devfest

Spring Webflux
vs
Spring MVC with VT

Google Developer Groups

Incheon/Songdo

이상준

(현) 카카오 엔터테인먼트
(전) 나이스평가정보
(전) 우리에프아이에스
(전) 롯데정보통신
(전) 플레이스5

Google Developer Groups

T.O.C

1. Motivation
2. 기존 자바 Thread 문제점
3. 리액티브 프로그래밍 & Spring Webflux
4. What is Virtual Thread & 구조
5. Performance Comparison
6. Conclusion

devfest

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
s.star,
r: Colors.yellow[500],
Text('23'),
```

Google Developer Groups

Incheon/Songdo

Motivation

Spring 기반 신규 프로젝트

Spring Webflux 쓸 때가 되었나**?**

Kotlin Coroutine으로 할까**?**

오**!!!! Virtual Thread** 정식 출시**?!** 그럼 **Webflux** 안써도 되는거 아냐**?**

하던데로 **Spring MVC + Virtual Thread** 로 할까**?**

일단 성능 비교 ㄱㄱ**!**

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

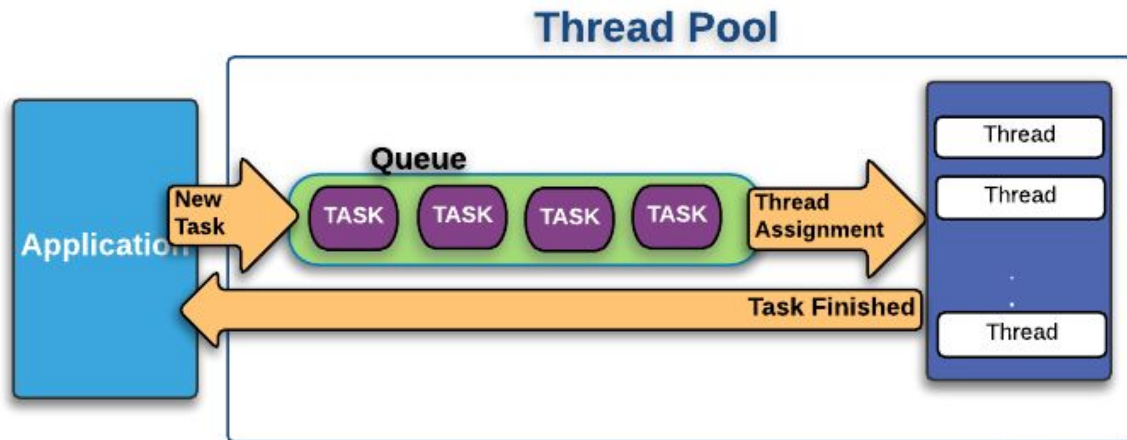
devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```

기존 자바 Thread 문제점

기존 자바의 Thread

기존 자바의 쓰레드는 실제 **OS**의 **Thread**와 **1:1** 구조
OS Thread는 갯수가 제한적이고 생성하고 유지하는 비용이 비쌈
생성과 유지비가 비싸니 **pool**을 만들어서 재활용 **(ex. db connection pool)**
Tomcat default max thread size : 200



Thread Per Request 모델

- Apache Httpd, Apache Tomcat
- 요청 1개당 1개의 Thread 사용
- Throughput을 늘리려면 Thread가 많아야
- OS Thread 무작정 늘릴 수 없음.

```
mbio@gdg-devfest-1:~$ uname -a
Linux gdg-devfest-1 6.2.0-1019-gcp #21~22.04.1-Ubuntu SMP Thu Nov 16 18:18:34 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
mbio@gdg-devfest-1:~$ grep -c processor /proc/cpuinfo
4
mbio@gdg-devfest-1:~$ cat /proc/meminfo | grep MemTotal
MemTotal:        16365492 kB
mbio@gdg-devfest-1:~$ cat /proc/sys/kernel/threads-max
127790
mbio@gdg-devfest-1:~$
```

GCP n2d-standard-4 (4코어, 16GB, 10G) 장비의 thread-max 사이즈

C10K Problem

그 유명한 **C10K Problem (1999년 Dan Kegel)**
Nonblocking I/O, Asynchronous I/O
요청을 처리하는 과정을 뜯어보면 **Blocking I/O**에서 **wait**에 대부분 시간을 소요
Ex. 먹는시간 **>** 대기시간

해결 방법

1. 장비 업그레이드

- 너무 비싸고, 가성비가 안나옴
- **OS Thread**는 동시에 **core** 수 만큼만 처리가능함
- **OS Thread**를 마구 늘리면 오히려 **context switching** 비용만 비싸짐

2. Event Driven

- 요청**(=**이벤트**)** 과 **I/O**처리 분리
- **I/O Multiplexing** 기반 싱글스레드 이벤트 루프
- **Blocking I/O**를 기다리는 시간 없이 요청 처리

Event Driven - I/O Multiplexing

커널레벨 프로그램

- select(), poll(), kqueue(), epoll()

크로스 플랫폼 라이브러리

- libuv (bsd계열 kqueue, linux계열 epoll, SunOS poll ...)

Python - FastAPI(uvloop)

Javascript - Node.js, nginx

JAVA - NIO, Netty

Redis

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

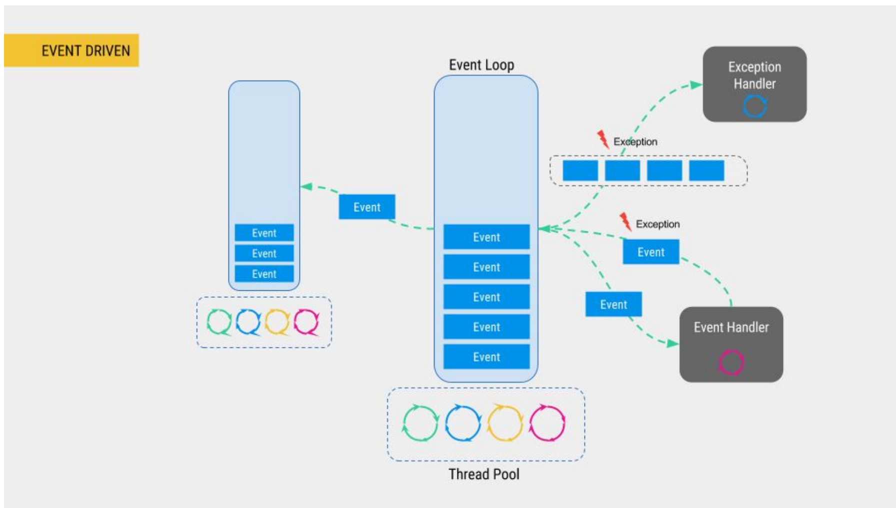
devfest

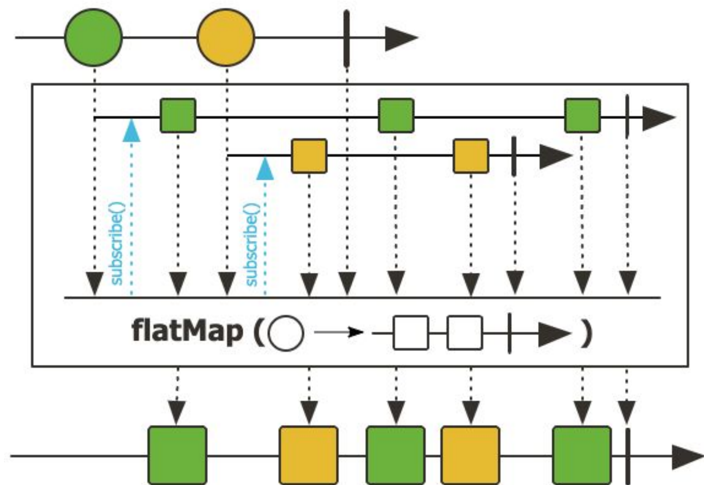
```
s.star,
r: Colors.yellow[500],

