Pinpoint 로 어플리케이션에 대한 Naver 수준의 가시성 확보하기
Index

- Microservice Architecture
- Observability in Naver
- Enhance Observability with Pinpoint
- Open source Pinpoint
- Open source Community
- Troubleshooting Distributed Systems
- How Pinpoint Works
- Troubleshooting with Pinpoint
- Future Plans
Microservice Architecture

- Development
- Maintenance
- Reliability
- Scalability
- Cost
- Deployment
- Releasing
Microservice Architecture

- Development
- Maintenance
- Reliability
- Scalability
- Cost
- Deployment
- Releasing

Microservices

From Wikipedia, the free encyclopedia

Some of this article’s listed sources may not be reliable. Please help this article by looking for better, more reliable sources. Unreliable citations may be challenged or deleted. (October 2018) (Learn how and when to remove this template message)

Microservices are a software development technique—a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services.[1] In a microservices architecture, services are fine-grained and the protocols are lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion.[2] It parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently.[3] It also allows the architecture of an individual service to emerge through continuous refactoring.[4] Microservice-based architectures facilitate continuous delivery and deployment.[5]
Microservice Architecture

- Development
- Maintenance
- Reliability
- Scalability
- Cost
- Deployment
- Releasing

Microservice-based architectures facilitate continuous delivery and deployment.
New Research Shows 63% of Enterprises Are Adopting Microservices Architectures

Despite new research, 50% of enterprises are unaware of microservices’ impact on revenue-generating business processes.

by Tom Smith · Sep. 20, 18 · Microservices Zone · Interview

Download DZone's 2019 Scaling DevOps Trend Report to learn how to ensure security as you scale DevOps, why “blameless retrospectives” are a myth, and the key considerations for getting the most out of DevOps as you scale.

Presented by DZone

https://dzone.com/articles/right-strategies-for-microservices-deployment
Adopting Microservices at Netflix: Lessons for Architectural Design

Tony Mauro of NSINX, Inc.  February 19, 2015

Blog  Tech

Microservices, containers, Netflix, cloud delivery network (CDN)

We recently blogged about our adoption of microservices at Netflix. It is becoming clear that if you keep using development processes and application architectures that worked just a few years ago, you simply can’t move fast enough to capture and hold the interest of Netflix users who can choose from an ever-growing number of apps.

According to a microservices architecture creates exciting opportunities in the marketplace for companies. For architects and developers, it promises an unprecedented level of control and speed as they deliver new experiences to customers.

In the real world, you can’t stop developing and deploying your apps as you prototype the processes.
Microservice Architecture

- Cryptocurrency Market
- Cloud Platform
- Search Engine
- Webtoon Platform
- Business Platform
- R & D Center
- Video Streaming Service
- Camera Mobile App
- AI Speaker
Microservice Architecture

- Deployment Management
- Debugging
Microservice Architecture

- Deployment Management
- Debugging

Containerization timeline

- Docker containerization debuted
- 20% of companies have containers deployed
- 50% of companies have containers deployed

SOURCE: GARTNER
Microservice Architecture

Docker Adoption Status by Infrastructure Size

Source: Datadog
Microservice Architecture
Microservice Architecture
Microservice Architecture

- Deployment Management
- Debugging
Observability in Naver

- Deployment Management
- Debugging
Observability in Naver

Observability

- Tracing
- Logging
- Metric
Observability in Naver

Observability

Tracing

Logging

Metric

그리고 사람
Observability in Naver

Observability

Logging

Tracing

Metric

NELO2
Observability in Naver

Observability

Tracing

Logging  Metric

PINPOINT
Observability in Naver

Observability

- Tracing
- Logging
- Metric

PINPOINT

NELO2

NPOT
Observability in Naver

Observability

Tracing

Logging  Metric

PINPOINT
Enhance Observability

Bird Eye View

Finding Slow Transactions
Distributed Tracing
DevOps
Scalable
Minimum Overload
Bird Eye View

- Server Map
Bird Eye View

- Server Map
Bird Eye View

- Server Map
Bird Eye View
Bird Eye View
Enhance Observability

Bird Eye View
Finding Slow Transactions
Distributed Tracing
DevOps
Scalable
Minimum Overload
Finding Slow Transaction

