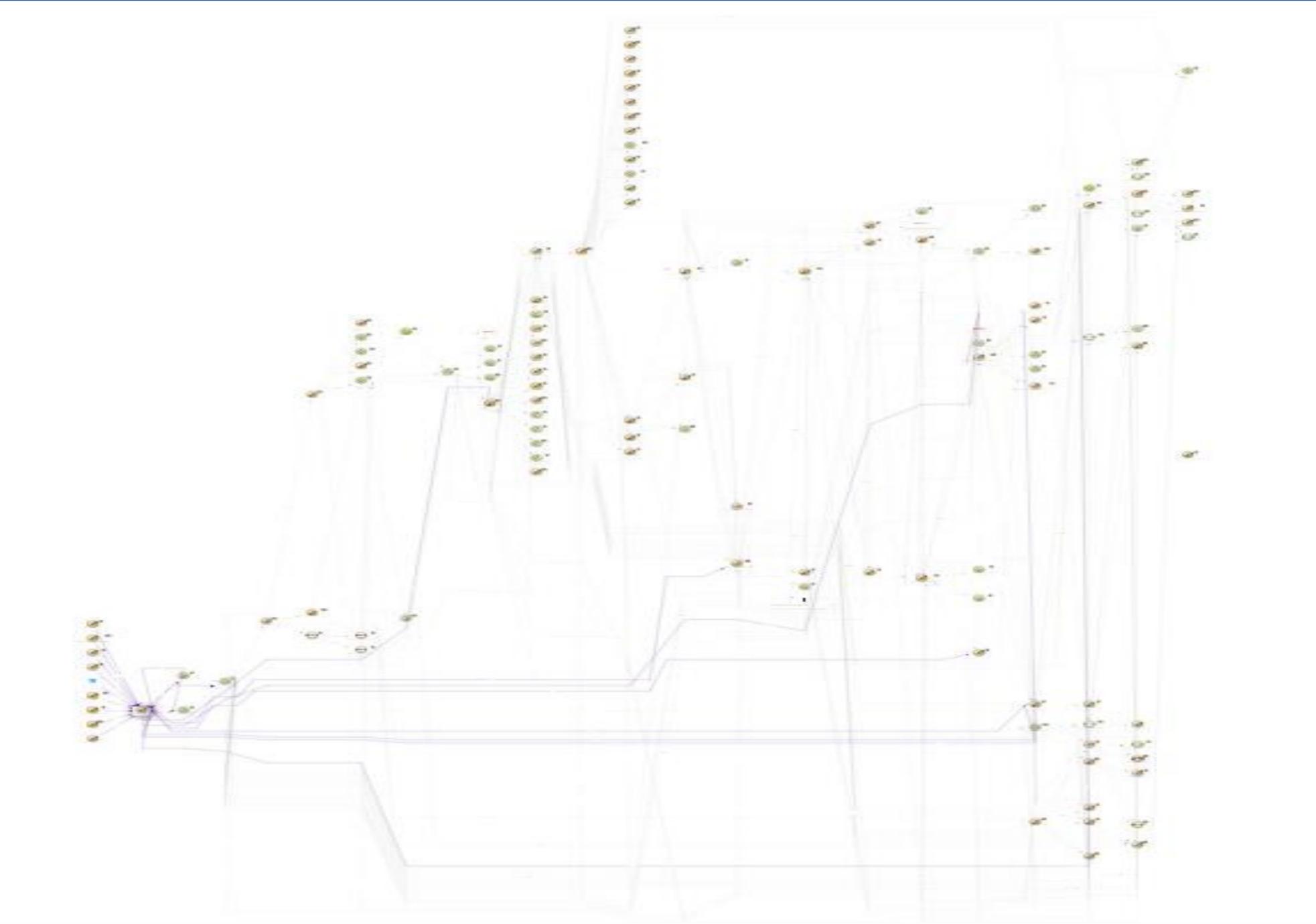
ServerMap

DEVIEW
2019

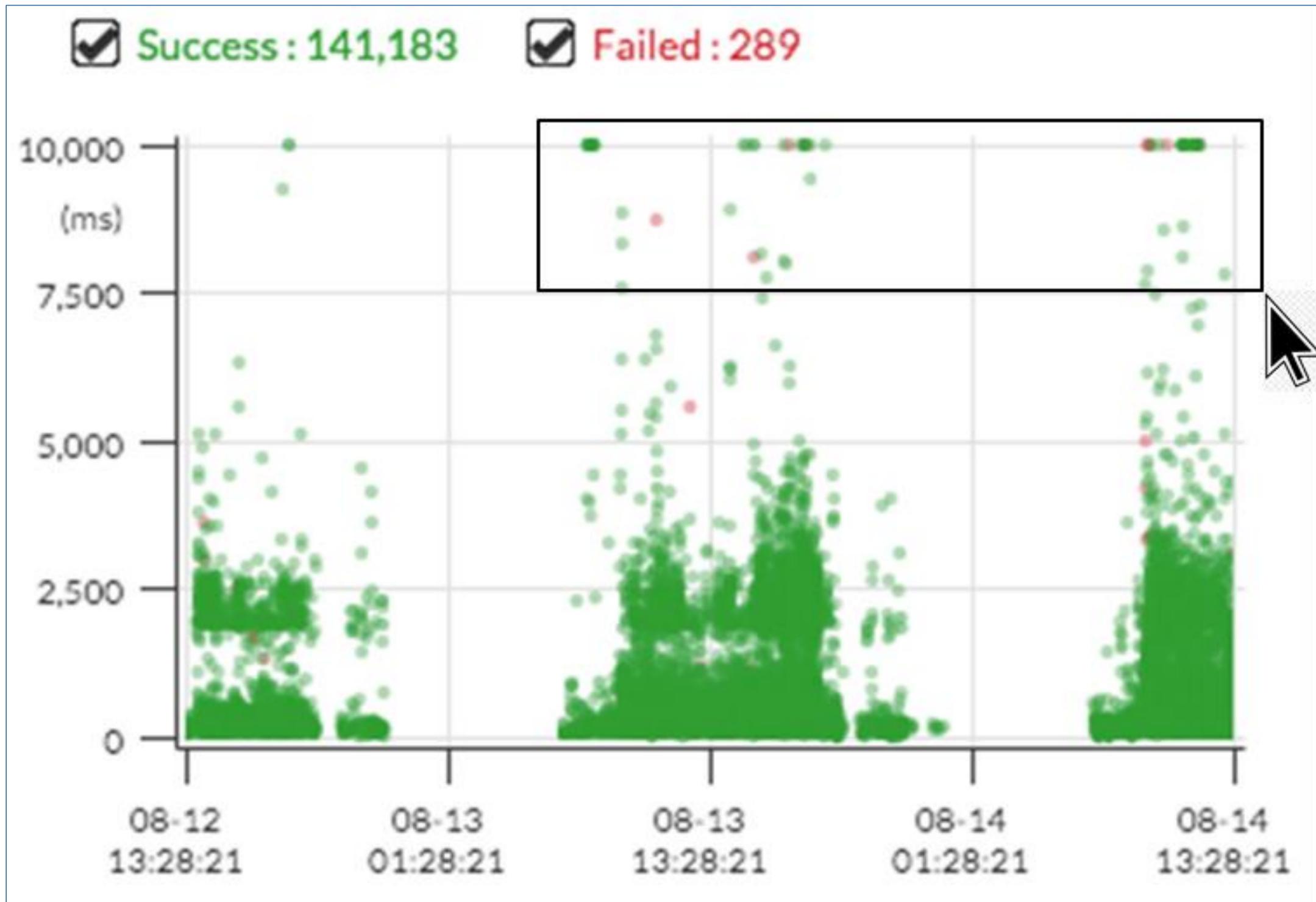


ServerMap

DEVIEW
2019



ScatterChart



Distributed Call tree

DEVIEW
2019

Transaction list

#	Start Time	Path	Res. (ms)	Exception	Agent	Client IP	Transaction
20	06/10 10:42:23 488	frontend pinpoint	728		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280917
19	06/10 10:42:23 477	frontend pinpoint	681		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280918
18	06/10 10:42:23 466	frontend pinpoint	602		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280919
17	06/10 10:42:18 591	backend pinpoint	81		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280920
21	06/10 10:42:20 824	frontend pinpoint	4		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280921
2	06/10 10:41:34 418	frontend pinpoint	3		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280922
13	06/10 10:42:04 775	frontend pinpoint	3		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280923
3	06/10 10:41:37 834	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280924
8	06/10 10:41:42 518	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280925
6	06/10 10:41:45 554	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280926
7	06/10 10:41:48 588	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280927
9	06/10 10:41:53 642	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280928
10	06/10 10:41:55 665	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280929
11	06/10 10:41:58 698	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280930
12	06/10 10:42:01 733	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280931
14	06/10 10:42:10 851	frontend pinpoint	2		FourWAS2	127.0.0.1	FourWAS2-1553844190736-280932

Code level call stack

Method	Argument	Start Time	Gap (ms)	Exec (ms)	Exec (%)	Self (ms)	Class	API	Agent	Application
Service Process	frontend pinpoint	10:42:20 447	0	602	100	0	StandardServlet	TOMCAT	FourWAS2	FRONT WEB
doGet	300	10:42:20 457	0	602	100	0	StandardServlet	TOMCAT	FourWAS2	FRONT WEB
doPost	127.0.0.1	10:42:20 457	0	602	100	4	ForwardController	SPRING	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:20 458	1	398	66	39	DispatchServlet	SPRING	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:20 458	0	4	1	1	Controller	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:20 458	0	1	1	1	AbstractServlet	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:20 459	0	2	1	2	HttpServletResponse	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:20 462	0	1	1	1	HttpServletResponse	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:21 054	39	2	1	0	Controller	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:21 054	0	1	1	1	AbstractServlet	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:21 055	0	1	1	1	HttpServletResponse	HTTP	FourWAS2	FRONT WEB
doPost	Request request, Response response	10:42:21 055	0	0	0	0	StandardServlet	TOMCAT	ApWAS2	BACKEND API
doPost	Request request, Response response	10:42:21 055	0	0	0	0	StandardServlet	TOMCAT	ApWAS2	BACKEND API

Distributed Call tree

DEVIEW
2019

Application : /emeroad.pinpoint TransactionId : FrontWAS2^1563931395270^189... AgentId : FrontWAS2 ApplicationName : FRONT-WEB

Call Tree Server Map Timeline Mixed View nelo Self >= 1000(ms) Complete

Method	Argument	Start Time	Gap(ms)	Exec(ms)	Exec(%)	Self(ms)	Class	API	Application	Agent
Servlet.service()	emeroad.pinpoint	17:30:48 004	0	295		0		TOMCAT	FRONT-WEB	FrontWAS2
http.statusCode	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		17:30:48 004	0	295		0	StandardHostValve	TOMCAT_ME...	FRONT-WEB	FrontWAS2
doGet(HttpServletRequest request, HttpServletResponse response)		17:30:48 004	0	295		2	FrameworkServlet	SPRING	FRONT-WEB	FrontWAS2
demo2()		17:30:48 006	2	293		280	DemoController	SPRING_BE...	FRONT-WEB	FrontWAS2
execute(HttpUriRequest request, Response response)		17:30:48 286	280	13		0	CloseableHttpCli...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
open(HttpRoute route, HttpContext context)	dev.pinpoint-workload	17:30:48 286	0	0		0	AbstractPooledCo...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
execute()	emeroad.pinpoint	17:30:48 286	0	13		13	HttpRequestExecu...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
http.io	write: 0ms read: 13m									
Servlet.service()	emeroad.pinpoint	17:30:48 286	0	12		0		TOMCAT	BACKEND-WEB	BackendWAS1
http.statusCode	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		17:30:48 286	0	12		0	StandardHostValve	TOMCAT_ME...	BACKEND-WEB	BackendWAS1
doPost(HttpServletRequest request, HttpServletResponse response)		17:30:48 286	0	12		9	FrameworkServlet	SPRING	BACKEND-WEB	BackendWAS1
backendweb()		17:30:48 295	9	3		0	DemoController	SPRING_BE...	BACKEND-WEB	BackendWAS1
cacheServiceTmel()		17:30:48 295	0	1		0	CacheServiceTmel	SPRING_BE...	BACKEND-WEB	BackendWAS1

Distributed Call tree

DEVIEW
2019

Application : /emeroad.pinpoint TransactionId : FrontWAS2^1563931395270^189... AgentId : FrontWAS2 ApplicationName : FRONT-WEB

Call Tree Server Map Timeline Mixed View nelo Self >= 1000(ms) 🔍 ? Complete 🔄

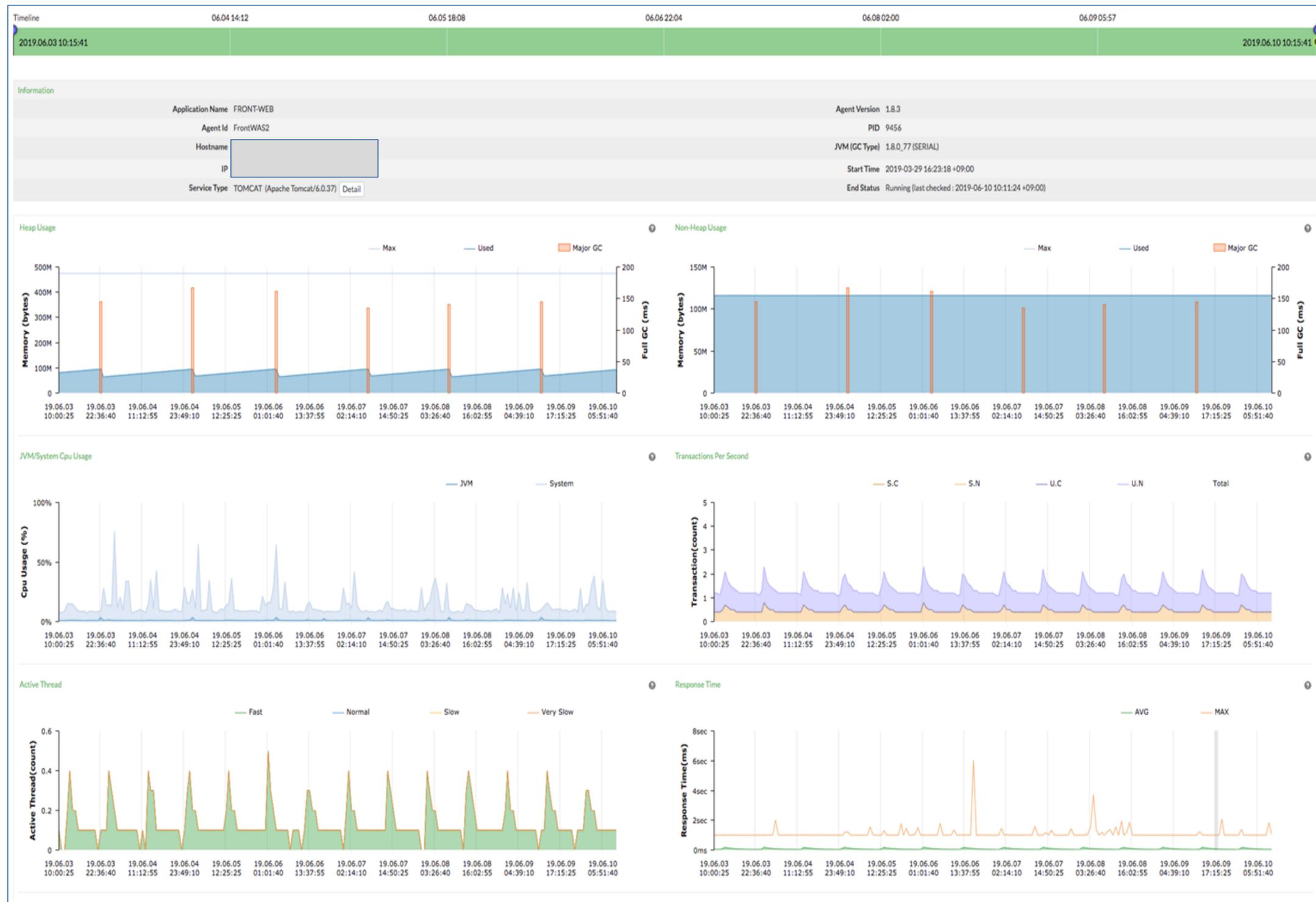
Method	Argument	Start Time	Gap(ms)	Exec(ms)	Exec(%)	Self(ms)	Class	API	Application	Agent
Servlet Process	/emeroad.pinpoint	17:30:48 004	0	295	100%	0		TOMCAT	FRONT-WEB	FrontWAS2
http.status.code	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		17:30:48 004	0	295	100%	0	StandardHostValve	TOMCAT_ME...	FRONT-WEB	FrontWAS2
doGet(HttpServletRequest request, HttpServletResponse response)		17:30:48 004	0	295	100%	2	FrameworkServlet	SPRING	FRONT-WEB	FrontWAS2
doDispatch(HttpServletRequest request, HttpServletResponse response)		17:30:48 006	2	293	100%	280	DemoController	SPRING_BE...	FRONT-WEB	FrontWAS2
execute(HttpUriRequest request, HttpResponse response)		17:30:48 286	280	13	100%	0	CloseableHttpCli...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
open(HttpRoute route, HttpContext context)	dev-pinpoint-workload	17:30:48 286	0	0	100%	0	AbstractPooledCo...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
execute(HttpRequest request, HttpResponse response)	/backendweb.pinpoint	17:30:48 286	0	13	100%	13	HttpRequestExecu...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
http.status.code	200									
http.io	write: 0ms, read: 13m									
Servlet Process	/backendweb.pinpoint	17:30:48 286	0	12	100%	0		TOMCAT	BACKEND-WEB	BackendWAS1
http.status.code	200									
REMOTE_ADDRESS	127.0.0.1									
doPost(HttpServletRequest request, HttpServletResponse response)		17:30:48 286	0	12	100%	0	StandardHostValve	TOMCAT_ME...	BACKEND-WEB	BackendWAS1
doPost(HttpServletRequest request, HttpServletResponse response)		17:30:48 286	0	12	100%	9	FrameworkServlet	SPRING	BACKEND-WEB	BackendWAS1
backendweb()		17:30:48 295	9	3	100%	0	DemoController	SPRING_BE...	BACKEND-WEB	BackendWAS1
CacheServiceImpl		17:30:48 295	0	1	100%	0	CacheServiceImpl	SPRING_BE...	BACKEND-WEB	BackendWAS1

FRONT-WEB (highlighted in black box)

BACKEND-WEB (highlighted in yellow box)

Inspector

DEVIEW
2019



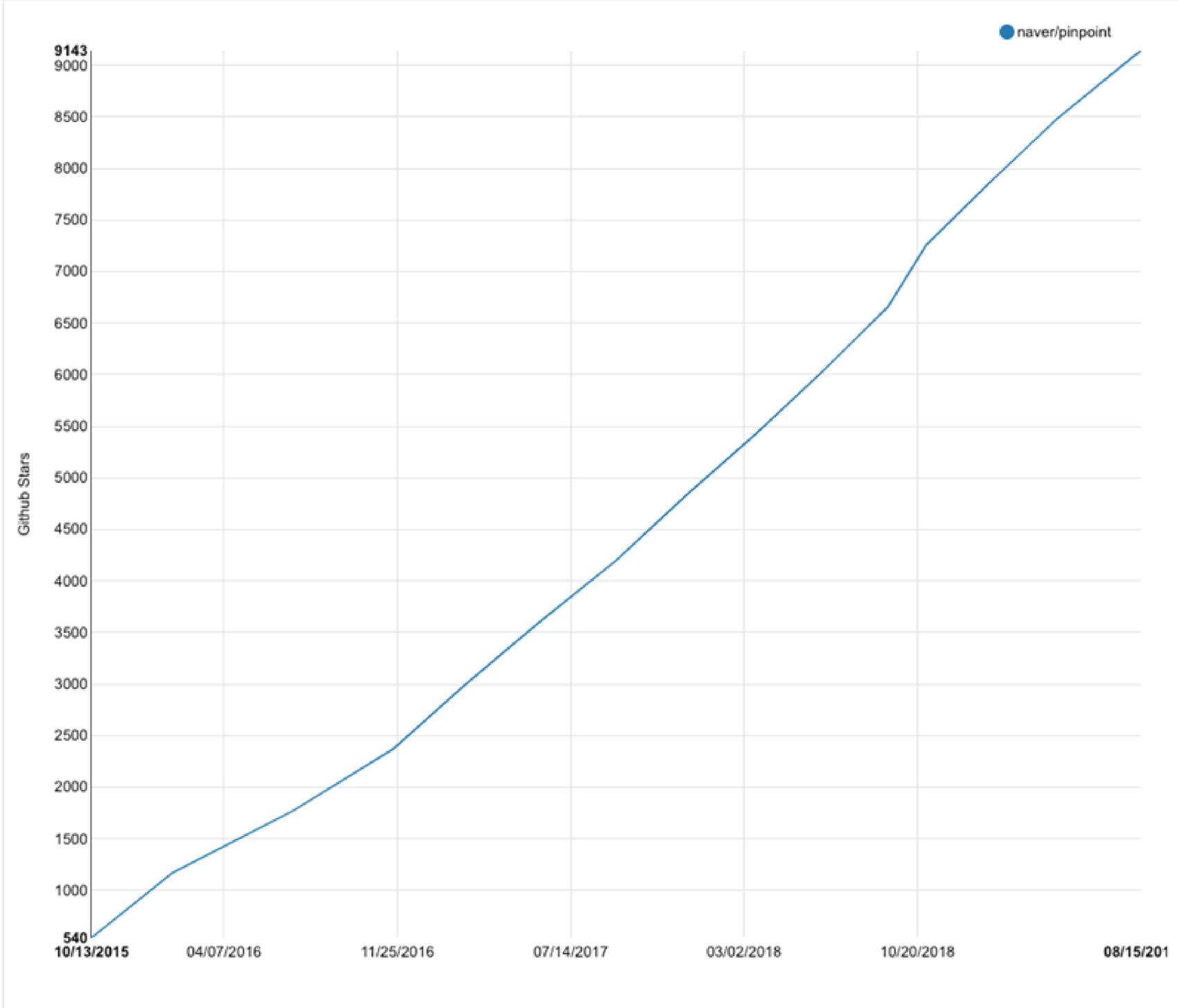
- Timeline
- Memory
- CPU
- JVM GC
- TPS
- Active Thread
- Response Time
- File Descriptor
- Buffer Usage
- Data Source

Pinpoint is

Opensource

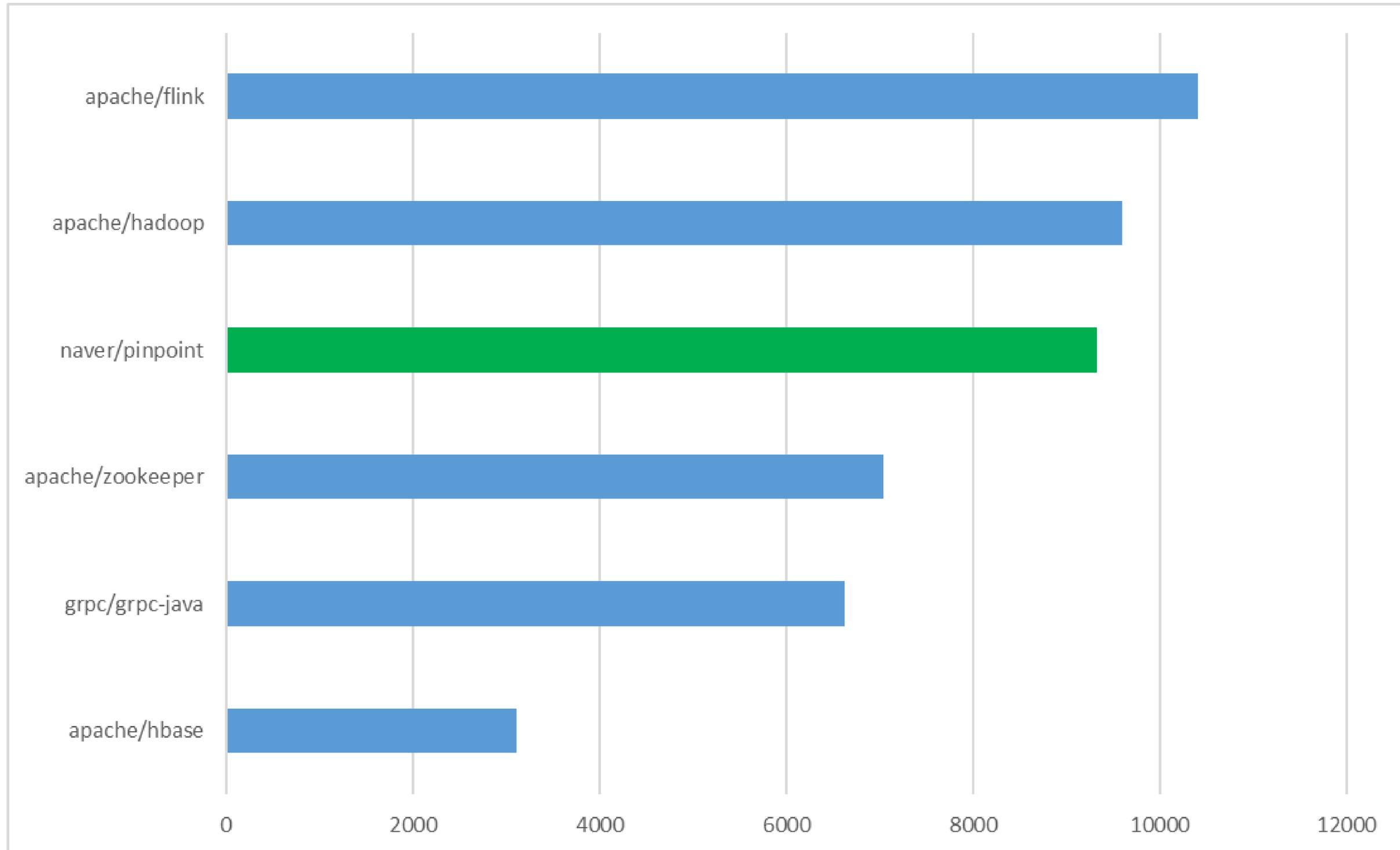
Github star

DEVIEW
2019

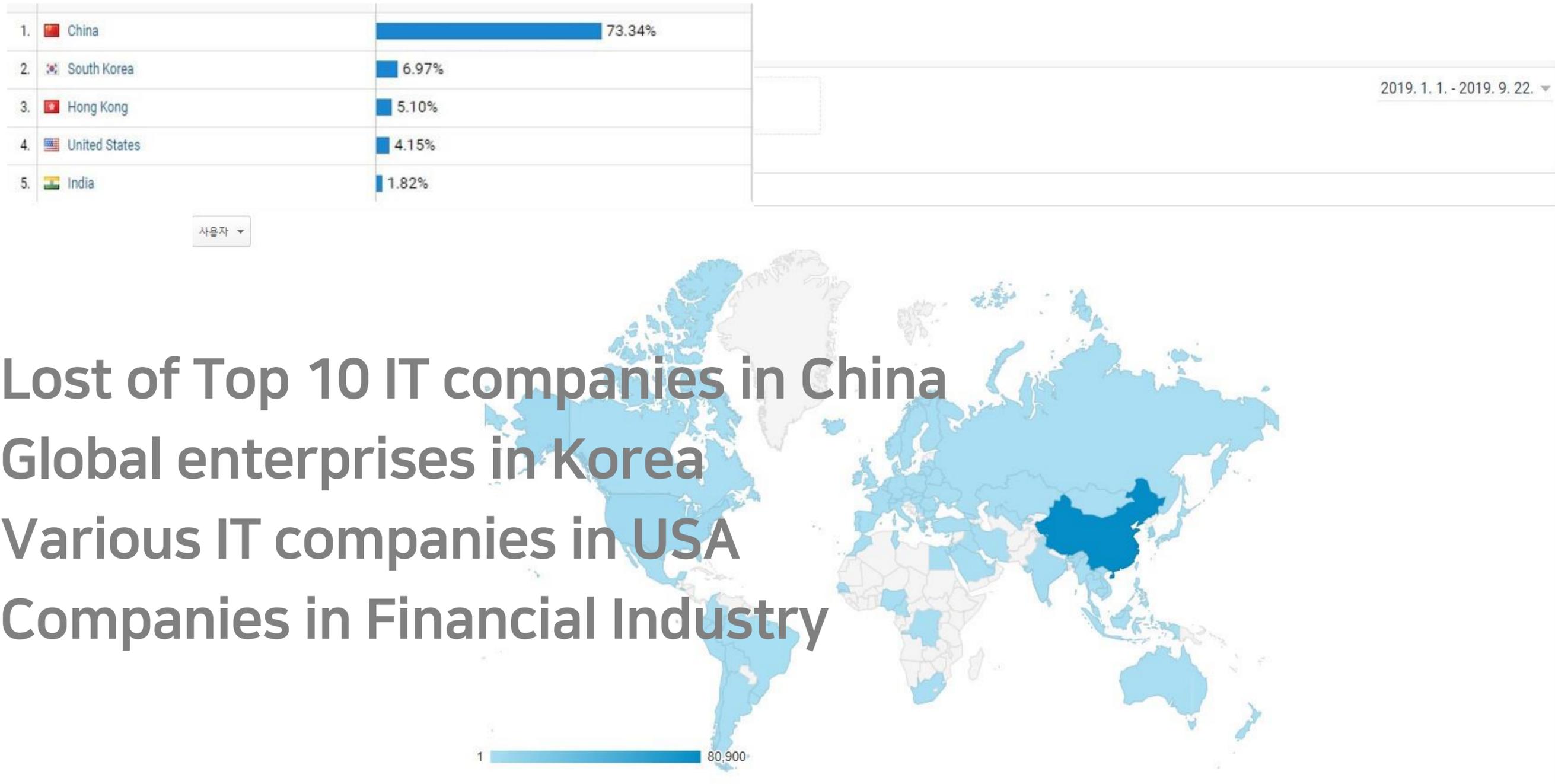


Github star

DEVIEW
2019



Active usage



- Lost of Top 10 IT companies in China
- Global enterprises in Korea
- Various IT companies in USA
- Companies in Financial Industry