Text('23'),
```

리액티브 프로그래밍 & Spring Webflux

Spring Webflux

Event-Driven, Event loop
Spring 기반 웹 프레임워크도 C10K Problem 이슈 해결!
Reactive Programming 패러다임을 따르는 Reactive Streams의 등장
Reactive Streams : Event Driven 을 구현하기 위한 API





리액티브 프로그래밍

In computing, **reactive programming** is a declarative programming paradigm concerned with data streams and the propagation of change. With this paradigm, it's possible to

Programming paradigms

- Action

-> 위키피디아 **:** 데이터 스트림과 변경사항의 전파에 중점을 둔 선언적 프로그래밍 패러다임

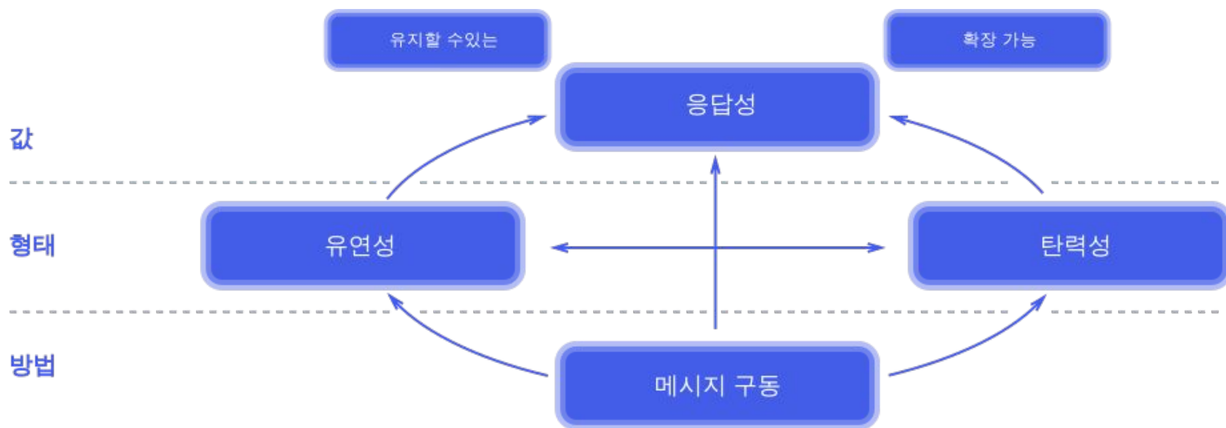
자바진영의 리액티브 프로그래밍 구현체 **RxJava, Reactor**
Reactor기반의 스프링 웹프로젝트가 **Spring Webflux**
그외 **Akka, Vert.x, Quassar ...**

Reactive Programming

데이터 스트림과 변경사항의 전파에 중점을 둔 선언적 프로그래밍 패러다임

명령형 -> 선언형 / 순차형 -> 동시성

리액티브 선언문



Reactive Programming

반응형 프로그래밍의 대표 **-** 스프레드 시트

- 선언적으로 함수를 선언

- A1, B1의 값이 변경되면 변경사항이 전파됨

- C1의 값이 자동으로 변경 **->** 전파 **-> D1** 값 변경

C1 fx =SUM(A1+B1)

	A	B	C	D
1	2	3	5	8
2				

D1 fx =SUM(C1+3)

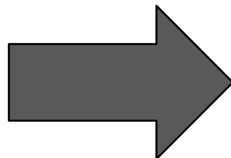
	A	B	C	D
1	2	3	5	8
2				

선언형 프로그래밍

원하는 결과를 프로그래밍

언제 어떤 방식으로 결과가 나올지는 추상화

```
kubectl run nginx --image=nginx
kubectl create deployment nginx --image=nginx
kubectl expose deployment nginx --port=80
kubectl edit deployment nginx
kubectl set image deployment nginx nginx=nginx:1.25
kubectl delete -f nginx.yaml
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
```

변경사항 전파 (이벤트, 메세징)

Pull -> Push

동기적인 요청과 응답대기에서 벗어남

데이터의 변경과 관련된 **이벤트 발행 -> 이벤트 구독**

앞에서본 **I/O Multiplexing**

Flutter Provider, React conext api

아키 레이어로 가면 **MQ**기반 이벤트 드라이븐

Flutter Provider / React Context API

```dart
class Person with ChangeNotifier {
  Person({this.name, this.age});
  final String name;
  int age;
  void increaseAge() {
    this.age++;
    notifyListeners();
  }
}

void main() {
  runApp(ChangeNotifierProvider(create: (_) => Person(name: "mbio", age: 32),child: MyApp(),),);
}

class MyHomePage extends StatelessWidget {
  const MyHomePage({Key key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Provider Class'),
      ),
      body: Center(
        child: Text(
          '''
          name :  ${Provider.of<Person>(context).name;}!
          age : ${Provider.of<Person>(context).age}''',
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => Provider.of<Person>(context, listen: false).increaseAge(),
      ),
    );
  }
}
```

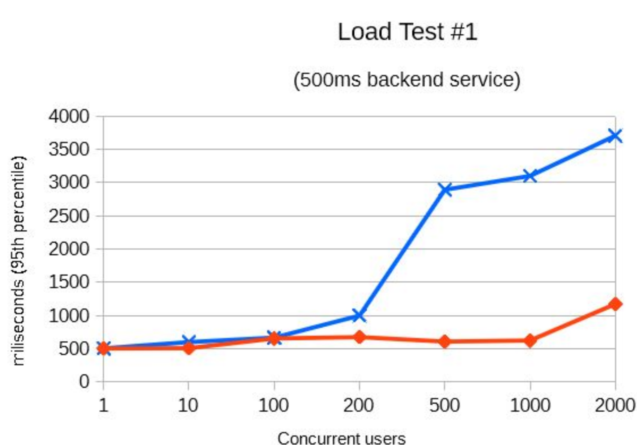
```jsx
export const PersonContext = createContext({
  name: '',
  age: 0,
  increaseAge: () => {},
});