- Scatter Chart

Response Time of each Transaction

Date & Time

Success: 107  Failed: 7
Finding Slow Transaction
Finding Slow Transaction

Slow Transactions

Success: 141,183
Failed: 289

Graph showing transaction times with a red circle highlighting slow transactions.
Finding Slow Transaction

![Graph showing Success: 3,895 and Failed: 2]
Finding Slow Transaction

![Chart showing success and failed transactions over time](chart.jpg)

*WHAT!*
Finding Slow Transaction

- Success: 4,906
- Failed: 0
Finding Slow Transaction

- Malfunctioning in Real Server
Finding Slow Transaction

- Malfunctioning in Real Server
Finding Slow Transaction

- Malfunctioning in Real Server

Reboot
Finding Slow Transaction

- Response Summary chart
- Load chart
Enhance Observability

Bird Eye View
Finding Slow Transactions
Distributed Tracing
DevOps
Scalable
Minimum Overload
Finding Slow Transaction
Distributed Tracing

- Call Stack

List of Selected Transaction

The Call Stack with Code Level
# Distributed Tracing

## PINPOINT Tracing Interface

### Tracing Information
- **Start Time**: Various times ranging from 08:17 to 22:53
- **Path**: URLs starting with `/shopping/orders/1340/` and `/shopping/orders/1339/`
- **Action**: Actions like `GET` and `POST` with varying parameters
- **Exception**: Various exceptions
- **Agent**: `apiapp01`
- **Client IP**: IP addresses
- **Transaction**: Unique transaction IDs

### Application Flow
- **Call Tree**: Diagram showing the flow from USER to ApiGateway, Shopping-Api, Shopping-Order, ApiGateway, and finally to MySQL.
- **Transaction ID**: `-apiapp01-15555100150090-3378633`
- **Agent ID**: `apiapp01`
- **Application Name**: `ApiGateway`
Distributed Tracing
## Distributed Tracing

The image shows a distributed tracing tool interface with details of transaction paths and call trees, highlighting specific components such as `Shopping.Order`, `Payment`, and `MySQL`. The trace data includes timestamps, durations, and other metrics related to the transaction process.
Enhance Observability

PINPOINT

Bird Eye View
Finding Slow Transactions
Distributed Tracing
DevOps
Scalable
Minimum Overload
DevOps

- Realtime Active Thread Chart

Can be used as HealthCheck
DevOps
DevOps
Clean =
DevOps

• Inspector

Basic Info of the Instance

• Heap, Non-Heap Memory
• JVM/SYSTEM CPU
• JVM GC
• TPS, Active Thread
• Response Time
• File Descriptor
• Direct/Mapped Buffer
• Data Source
Enhance Observability

Bird Eye View
Finding Slow Transactions
Distributed Tracing
DevOps
**Scalable**
Minimum Overload
Scalable

- Fully functioning in Naver

70 billion Span Chunk (trace segment) per day
Scalable

- Fully functioning in Naver

12,000+ Pinpoint Agent

17 Collectors

63 region server

Hbase

70 billion Span Chunk (trace segment) per day
Scalable

• Fully functioning in Naver

12,000+ Pinpoint Agent

1,800+ Application

17 Collectors

Peek TPS : 870K

Write TPS : 1M

63 region server

Hbase

70 billion Span Chunk (trace segment) per day
Scalable

• Fully functioning in Naver

12,000+ Pinpoint Agent

1,800+ Application

70 billion Span Chunk (trace segment) per day

17 Collectors

Write TPS : 1M

63 region server

Hbase

Peek TPS : 870K

Application

Application

Application

Application

agent

agent

agent

agent
Enhance Observability

PINPOINT

Bird Eye View
Finding Slow Transactions
Distributed Tracing
DevOps
Scalable

Minimum Overload
Minimum Overload

- Execute Integration Tests Periodically
  - **Less than 3%** difference in performance
  - Sampling ‘No-Agent’, ‘5%’, ‘100%’

---

<table>
<thead>
<tr>
<th>VUser 4</th>
<th>VUser 32</th>
<th>VUser 64</th>
<th>VUser 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration (ms)</td>
<td>Tests</td>
<td>Error</td>
<td>Mean Test Time (ms)</td>
</tr>
<tr>
<td>300000</td>
<td>135003</td>
<td>0</td>
<td>6.87</td>
</tr>
<tr>
<td>300000</td>
<td>123282</td>
<td>0</td>
<td>7.42</td>
</tr>
<tr>
<td>300000</td>
<td>127462</td>
<td>0</td>
<td>7.29</td>
</tr>
</tbody>
</table>
Open Source Pinpoint