Pinpoint 코어가

궁금하다면 데뷰 2015

다시보기

2. 실시간 observability 확보하기

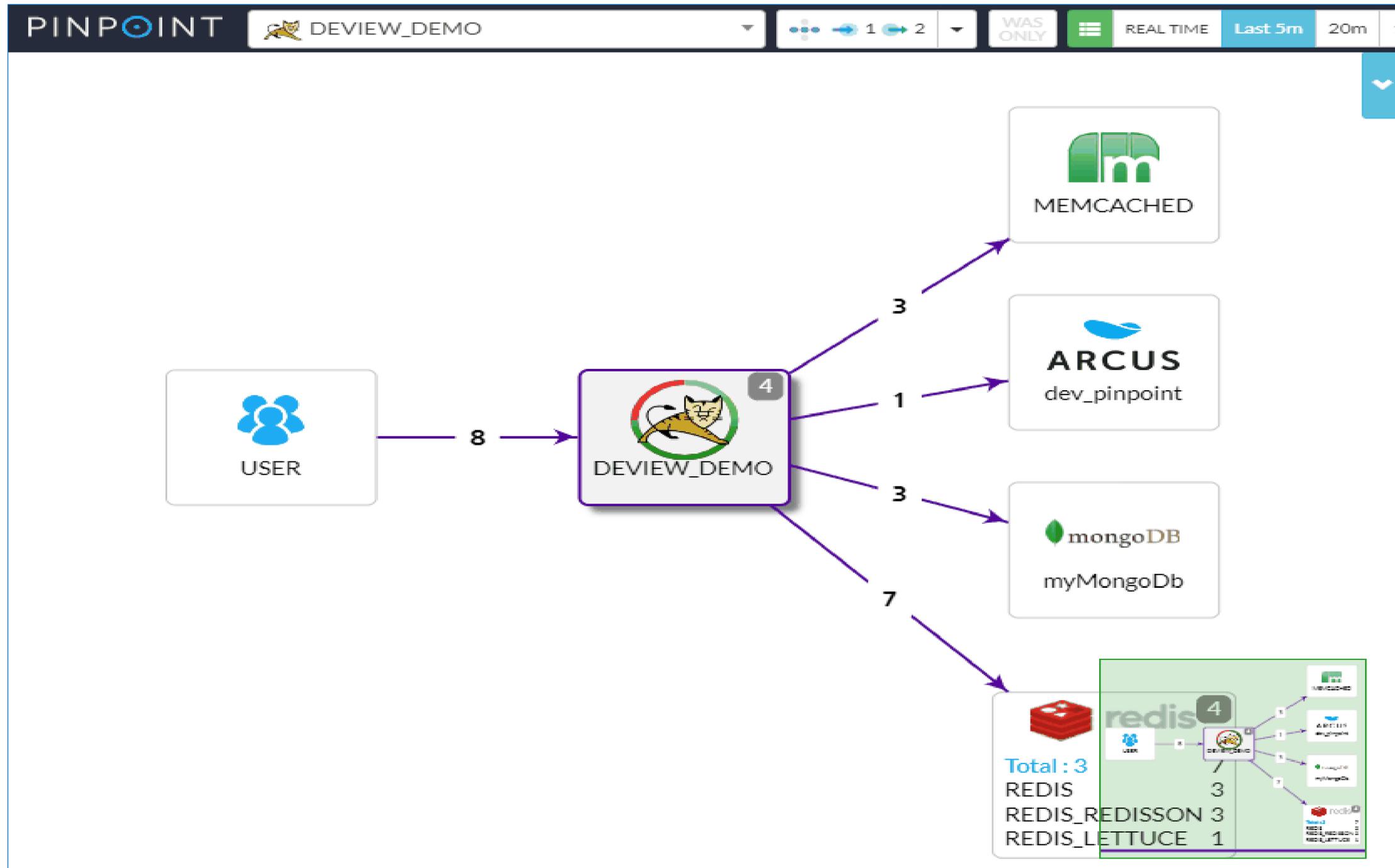
해결해야 하는 문제

현재 각 에이전트가 처리하고 있는 요청은 얼마나 되는지?

예상하지 못한 문제로 끝나지 않는 요청이 있다면 어떻게 확인하는지?

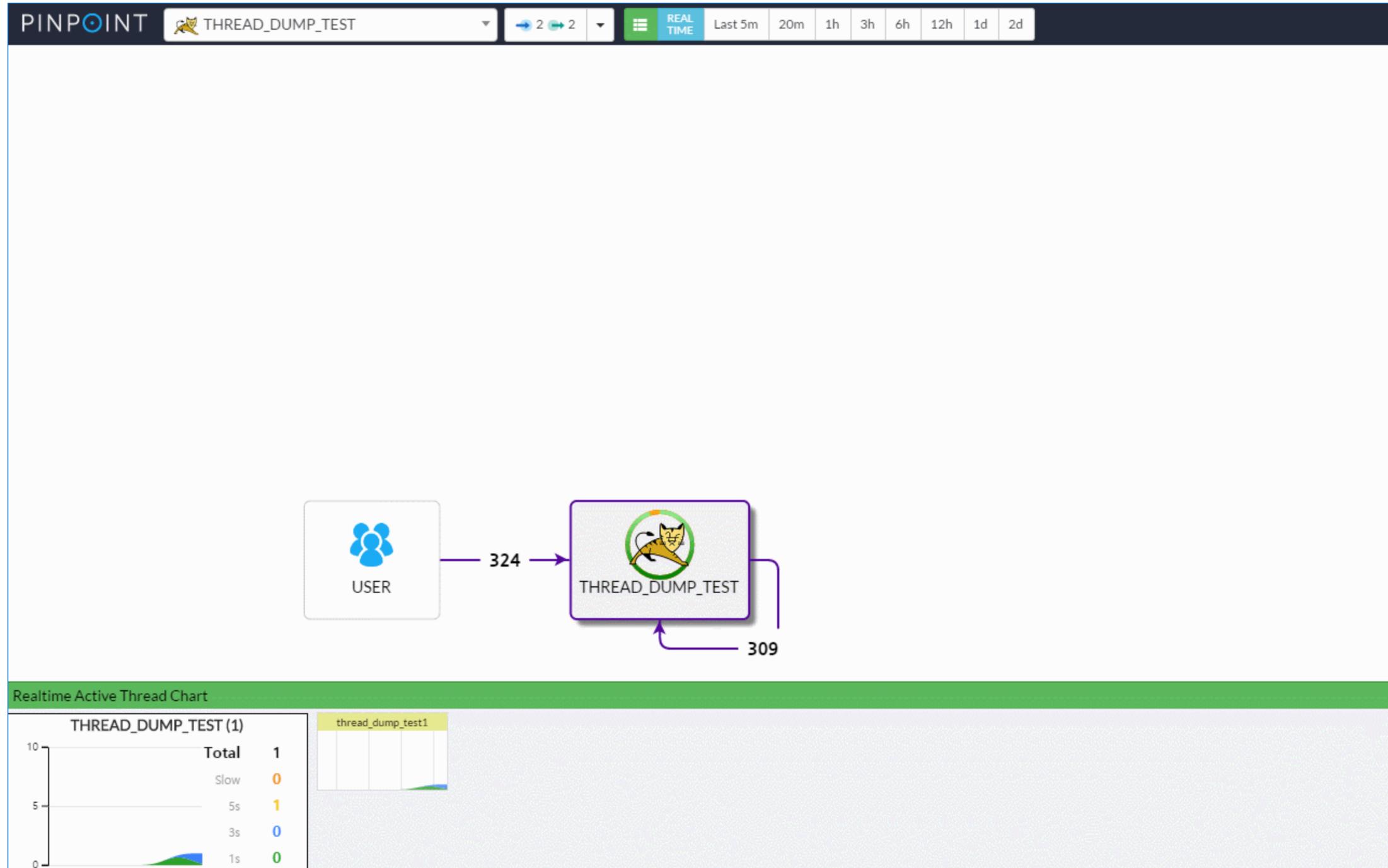
해결 (리얼타임기능)

DEVIEW
2019



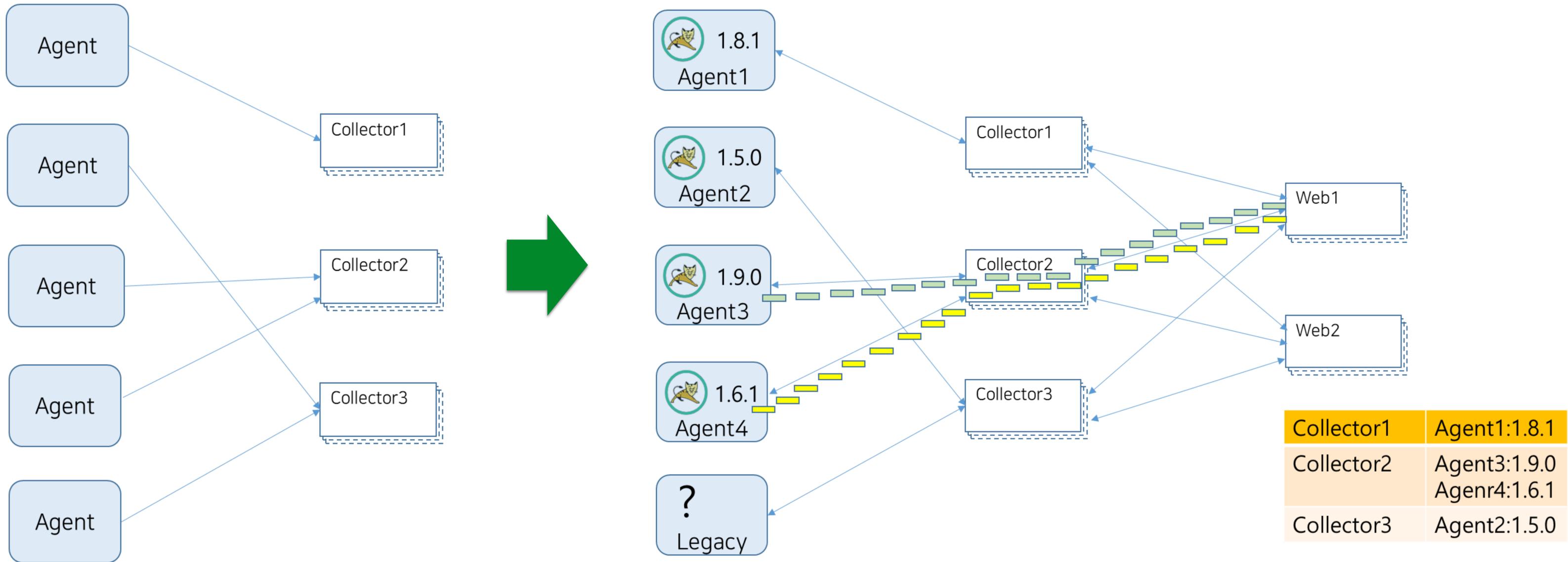
해결 (실시간 스레드덤프 기능)

DEVIEW
2019



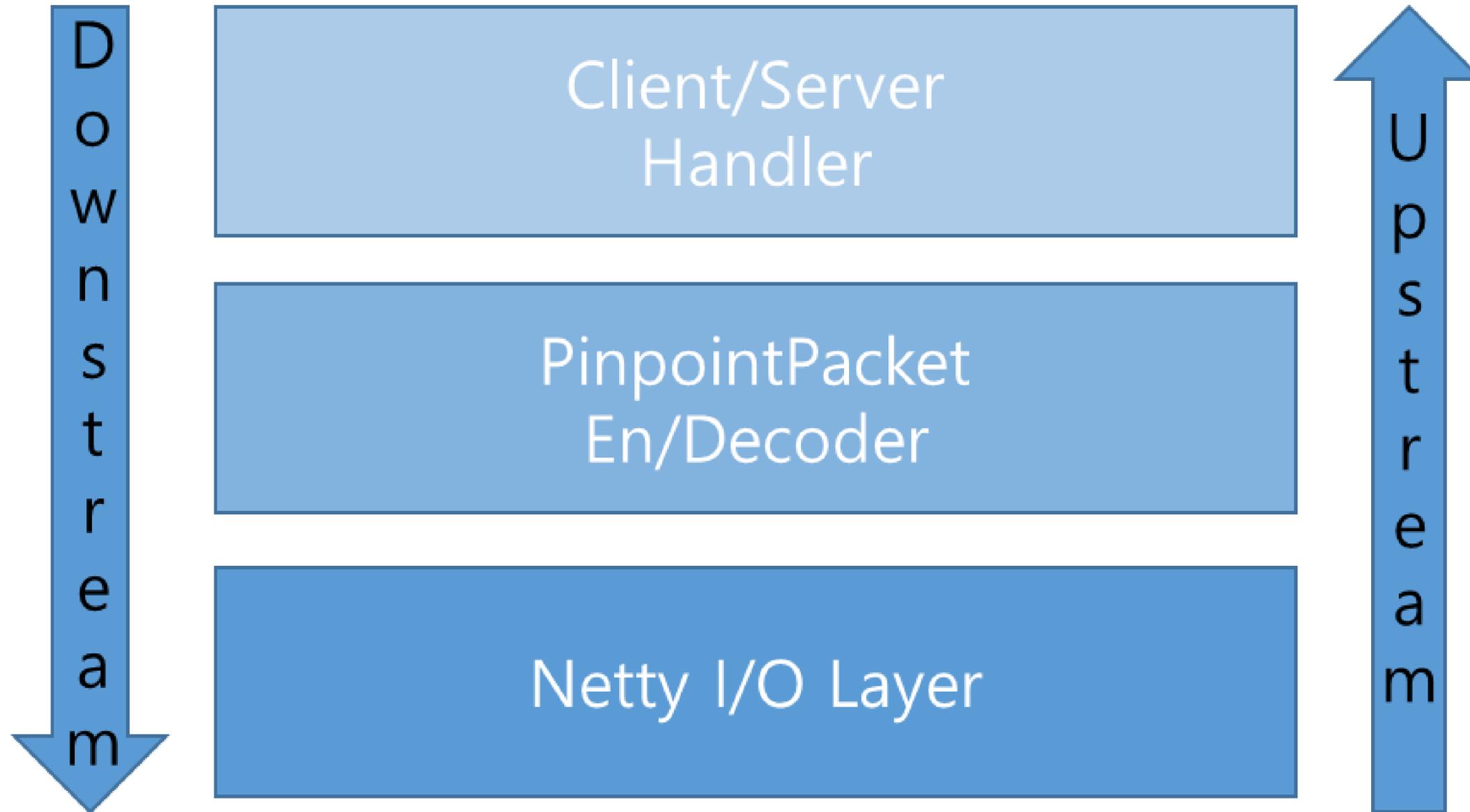
해결 방법 : 구조 변경

DEVIEW
2019



1. RPC 구조

DEVIEW
2019



1. PinpointPacket 이란?

- 핀포인트내에서 데이터를 주고 받는 최소 단위
- 각 패킷은 역할을 가짐 (Send, Request, ... 등)
- Packet Format (Binary Message)

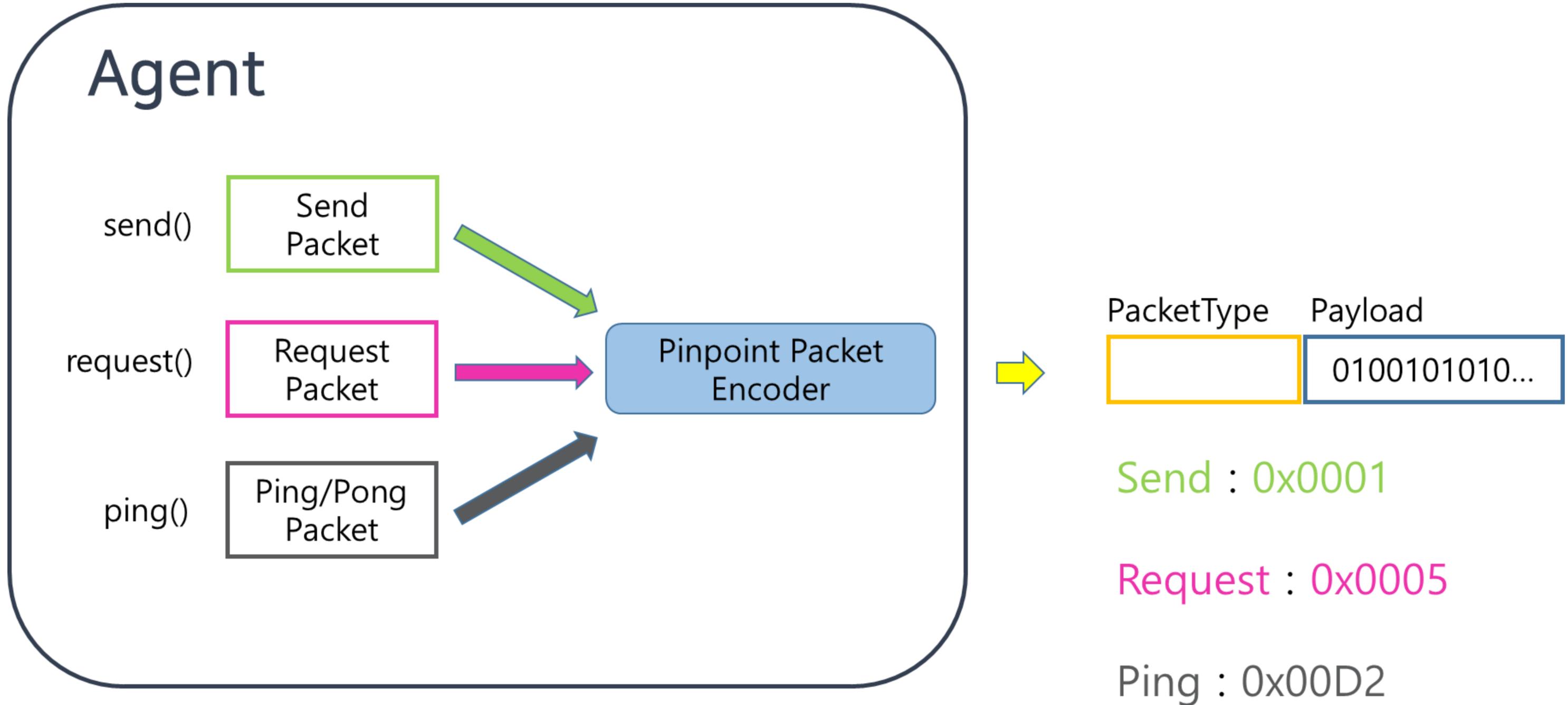
2byte

Size Not defined

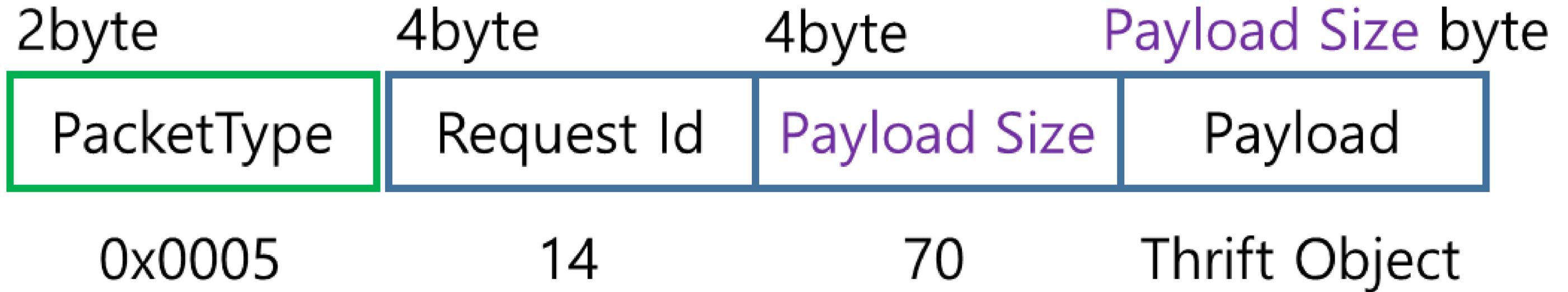
PacketType

Depends on Packet

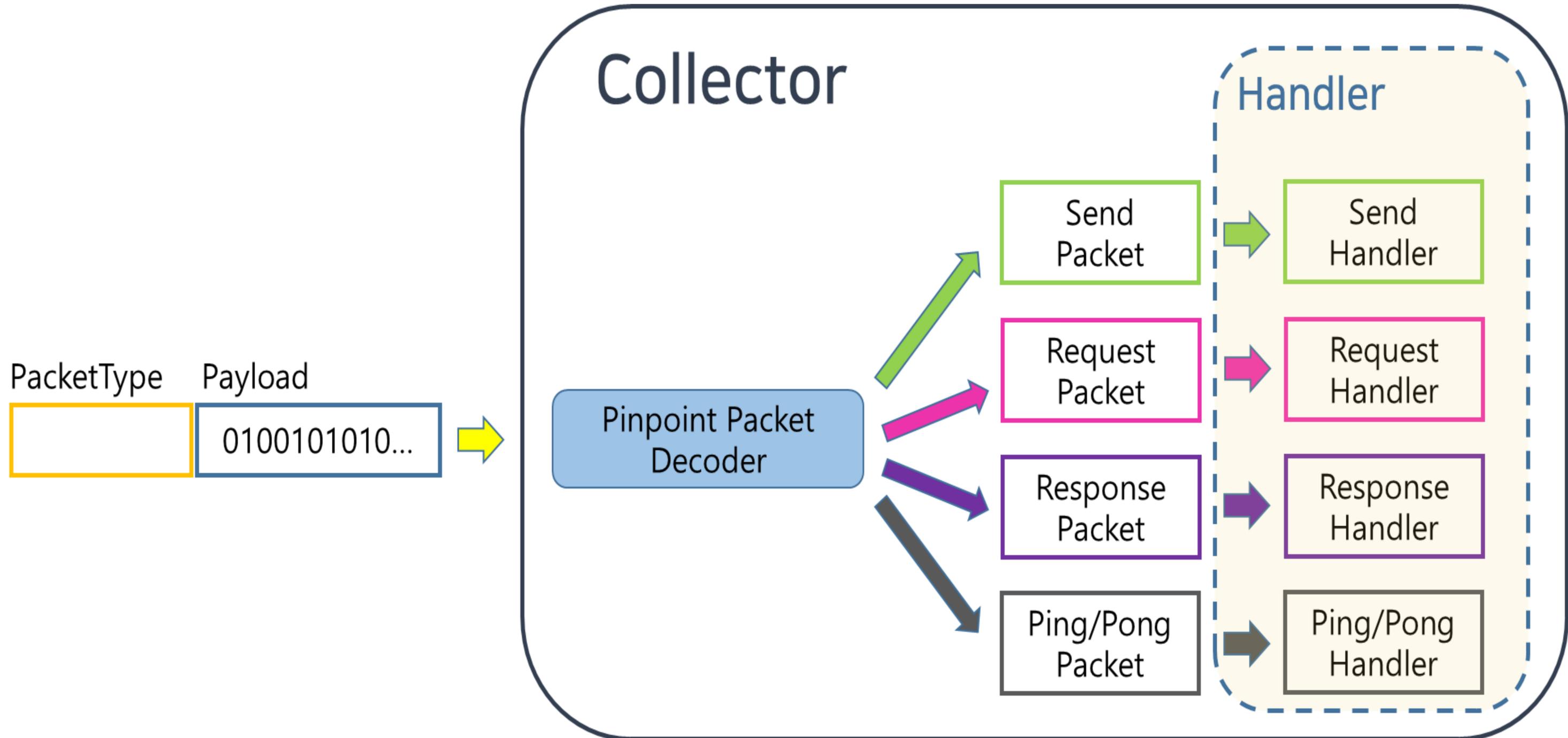
1. 에이전트에서 Packet 송신



1. RequestPacket 구조

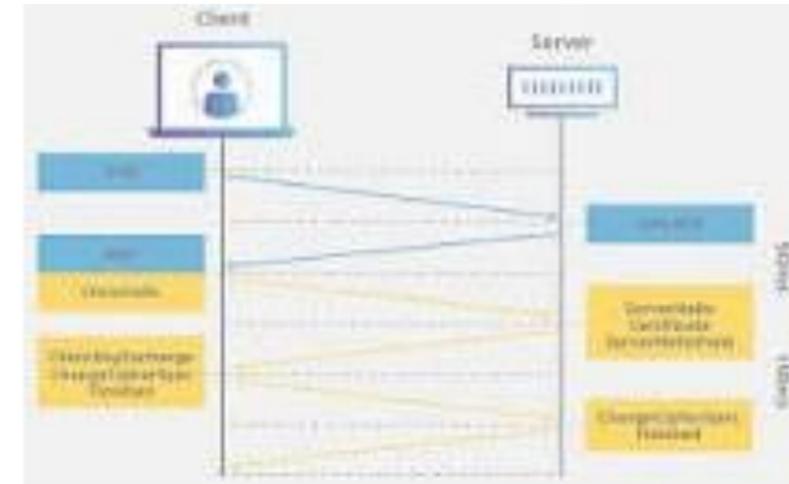


1. 콜렉터에서 Packet 수신



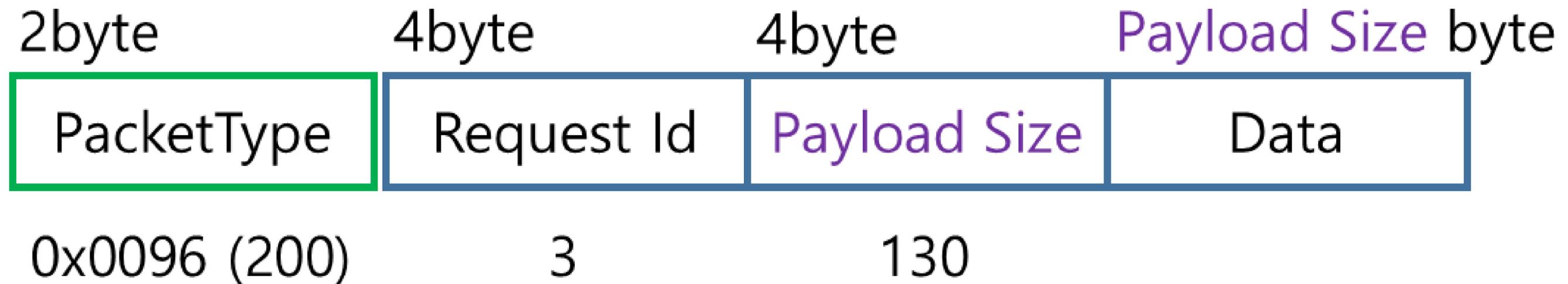
2. Handshake

A **TLS handshake** is the process that kicks off a communication session that uses **TLS** encryption. During a **TLS handshake**, the two communicating sides exchange messages to acknowledge each other, verify each other, establish the encryption algorithms they will use, and agree on session keys.