const PersonProvider = ({ children }) => {
  const [name, setName] = useState("mbio");
  const [age, setAge] = useState(32);
  const increaseAge = () => setAge(age => age + 1);
  return (<PersonContext.Provider value={{ name, age, increaseAge }}>{children}</PersonContext.Provider>);
};

const MyHomePage = () => {
  const { name, age, increaseAge } = useContext(PersonContext);

  return (
    <div>
      <h1>Provider Class</h1>
      Name: <b>{name}</b>
      Age: <b>{age}</b>
      <button onClick={increaseAge}>Increase Age</button>
    </div>
  );
};

function App() {
  return (
    <PersonProvider>
      <MyHomePage />
    </PersonProvider>
  );
}
```

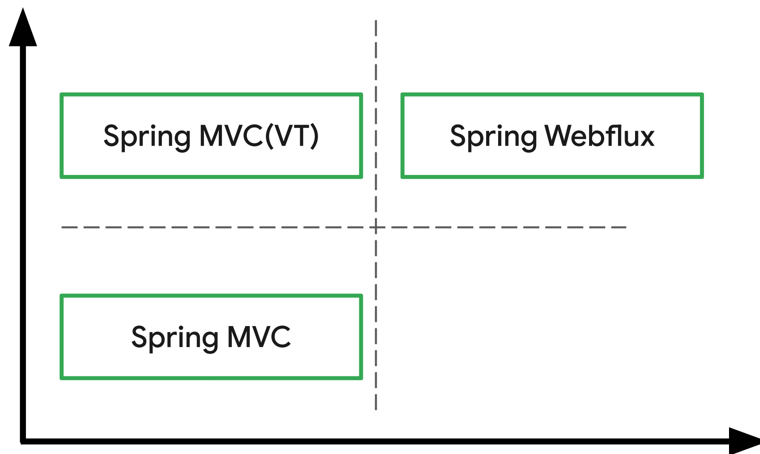
What is Webflux Problem

- C10K 해결, 리액티브 프로그래밍 스타일도 맘에 드는데 왜?
- 코드 어려움
- Thread를 중심으로 하는 Java의 디자인과 어울리지 않음
- Exception Stack Trace, Debugger 등이 전부 스레드 기반
- 1개의 요청을 처리하는 1개의 스레드를 추적하는 개념으로 트레이싱 했었는데 스레드가 계속 바뀌니 추적이 안됨

There is no silver bullet

- **reactive streams**가 등장하고 **Webflux** 같은 프레임워크가 생긴건
  사실 자바가 비동기 논블럭킹 지원을 잘 안해줘서 **(kotlin coroutine**도 마찬가지**)**
- **Blocking** 시점에 기다리지 않게 스케줄링을 하는게 쟁점인데 개발자가 직접 해야함
- **Framework** 같은걸 왜 쓰는데**?** 비즈니스 로직에 집중