Jun 17, 2012 – Oct 14, 2019

Contributions to master, excluding merge commits

2012.07 Development Start
2014.01 First use of Pinpoint
2014.12 90% teams switched from commercial APM
2015.01 Starting open source
2017.12 Reached 5000 stars

PINPOINT

NAVER
Open Source Pinpoint

APM, (Application Performance Management) tool for large-scale distributed systems written in Java.

https://naver.github.io/pinpoint/

⚠️ We found potential security vulnerabilities in your dependencies.
You can see this message because you have been granted access to security alerts for this repository.
Open Source Pinpoint

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

COSCon 2019 Coming!

- **Date**: 2019-11-02 09:00 ~ 11-03 17:00
- **Location**: East China Normal University (Zhongshan North Road Campus), Putuo District, Shanghai, Putuo, Shanghai

The event is organized by KAIYUANSHE
Open Source Community
Open Source Community

Topics
- Service Signals
- Service Status
- Service Sentiment

- Long history of distributed systems
- 100s of different services built by 100s of teams
- Borgmon and Monarch
- Efficient instrumentation stack
Open Source Community
GOOD HOSTING!!!!
Open Source Community

Subtle, but if you look carefully, Naver made @Pinpoint_APM @ASFSkyWalking and @zipkin project cookies. They could have done iced cookies, but that’s another story.

Chris K Wensel @cwensel · 7월 13일
Had some serious FOMO this week.

Wu Sheng 吴荣 @wusheng1108 · 7월 12일
My daughter is the center of event. Haha
Glad to meet old and new friends there. Thanks everyone.

Pinpoint @Pinpoint_APM · 7월 12일
The two days of "Pinpoint Open House" event has finished. Thanks to everyone who has joined us in the event.

Special Thanks to @adriancole @autoletics @wusheng1108 @rakyll @jcchavez @TommyLudwig @dttorikae, Chenguoxi, Liumingyi

and of course, members of Team Pinpoint

Our team member @wusheng1108 is in Seoul, Korea at @Pinpoint_APM’s #opensource conference. He’ll be presenting on the core concepts of @ASFSkyWalking and sharing with the crowd the latest look of SkyWalking’s UI today.
**Scaling Distributed Tracing**

Distributed Tracing Workshop — Seoul — July 2019

Below I've listed the slides I presented on the first day of a distributed tracing workshop organized by Naver Corp. and held in Seoul, South Korea.
Open Source Community
Open Source Community
Microservice Challenges

More areas that can fail

99% success rate over 10 components -> 90.5% success rate
Network error much more relevant

Increase in latency

1s 99th percentile over 10 components -> approx. 80th percentile

Observability greatly reduced

Individual servers does not tell the whole story
Traditional way of troubleshooting no longer works
Troubleshooting Microservices
Troubleshooting Microservices

Your API is slow

USER  ApiGateway  Shopping-API
Troubleshooting Microservices
Troubleshooting Microservices

USER ➔ ApiGateway ➔ Shopping-API ➔ Order, Product
Troubleshooting Microservices
Troubleshooting Microservices

Distributed Transactions

- GET /shopping/products/12345
- POST /shopping/orders
Troubleshooting Microservices

Monolithic vs Microservices

We replaced our monolith with microservices so that every outage could be more like a murder mystery.

4:10 PM - 7 Oct 2015

www.slideshare.net/alvarosanchezmariscal/stateless-authentication-for-microservices
So what do we need?