2. HandshakePacket 추가

- ControlHandshakePacket



- ControlHandshakeResponsePacket

2. Handshake Data

Handshake Data

ApplicationName

AgentId

StartTimeStamp

Version

에이전트 식별자

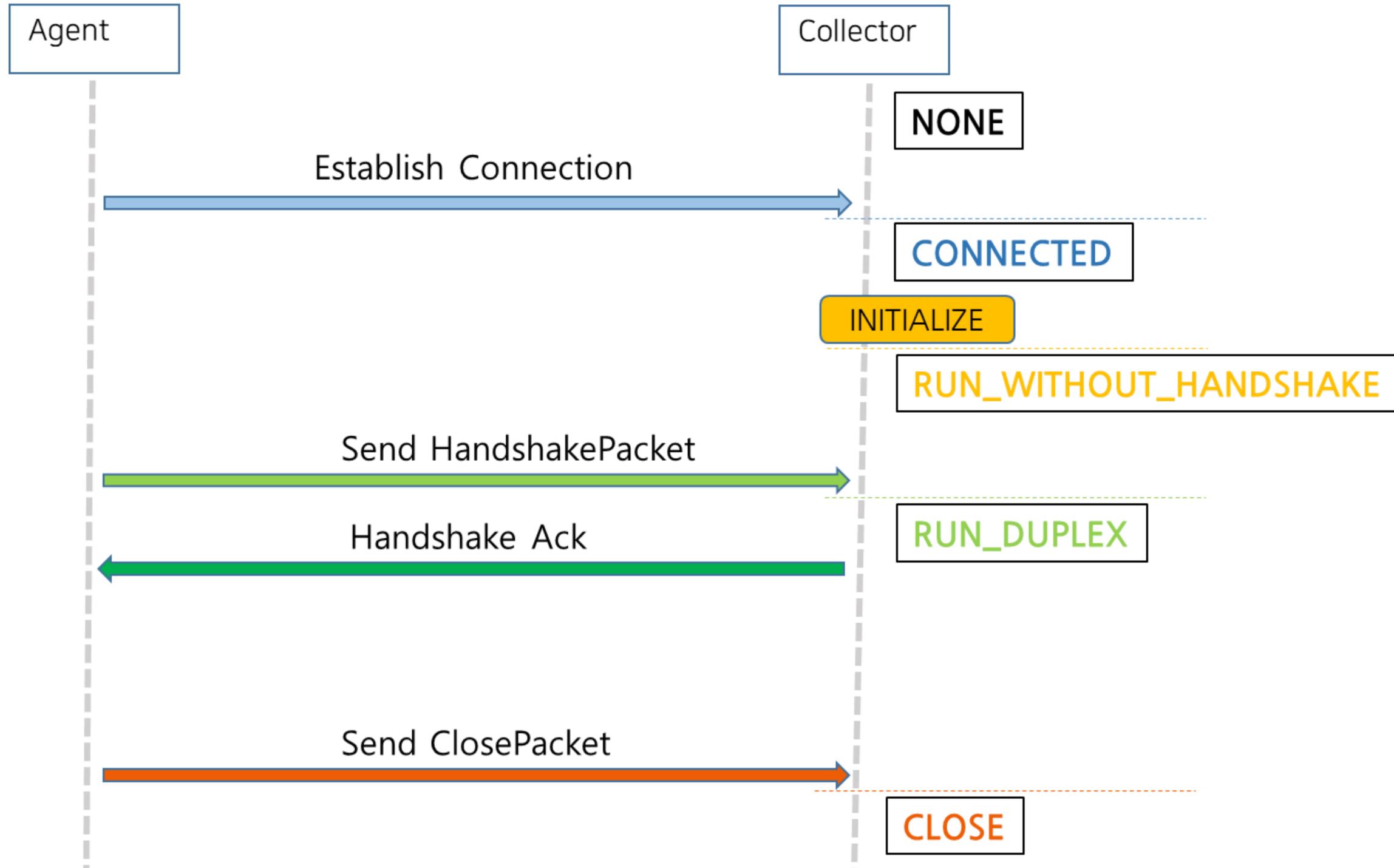
SupportServer

SupportCommandList

요청 지원 여부 및 지원 API

2. 상태값 생성 및 변경

DEVIEW
2019

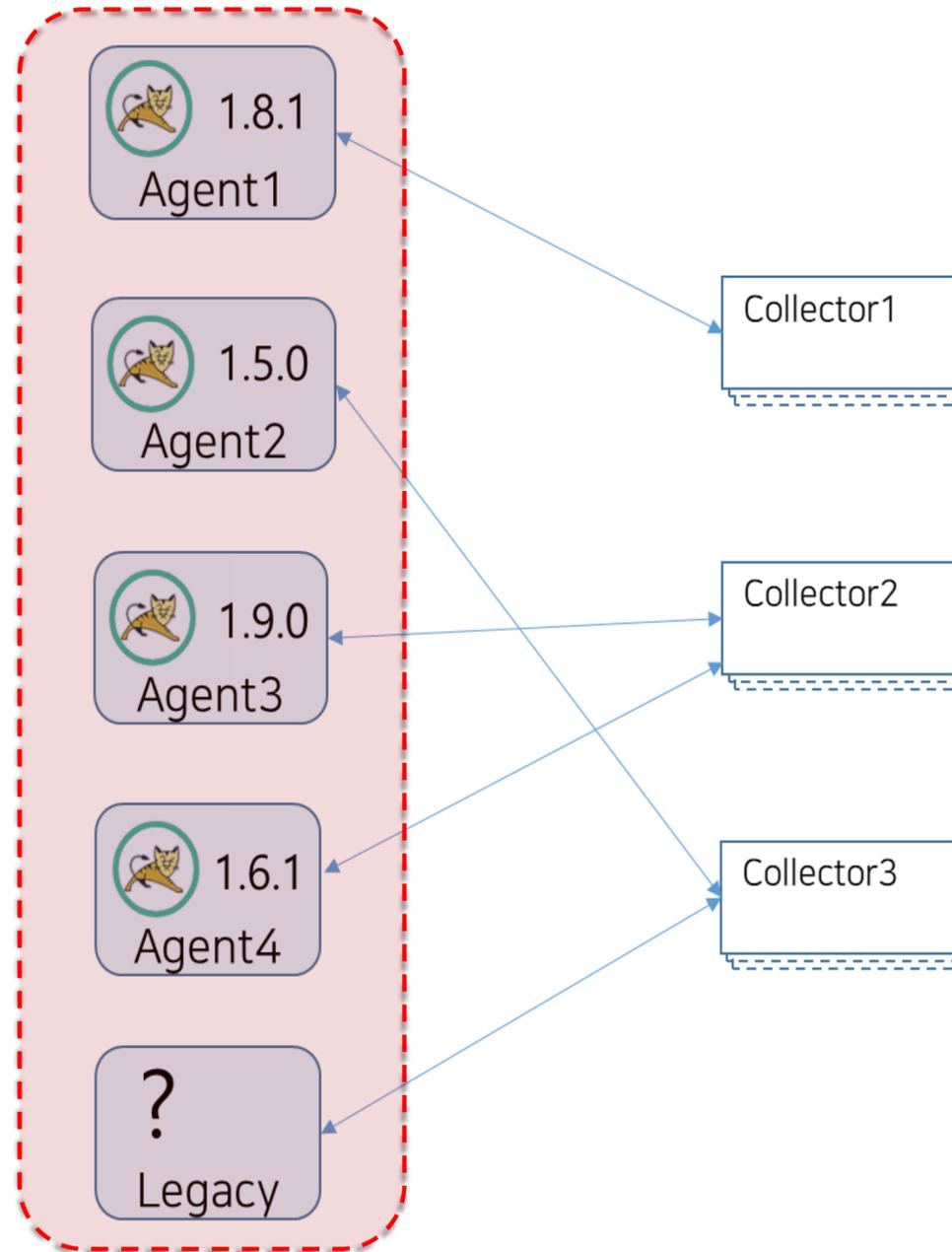


2. 상태 처리 리스너 생성

- StateChangeListener 생성

```
public interface StateChangeListener<S extends PinpointSocket> {  
    void stateUpdated(S pinpointSocket, SocketStateCode updatedStateCode) throws Exception;  
}
```

2. Handshake 결과



3. 에이전트 연결 정보 공유

What is ZooKeeper?



Apache ZooKeeper™

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they

3. 상태 처리 리스너 구현

```
@Override
public void stateUpdated(PinpointServer pinpointServer, SocketStateCode updatedStateCode) {
    logger.info("stateUpdated() started. (PinpointServer={}, updatedStateCode={})", pinpointServer, updatedStateCode);

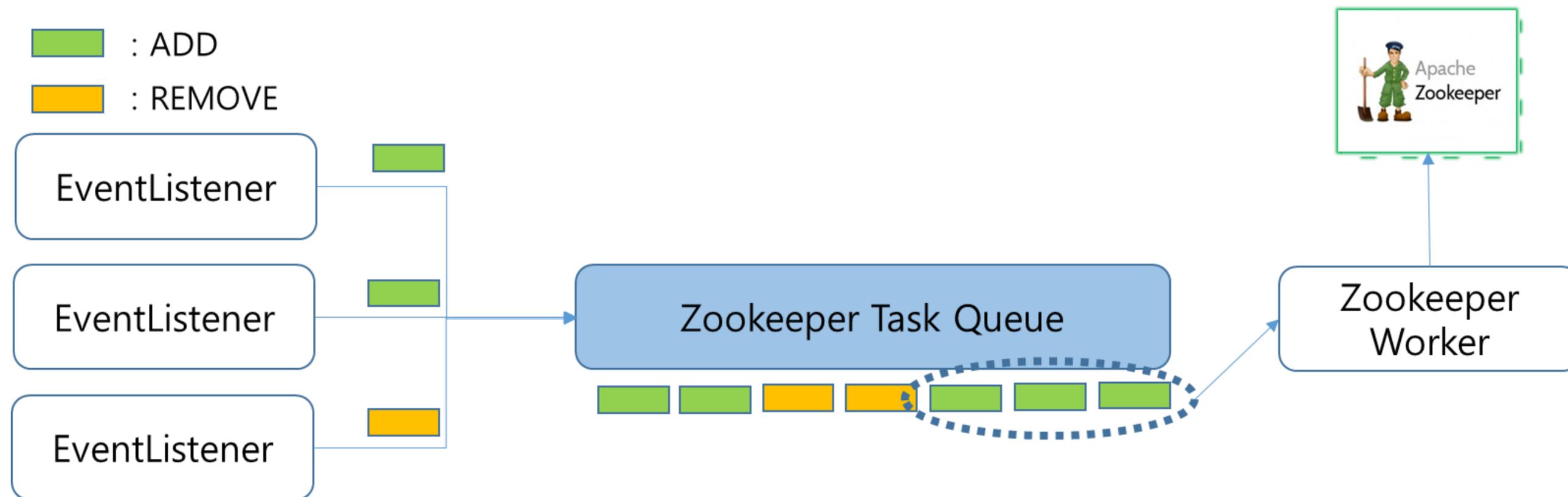
    Map<Object, Object> channelPropertiesMap = pinpointServer.getChannelProperties();
    ChannelProperties channelProperties = channelPropertiesFactory.newChannelProperties(channelPropertiesMap);

    if (SocketStateCode.RUN_DUPLEX == updatedStateCode) {
        ClusterPoint<byte[]> pinpointServerClusterPoint = newClusterPoint(pinpointServer, channelProperties);
        profilerClusterManager.register(pinpointServerClusterPoint);
    } else if (SocketStateCode.isClosed(updatedStateCode)) {
        ClusterPoint<byte[]> pinpointServerClusterPoint = newClusterPoint(pinpointServer, channelProperties);
        profilerClusterManager.unregister(pinpointServerClusterPoint);
    }
}
```

3. ZooKeeper 이벤트 저장

- Worker Queue 형태

ZooKeeper 부하를 줄이기 위해서 같은 형태의 이벤트는 함께 처리



3. 콜렉터 노드 생성

- 연결되어 있는 에이전트의 정보를 저장하기 위한 노드 생성

```
/pinpoint-cluster/collector
```

```
- $serverIdentifier (ephemeral)
```

```
- $applicationName:$agentId:$startTimeStamp  
- $applicationName:$agentId:$startTimeStamp  
- $applicationName:$agentId:$startTimeStamp
```

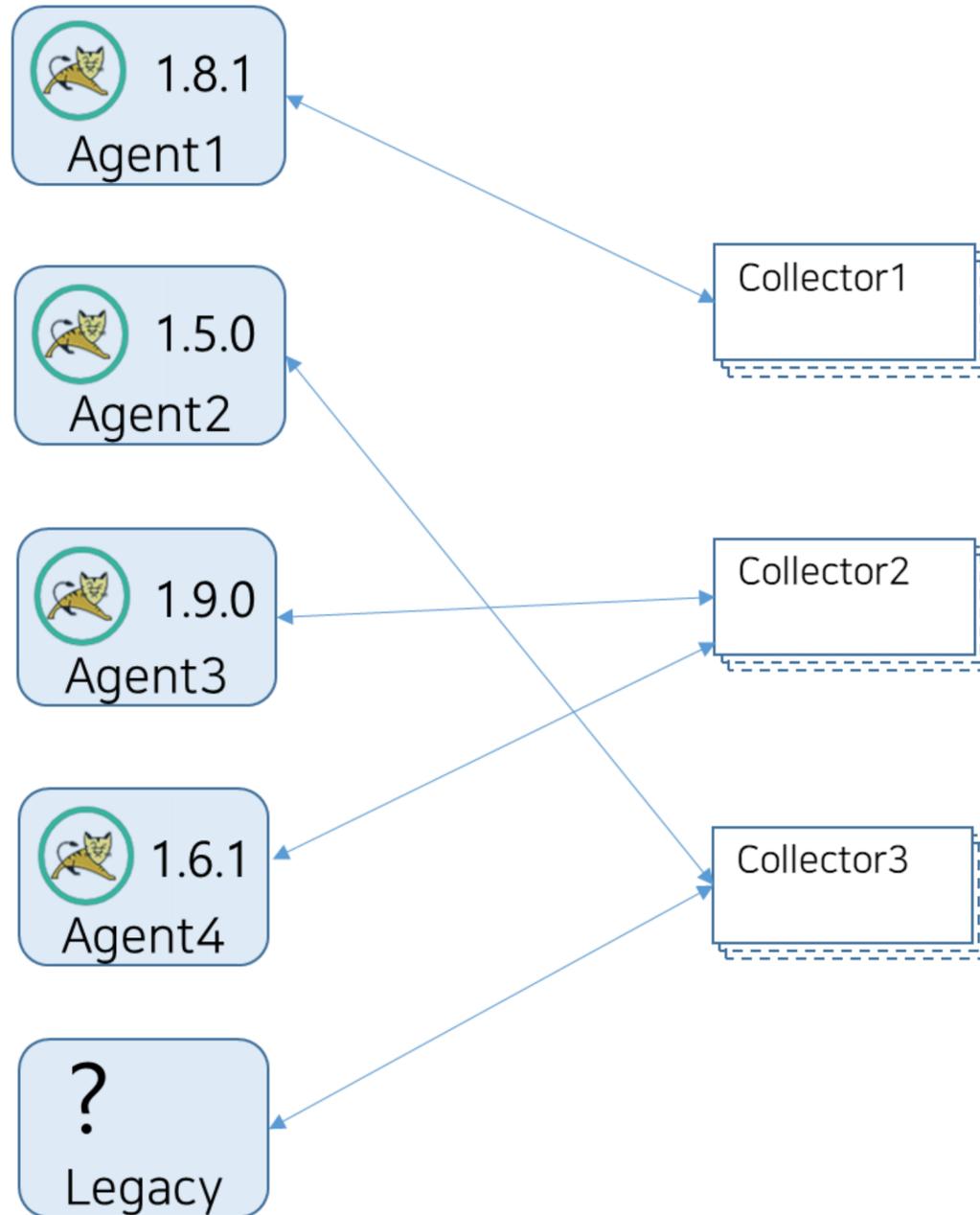
```
- $serverIdentifier (ephemeral)
```

```
- $applicationName:$agentId:$startTimeStamp  
- $applicationName:$agentId:$startTimeStamp  
- $applicationName:$agentId:$startTimeStamp
```

에이전트 식별자

- 노드 이름 “\${pid}@{HostName}”
- 쥬키퍼의 ephemeral 노드 생성
- 노드에 콜렉터와 연결되어있는 모든 에이전트 정보 저장

3. 에이전트 연결 정보 공유 결과



A diagram showing the configuration for a Pinpoint cluster in ZooKeeper. It features a ZooKeeper icon and the text "/pinpoint-cluster". Below it, a circle represents the "/collector" node. A table lists the registered collectors and their associated agents.

Collector1	Agent1:1.8.1
Collector2	Agent3:1.9.0 Agent4:1.6.1
Collector3	Agent2:1.5.0

4. 웹 콜렉터 연동

What is ZooKeeper?



Apache ZooKeeper™

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they

쥬키퍼에 웹 장비의 정보를 저장하여 연결정보 공유

4. 웹 노드 생성

- 웹 모듈의 네트워크 정보 저장을 위한 노드 생성

```
/pinpoint-cluster/web
```

```
-${represtationIp}:port (ephemeral)
```

```
- $serverAddress1  
- $serverAddress2  
- $serverAddress3  
- ...
```

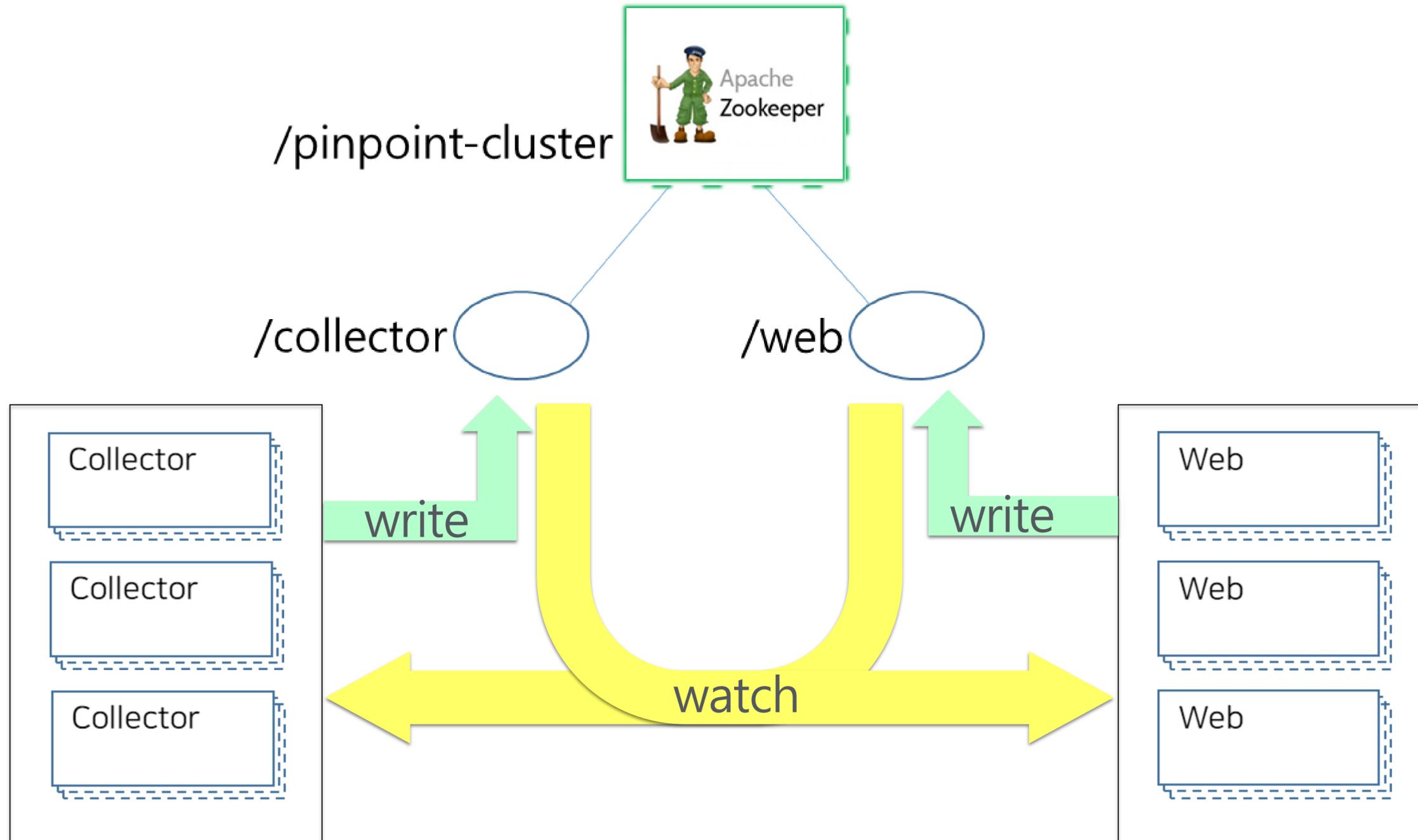
```
-${represtationIp}:port (ephemeral)
```

```
- $serverAddress1  
- $serverAddress2  
- $serverAddress3  
- ...
```