```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```

Google Developer Groups

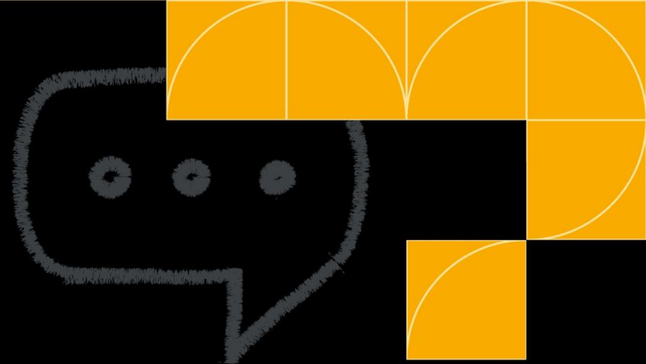
Incheon/Songdo

What is Virtual Thread?

```
Text(
  'Simple Statement or URL',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

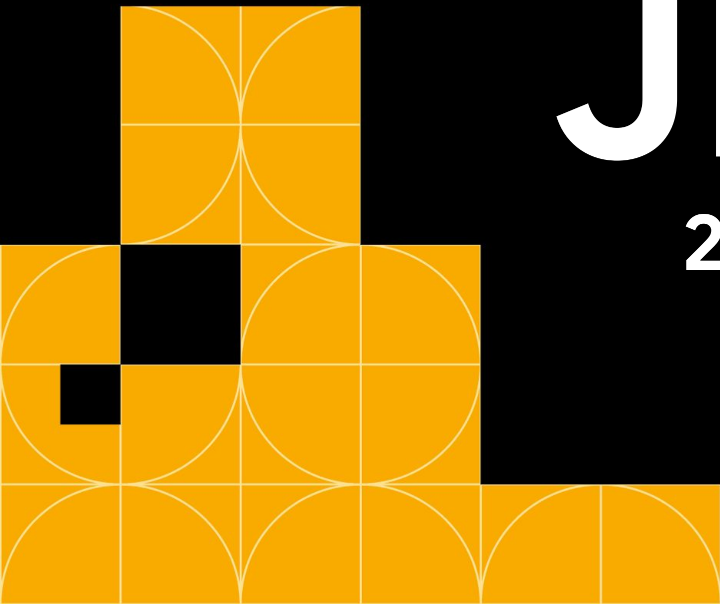
devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```



JDK 21

2023.09.19 LTS



Virtual Thread가 뭐에요?

JDK 21에 정식 출시된 경량**(lightweight)** 쓰레드

Project Loom의 결과물 **(thread**보다 작은 단위의 **task** 지원**)**

JVM이 동작하는 **OS Thread**를 사용하지 않음

Virtual Machine에서 자체적으로 스케줄링하여

수백만개의 쓰레드를 사용할 수 있게 해줌

User Thread vs Kernel Thread

User Thread

- Lightweight thread, Green thread, Virtual Thread, Fiber Thread

Kernel Thread

- OS Thread, Platform Thread, Carrier Thread

가상 쓰레드는 새로운 개념이 아님

Golang Goroutine = “A goroutine is a lightweight thread managed by the Go runtime. “

Haskell GHC, Erlang, Elixir ... High Throughput으로 유명한 언어들은 이전부터 제공

자바의 Thread

Green Thread (싱글코어 시절에 설계)

- JVM 1.1 ~ 1.3
- Java Threads: Native Thread = M:1

Platform Thread

- JVM 1.3~
- Java Thread: Native Thread = 1:1

Virtual Thread

- JVM 21~
- Java Threads: Native Threads = M:N (M > N)

Spring Boot 3.2.0 - 2023.11.23 Release

JDK 21 Virtual Thread 공식 지원

사용방법

- spring.threads.virtual.enabled=true

Thread.currentThread().toString()

- Spring MVC
  Thread[#43,http-nio-8080-exec-1,5,main]

- Spring MVC- Virtual Thread
  VirtualThread[#58,tomcat-handler-0]/runnable@ForkJoinPool-1-worker-1

- Spring Webflux
  Thread[#68,reactor-http-nio-3,5,main]

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

devfest

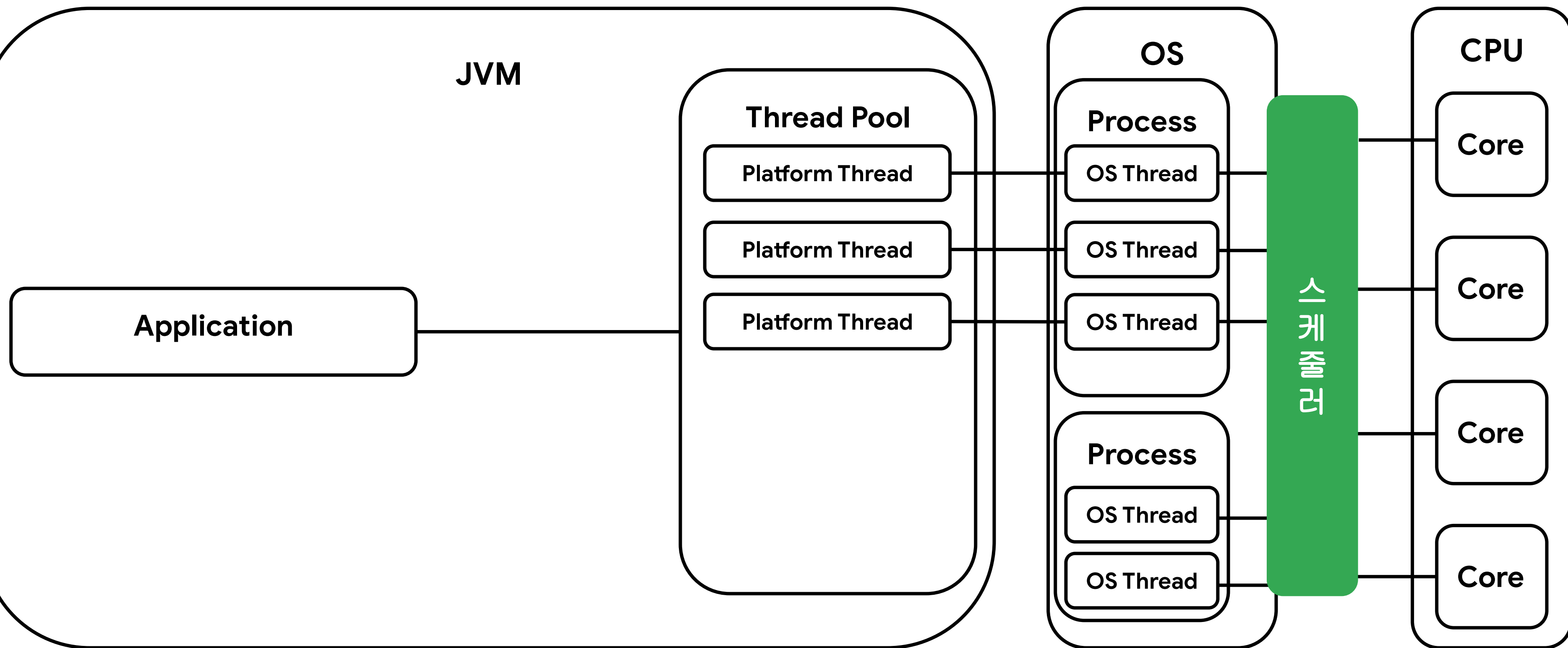
```
s.star,
r: Colors.yellow[500],