**Context**
- Identify as part of the same request within a single service instance
- Identify as part of the same transaction within distributed service instances
- Context propagation across thread, or process boundary

**Order**
- Timestamp not enough
- Distributed Nodes - time skew
- Asynchronous Processing - delayed execution

**Structure**
- Order - 1 dimensional
- Call tree requires depth as well
Pinpoint

**Call Stack Trace**
Traces everything that happens in a single instance
Context propagated via thread-local
Order and structure inherently provided by emulating the call stack

**Distributed Transaction Trace**
Stitches multiple call stacks under the same transaction
Context propagated via RPC
Order and structure via parent-child relationship of distributed call stacks
Server Map
Server Map
Distributed Call Tree

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Start Time</th>
<th>Gap (ms)</th>
<th>Exec (ms)</th>
<th>Exec (%)</th>
<th>Self (ms)</th>
<th>Class</th>
<th>API</th>
<th>Application</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlet Process</td>
<td>/emeroad.pinpoint</td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>StandardFilter</td>
<td>TOMCAT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td>http.status.code</td>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REMOTE_ADDRESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoke(Request request, Response response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>StandardFilter</td>
<td>TOMCAT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doGet(HttpservletRequest request, HttpservletResponse response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>FrameworkServlet</td>
<td>SPRING</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>demo()</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>DemoController</td>
<td>SPRING</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpUriRequest request, Response response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>CloseableHttpResponse</td>
<td>HTTP_CLIENT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>open(HttpRoute route, HttpContext context)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>AbstractHttpClient</td>
<td>HTTP_CLIENT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpservletRequest request, HttpservletResponse response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>HttpRequestExecutor</td>
<td>HTTP_CLIENT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>http.status.code</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td>0</td>
<td>StandardFilter</td>
<td>TOMCAT</td>
<td>BACKEND-WEB</td>
<td>BackendWAS1</td>
</tr>
<tr>
<td></td>
<td>http.io</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Servlet Process</td>
<td>/backendWeb.pinpoint</td>
<td>17:30:48 004</td>
<td>0</td>
<td>12</td>
<td></td>
<td>0</td>
<td>StandardFilter</td>
<td>BACKEND-WEB</td>
<td>BACKEND-WEB</td>
<td>BackendWAS1</td>
</tr>
<tr>
<td></td>
<td>http.status.code</td>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REMOTE_ADDRESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoke(Request request, Response response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>12</td>
<td></td>
<td>0</td>
<td>StandardFilter</td>
<td>BACKEND-WEB</td>
<td>BACKEND-WEB</td>
<td>BackendWAS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doPost(HttpservletRequest request, HttpservletResponse response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>12</td>
<td></td>
<td>0</td>
<td>FrameworkServlet</td>
<td>BACKEND-WEB</td>
<td>BACKEND-WEB</td>
<td>BackendWAS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>backendWeb()</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>7</td>
<td></td>
<td>0</td>
<td>DemoController</td>
<td>BACKEND-WEB</td>
<td>BACKEND-WEB</td>
<td>BackendWAS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Distributed Call Tree

### Call Tree

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Start Time</th>
<th>Gap (ms)</th>
<th>Exec (ms)</th>
<th>Exec (%)</th>
<th>Self (ms)</th>
<th>Class</th>
<th>API</th>
<th>Application</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlet Process</td>
<td>/emerroad.pinpoint</td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoke(Request request, Response response)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doGet(HttpServletRequest request, HttpServletResponse response)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>demo()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpUriRequest request, HttpServletRequest request, HttpServletResponse response)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>open(ServletContext context, String resource)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpRequest request, String resource)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Application: /emerroad.pinpoint

Transaction Id: FrontWAS2-1563931395270^189...
AgentId: FrontWAS2
ApplicationName: FRONT-WEB

### Server Map

Timeline
Mixed View
elo
Self >= 1000 (ms)