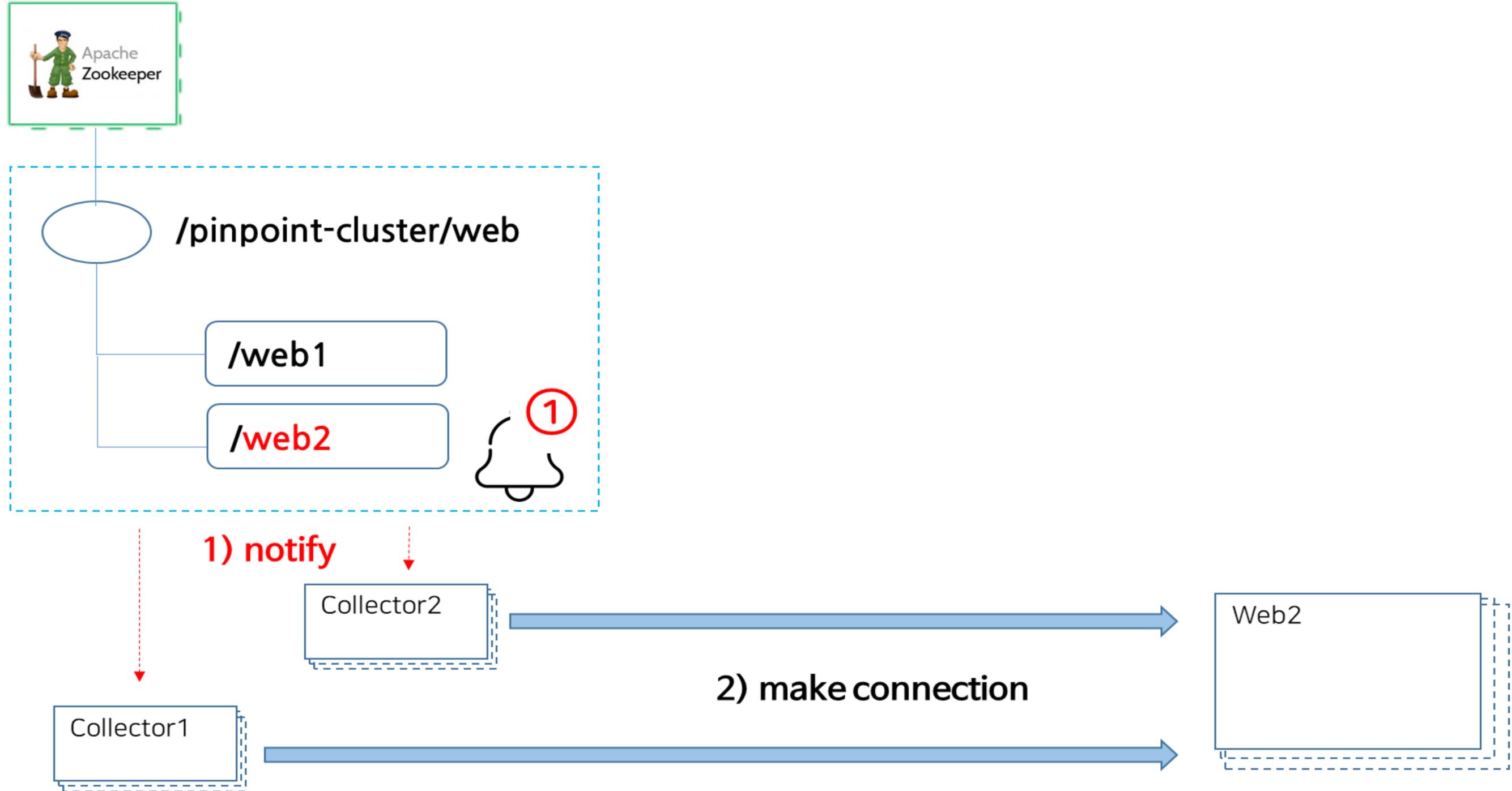
모든 네트워크 주소목록

- 노드 이름 “\${대표IP}:{Port}”
- 쥬키퍼의 ephemeral 노드 생성
- 내용으로 웹 장비의 모든 네트워크 주소 정보 저장

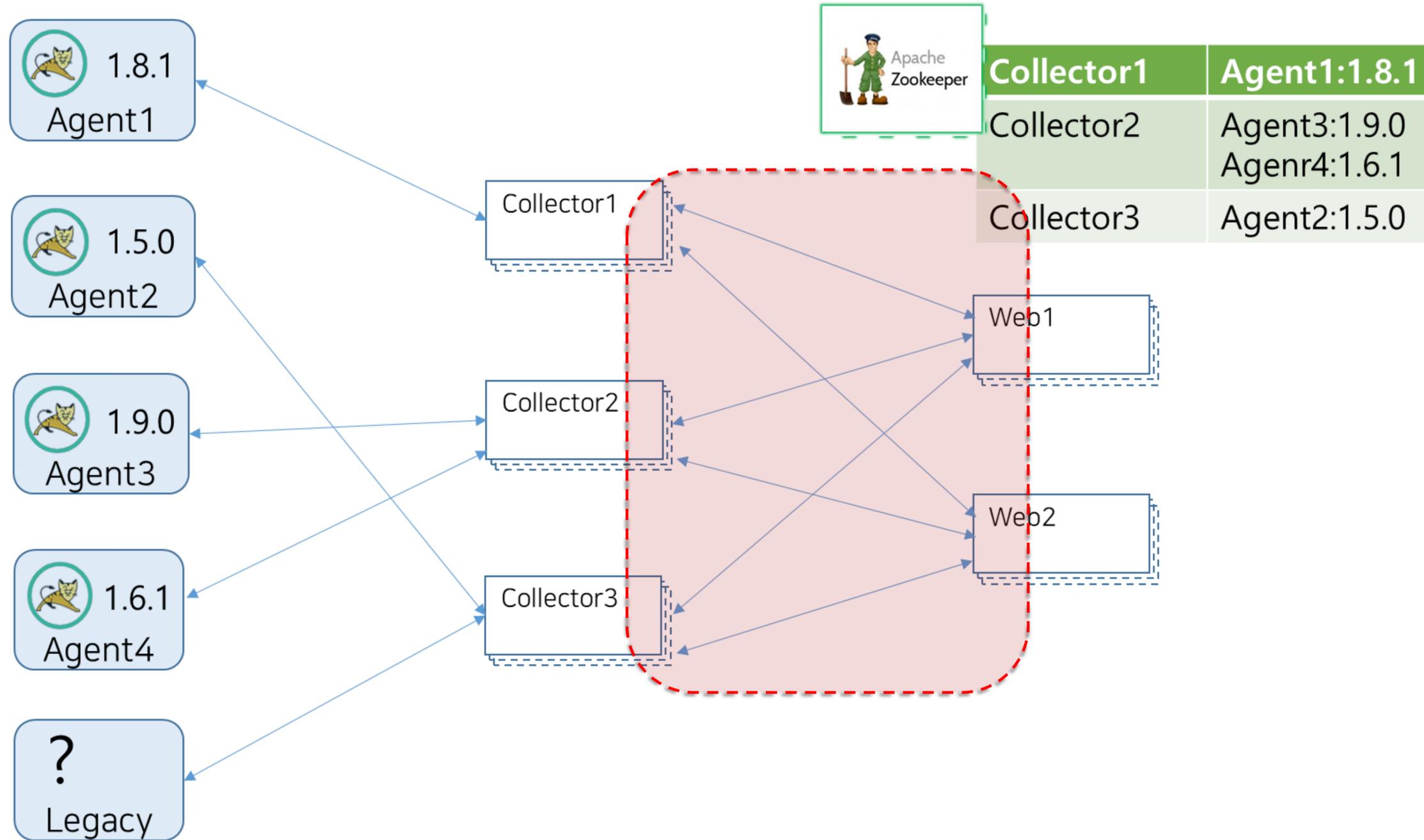
4. 웹 콜렉터간 노드 정보 공유



4. 콜렉터에서 웹으로 연결 구조



4. 웹 콜렉터 연동 결과



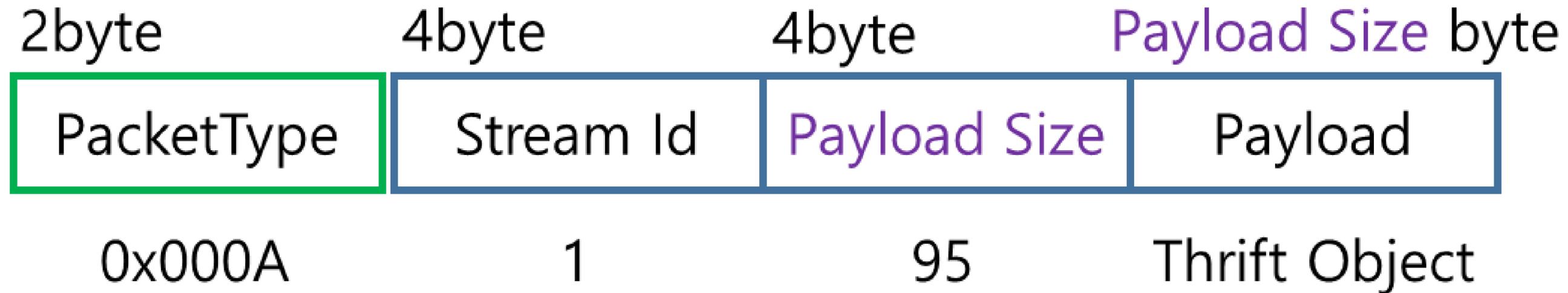
5. Req/Res 한계점

- 현재의 구조에서 ActiveThread를 가져오려면



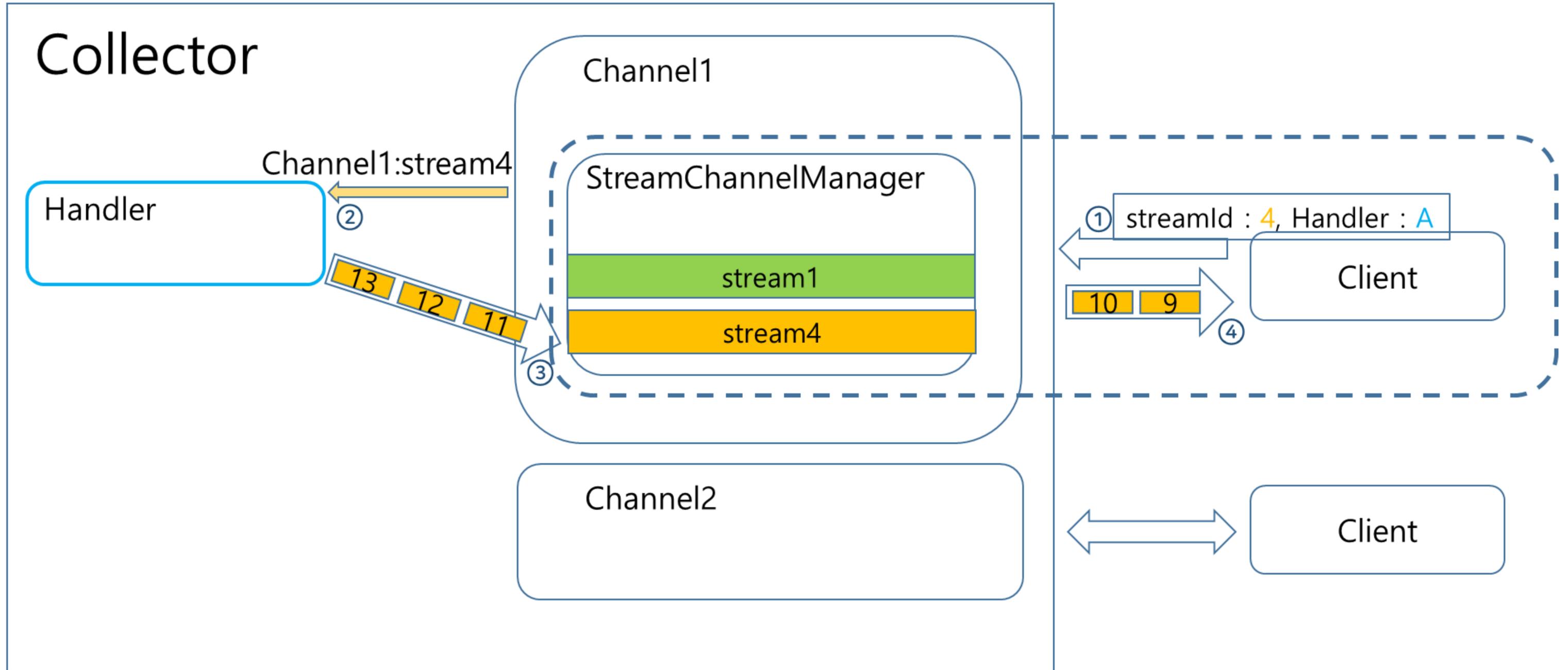
5. StreamPacket 추가

- StreamCreatePacket



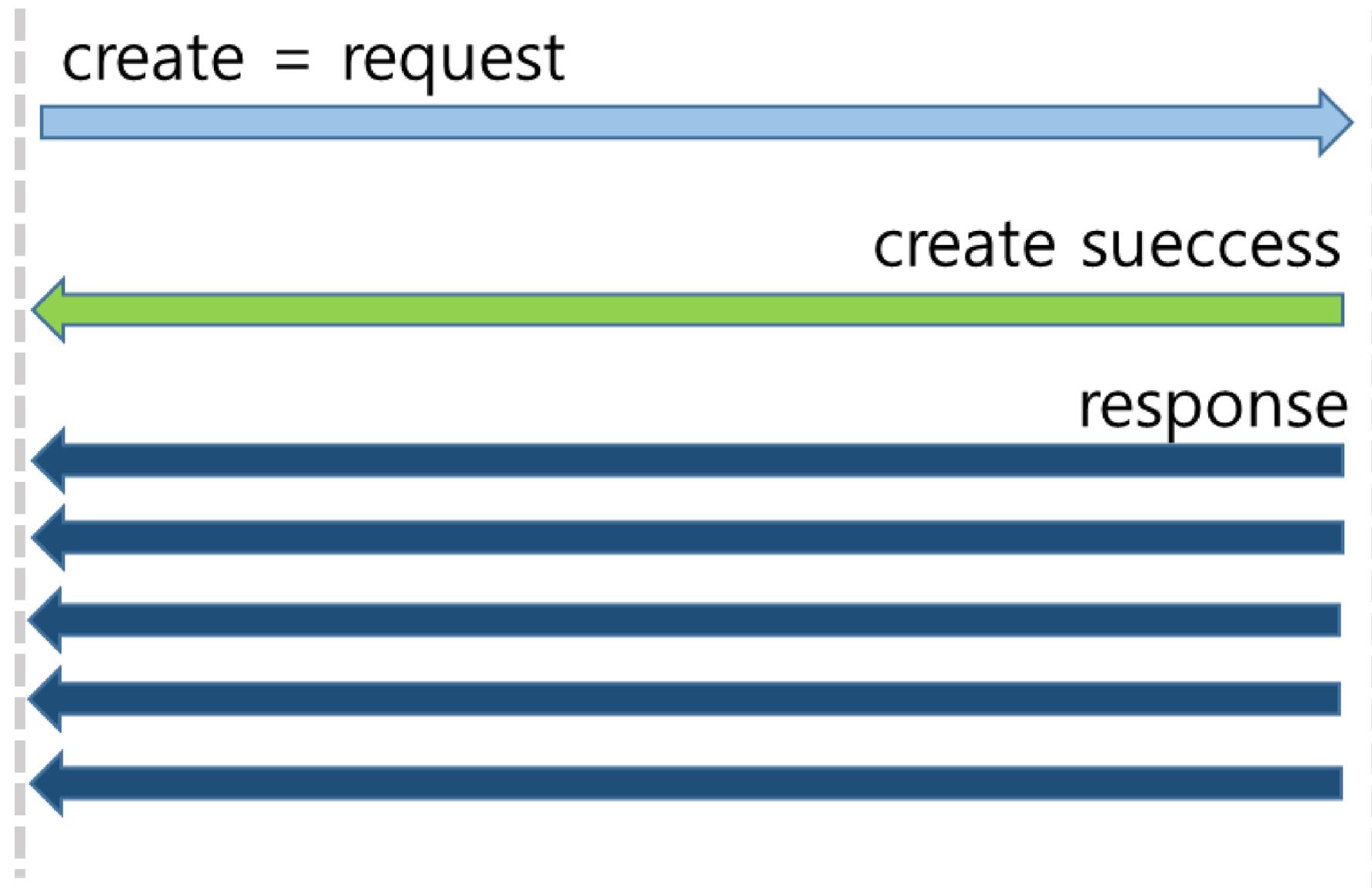
- StreamCreateSuccessPacket
- StreamCreateFailPacket
- StreamResponsePacket

5. Stream 동작 구조

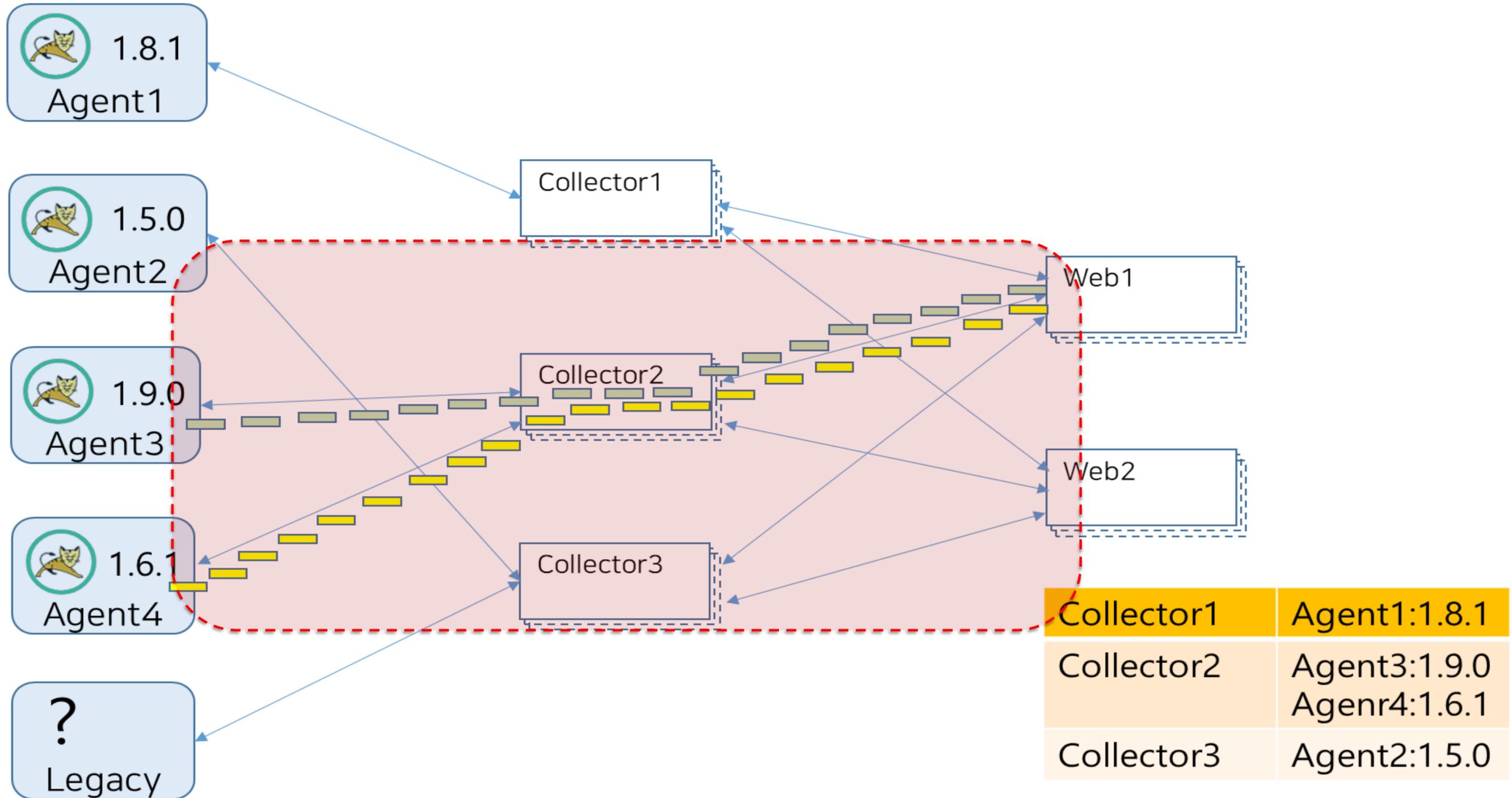


5. Stream 기능 지원 결과

- 에이전트의 부담이 줄어든 1 Req / Multi Res



결과



3. 효율적으로 observability 정보 저장하기

해결해야 하는 문제

DEVIEW
2019

사용자 늘어나

⇒ 저장공간 부족

⇒ 네트워크 대역 부족

⇒ 조회 시간 증가

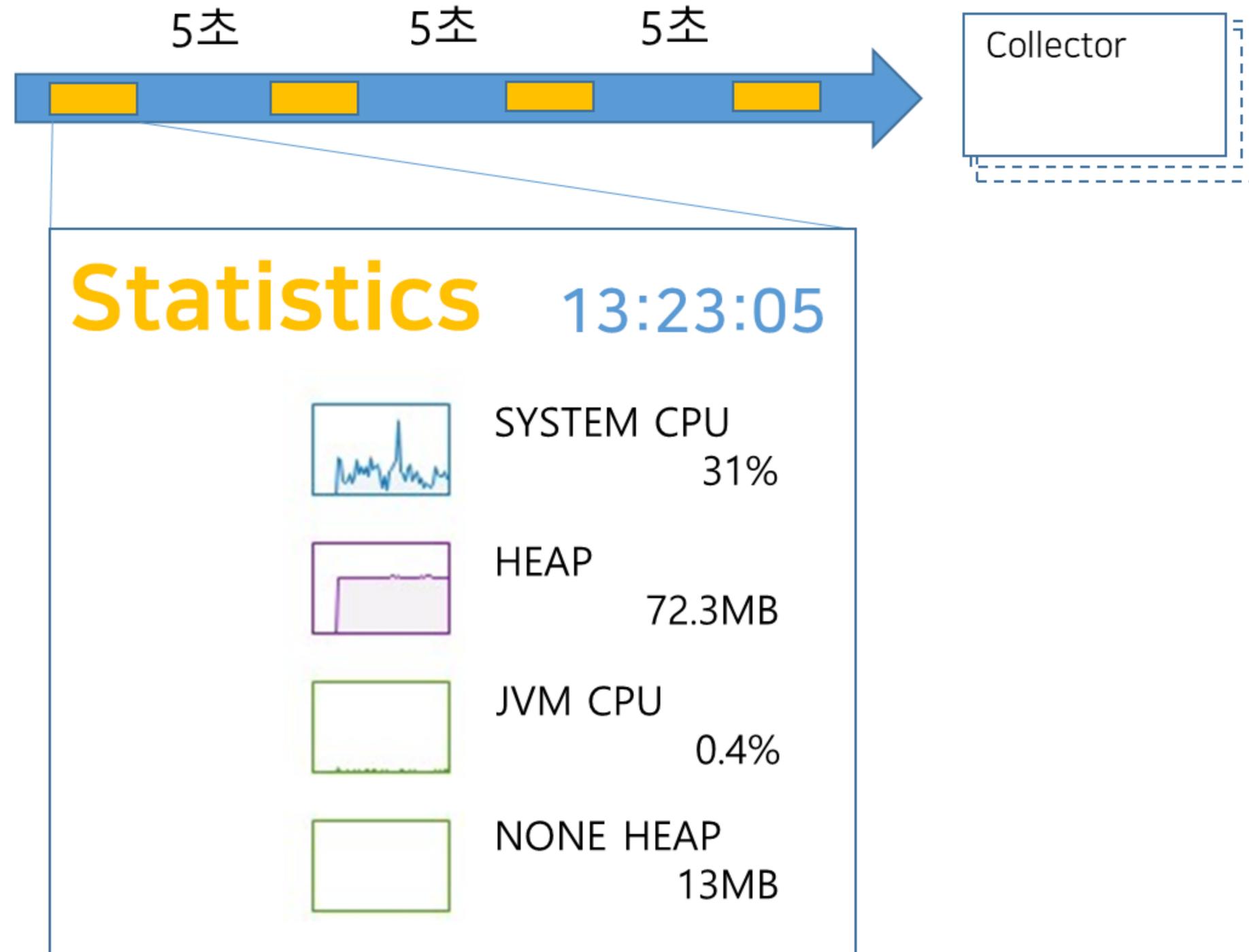
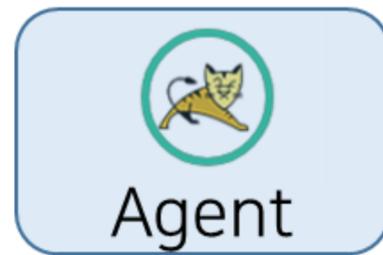
必 개선 필요

1. Inspector 개선

DEVIEW
2019



1. 데이터 전달 구조



1. 기존 데이터 저장방식

- 5초 단위로 데이터를 받고 이를 즉시 Hbase에 저장

rowkey :

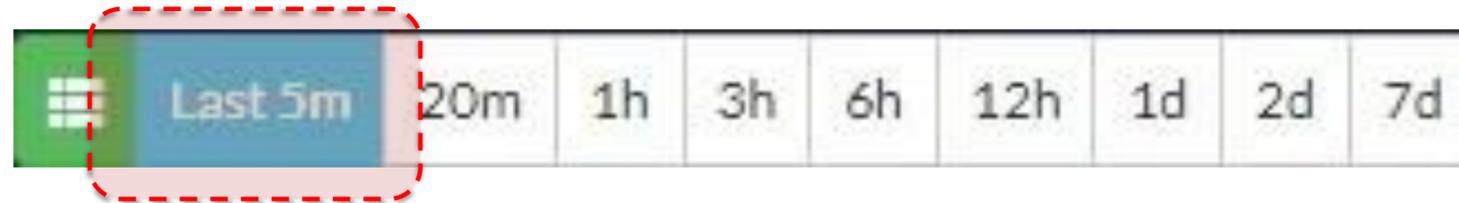
1byte	24byte	8byte
SaltKey	AgentId	EventTime

value :

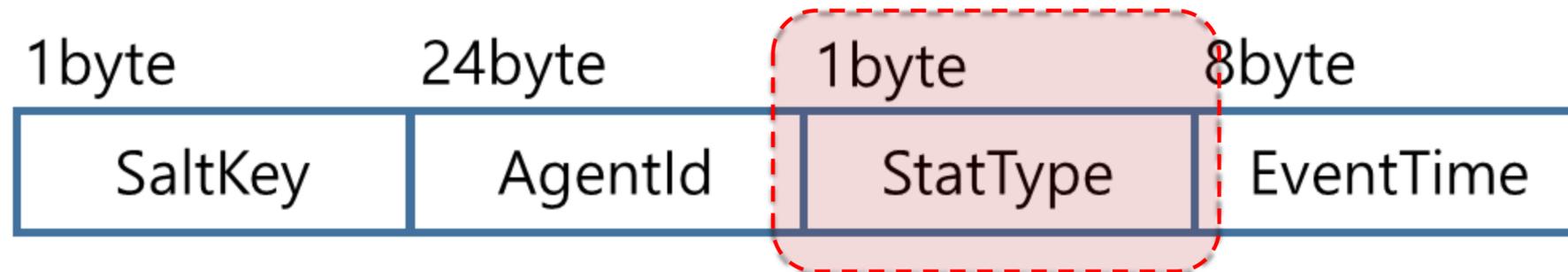
CPU	System CPU	HEAP	NONE HEAP
-----	------------	------	-----------

1. 데이터 저장방식 변경

- 5분 단위 저장



- 각각의 메트릭 정보를 나누어서 저장 (메트릭 6개)



1. 데이터 저장방식 변경 중간 결과

- rowkey 갯수 (5분 기준, 메트릭 6개)

60 → 6

- 조회속도

parallel 요청 가능 (성능개선)

*Type*을 나누고 시간을 뭉쳐보니

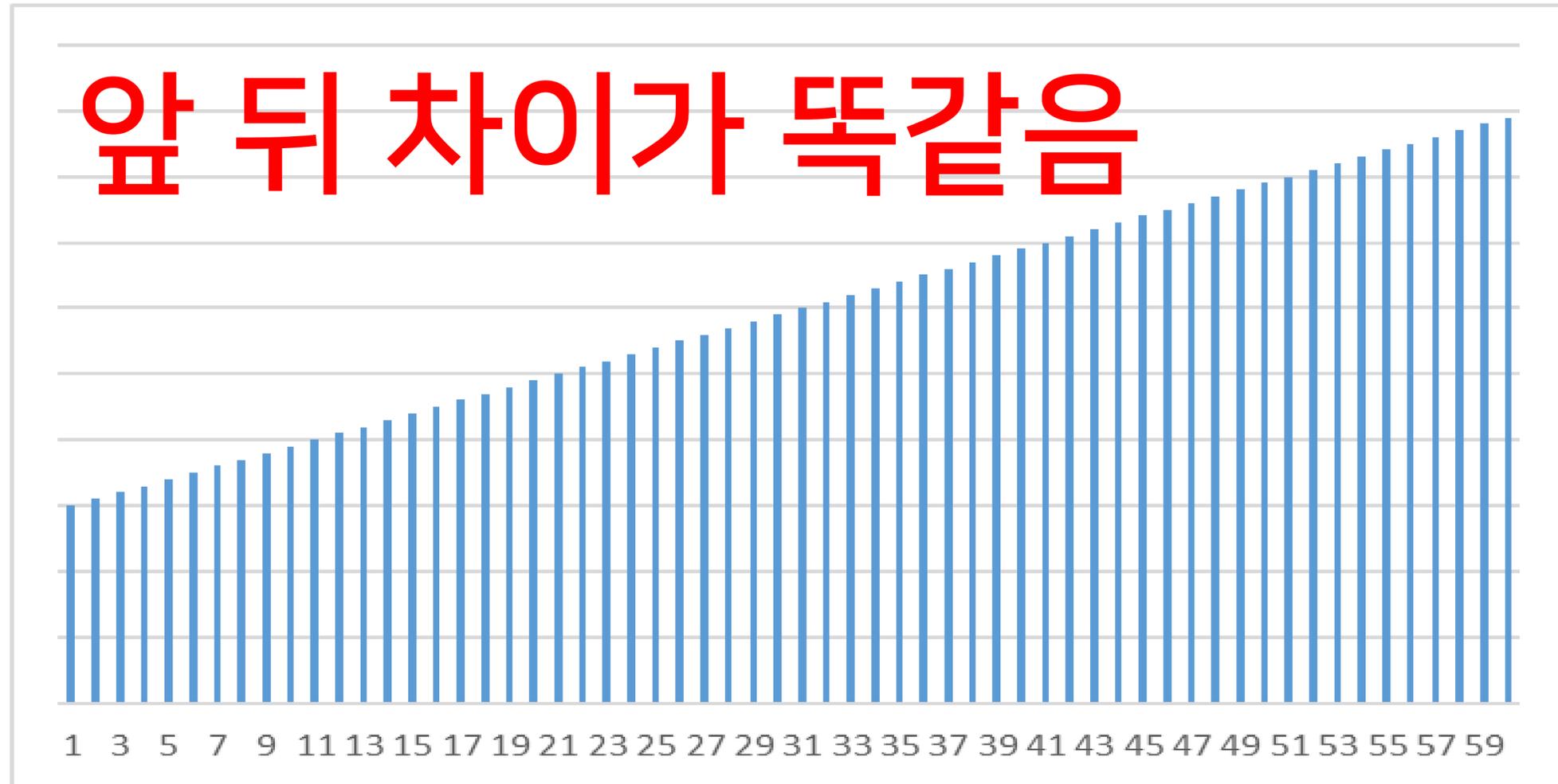
1. 데이터를 가만히 바라보니



1. 데이터를 가만히 바라보니



1. 데이터를 가만히 바라보니



1. Dynamic Encoding 지원

- Delta
- Delta of Delta
- Repeat
- SAME
- RAW
- ...