Text('23'),
```

Google Developer Groups

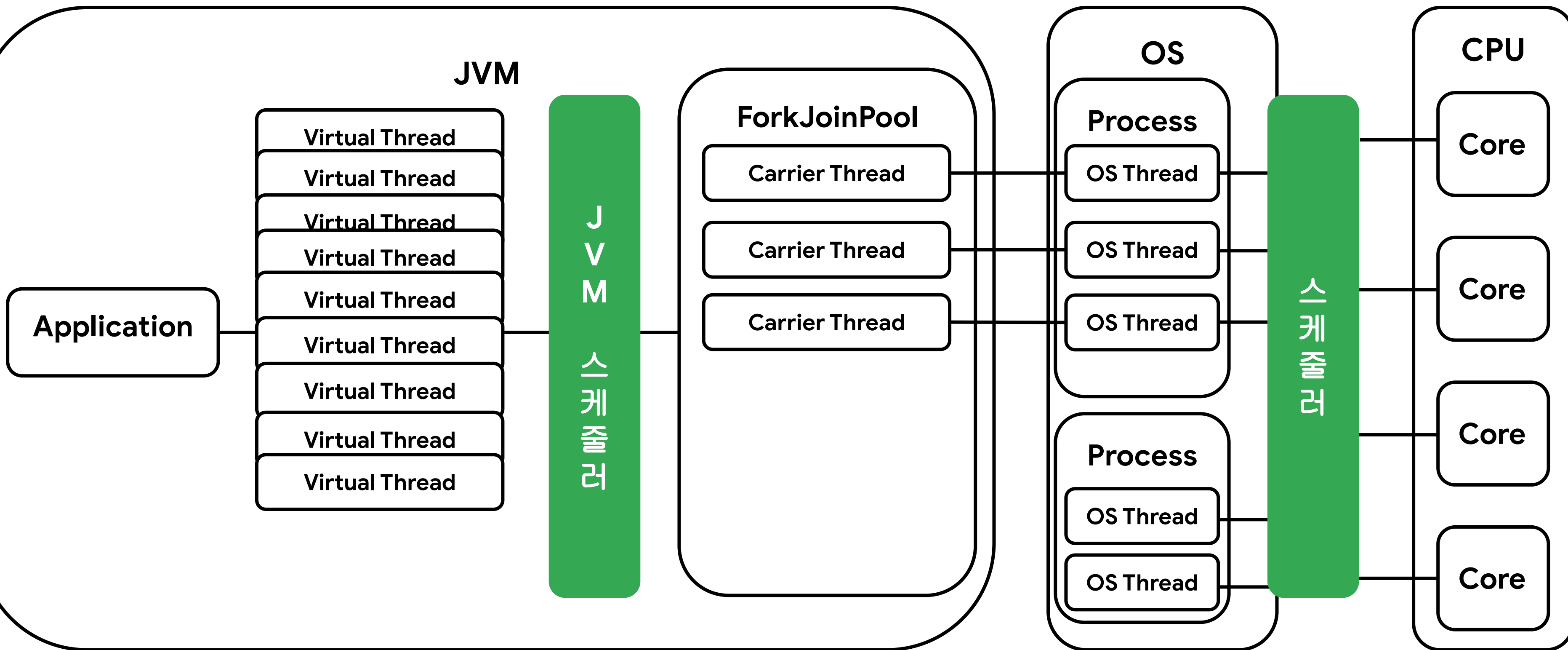
Incheon/Songdo

Virtual Thread 구조

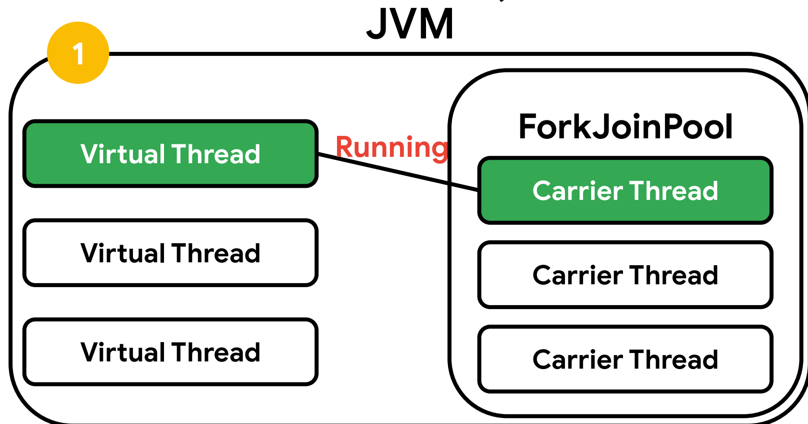
Virtual Thread (Fiber Thread)

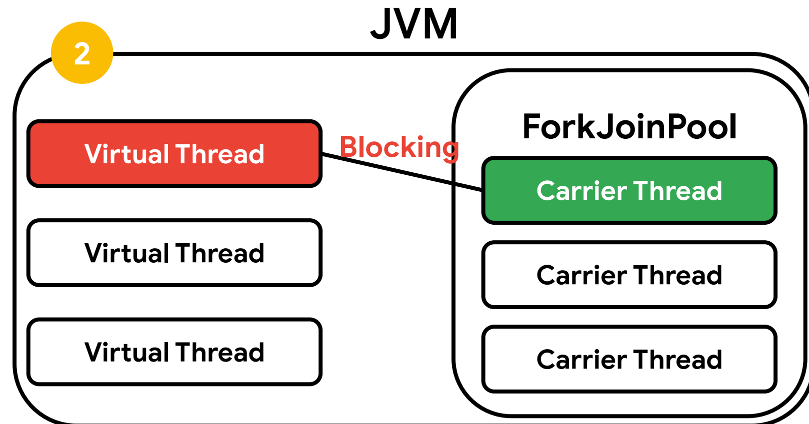


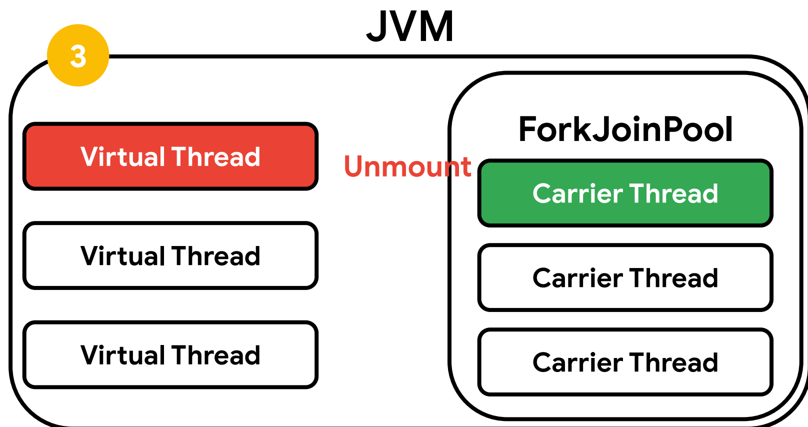
Virtual Thread (Fiber Thread)

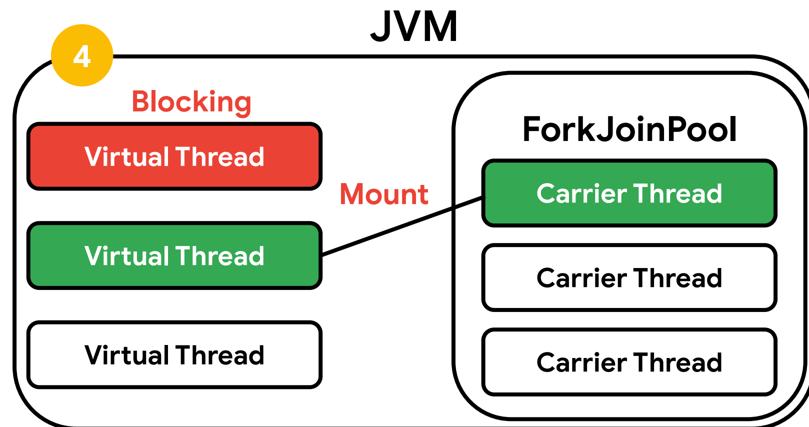


Virtual Thread (Fiber Thread)









Carrier Thread

기존 **OS Thread**와 **1:1**로 매핑되던 **Platform Thread**

Virtual Thread에서는 **OS**와 매칭되는 쓰레드의 이름이 **Carrier Thread**로 변경

ForkJoinPool 안에서 동작하는 **Worker Thread**가 **Carrier Thread**

- **CPU** 코어 갯수 만큼 생성
- **Available Processor** 만큼
- **Like, Webflux**

Platform Thread Count Monitor

```
Tasks: 43, 132 thr; 1 running
Load average: 5.64 2.34 0.93
Uptime: 7 days, 01:26:05