---

Naver
### Distributed Call Tree

#### HttpClient.execute()

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Start Time</th>
<th>Gap(ms)</th>
<th>Exec(ms)</th>
<th>Exec(%)</th>
<th>Self(ms)</th>
<th>Class</th>
<th>API</th>
<th>Application</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlet Process</td>
<td>/emeroad.pinpoint</td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td>0</td>
<td>0</td>
<td>TONCAT</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>invoke(Request request, Response response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td>0</td>
<td>0</td>
<td>Standard::http::Client</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>doGet(HttpServletRequest request, HttpServletResponse response)</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td>0</td>
<td>0</td>
<td>Framework::Servlet</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>demo2()</td>
<td></td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td>0</td>
<td>0</td>
<td>DemoController</td>
<td>FRONT-WEB</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>execute(HttpUriRequest request, HttpResponse response)</td>
<td></td>
<td>17:30:48 200</td>
<td>200</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>CloseableHttpClient</td>
<td>HTTP_CLIENT</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>openHttpRoute(true, HttpRouteConfig for default workload)</td>
<td></td>
<td>17:30:48 200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>AbstractPooledHttpClient</td>
<td>HTTP_CLIENT</td>
<td>FrontWAS2</td>
<td></td>
</tr>
<tr>
<td>HttpClient.execute()</td>
<td></td>
<td>17:30:48 200</td>
<td>12</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>TONCAT</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>http.io</td>
<td></td>
<td>17:30:48 200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>TONCAT</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>Servlet Process</td>
<td>/backendweb.pinpoint</td>
<td>17:30:48 200</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>TONCAT</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>http.status.code</td>
<td></td>
<td>17:30:48 200</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>TONCAT</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>invoke(Request request, Response response)</td>
<td></td>
<td>17:30:48 200</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>Framework::Servlet</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>doPost(HttpServletRequest request, HttpServletResponse response)</td>
<td></td>
<td>17:30:48 200</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>DemoController</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
<tr>
<td>backendweb()</td>
<td></td>
<td>17:30:48 200</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>Framework::Servlet</td>
<td>BACKEND-WEB</td>
<td>BackenWAS1</td>
<td></td>
</tr>
</tbody>
</table>
Distributed Call Tree

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

Method: HttpClient.execute()
Distributed Call Tree

HttpClient.execute()

Tomcat.receive()
Distributed Call Tree

<table>
<thead>
<tr>
<th>Method</th>
<th>Argument</th>
<th>Start Time</th>
<th>Gap(ms)</th>
<th>Exec(ms)</th>
<th>Exec(%)</th>
<th>Self(ms)</th>
<th>Class</th>
<th>API</th>
<th>Application</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke(Request request, Response response)</td>
<td>/emeredoc.pinpoint</td>
<td>17:30:48 004</td>
<td>0</td>
<td>295</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doGet(HTTPRequest request, HttpServletResponse response)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpUriRequest request, HttpsURLConnection connection)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute(HttpRequest request, HttpServletResponse response)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servletProcess</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servletProcess</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BACKEND-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createPinpointWorker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createPinpointWorker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BACKEND-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BACKEND-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRONT-WEB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How it all works

Call Stack Trace

Distributed Transaction Trace
invoke(); -> doGet(); -> demo2();
invoke(); -> doGet(); -> demo2();

invoke() {
    doGet() {
        demo2() {
            doGet();
            demo2();
        }
    }
}

demo2() {
invoke(); -> doGet(); -> demo2();
invoke(); -> doGet(); -> demo2();

invoke() {
    doGet() {
        demo2() {
            doGet();
        }
    }
}

demo2() {
}

doGet() {
    demo2() {
        doGet();
    }
}
## Call Stack Trace

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke()</td>
<td>0</td>
</tr>
<tr>
<td>doGet()</td>
<td>1</td>
</tr>
<tr>
<td>demo2()</td>
<td>2</td>
</tr>
</tbody>
</table>
## Call Stack Trace

<table>
<thead>
<tr>
<th>Method</th>
<th>Sequence</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke()</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doGet()</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>demo2()</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Call Stack Trace

<table>
<thead>
<tr>
<th>Method</th>
<th>Sequence</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke()</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doGet()</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>demo2()</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Sequence</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke()</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doGet()</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>demo2()</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Call Stack Trace

foo() {
    fooInterceptor.before();
    ...
    fooInterceptor.after();
}
invoke() {
    invokeInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        }
        doGetInterceptor.after();
    }
    invokeInterceptor.after();
}
invoke() {
    invokeInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        }
        doGetInterceptor.after();
    }
    invokeInterceptor.after();
}
involve() {
    involveInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        }
        doGetInterceptor.after();
    }
    involveInterceptor.after();
}

before()
Increment sequence
Push on to the call stack

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Event</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>invoke()</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>doGet()</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
invoke() {
    invokeInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        }
        doGetInterceptor.after();
    }
    invokeInterceptor.after();
}
invoke() {
    invokeInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        }
        doGetInterceptor.after();
    }
    invokeInterceptor.after();
}
invoke() {
    invokeInterceptor.before();
    doGet() {
        doGetInterceptor.before();
        demo2() {
            demo2Interceptor.before();
            demo2Interceptor.after();
        } 
        doGetInterceptor.after();
    } 
    invokeInterceptor.after();
}
Call Stack Trace