1. 데이터가 들어오면



1. 데이터가 들어오면 (기존)

rowkey	heapUsed	heapMax	
salt(1)agentId(24) 1497029500000(8)	hpU(3):300,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes
salt(1)agentId(24) 1497029505000(8)	hpU(3):301,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes
salt(1)agentId(24) 1497029510000(8)	hpU(3):303,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes
salt(1)agentId(24) 1497029515000(8)	hpU(3):290,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes
salt(1)agentId(24) 1497029520000(8)	hpU(3):291,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes
salt(1)agentId(24) 1497029525000(8)	hpU(3):300,000,000(5)	hpM(3):2,000,000,000(5)	49 bytes

Before : 294 bytes

이벤트시간 <=

힙사용량 <=

힙맥스값 <=

294B

1. 데이터가 들어오면 (변경)

- Rowkey :

1byte SaltKey	24byte AgentId	1byte StatType	8byte EventTime
------------------	-------------------	-------------------	--------------------

34 bytes

- 이벤트 시간 : DELTA_OF_DELTA(1) 5000(2) 0000(4) 7 bytes

- 힙사용량 : DELTA(1) 300m(5) 1m(3) 2m(4) 24 bytes
 -13m(4) 1m(3) 9m(4)

- 최대힙량 : REPEAT(1) 2g(5) 6(1)

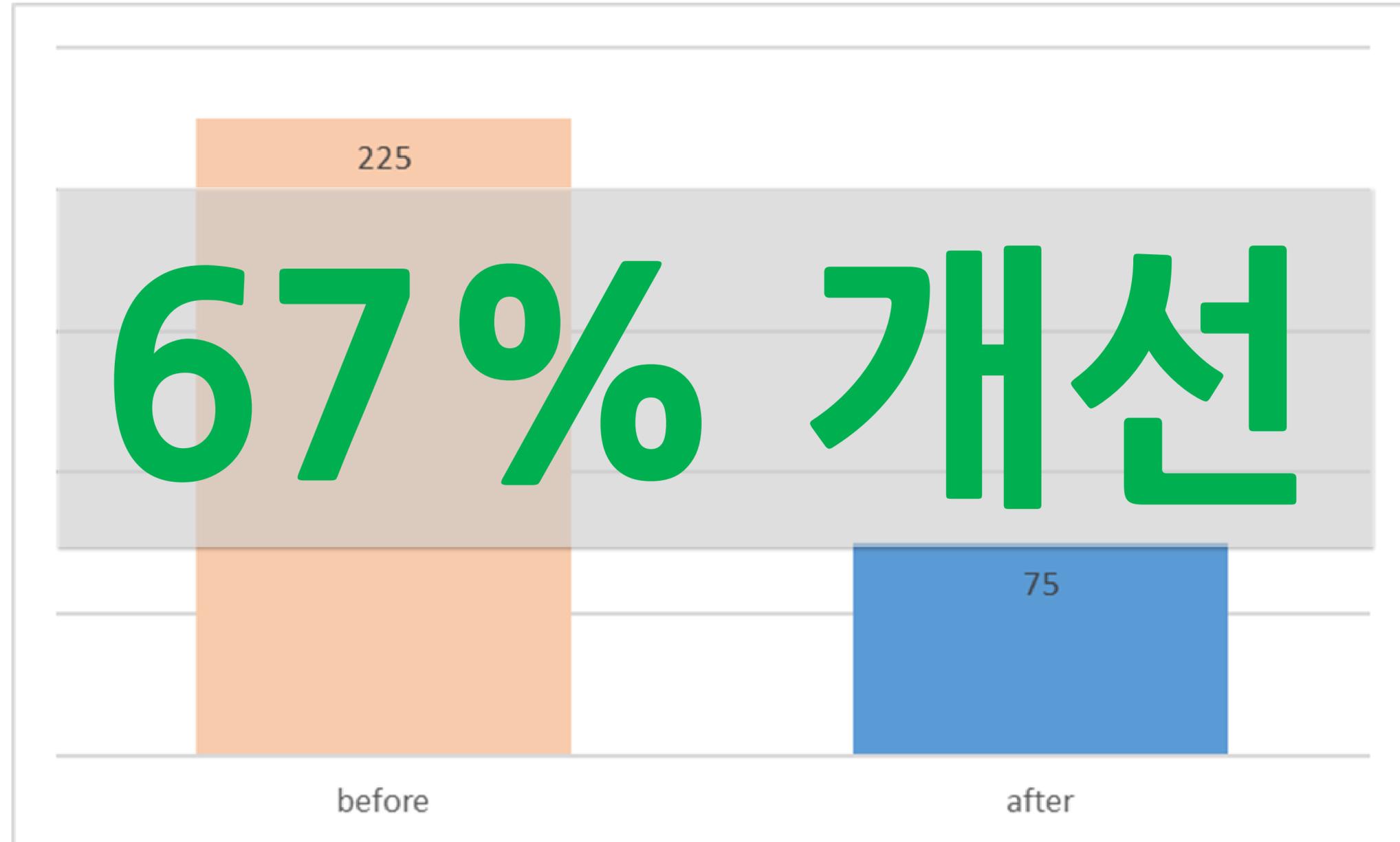
72B

7 bytes

1. 데이터가 들어오면 (결과)



1. 저장용량 개선



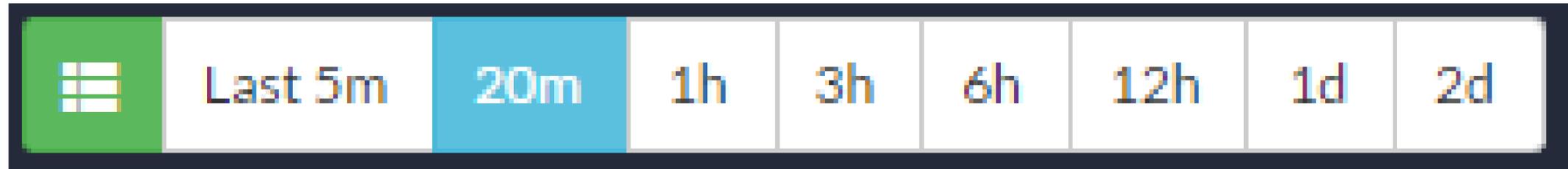
일부 에이전트 데이터 1주일간 DualWrite로 저장 후 비교

1. 조회시간 개선

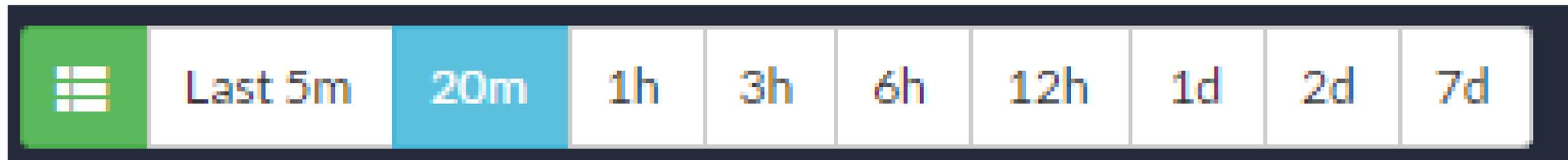
- 로컬 기준 2일치 (기존 Max값)
3.5 sec -> 350 ms

90% 개선

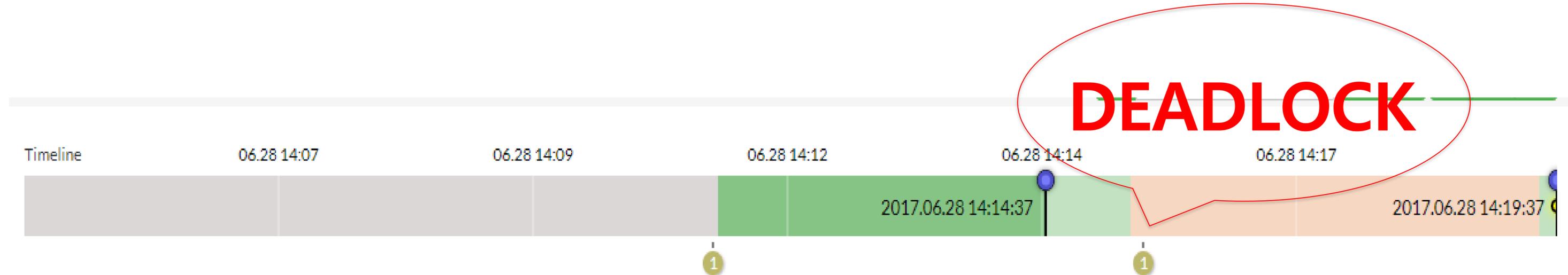
1. 더 긴 조회시간 추가



+ 7d



1. 서버 타임라인 이벤트 추가

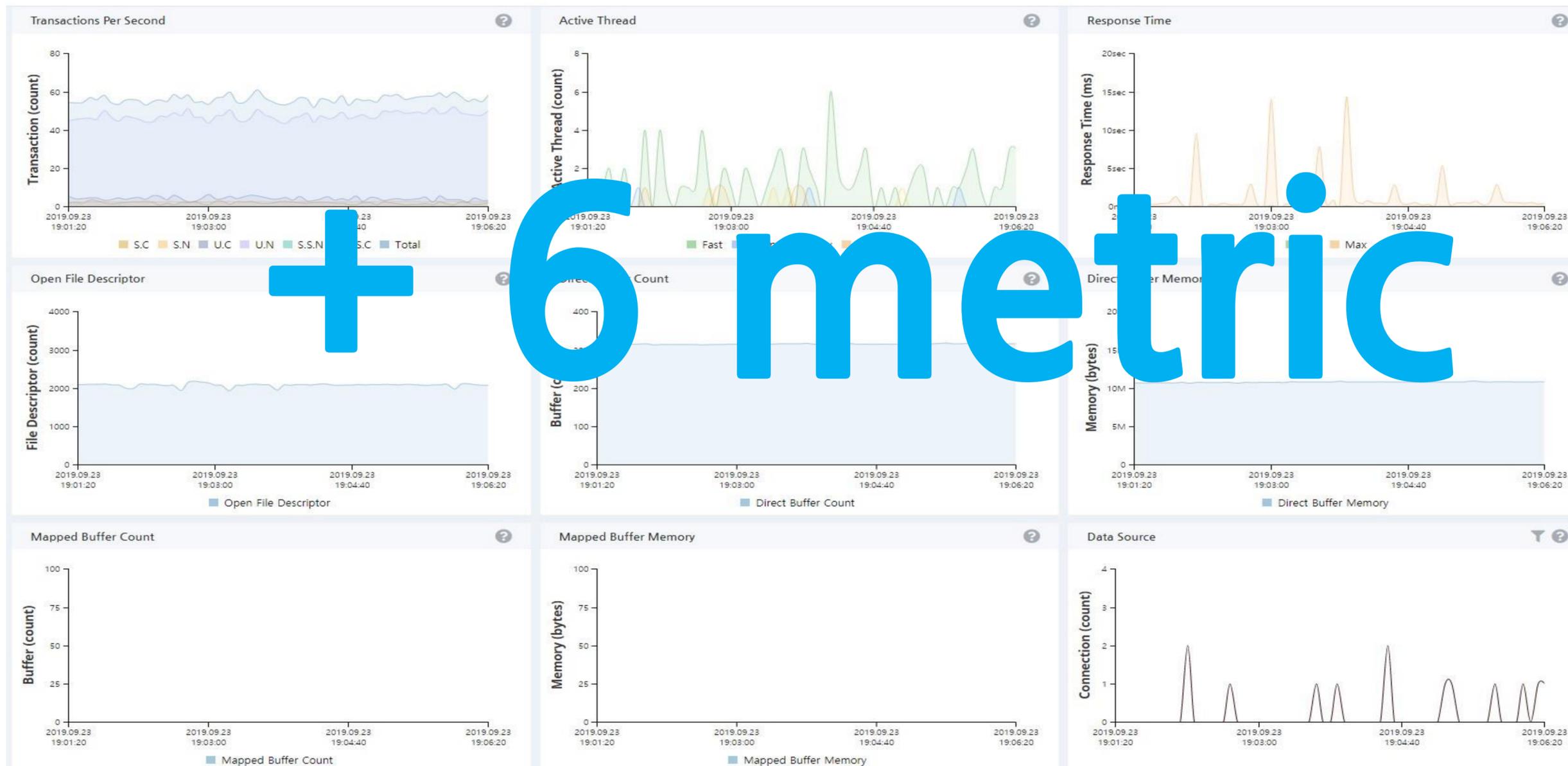


Event - 1 ✕

Time	Description	Message
2017-06-28 14:15:27 +09:00	Agent deadlock detected	"DEADLOCK THREAD-1" Id=0x9c BLOCKED on java.lang.Object@1dbc0490 owned by "DEADLOCK THREAD-2" Id=157 at com.navercorp.test.pinpoint.testweb.controller.ThreadDeadLockController\$Thread1.run(ThreadDeadLockController.java:85) - blocked on java.lang.Object@1dbc0490 "DEADLOCK THREAD-2" Id=0x9d BLOCKED on java.lang.Object@5246ea93 owned by "DEADLOCK THREAD-1" Id=156 at com.navercorp.test.pinpoint.testweb.controller.ThreadDeadLockController\$Thread1.run(ThreadDeadLockController.java:85) - blocked on java.lang.Object@5246ea93

1. Inspector 개선 결과

- DataSource, ResponseTime, OpenFD, ByteBuffer등 추가



2. Trace 개선

DEVIEW
2019

Application : /emeroad.pinpoint TransactionId : FrontWAS2^1563931395270^189... AgentId : FrontWAS2 ApplicationName : FRONT-WEB

Call Tree Server Map Timeline Mixed View nelo Self >= 1000(ms) [Search] [Complete] [Refresh]

Span

Method	Argument	Start Time	Gap(ms)	Exec(ms)	Exec(%)	Self(ms)	Class	API	Application	Agent
Servlet Process	/emeroad.pinpoint	17:30:48 004	0	295		0		TOMCAT	FRONT-WEB	FrontWAS2
http.status.code	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		17:30:48 004	0	295		0	StandardHostValve	TOMCAT_ME...	FRONT-WEB	FrontWAS2
doGet(HttpServletRequest request, HttpServletResponse response)		17:30:48 004	0	295		2	FrameworkServlet	SPRING	FRONT-WEB	FrontWAS2
demo2()		17:30:48 006	2	293		280	DemoController	SPRING_BE...	FRONT-WEB	FrontWAS2
execute(HttpUriRequest request, Response response)		17:30:48 286	280	13		0	CloseableHttpCli...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
open(HttpRoute route, HttpContext context)	dev-pinpoint-workload	17:30:48 286	0	0		0	AbstractPooledCo...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
execute(HttpRequest request, HttpResponse response)	/backendweb.pinpoint	17:30:48 286	0	13		13	HttpRequestExecu...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
http.status.code	200									
http.io	write: 0ms, read: 13m									
Servlet Process	/backendweb.pinpoint	17:30:48 286	0	12		0		TOMCAT	BACKEND-WEB	BackendWAS1
http.status.code	200									
REMOTE_ADDRESS	10.106.147.206									
invoke(Request request, Response response)		17:30:48 286	0	12		0	StandardHostValve	TOMCAT_ME...	BACKEND-WEB	BackendWAS1
doPost(HttpServletRequest request, HttpServletResponse response)		17:30:48 286	0	12		9	FrameworkServlet	SPRING	BACKEND-WEB	BackendWAS1
backendweb()		17:30:48 295	9	3		0	DemoController	SPRING_BE...	BACKEND-WEB	BackendWAS1
cacheService()		17:30:48 295	0	1		0	CacheServiceImpl	SPRING_BE...	BACKEND-WEB	BackendWAS1

2. SpanEvent란?

SpanEvent

The screenshot shows a Pinpoint call tree for a transaction. The top bar indicates the application is '/emeroad.pinpoint', the transaction ID is 'FrontWAS2^1563931395270^189...', the agent ID is 'FrontWAS2', and the application name is 'FRONT-WEB'. The interface includes tabs for 'Call Tree', 'Server Map', 'Timeline', and 'Mixed View', along with a filter 'Self >= 1000(ms)'. The main table displays the call tree structure with columns for Method, Argument, Start Time, Gap(ms), Exec(ms), Exec(%), Self(ms), Class, API, Application, and Agent. A specific span event, 'invoke(Request request, Response response)', is highlighted with a dashed purple box. This event is associated with the 'StandardHostValve' class in the 'FRONT-WEB' application. Below it, the tree shows the execution of 'doGet' by 'FrameworkServlet' and 'demo2()' by 'DemoController'. Further down, it shows a call to 'execute' by 'CloseableHttpClient' to 'BackendWAS1' for the URL '/backendweb.pinpoint', which includes sub-events for 'Servlet Process', 'doPost', and 'backendweb()'.

Method	Argument	Start Time	Gap(ms)	Exec(ms)	Exec(%)	Self(ms)	Class	API	Application	Agent
Servlet Process	/emeroad.pinpoint	17:30:48 004	0	295		0		TOMCAT	FRONT-WEB	FrontWAS2
doGet(HttpServletRequest request, HttpServletResponse response)		17:30:48 004	0	295		2	FrameworkServlet	SPRING	FRONT-WEB	FrontWAS2
demo2()		17:30:48 006	2	293		280	DemoController	SPRING_BE...	FRONT-WEB	FrontWAS2
execute(HttpUriRequest request, HttpResponseHandler<HttpEntity> handler)		17:30:48 286	280	13		0	CloseableHttpCli...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
open(HttpRoute route, HttpContext context)	dev-pinpoint-workload	17:30:48 286	0	0		0	AbstractPooledCo...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
execute(HttpRequest request, HttpResponseHandler<HttpEntity> handler)	/backendweb.pinpoint	17:30:48 286	0	13		13	HttpRequestExecu...	HTTP_CLIE...	FRONT-WEB	FrontWAS2
Servlet Process	/backendweb.pinpoint	17:30:48 286	0	12		0		TOMCAT	BACKEND-WEB	BackendWAS1
doPost(HttpServletRequest request, HttpServletResponse response)		17:30:48 286	0	12		9	FrameworkServlet	SPRING	BACKEND-WEB	BackendWAS1
backendweb()		17:30:48 295	9	3		0	DemoController	SPRING_BE...	BACKEND-WEB	BackendWAS1

2. 저장 단위 (기존)

- 기존 : **SpanEvent** 단위로 저장

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code	200							
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	3	DemoController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* testquery */ select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

2. 저장 단위 (변경)

- SpanChunk(임의의 SpanEvent 묶음) 단위로 저장

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code	200							
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	3	DemoController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* testquery */ select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

- rowkey 갯수 : 10 -> 1

가까운 *SpanEvent*를 뭉쳐보니

2. 데이터를 또 가만히 바라보니

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
시작시간이 같거나 비슷								
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code								
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	3	DemoController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* testquery */ select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

2. 데이터를 또 가만히 바라보니

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code	200							
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	3	DemoController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	0	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	0	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* customery / select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

API 타입이 앞에 것과

같은 경우 많음

2. 데이터를 또 가만히 바라보니

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code	200							
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	0	HandlerController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	0	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	0	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* testquery */ select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

Argument가
없는 경우 많음

2. 데이터를 또 가만히 바라보니

	Argument	Start Time	Gap(ms)	Exec(m...	Class	API	Agent	Application
Servlet Process	/backendapi.pinpoint	11:00:17 752	0	5		TOMCAT	ApiWAS1	BACKEND-API
http.status.code	200							
REMOTE_ADDRESS								
invoke(Request request, Response response)		11:00:17 752	0	5	StandardHostValve	TOMCAT_METHOD	ApiWAS1	BACKEND-API
doPost(HttpServletRequest request, HttpServletResponse response)		11:00:17 752	0	5	FrameworkServlet	SPRING	ApiWAS1	BACKEND-API
backendapi()		11:00:17 753	1	3	DemoController	SPRING_BEAN	ApiWAS1	BACKEND-API
getConnection()		11:00:17 753	0	0	BasicDataSource	DBCP2	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	false	11:00:17 753	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberServiceImpl	SPRING_BEAN	ApiWAS1	BACKEND-API
list()		11:00:17 753	0	1	MemberDaoJdbc	SPRING_BEAN	ApiWAS1	BACKEND-API
executeQuery(Statement stmt, String sql)		11:00:17 753	0	1	StatementImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
SQL	/* testquery */ select * from member							
commit()		11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
setAutoCommit(boolean autoCommitFlag)	true	11:00:17 754	0	0	ConnectionImpl	MYSQL(pinpoi...	ApiWAS1	BACKEND-API
close()		11:00:17 754	0	0	PoolingDataSource...	DBCP2	ApiWAS1	BACKEND-API

depth가 같은 경우가

많음

2. 필드별 자주 사용하는 규칙 적용

- 시작시간 (Equal or Delta)
- Argument (Non exist or Raw)
- Api Type (Equal or Raw)
- Depth (Equal or Raw)
- ...

2. 규칙 + 비트패킹 적용

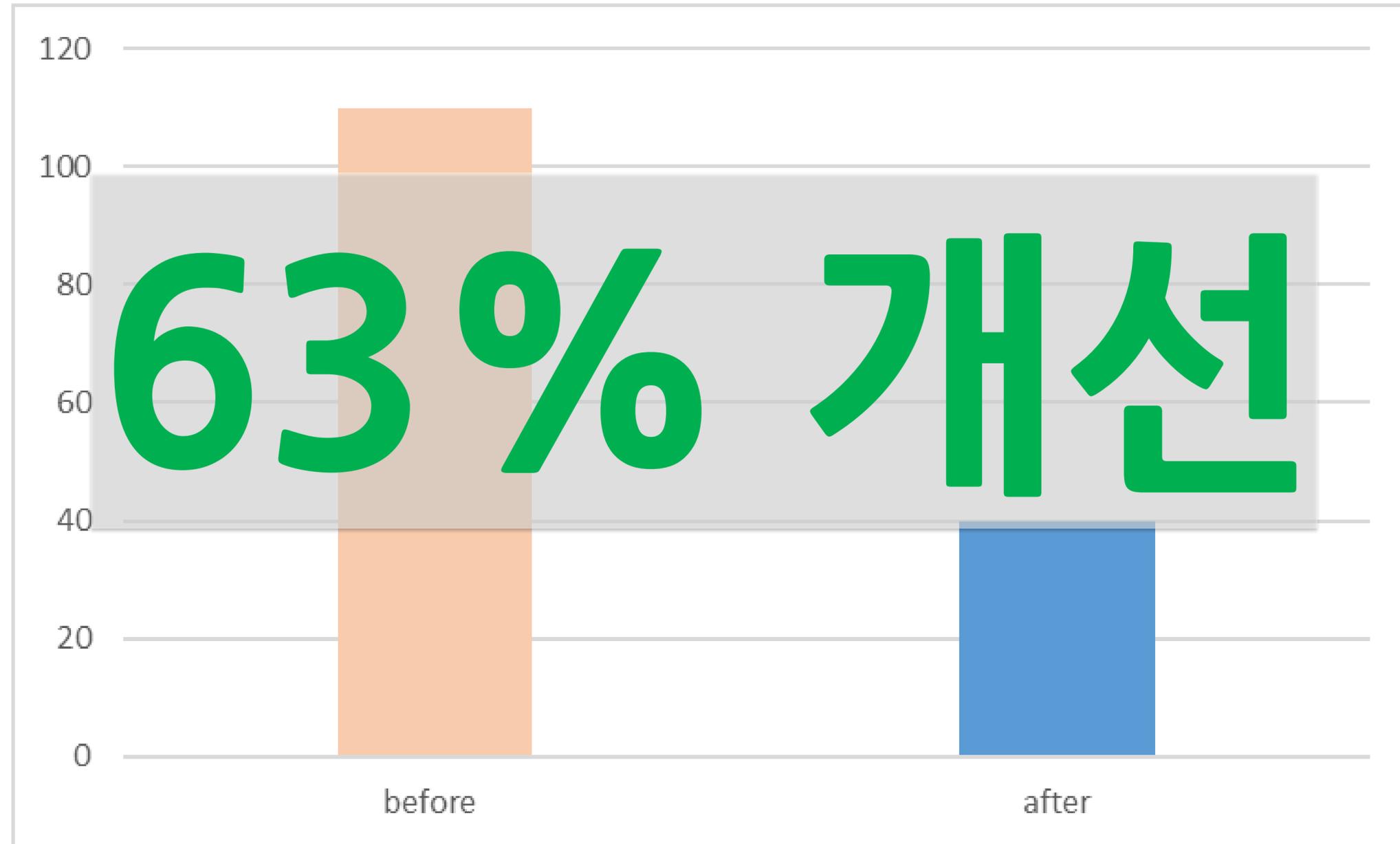
- 기존 : 모든 필드 정보 기록



- 변경 : 미리 정해진 규칙으로 저장되는 경우 1bit만으로 기록



2. Trace 개선 결과



일부 에이전트 데이터 1주일간 DualWrite로 비교시

3. Hbase Parallel Scanner 개발

- salting for rowkey

37. Rowkey Design

37.1. Hotspotting

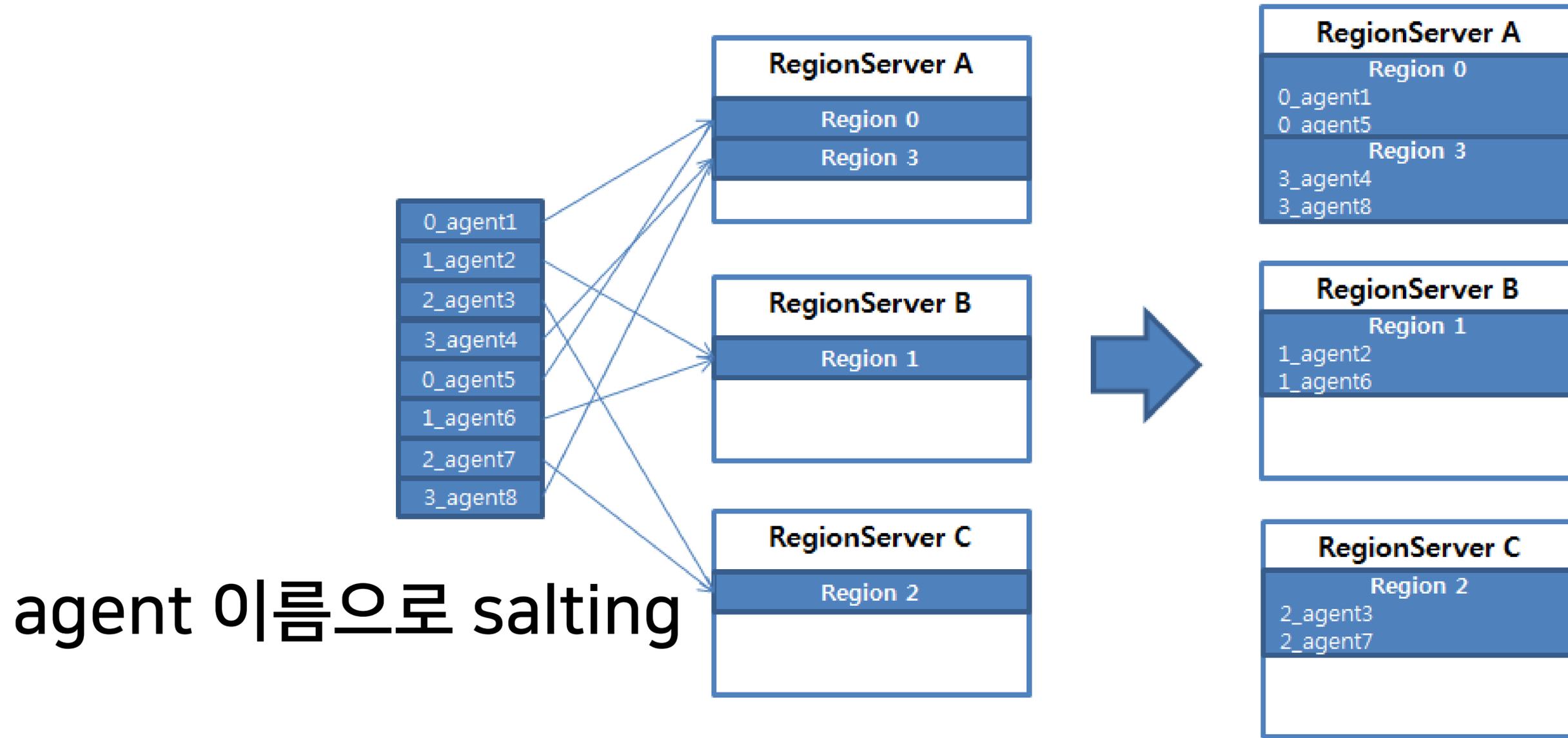
<https://hbase.apache.org/book.html#rowkey.design>

Rows in HBase are sorted lexicographically by row key. This design optimizes for scans, allowing you to store related rows, or rows that will be read together, near each other. However, poorly designed row keys are a common source of *hotspotting*. Hotspotting occurs when a large amount of client traffic is directed at one node, or only a few nodes, of a cluster. This traffic may represent reads, writes, or other operations. The traffic overwhelms the single machine responsible for hosting that region, causing performance degradation and potentially leading to region unavailability. This can also have adverse effects on other regions hosted by the same region server as that host is unable to service the requested load. It is important to design data access patterns such that the cluster is fully and evenly utilized.

To prevent hotspotting on writes, design your row keys such that rows that truly do need to be in the same region are, but in the bigger picture, data is being written to multiple regions across the cluster, rather than one at a time. Some common techniques for avoiding hotspotting are described below, along with some of their advantages and drawbacks.