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
598301 mbio 20 0 7752M 421M 24904 S 2.0 2.6 0:16.22 java -Xms4096m -Xmx4096m -jar jdk21-servlet-1.0-SNAPSHOT.jar --spring.profiles
```

```
Every 0.5s: ps -o thcount 598301          gdg-devfest-1: Sat Dec  9 02:54:28 2023

THCNT
   39
```

```
mbio@gdg-devfest-test:~$ k6 run --vus 2000 --duration 5s k6-string.js
```

Virtual Thread Count Monitor

```
Tasks: 43, 126 thr; 1 running
Load average: 0.30 0.26 0.14
Uptime: 7 days, 01:22:01

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
588282 mbio      20   0 17216  7796  5504 S  0.0  0.0  0:00.03  └─ sshd: mbio@pts/1
588283 mbio      20   0 10136  6144  3584 S  0.0  0.0  0:00.05     └─ -bash
595335 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  1:18.21        ├─ java -Xms2048m -Xmx4096m -jar jdk21-servlet-1.0-SNAPSHOT.jar
595336 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:03.45        │  ├─ java
595337 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.08        │  ├─ GC Thread#0
595338 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ G1 Main Marker
595339 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.05        │  ├─ G1 Conc#0
595340 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.08        │  ├─ G1 Refine#0
595341 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ G1 Service
595342 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.08        │  ├─ VM Periodic Tas
595343 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.01        │  ├─ VM Thread
595344 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Reference Handl
595345 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Finalizer
595346 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Signal Dispatch
595347 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Service Thread
595348 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.01        │  ├─ Monitor Deflati
595349 mbio      20   0 7746M 1556M 25160 S  0.7  9.7  0:11.96        │  ├─ C2 CompilerThre
595350 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:03.03        │  ├─ C1 CompilerThre
595351 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.02        │  ├─ Notification Th
595352 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Common-Cleaner
595356 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ Cleaner-0
595365 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.07        │  ├─ GC Thread#1
595366 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.06        │  ├─ GC Thread#2
595367 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.06        │  ├─ GC Thread#3
595375 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.02        │  ├─ Catalina-utilit
595376 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.02        │  ├─ Catalina-utilit
595377 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ container-0
595378 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.02        │  ├─ mysql-cj-abando
595385 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.00        │  ├─ cp-mbio houseke
595390 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:06.26        │  ├─ http-nio-8080-P
595391 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:00.21        │  ├─ http-nio-8080-A
595405 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:08.12        │  ├─ ForkJoinPool-1-
595406 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:08.23        │  ├─ ForkJoinPool-1-
595407 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:08.08        │  ├─ ForkJoinPool-1-
595408 mbio      20   0 7746M 1556M 25160 S  0.0  9.7  0:08.25        │  └─ ForkJoinPool-1-
595483 mbio      20   0  6220  2048  1920 S  0.0  0.0  0:00.00        └─ tail -f nohup.out
593305 root      20   0 16916 10752  8704 S  0.0  0.1  0:00.00  ├─ sshd: mbio [priv]
593360 mbio      20   0 17212  7924  5504 S  0.0  0.0  0:00.04  │  └─ sshd: mbio@pts/0
593361 mbio      20   0  9856  5760  3456 S  0.0  0.0  0:00.02  │     └─ -bash
595582 mbio      20   0  7148  3200  2432 S  0.0  0.0  0:00.31  │        └─ watch -n 0.5 ps -o thcount 595335
593985 root      20   0 16912 10624  8576 S  0.0  0.1  0:00.01  ├─ sshd: mbio [priv]
594055 mbio      20   0 17208  7920  5504 S  0.0  0.0  0:00.14  │  └─ sshd: mbio@pts/2
594056 mbio      20   0  9856  5888  3456 S  0.0  0.0  0:00.02  │     └─ -bash
594764 mbio      20   0  9400  4992  3328 R  0.7  0.0  0:01.57  │        └─ htop
   856 root      20   0 16296  8448  6784 S  0.0  0.1  0:01.63  ├─ /lib/systemd/systemd-logind
   864 root      20   0  7284  2816  2560 S  0.0  0.0  0:00.71  ├─ /usr/sbin/cron -f -P
  1346 _chrony   20   0 18916  3356  2816 S  0.0  0.0  0:00.15  ├─ /usr/sbin/chronyd -F 1
  1347 _chrony   20   0 10588  1856  1408 S  0.0  0.0  0:00.00  │  └─ /usr/sbin/chronyd -F 1
  2293 root      20   0  289M 20096 17408 S  0.0  0.1  0:03.21  ├─ /usr/libexec/packagekitd
  2294 root      20   0  289M 20096 17408 S  0.0  0.1  0:00.00  │  ├─ gmain
  2295 root      20   0  289M 20096 17408 S  0.0  0.1  0:00.41  │  └─ gdbus
  9548 root      20   0 1421M 59628 45184 S  0.0  0.4  0:45.87  ├─ /opt/google-cloud-ops-agent/libexec/google_cloud_ops_agent_diagnostics -config /etc/google
  9554 root      20   0 1421M 59628 45184 S  0.0  0.4  0:09.12  │  ├─ google_cloud_op
  9555 root      20   0 1421M 59628 45184 S  0.0  0.4  0:00.00  │  ├─ google_cloud_op
  9556 root      20   0 1421M 59628 45184 S  0.0  0.4  0:06.98  │  ├─ google_cloud_op
  9557 root      20   0 1421M 59628 45184 S  0.0  0.4  0:00.00  │  ├─ google_cloud_op
  9558 root      20   0 1421M 59628 45184 S  0.0  0.4  0:00.00  │  ├─ google_cloud_op
  9564 root      20   0 1421M 59628 45184 S  0.0  0.4  0:07.50  │  ├─ google_cloud_op
  9565 root      20   0 1421M 59628 45184 S  0.0  0.4  0:00.00  │  ├─ google_cloud_op
  9568 root      20   0 1421M 59628 45184 S  0.0  0.4  0:07.14  │  ├─ google_cloud_op
  9569 root      20   0 1421M 59628 45184 S  0.0  0.4  0:08.13  │  └─ google_cloud_op
  9588 root      20   0 1418M 50176 38272 S  0.0  0.3  0:25.24  ├─ /opt/google-cloud-ops-agent/libexec/google_cloud_ops_agent_wrapper -config_path /etc/googl
  9591 root      20   0 1418M 50176 38272 S  0.0  0.3  0:02.35  │  ├─ google_cloud_op
  9592 root      20   0 1418M 50176 38272 S  0.0  0.3  0:04.08  │  ├─ google_cloud_op
  9593 root      20   0 1418M 50176 38272 S  0.0  0.3  0:05.81  │  ├─ google_cloud_op
  9594 root      20   0 1418M 50176 38272 S  0.0  0.3  0:05.72  │  ├─ google_cloud_op
  9595 root      20   0 1418M 50176 38272 S  0.0  0.3  0:00.00  │  ├─ google_cloud_op
  9596 root      20   0 1418M 50176 38272 S  0.0  0.3  0:00.00  │  ├─ google_cloud_op
  9604 root      20   0 1418M 50176 38272 S  0.0  0.3  0:00.00  │  └─ google_cloud_op
  9605 root      20   0 1416M 24636 14888 S  0.0  0.2  5:09.55  │  /opt/google-cloud-ops-agent/subagents/fluent-bit/bin/fluent-bit --config /run/google-cl
  9606 root      20   0 1416M 24636 14888 S  0.0  0.2  2:20.41  │  ├─ flb-pipeline
F1Help F2Setup F3Search F4Filter F5List F6SortBy F7Nice - F8Nice + F9Kill F10Quit
[0] 0:Python*
```