invoke() {
  invokeInterceptor.before();
  doGet() {
    doGetInterceptor.before();
    demo2() {
      demo2Interceptor.before();
      demo2Interceptor.after();
    }
    doGetInterceptor.after();
  }
  invokeInterceptor.after();
}

after()
Pop off the call stack
Buffer to write queue

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Event</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>demo2()</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>doGet()</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>invoke()</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>invoke()</td>
<td>0</td>
</tr>
</tbody>
</table>

Write Queue
- demo2()
- doGet()
- invoke()
## Call Stack Trace

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoke()</td>
<td>0</td>
</tr>
<tr>
<td>doGet()</td>
<td>1</td>
</tr>
<tr>
<td>demo2()</td>
<td>2</td>
</tr>
</tbody>
</table>
Find out the relationship between nodes connected by RPCs for a given transaction

Trace tag added to each requests
- HTTP : HttpHeaders
Distributed Transaction Trace

TraceId
- TransactionId
- SpanId
- Parent SpanId

Node 1
TxId = Node1^Time^1
SpanId = 1
Parent SpanId = -1

RPC 1

Node 2
TxId = Node1^Time^1
SpanId = 2
Parent SpanId = 1

RPC 2
Node 3
TxId = Node1^Time^1
SpanId = 3
Parent SpanId = 2

RPC 3
Node 4
TxId = Node1^Time^1
SpanId = 4
Parent SpanId = 2
Distributed Transaction Trace

**TransactionId (TxId)**
Globally unique ID for a single transaction

- **Node 1**: $TxId = Node1^\text{Time}^1$
  - SpanId = 1
  - Parent SpanId = -1

- **Node 2**: $TxId = Node1^\text{Time}^1$
  - SpanId = 2
  - Parent SpanId = 1

- **Node 3**: $TxId = Node1^\text{Time}^1$
  - SpanId = 3
  - Parent SpanId = 2

- **Node 4**: $TxId = Node1^\text{Time}^1$
  - SpanId = 4
  - Parent SpanId = 2
SpanId, Parent SpanId
Id used to encode parent-child relationship between nodes
Distributed Call Tree

<table>
<thead>
<tr>
<th>Method</th>
<th>Server Map</th>
<th>Timeline</th>
<th>Mixed View</th>
<th>neko</th>
<th>Self &gt;=</th>
<th>1000(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlet Process</td>
<td>application</td>
<td>Transaction Id: FrontWAS2*:1563931395270*189...</td>
<td>AgentId: FrontWAS2</td>
<td>ApplicationName: FRONT-WEB</td>
<td>neko</td>
<td>Self &gt;=</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Servlet Process
  - `/emeroad.pinpoint`
  - `http.status.code`
  - `REMOTE_ADDRESS`
  - `127.0.0.1`
  - `invoke(Request request, Response response)`
  - `doGet(HttpServletRequest request, HttpServletResponse response)`
  - `demo2()`
  - `execute(HttpUriRequest request, Response response)`
  - `open(HttpRoute route, HttpContext dev-pinpoint.workload)`
  - `execute(HttpRequest request, HttpUriRequest uriRequest)`
  - `http.status.code`
  - `http.io`
  - `write: 0ms, read: 12ms`
  - Servlet Process
    - `/backendweb.pinpoint`
    - `http.status.code`
    - `REMOTE_ADDRESS`
    - `invoke(Request request, Response response)`
    - `doPost(HttpServletRequest request, HttpServletResponse response)`
    - `backendweb()`

- PINPOINT: FRONT-WEB

- NAVER
Troubleshooting with Pinpoint

Let’s troubleshoot our system using Pinpoint
(There’s a link @ www.github.com/naver/pinpoint)
Troubleshooting with Pinpoint
Future plans
Future Plans

Node.js

Golang

PHP
Future Plans
Thank you

Q & A

Homepage: https://naver.github.io/pinpoint
Github: https://github.com/naver/pinpoint
Twitter: https://twitter.com/Pinpoint_APM
E-mail: roy.kim@navercorp.com, hyungil.jeong@navercorp.com