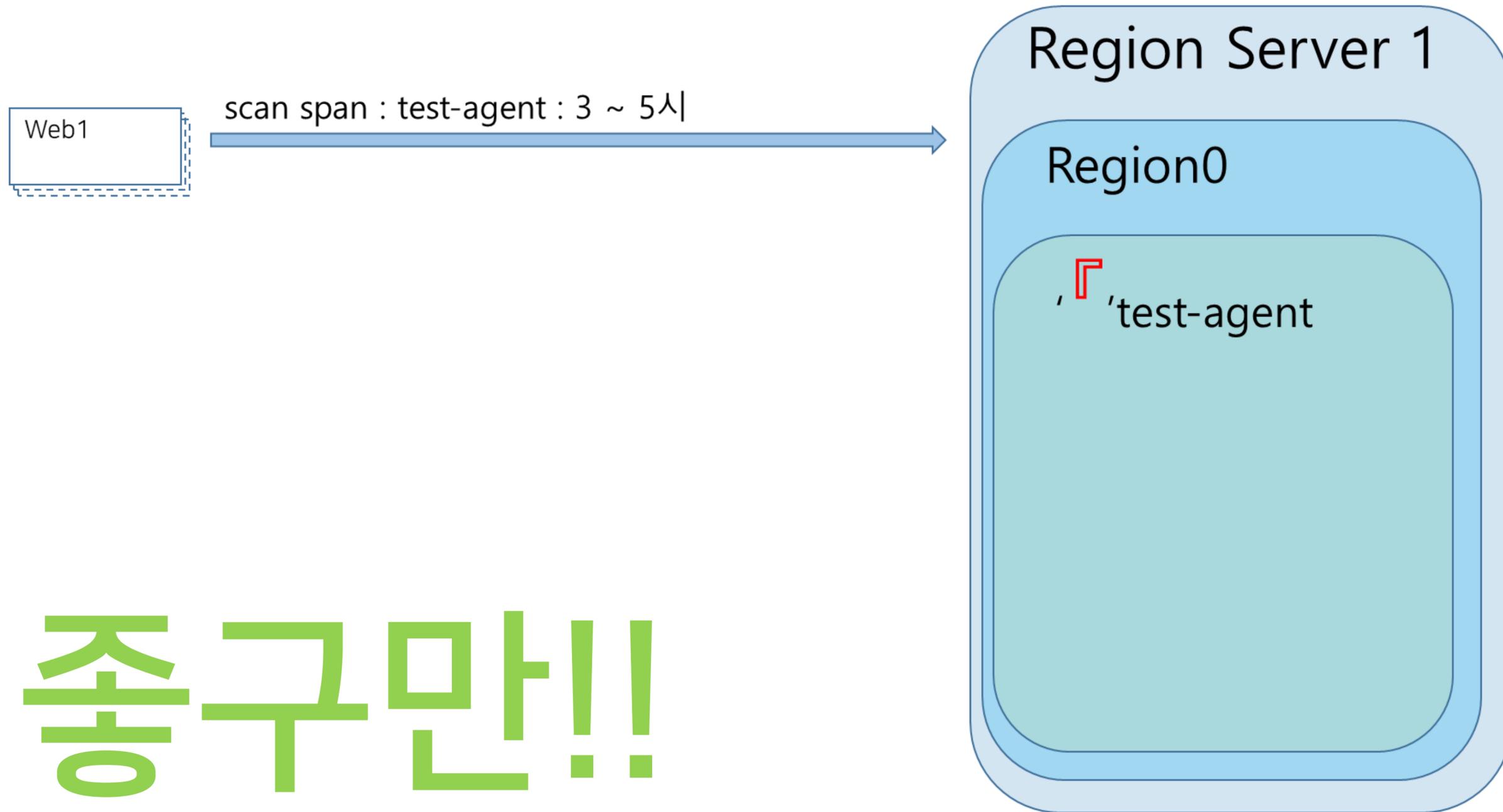
Salting

3. Hbase 저장 (기존)



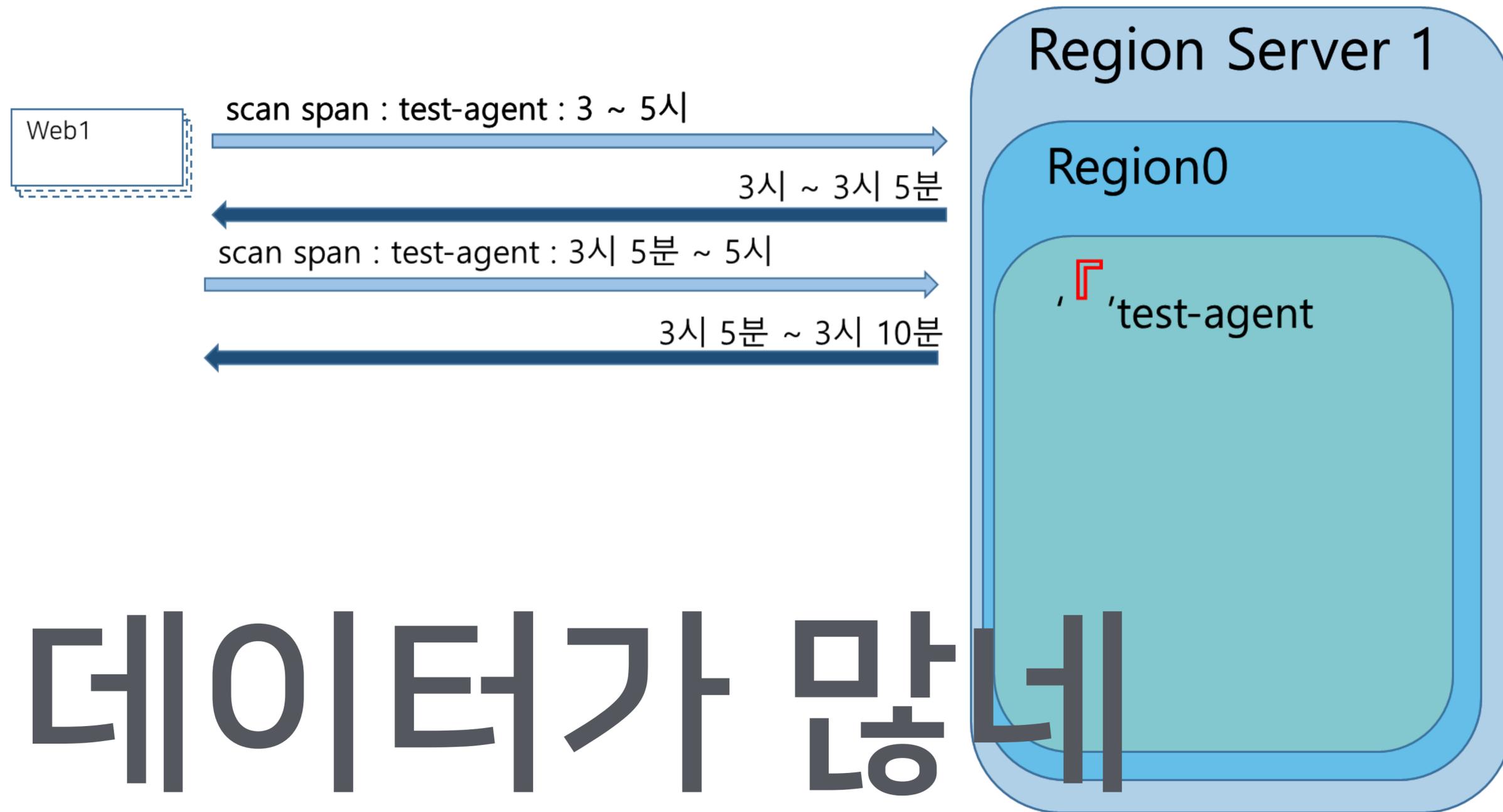
Hotspotting은 피하고, 연관성 있는 데이터는 모으고

3. Trace 데이터 조회 (기존)

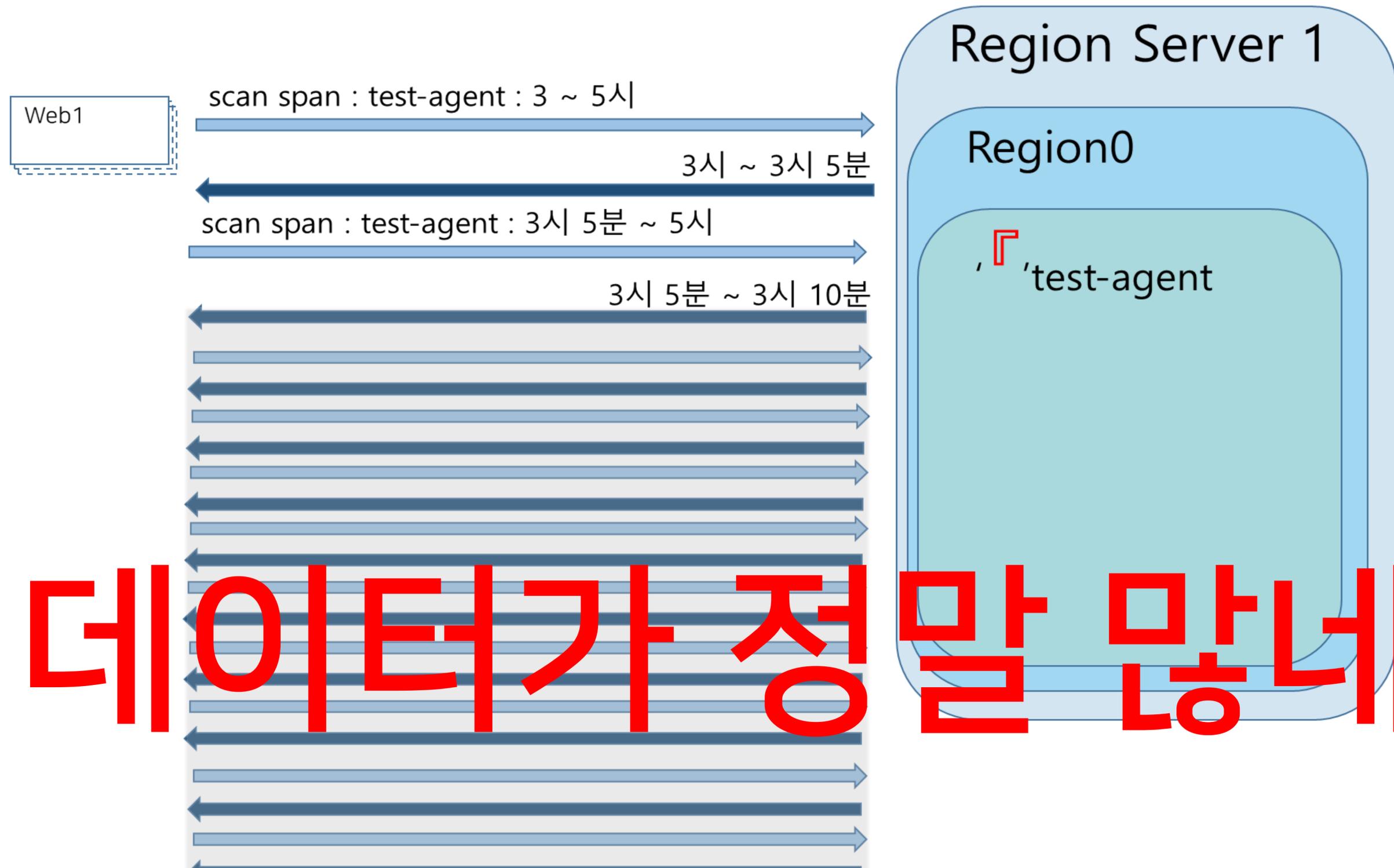


종구만!!

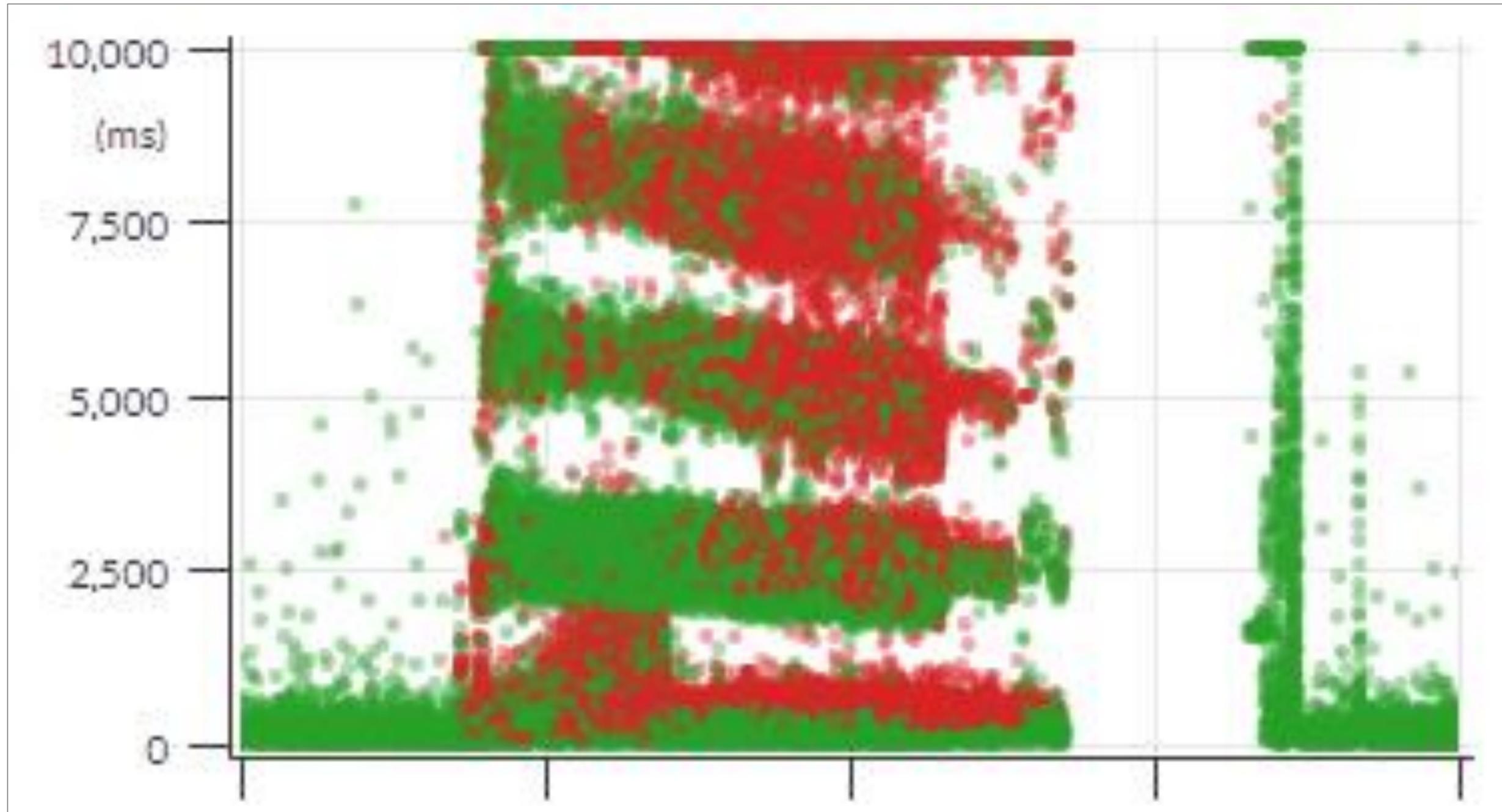
3. Trace 데이터 조회 (기존)



3. Trace 데이터 조회 (기존)



3. Trace 데이터는 정말 많았다



3. Trace 데이터는 정말 많았다



3. 이게 우리한테 맞는건가?

정말 많은 데이터를

1Region에 모으고

순차적으로 조회

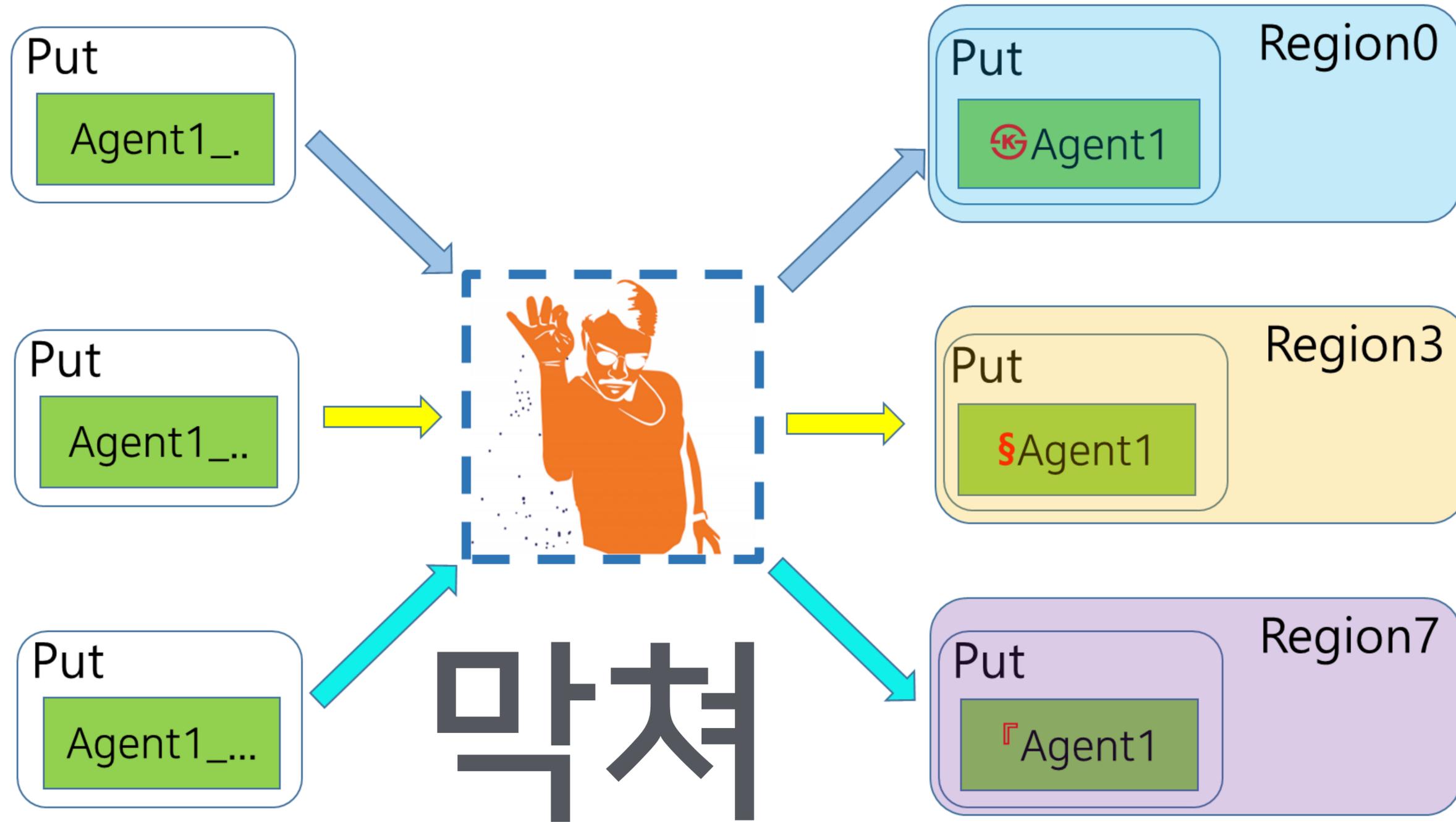
VS

정말 많은 데이터를

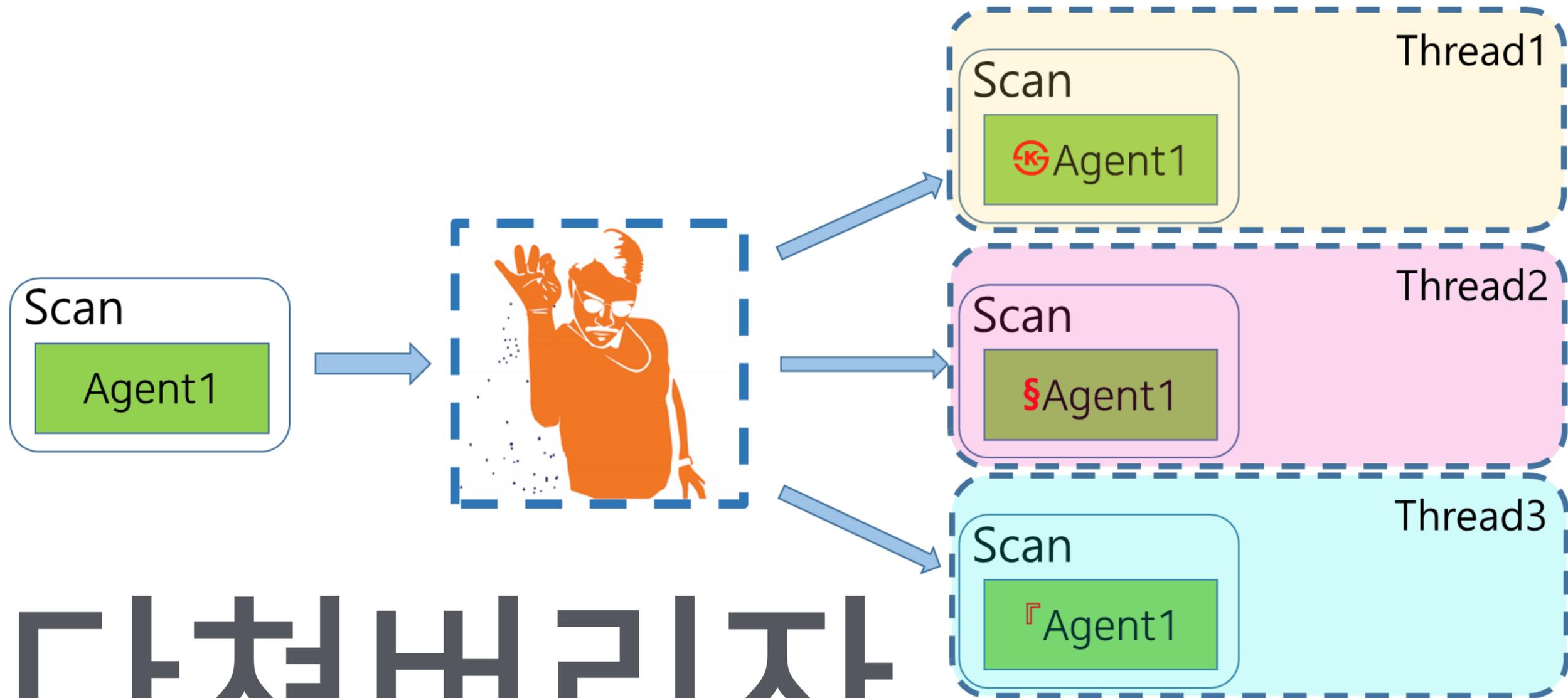
여러 Region 넣고

병렬적으로 조회

3. 넣을때 무작위로

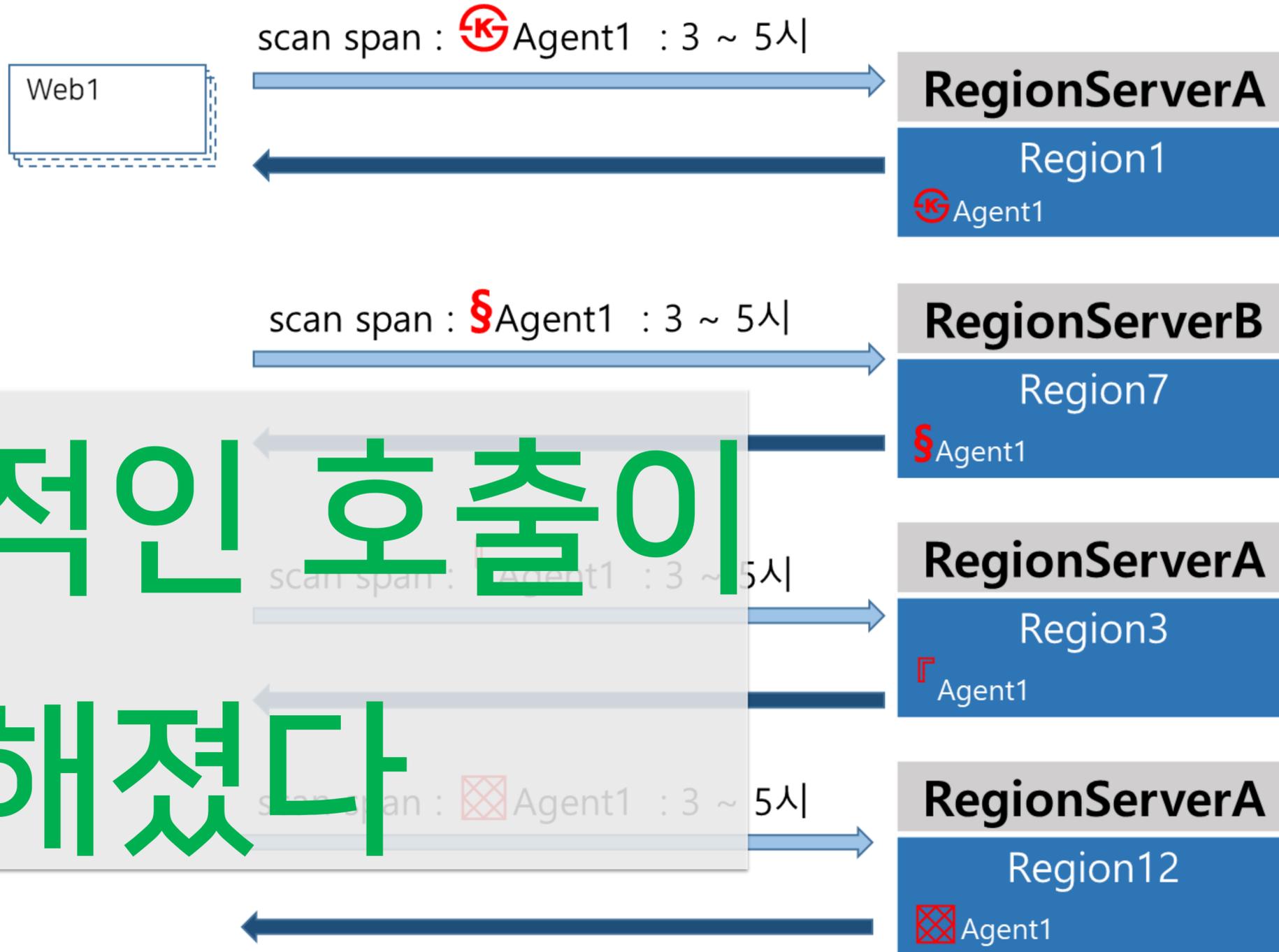


3. 가지고 올때는 다 가져오고



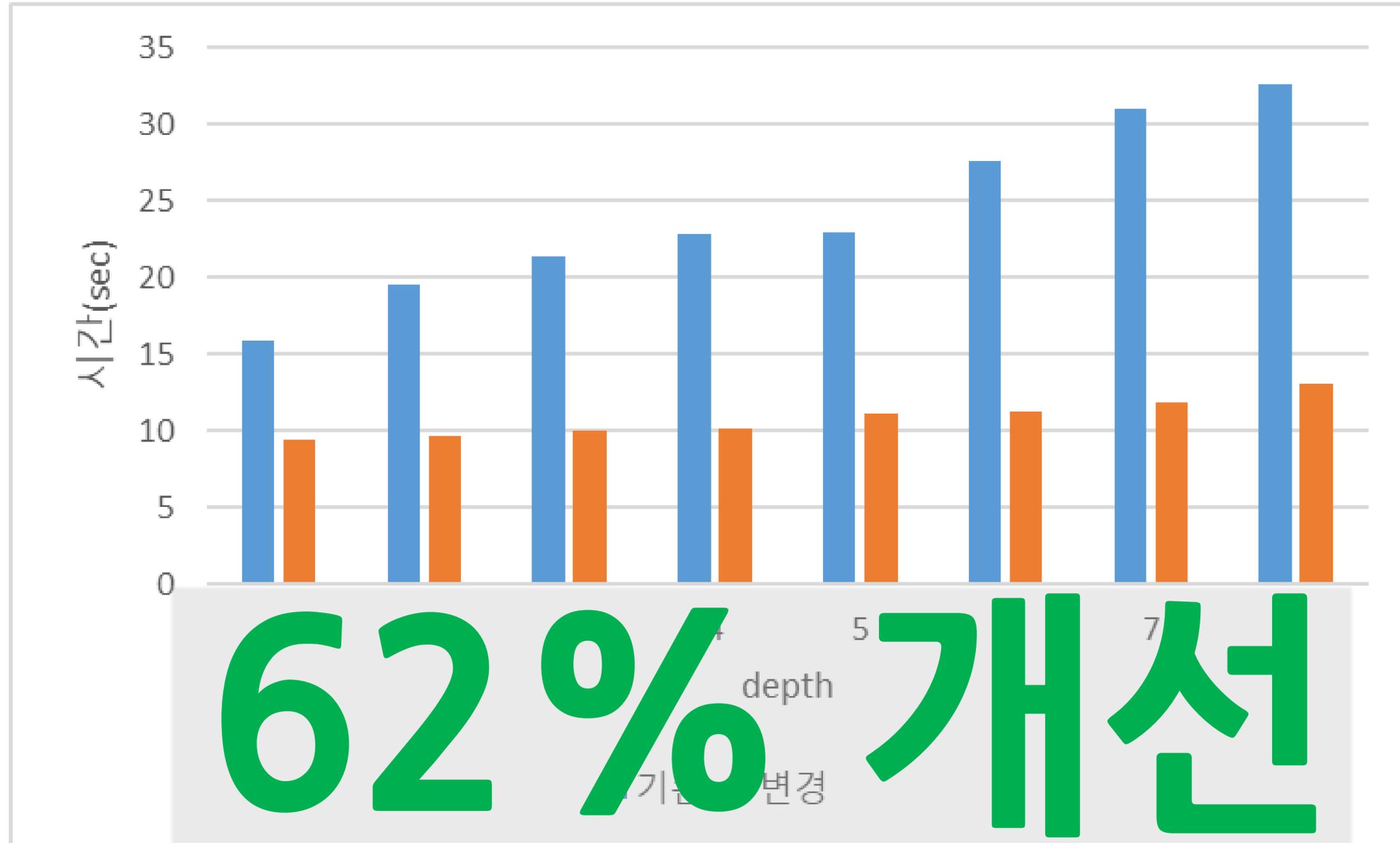
다 가져버리자

3. 무작위 salting 결과



병렬적인 호출이
가능해졌다

3. 무작위 salting 결과



3. Hbase Parallel Scanner 회고

- Salting Key를 이 처럼 쓰는건 매우 데이터가 많이 들어오는 경우에만 권장합니다.
- Hbase client 2.0 부터는 유사한 기능을 지원하고 있습니다.