```
Every 0.5s: ps -o thcount 595335                    gdg-devfest-1: Sat Dec  9 02:50:23 2023

THCNT
   33
```

```
mbio@gdg-devfest-test:~$ k6 run --vus 2000 --duration 5s k6-string.js
```

```
"mbioui-MacBookAir.loc" 11:50 09-Dec-23
```

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```

Google Developer Groups

Incheon/Songdo

Performance Test

성능 테스트 환경과 툴

Grafana K6

- VUser 3000, Duration 30s, ResponseTime (P90)

Java 실행 환경

- Google Cloud, n2d-standard-4 (4코어, 16GB, 7Gbps)
- MaxHeap 4GB
- Tomcat default max thread size : 200

Database 별도 서버

- Google Cloud SQL - Mysql 8.0.35
- Enterprise-개발설정
- Connection pool : 150

요약

리전	asia-northeast3(서울)
DB 버전	MySQL 8.0.35
vCPU	4 vCPU
메모리	16GB
데이터 캐시	사용 중지됨
스토리지	100GB
연결	공개 IP
백업	자동
가용성	단일 영역
point-in-time recovery	사용 설정됨
네트워크 처리량(MB/초)	1,000/1,000
디스크 처리량(MB/초)	읽기: 48.0/240.0 쓰기: 48.0/240.0
IOPS	읽기: 3,000/15,000 쓰기: 3,000/15,000

Cloud SQL 인스턴스

성능 테스트 Application & API

성능 테스트 **Application**

- **JDK8 + Spring Boot 2.7.13 (Jetbrain** 조사결과 아직 **JDK8 1**등**)**
- **JDK21 + Spring Boot 3.2.0 + Virtual Thread**
- **JDK21 + Spring Boot 3.2.0 + Platform Thread**
- **JDK21 + Spring Boot 3.2.0 + webflux**

성능 테스트 **API**

- 단순 문자열 반환
- **1**초 **blocking API**
- **1**초 **blocking +** 고속 **select**
- **1**초 응답 쿼리 **(slow query)**

성능 테스트 API

Spring MVC

```java
@RestController
@RequiredArgsConstructor
@Slf4j
public class HelloController {
    private final JdbcTemplate jdbcTemplate;
    new *
    @GetMapping("/thread-name")
    public String getThreadName() {
        return Thread.currentThread().toString();
    }
    new *
    @GetMapping("/block")
    public String block() throws InterruptedException {
        Thread.sleep( millis: 1000);
        return Thread.currentThread().toString();
    }
    new *
    @GetMapping("/block/query")
    public String blockingAndQuery() throws InterruptedException {
        Thread.sleep( millis: 1000);
        jdbcTemplate.queryForList( sql: "select now()");
        return Thread.currentThread().toString();
    }
    new *
    @GetMapping("/slow-query")
    public String slowQuery() {
        jdbcTemplate.queryForList( sql: "select sleep(1);");
        return Thread.currentThread().toString();
    }
}
```

Spring Webflux