4. 다양한 컴포넌트에 대한 observability 확보하기

해결해야 하는 문제

- 많고 다양한 플러그인을 지원하려면
- 지속적으로 릴리즈 되는 라이브러리의 플러그인을 지원하려면

doc/modules.md 확인가능

Title	Instrumented Library	Min	Max	Comment
Tomcat		6.x	9.x	
Jetty		8.x	9.x	
JBoss		6.x	7.x	
Resin		4.x	4.x	
Websphere		6.x	8.x	

어제 나왔는데 오늘 지원함!

Vertx		3.3	3.5	
Weblog		1.0	1.2	
Undertow				
Jasper				
OpenWhisk	whisk.core			

SpringMVC Framework	spring-webmvc	3.0.7	5.1.9	
Spring Web	spring-web	4.1.2	4.3.25	
Spring RabbitMQ	spring-rabbit	1.3.3	2.1.10	
Spring IBatis	spring-ibatis	2.0.7	2.0.8	
Spring MyBatis	mybatis-spring	1.1.0	1.3.3	
Spring Boot	spring-boot-autoconfigure			
MyBatis	mybatis	3.0.3	3.3.1	
Hystrix	hystrix-core	1.4.0	1.5.18	
JDKHTTP				
HttpClient3	commons-httpclient	3.0	3.1	
HttpClient4	httpclient	4.0	4.5.4	
Thrift	libthrift	0.9.1	0.12.0	
Google HTTP Client	google-http-client	1.19.0	1.32.1	

The screenshot shows the GitHub release page for the 'google-http-client' repository. The latest release is v1.32.1, released by yoshi-automation 4 days ago. The release notes mention updates to dependencies: 'update dependency com.google.protobuf:protobuf-java to v3.10.0 (#824) (c51b62f)' and 'update guava to 28.1-android (#817) (e05b6a8)'. There are two assets: 'Source code (zip)' and 'Source code (tar.gz)'. A red dashed box highlights the version number 'v1.32.1' and the release date '1.32.1 (2019-09-20)'. Another red dashed box highlights the 'Google HTTP Client' entry in the dependency table on the right side of the image.

必

블로그인 테스트

1. 플러그인 어떻게 테스트 할까?

플러그인 테스트는
최대한 실상황에서

⇒ **그럴려면 에이전트를
활성화 해야 하는데**

⇒ **그럴려면 프로세스를
실행 해야 하는데**

1. Custom TestRunner 생성

```
public class PinpointTestRunner extends BlockJUnit4ClassRunner {  
  
    public PinpointTestRunner(Class<?> klass) throws InitializationError {  
        super(klass);  
    }  
  
    @Override  
    public void run(RunNotifier notifier) {  
        // 여기서 에이전트를 활성화 하는 프로세스를 실행해서 결과값을 받으면;;;  
    }  
  
}
```

1. 테스트에서 외부 프로세스를 실행

- ProcessBuilder

```
ProcessBuilder builder = new ProcessBuilder();  
// 실행할 커맨드를 넣고  
// builder.command("rm", "-rf", "/");  
builder.command(getCommands()); <= "java ${runClass} ${args}"  
// 커맨드를 실행  
Process process = builder.start();  
// 실행된 커맨드의 결과값을 획득  
InputStream inputStream =  
process.getInputStream();
```

1. 자바 프로세스는 main부터

- `runClass` // main에서 테스트를 실행하는 클래스 생성

```
public static void main(String[] args) throws ClassNotFoundException, InitializationError {  
    // 인자로 Test할 ClassName을 받아  
    String testClazzName = args[1];  
  
    // 클래스 생성후  
    ClassLoader classLoader = RunTestMain.class.getClassLoader();  
    Class<?> testClazz = classLoader.loadClass(testClazzName);  
  
    Runner runner = new BlockJUnit4ClassRunner(testClazz);  
    // junit 실행  
    JUnitCore junit = new JUnitCore();  
    junit.run(runner);  
}
```

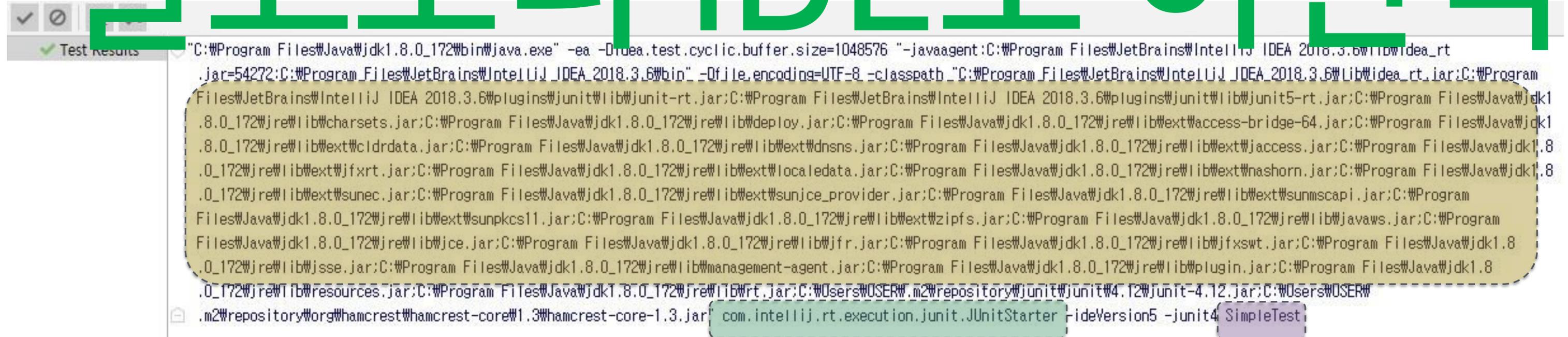
1. 테스트에서 외부 프로세스를 실행

java -cp `${classpath}`

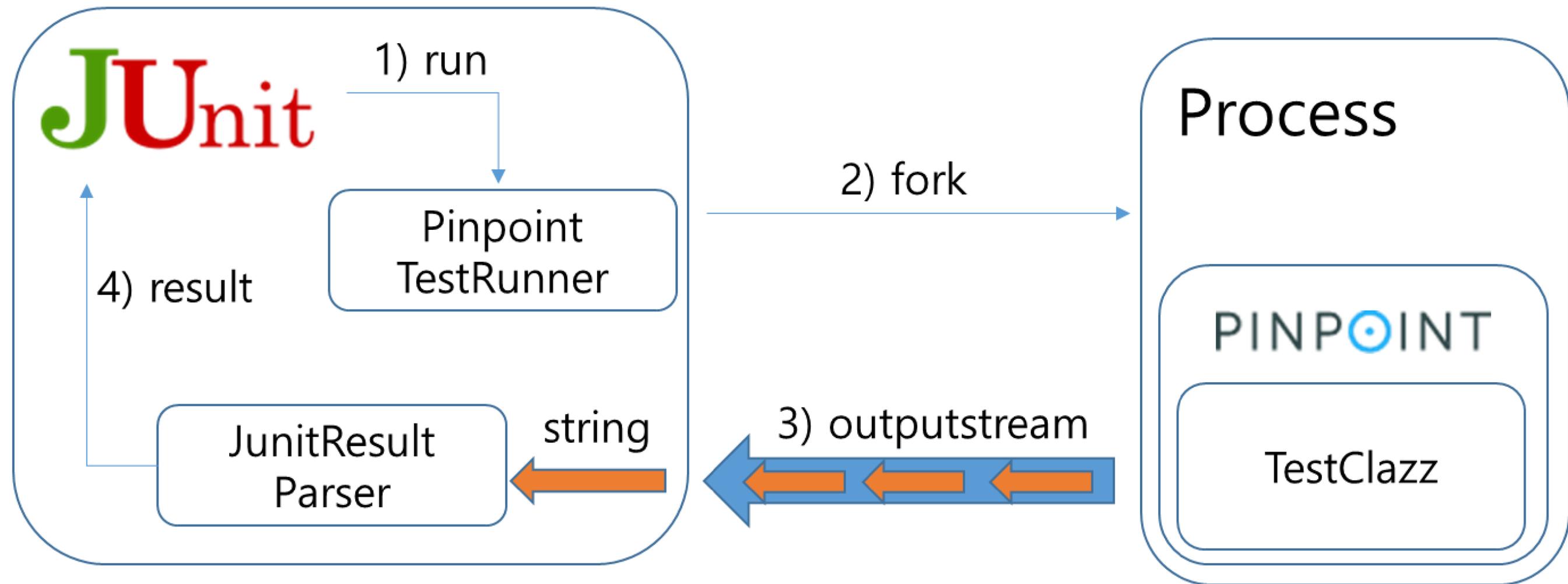
`${PINPOINT_OPTS}`

RunMainTest test.JacksonPluginIT

알고보니 IDE도 이런식



1. 플러그인 테스트 구조



2. 다양한 버전 테스트

DEVIEW
2019

Netty/All In One
Netty/All In One

License	Apache 2.0
Categories	Network App Frameworks
Tags	network socket framework netty io
Used By	1,720 artifacts

Central (129) Spring Library (3) Redhat GA (15) Redhat EA (2) ICM (1)

Version	
5.0.x	5.0.0.Alpha2
	5.0.0.Alpha1
	4.1.41.Final
	4.1.40.Final
	4.1.39.Final

129개!!!!!!

2. 겁쟁이가 되기 싫어졌다

- netty 버전 테스트를 하려면 pom.xml을 129번 바꾸면 되긴하는데.

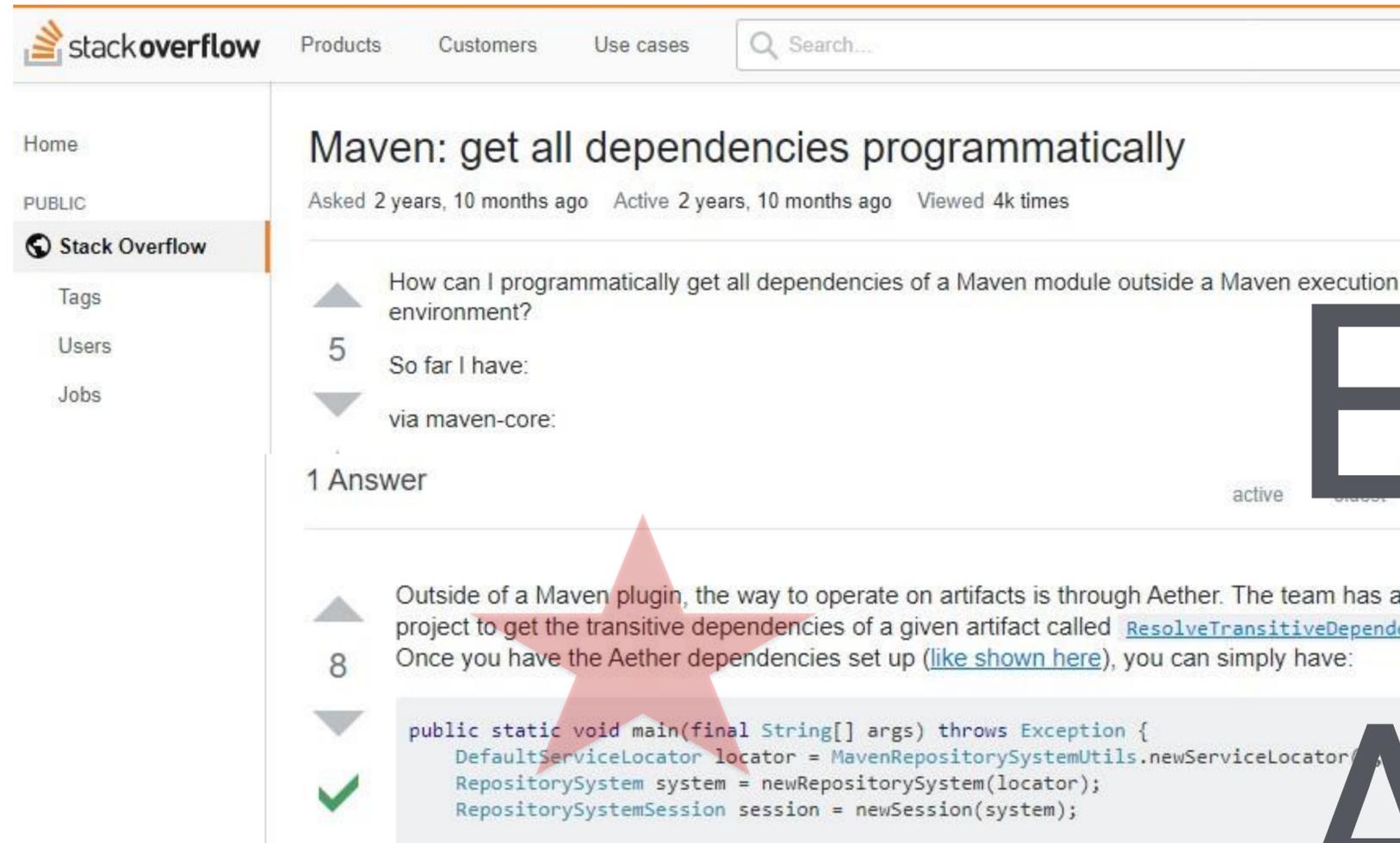


겁쟁이 전용 테스트

용감한 코더는 이 단계를 건너뛴다. 상사를 무서워하고, 직장을 잃을까
고, 고객의 불평 메일을 받거나 고소당하는 일을 걱정하는 프로그래머가
다. 이러한 두려움 때문에 행동에 제약을 받고, 생산성이 떨어진다는 것
아는 사실이다. 연구에 따르면 테스트 단계를 없애는 것이 관리자로서 하
을 미리 결정할 수 있게 하는 등 계획 단계부터 많은 도움이 된다. 두려움
다면 혁신과 실험정신이 창결할 수 있다. 프로그래머는 코드를 생산하는
수 있고, 헬프 데스크와 기존 유지보수 그룹이 협력해서 디버깅을 담당할
우리의 코딩 능력을 온전히 믿는다면 테스트 따위는 더 이상 불필요하다.

2. Aether

DEVIEW
2019



The screenshot shows a Stack Overflow page for the question "Maven: get all dependencies programmatically". The question asks how to programmatically get all dependencies of a Maven module outside a Maven execution environment. The top answer, with 8 votes, explains that Aether is the way to operate on artifacts outside a Maven plugin and provides a code snippet for using Aether to resolve transitive dependencies.

stackoverflow Products Customers Use cases Search...

Home
PUBLIC
Stack Overflow
Tags
Users
Jobs

Maven: get all dependencies programmatically

Asked 2 years, 10 months ago Active 2 years, 10 months ago Viewed 4k times

▲ How can I programmatically get all dependencies of a Maven module outside a Maven execution environment?
5 So far I have:
▼ via maven-core:
1 Answer

▲ 8 Outside of a Maven plugin, the way to operate on artifacts is through Aether. The team has a sample project to get the transitive dependencies of a given artifact called [ResolveTransitiveDependencies](#). Once you have the Aether dependencies set up ([like shown here](#)), you can simply have:

```
public static void main(final String[] args) throws Exception {  
    DefaultServiceLocator locator = MavenRepositorySystemUtils.newServiceLocator();  
    RepositorySystem system = newRepositorySystem(locator);  
    RepositorySystemSession session = newSession(system);
```

Eclipse

Aether

2. Version Scheme

- 특정 버전 명시 : $[1.3.2] \Rightarrow 1.3.2$
- 특정 버전 여러 개 명시 : $[1.3.2], [1.3.5] \Rightarrow 1.3.2, 1.3.5$
- 버전 범위 : $[1.3.2, 1.3.10] \Rightarrow 1.3.2 \sim 1.3.10$
- 버전 범위 : $[1.3.2, 1.3.10) \Rightarrow 1.3.2 \sim 1.3.9$
- 버전 범위 : $(1.3.2, 1.3.10] \Rightarrow 1.3.3 \sim 1.3.10$
- 버전 범위 : $(1.3.2, 1.3.10) \Rightarrow 1.3.3 \sim 1.3.9$

2. Version Scheme

- 버전 범위 : $[1.3.2, 1.3.max]$ => 1.3.2부터 1.3.x 최신 버전 까지
1.3.2 ~ 1.3.33
- 버전 범위 : $[1.3.min, 1.3.max]$ => 1.3.0 ~ 1.3.33 (1.3.가장최신)
- 버전 범위 : $[1.3.min,)$ => 1.3.0 부터 최신 버전 까지
1.3.0 ~ 2.0.27

2. artifact로 버전 가져오기

@Test

```
public void getPinpointVersionTest() throws VersionRangeResolutionException {  
    // 메이븐 아티팩트 생성  
    DefaultArtifact artifact = new DefaultArtifact("com.navercorp.pinpoint", "pinpoint", "jar", "[1.min,)");  
  
    // Request 파라미터로 아티팩트와 메이븐 리파지토리 포함  
    VersionRangeRequest rangeRequest = new VersionRangeRequest();  
    rangeRequest.setArtifact(artifact);  
    rangeRequest.setRepositories(getLocalRepository());  
  
    // 호출  
    VersionRangeResult rangeResult = getSystem().resolveVersionRange(getSession(), rangeRequest);  
    List<Version> versionList = rangeResult.getVersions();  
  
    System.out.println(versionList);  
}
```

```
[1.6.0, 1.6.1, 1.6.2, 1.7.0, 1.7.1, 1.7.2, 1.7.3-p1, 1.8.0-RC1, 1.8.0-SNAPSHOT, 1.8.0, 1.8.1-RC1,  
1.8.1-SNAPSHOT, 1.8.1, 1.8.2-SNAPSHOT, 1.8.3-SNAPSHOT, 1.8.3, 1.8.4-SNAPSHOT, 1.8.5-SNAPSHOT, 1.9.0-SNAPSHOT]
```

2. 의존 라이브러리 가져오기

@Test

```
public void getPinpointDependencyTest() throws Exception{
```

```
    // 메이븐 아티팩트 생성
```

```
    DefaultArtifact artifact = new DefaultArtifact("com.navercorp.pinpoint", "pinpoint-rpc", "jar", "1.8.3");
```

```
    // 디펜던시 요청 메이븐 RUNTIME 범위만 가져오게함 ,
```

```
    Dependency dependency = new Dependency(artifact, JavaScopes.RUNTIME);
```

```
    DependencyFilter classpathFilter = DependencyFilterUtils.classpathFilter(JavaScopes.RUNTIME);
```

```
    // 파라미터로 디펜던시, 리파지토리
```

```
    CollectRequest collectRequest = new CollectRequest((Dependency)null, Arrays.asList(dependency), getLocalRepository());
```

```
    DependencyRequest dependencyRequest = new DependencyRequest(collectRequest, classpathFilter);
```

```
    // 호출 (로컬에 라이브러리가 없는 경우 로컬 리파지토리에 저장) & 결과값 출력
```

```
    DependencyResult result = getSystem().resolveDependencies(getSession(), dependencyRequest);
```

```
    System.out.println(getFiles(result));
```

```
}
```

```
[C:\Users\USER\.m2\repository\com\navercorp\pinpoint\pinpoint-rpc\1.8.3\pinpoint-rpc-1.8.3.jar,  
C:\Users\USER\.m2\repository\com\navercorp\pinpoint\pinpoint-annotations\1.8.3\pinpoint-annotations-1.8.3.jar,  
C:\Users\USER\.m2\repository\com\navercorp\pinpoint\pinpoint-commons\1.8.3\pinpoint-commons-1.8.3.jar,  
C:\Users\USER\.m2\repository\io\netty\netty\3.10.6.Final\netty-3.10.6.Final.jar]
```

2. 어노테이션으로 기능 구현

```
@RunWith(PinpointPluginTestSuite.class)
@PinpointAgent (AgentPath.PATH)
@JvmVersion({6, 7})
@Dependency({"com.zaxxer:HikariCP-java6:(2.3.11,]", "com.h2database:h2:1.4.191"})
public class HikariCpIT {
```

2. 다양한 버전 테스트 결과

✓ HikariCpIT	12 s 947 ms
✓ [HikariCP-java6-2.3.12:child:7]	3 s 273 ms
✓ defaultTest1[HikariCP-java6-2.3.12:child:7]	1 s 258 ms
✓ defaultTest3[HikariCP-java6-2.3.12:child:7]	1 s 10 ms
✓ defaultTest2[HikariCP-java6-2.3.12:child:7]	1 s 5 ms
✓ [HikariCP-java6-2.3.12:child:6]	3 s 246 ms
✓ defaultTest1[HikariCP-java6-2.3.12:child:6]	1 s 240 ms
✓ defaultTest3[HikariCP-java6-2.3.12:child:6]	1 s 8 ms
✓ defaultTest2[HikariCP-java6-2.3.12:child:6]	998 ms
✓ [HikariCP-java6-2.3.13:child:7]	3 s 239 ms
✓ defaultTest1[HikariCP-java6-2.3.13:child:7]	1 s 225 ms
✓ defaultTest2[HikariCP-java6-2.3.13:child:7]	1 s 8 ms
✓ defaultTest3[HikariCP-java6-2.3.13:child:7]	1 s 6 ms
✓ [HikariCP-java6-2.3.13:child:6]	3 s 189 ms
✓ defaultTest1[HikariCP-java6-2.3.13:child:6]	1 s 177 ms
✓ defaultTest2[HikariCP-java6-2.3.13:child:6]	1 s 7 ms
✓ defaultTest3[HikariCP-java6-2.3.13:child:6]	1 s 5 ms

1.4.191"))

잘된다!!

3. github으로 pull request

DEVIEW
2019

Code Issues 95 Pull requests 19 Wiki Security Insights Settings

[Doc] Modules Update #6018

Open RoySRose wants to merge 1 commit into naver:master from RoySRose:BranchForModuleDoc

Conversation 1 Commits 1 Checks 0 Files changed 1

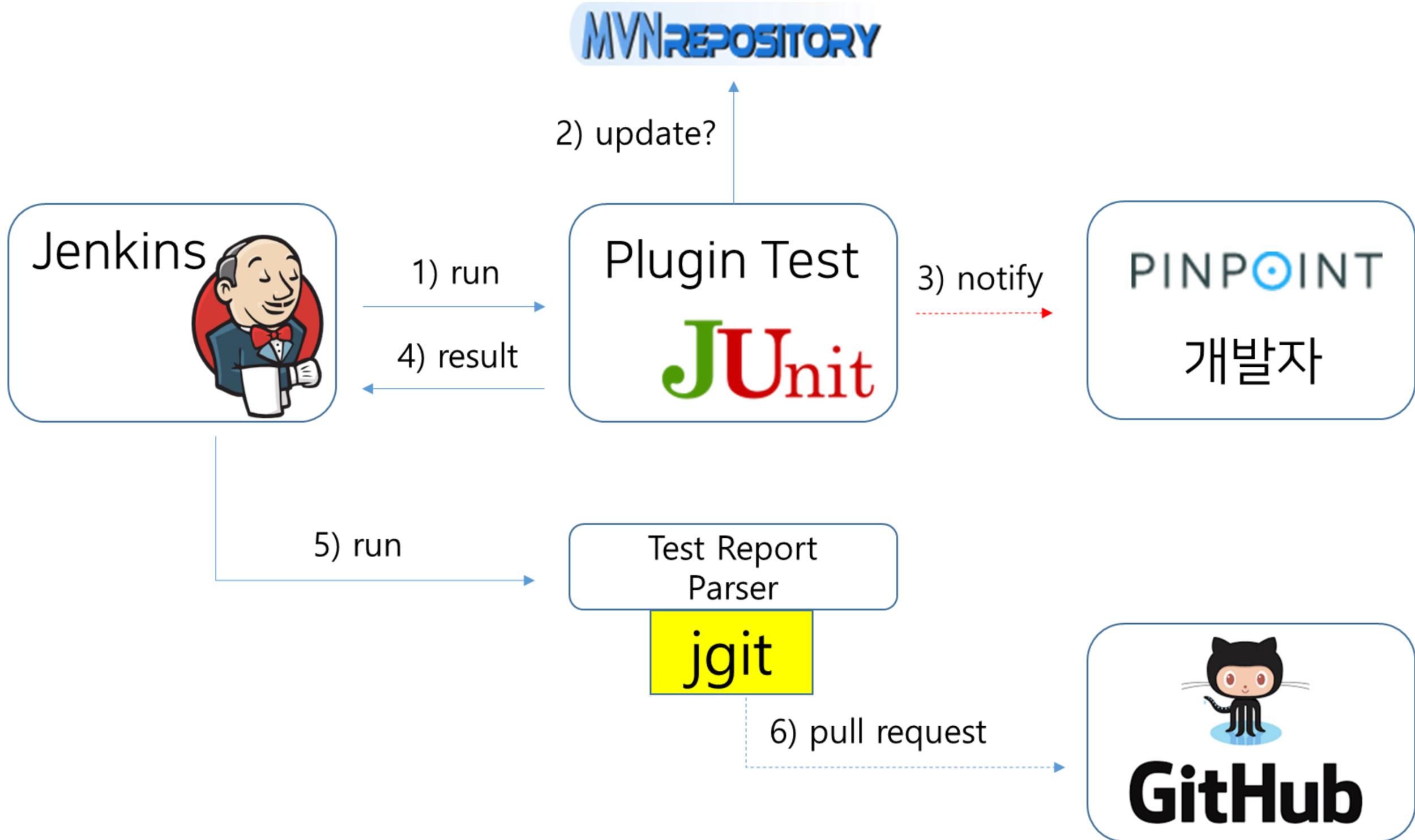
Changes from all commits File filter... Jump to... ⚙

6 doc/modules.md

```
@@ -29,7 +29,7 @@
- | Httpclient3
+ | Httpclient3
| 3.0 | 3.1 | <AG>
- | Httpclient4
+ | Httpclient4
| 4.0 | 4.5.4 | <AG>
- | Thrift
+ | Thrift
| 0.9.1 | 0.12.0 | <AG>
- | Google HTTP Client
+ | Google HTTP Client
| 1.19.0 | 1.32.0 | <AG>
+ | 1.19.0 | 1.32.1 | <AG>
- | AsyncHttpClient
+ | AsyncHttpClient
| 1.7.24 | 1.8.17 | <AG>
- | OkHttp
+ | OkHttp
| 2.0.0 | 3.10.0 | <AG>
- | Apache HttpAsyncClient
+ | Apache HttpAsyncClient
| 4.0 | 4.1.3 | <AG>
@@ -64,13 +64,13 @@
- | \*[Redis](https://github.com/naver/pinpoint/tree/master/plugins/redis-redisson)
+ | \*[Redis](https://github.com/naver/pinpoint/tree/master/plug
| 3.10.0 | 3.10.4 |
- |
+ |
|
- | Apache CXF
+ | Apache CXF
| 3.0.0 | 3.3.3 | <AG>
- | Netty
+ | Netty
| 4.1.0 | 4.1.41 | <AG>
+ | 4.1.0 | 4.1.42 | <AG>
```

1.32.0 => 1.32.1

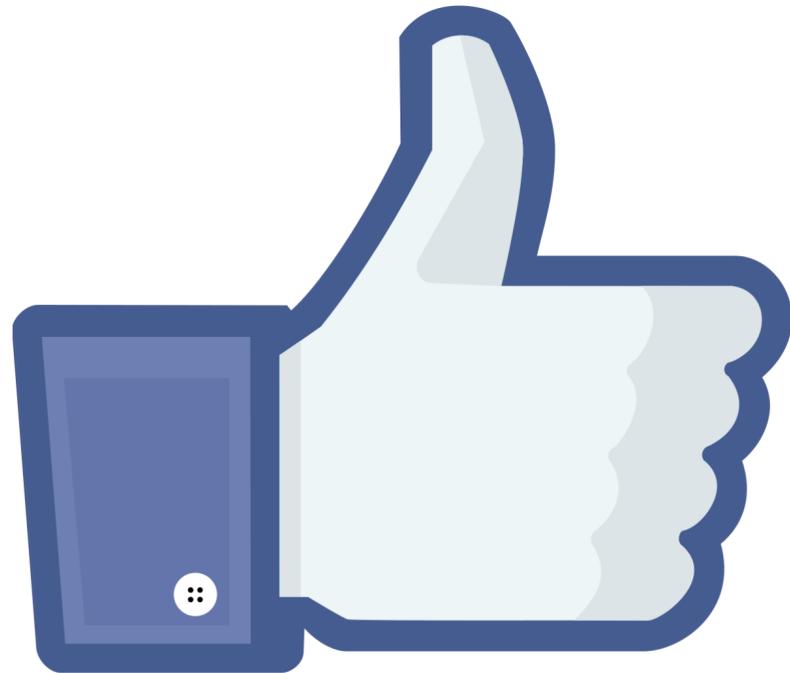
3. 지원 라이브러리 업데이트 구조



3. 테스트의 자가증식

- 코드는 그대론데 테스트가 계속 늘어난다.

테스트 작성 252개

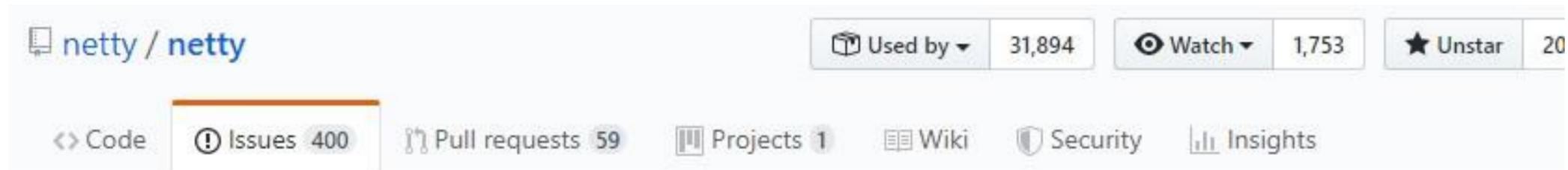


-> 처음 1300개

-> 지금 4512개

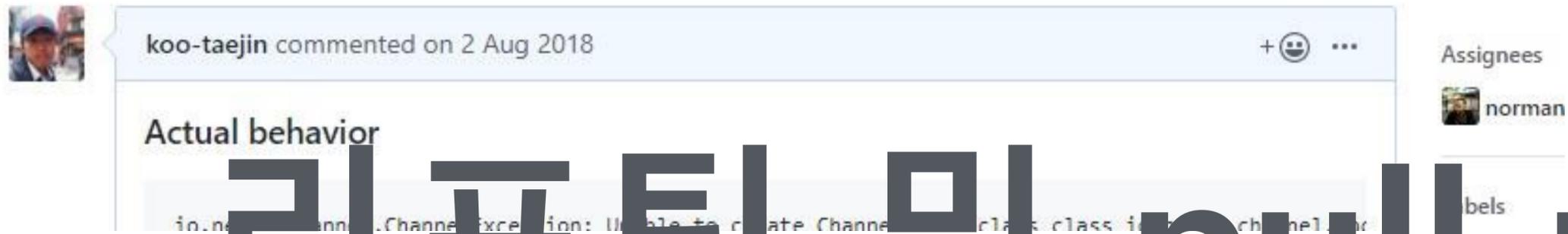
3. 핀포인트 테스트를 넘어서

- netty 플러그인 테스트 에러가 났는데 아무리 봐도 이상함



In netty-4.1.28.Final version, an error occurs when running on jdk6. #8166

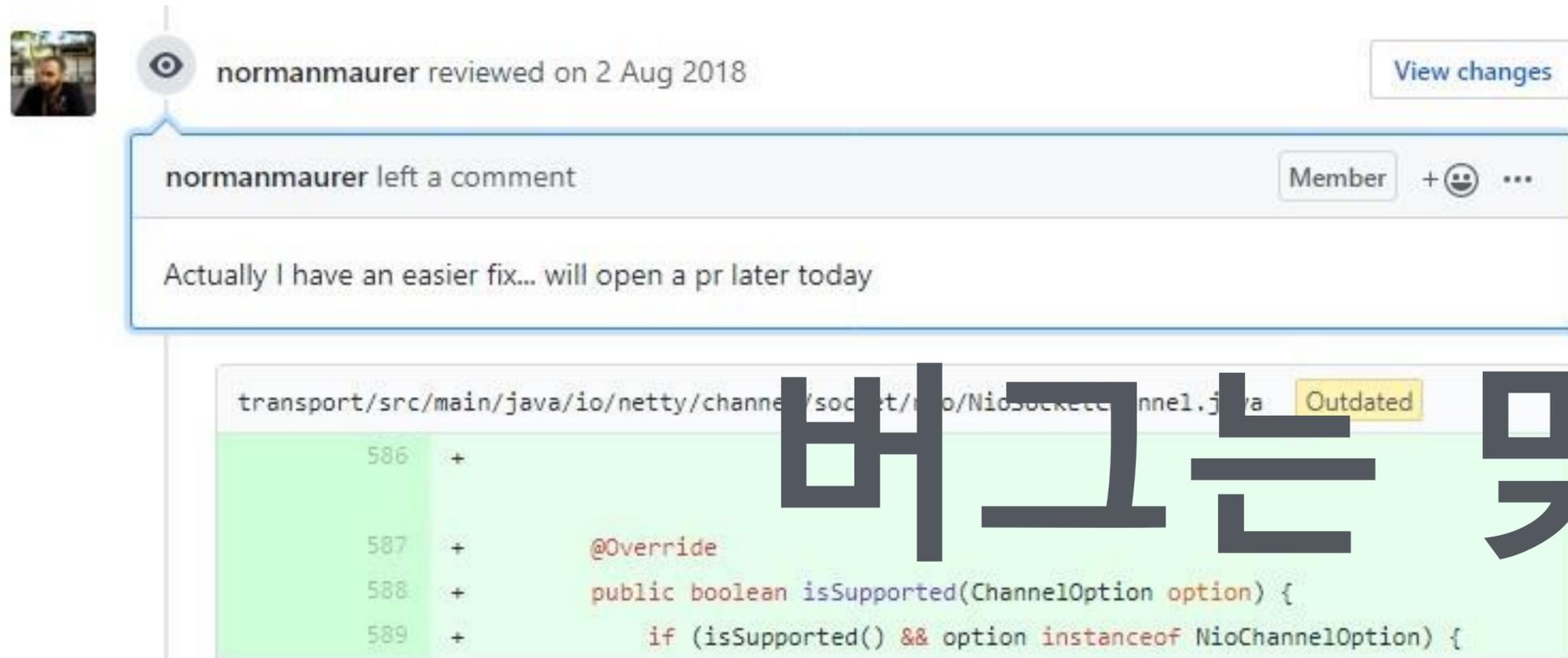
Closed koo-taejin opened this issue on 2 Aug 2018 · 1 comment



리포팅 및 pull request

3. 핀포인트 테스트를 넘어서

- netty 플러그인 테스트 에러가 났는데 아무리 봐도 이상함



버그는 맞는데

직접 고친다고함

3. 핀포인트 테스트를 넘어서

Netty 4.1.29.Final released

by normanmaurer
on 24-Aug-2018

Today we announce the release of netty 4.1.29.Final. This release contains bugfixes and some improvements.

✦ **develar** commented on 14 Aug 2018 Contributor + 😊 ...

Thanks a lot for fix, because master of IntelliJ IDEA community is affected (183). Do you have any plans to release 4.1.29 soon?

for NS servers that have no ADDITIONAL records (#8177)
 of the same netty artifact can be loaded as long as the shaded prefix is different (#8207)
 creating an OpenSSL engine in client mode (#8178)
 ch domain results in invalid hostname (#8180)
 use an AssertionError when calling ServerSocketChannel.config().getOptions() (#8183)
 ed on Java6 again (#8168)
 packetArray and lovArray in EpollEventLoop (#8160)
 e browse our issue tracker for 4.1.29.Final.

intellij 개발자도 빨리 고쳐주세요라고

Thank You

기여 & 뿌듯

Every idea and bug-report counts and so we thought it is worth mentioning those who helped in this area. Please report an unintended omission.

- @hc-codersatlas
- @jianglai
- @koo-taejin

THANK YOU

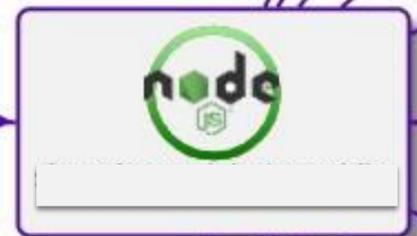
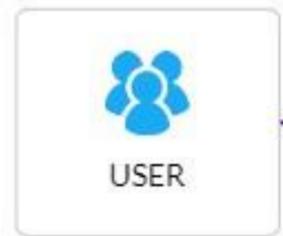
Every idea and bug-report counts and so we thought it is worth mentioning those who helped in this area. Please report an unintended omission.

- @hc-codersatlas
- @jianglai
- @koo-taejin

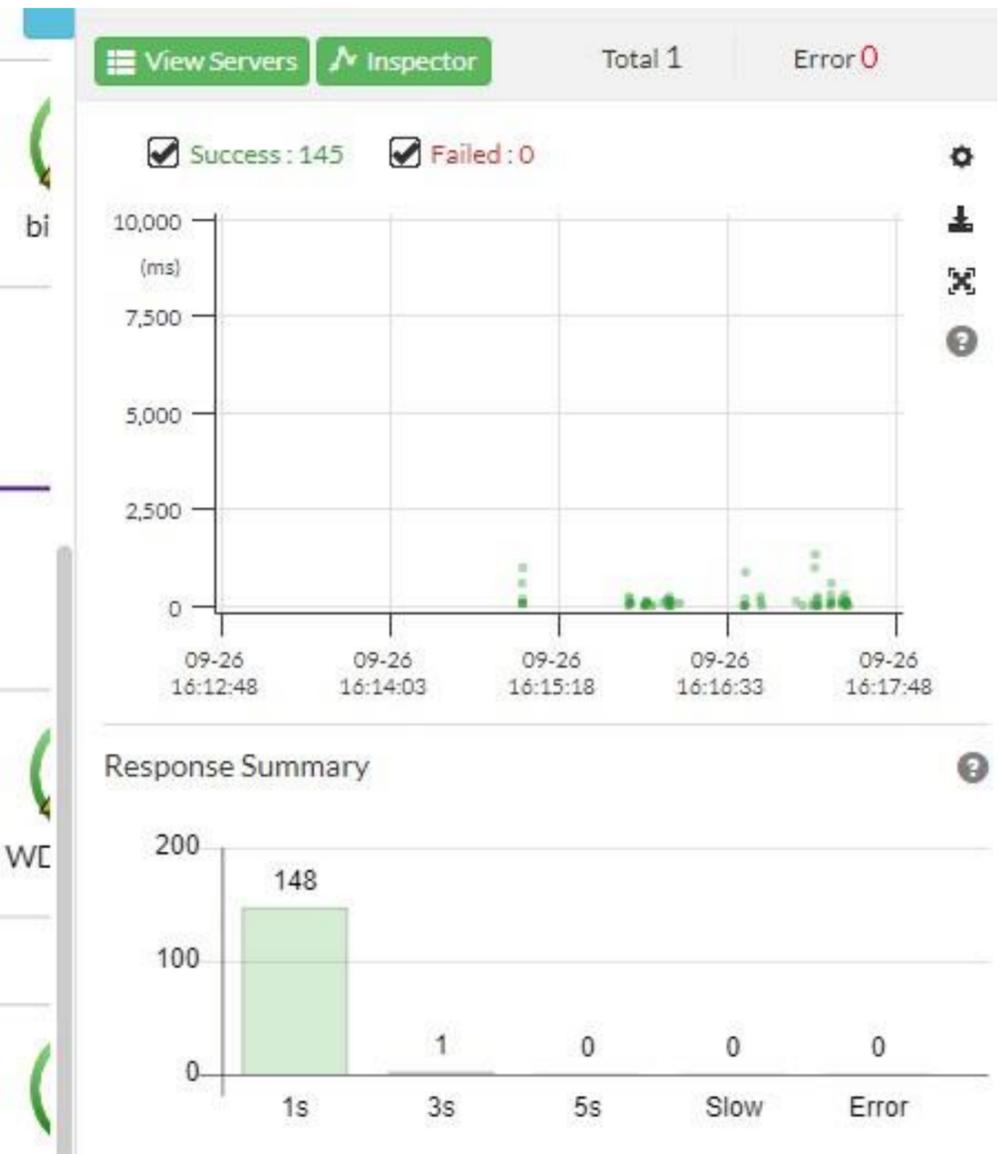
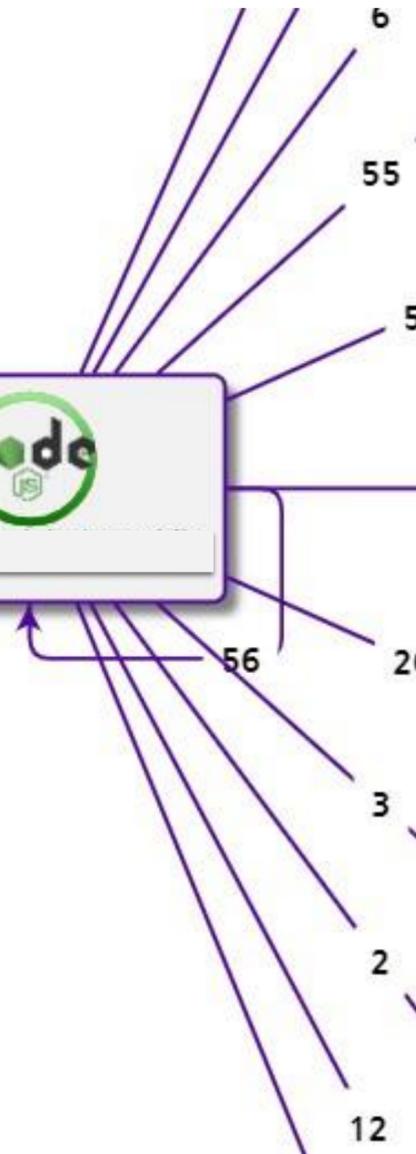
5. What's next

Multi language support

DEVIEW
2019

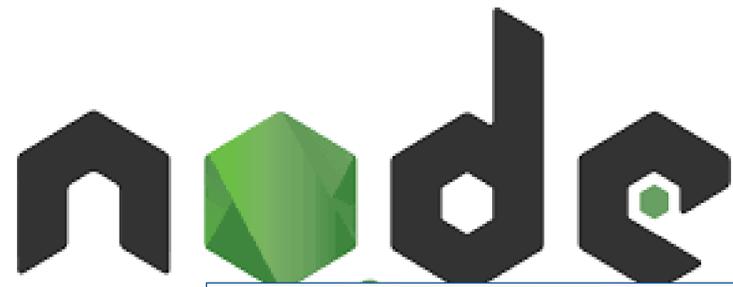


93



Multi language support

DEVIEW
2019



The screenshot displays the DEVIEW monitoring interface. At the top, there are buttons for 'View Servers' and 'Inspector', along with status indicators 'Total 1' and 'Error 0'. Below these, it shows 'Success : 145' and 'Failed : 0'. A central panel contains the following text:

TRACK 3 DAY 2 / 16:00~16:45

```
console.warn('좀 불안하지?') //node.js 모니터  
링을 위한 pinpoint node agent 개발기
```

At the bottom right, a bar chart shows performance metrics. The x-axis categories are 1s, 3s, 5s, Slow, and Error. The y-axis ranges from 0 to 200. The values for each category are: 1s: 148, 3s: 1, 5s: 0, Slow: 0, Error: 0.

Category	Count
1s	148
3s	1
5s	0
Slow	0
Error	0

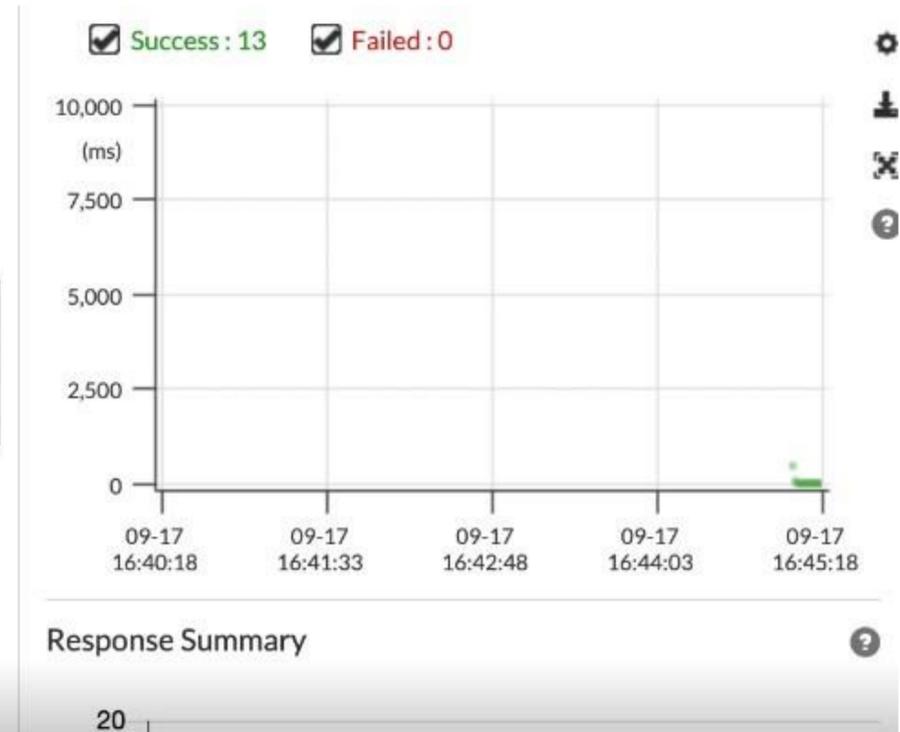
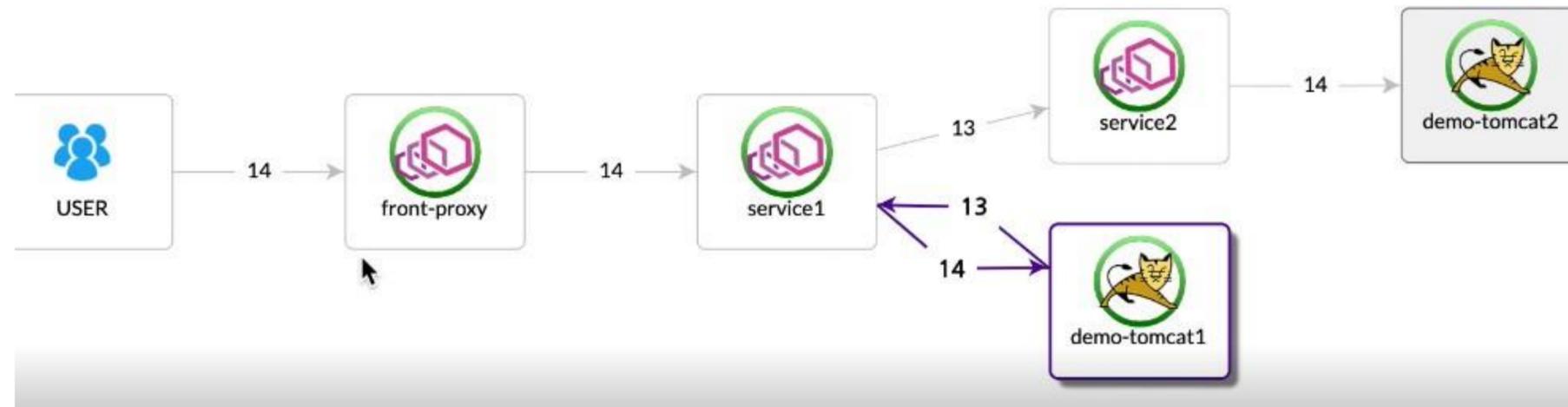
Multi language support (gRpc)

DEVIEW
2019



Multi language support (gRpc)

DEVIEW
2019



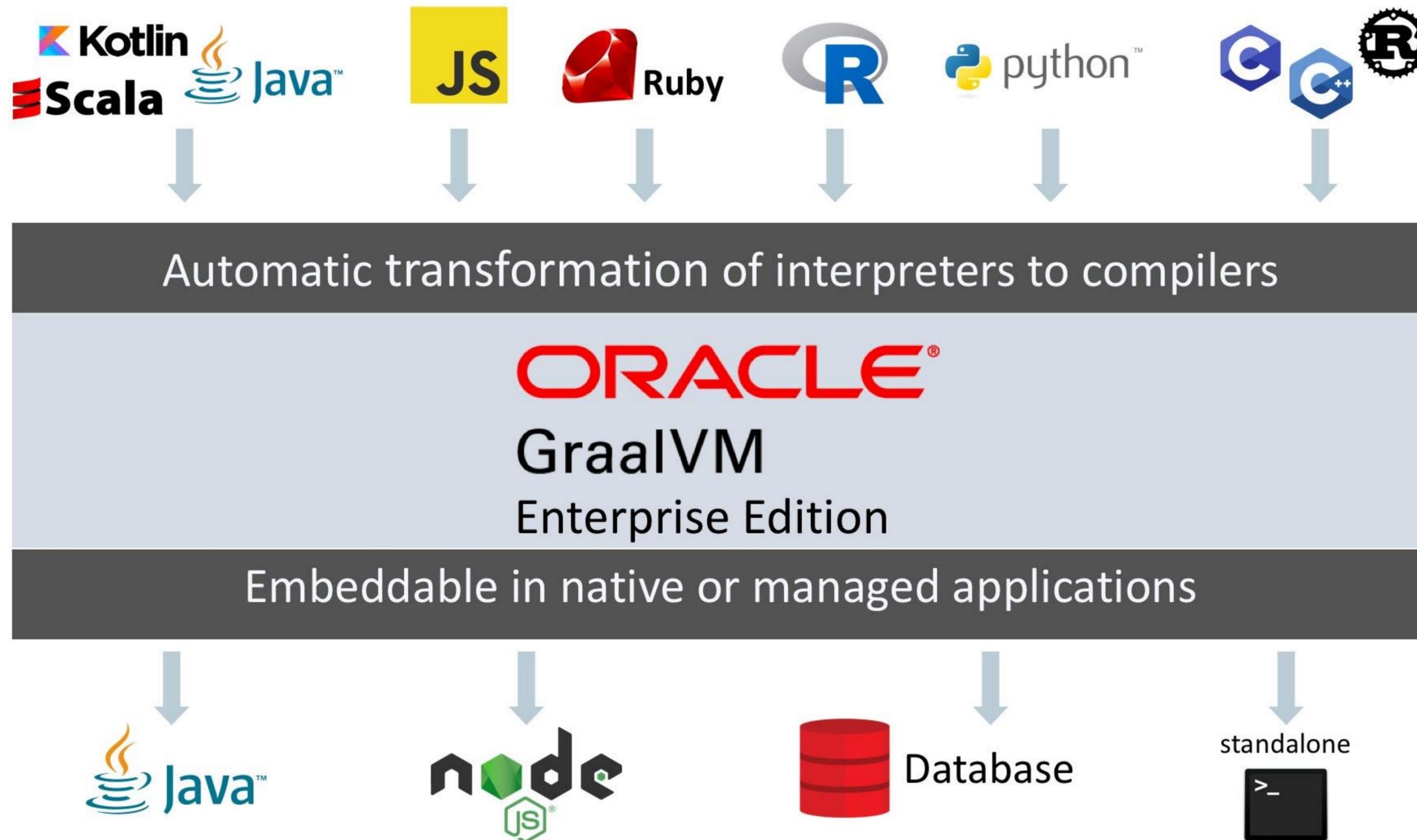
Multi language support (GraalVM)

DEVIEW
2019



Multi language support (GraalVM)

DEVIEW
2019



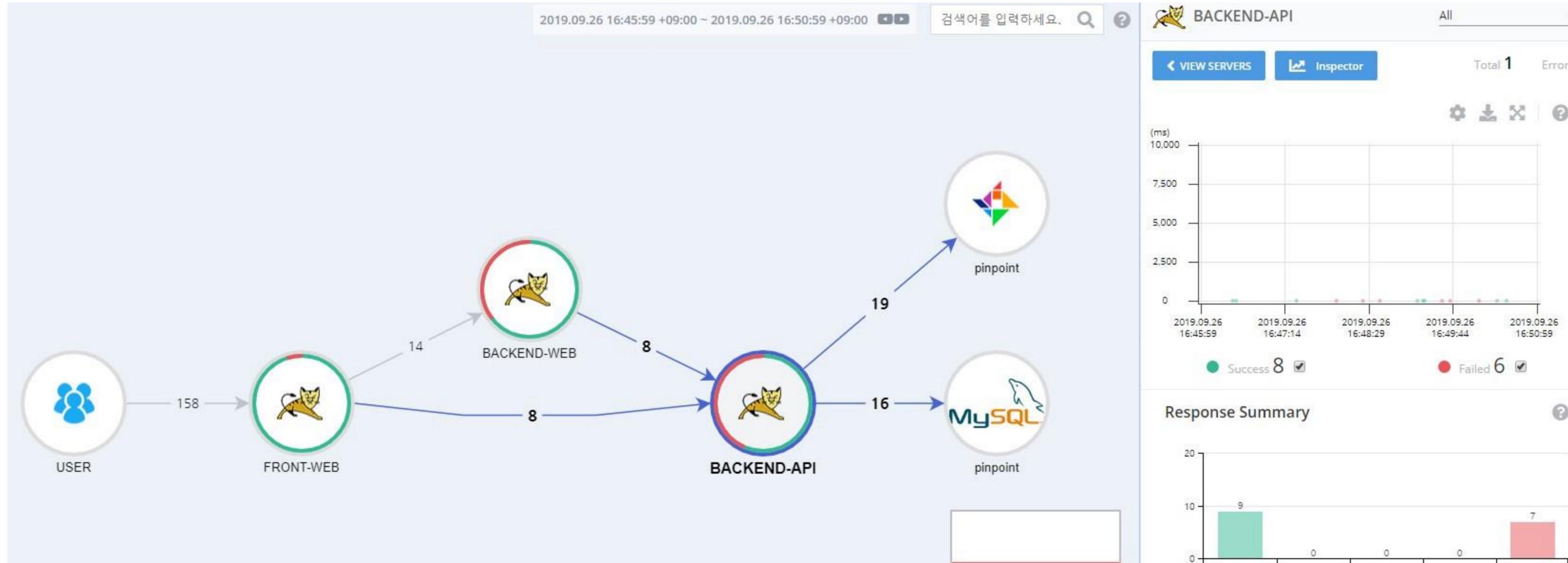
Integration with k8s

DEVIEW
2019

PINPOINT

New UI

DEVIEW
2019



서버맵

New UI

DEVIEW
2019



인스펙터

Community

DEVIEW
2019



홍콩, 대만
오픈소스 컨퍼런스
참가 및
세션 진행

Community

DEVIEW
2019



글로벌

APM 리더

초대 및 세미나

진행

@강남



Community

DEVIEW
2019



APM 주요 이슈

공유 및 토론

@ 그린팩토리

Community

DEVIEW
2019



글로벌

APM 리더

한국 문화

공유 및

실습

Q & A

Thank You