```java
@RestController
@RequiredArgsConstructor
@Slf4j
public class HelloController {
    private final DatabaseClient databaseClient;
    new *
    @GetMapping("/thread-name")
    public Mono<String> getThreadName() {
        return Mono.just(Thread.currentThread().toString());
    }
    new *
    @GetMapping("/block")
    public Mono<String> block() {
        return Mono.delay(Duration.ofSeconds(1)).then(Mono.just(Thread.currentThread().toString()));
    }
    new *
    @GetMapping("/block/query")
    public Mono<String> blockingAndQuery() {
        return Mono.delay(Duration.ofSeconds(1))
                .then(databaseClient.sql("select now();")
                        .map(row -> Thread.currentThread().toString()).first());
    }
    new *
    @GetMapping("/slow-query")
    public Mono<String> query() {
        return databaseClient.sql("select sleep(1);")
                .map(row -> Thread.currentThread().toString()).first();
    }
}
```

단순 문자열 반환

MVC-VT 회차	TPS	Response Time
1	10198.4	527 ms
2	11211.9	510 ms
3	10836.8	522 ms

MVC-PT 회차	TPS	Response Time
1	10906.2	517.17 ms
2	10940.9	512.64 ms
3	11067.2	509.18 ms

Webflux 회차	TPS	Response Time
1	11416.2	500 ms
2	11334.3	512.04 ms
3	11291.4	513.28 ms

JDK8 회차	TPS	Response Time
1	11220.2	500.46 ms
2	11040.9	515.19 ms
3	11046.7	520 ms

1초 blocking

MVC-VT 회차	TPS	Response Time
1	2855.4	1018 ms
2	2858.5	1003 ms
3	2853.1	1012 ms

MVC-PT 회차	TPS	Response Time
1	197.0	15007 ms
2	197.2	15010 ms
3	197.2	15132 ms

Webflux 회차	TPS	Response Time
1	2863.2	1003 ms
2	2870.8	1002 ms
3	2853.4	1005 ms

JDK8 회차	TPS	Response Time
1	198.0	15046 ms
2	197.9	15213 ms
3	197.8	15123 ms

1초 blocking + 고속 select

MVC-VT 회차	TPS	Response Time
1	1987.5	1873 ms
2	2013.5	1701 ms
3	1948.1	1772 ms

MVC-PT 회차	TPS	Response Time
1	196.2	15153 ms
2	195.9	15013 ms
3	195.2	15211 ms

Webflux 회차	TPS	Response Time
1	2851.2	1014 ms
2	2847	1013 ms
3	2887.5	1005 ms

JDK8 회차	TPS	Response Time
1	196.8	15144 ms
2	197.6	15034 ms
3	198.68	15135 ms

Slow Query (응답 1초)

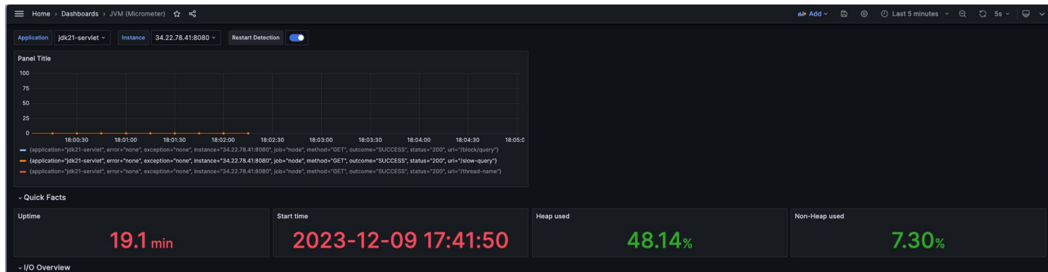
MVC-VT 회차	TPS	Response Time
1	4	262초
2	-	-
3	-	-

MVC-PT 회차	TPS	Response Time
1	148.2	20.06 s
2	149.1	20.05 s
3	149.0	20.04 s

Webflux 회차	TPS	Response Time
1	147.9	20.03 s
2	148.7	20.06 s
3	148.5	20.03 s

JDK8 회차	TPS	Response Time
1	148.2	20.76 s
2	148.8	21.03 s
3	148.4	20.25 s

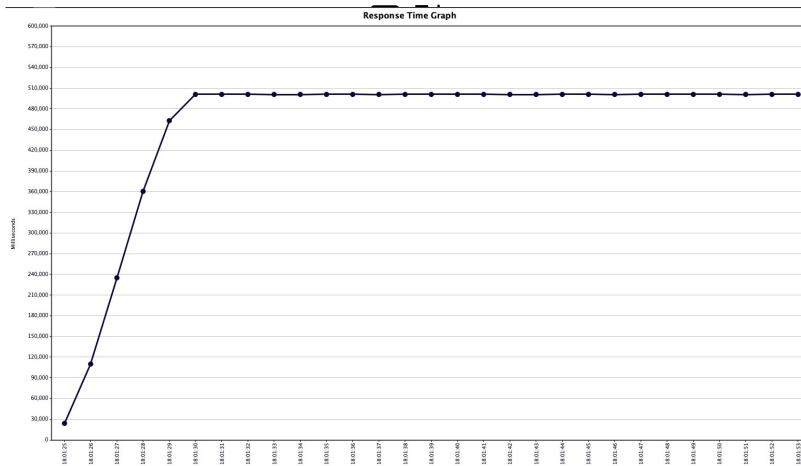
VT-Slow Query (응답 1초)




```
:: Spring Boot ::                (v3.2.0)

2023-12-09T08:41:51.542Z  INFO 750388 --- [jdk21-servlet] [           main] com.mbio.Jdk21ServletApplication         : Starting Jdk21ServletApplication v1.0-SNAPSHOT using Ja
2023-12-09T08:41:51.545Z  INFO 750388 --- [jdk21-servlet] [           main] com.mbio.Jdk21ServletApplication         : The following 1 profile is active: "default"
2023-12-09T08:41:52.567Z  INFO 750388 --- [jdk21-servlet] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DEFAULT
2023-12-09T08:41:52.581Z  INFO 750388 --- [jdk21-servlet] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 10 ms. Foun
2023-12-09T08:41:53.050Z  INFO 750388 --- [jdk21-servlet] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2023-12-09T08:41:53.063Z  INFO 750388 --- [jdk21-servlet] [           main] o.apache.catalina.core.StandardService    : Starting service [Tomcat]
2023-12-09T08:41:53.063Z  INFO 750388 --- [jdk21-servlet] [           main] o.apache.catalina.core.StandardEngine     : Starting Servlet engine: [Apache Tomcat/10.1.16]
2023-12-09T08:41:53.096Z  INFO 750388 --- [jdk21-servlet] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]        : Initializing Spring embedded WebApplicationContext
2023-12-09T08:41:53.098Z  INFO 750388 --- [jdk21-servlet] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2023-12-09T08:41:53.828Z  INFO 750388 --- [jdk21-servlet] [           main] com.zaxxer.hikari.HikariDataSource        : cp-mbio - Starting...
2023-12-09T08:41:54.265Z  INFO 750388 --- [jdk21-servlet] [           main] com.zaxxer.hikari.pool.HikariPool         : cp-mbio - Added connection com.mysql.cj.jdbc.Connection
2023-12-09T08:41:54.268Z  INFO 750388 --- [jdk21-servlet] [           main] com.zaxxer.hikari.HikariDataSource        : cp-mbio - Start completed.
2023-12-09T08:41:54.399Z  INFO 750388 --- [jdk21-servlet] [           main] o.s.b.a.e.web.EndpointLinksResolver       : Exposing 4 endpoint(s) beneath base path '/actuator'
2023-12-09T08:41:54.496Z  INFO 750388 --- [jdk21-servlet] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '
2023-12-09T08:41:54.513Z  INFO 750388 --- [jdk21-servlet] [           main] com.mbio.Jdk21ServletApplication         : Started Jdk21ServletApplication in 3.534 seconds (proc
2023-12-09T08:41:56.875Z  INFO 750388 --- [jdk21-servlet] [omcat-handler-0] o.a.c.c.C.[Tomcat].[localhost].[/]        : Initializing Spring DispatcherServlet 'dispatcherServle
2023-12-09T08:41:56.876Z  INFO 750388 --- [jdk21-servlet] [omcat-handler-0] o.s.web.servlet.DispatcherServlet         : Initializing Servlet 'dispatcherServlet'
2023-12-09T08:41:56.877Z  INFO 750388 --- [jdk21-servlet] [omcat-handler-0] o.s.web.servlet.DispatcherServlet         : Completed initialization in 1 ms
```

Micrometer 로 데이터 전달

에러로그 없음



Jdbc connection 경합

Synchronized - Carrier Thread Blocking

Synchronized 사용 시 Virtual Thread에 연결된 Carrier Thread가 Blocking.

이런 경우를 **pinned**

JEP 444: Virtual Threads

There are two scenarios in which a virtual thread cannot be unmounted during blocking operations because it is *pinned* to its carrier:

1. When it executes code inside a synchronized block or method, or
2. When it executes a native method or a foreign function.

Synchronized - Carrier Thread Blocking

라이브러리들에서 **virtual thread** 대응을 위해 작업중인것으로 보임

Merged — Replace synchronized with ReentrantLock #2

Changes from 1 commit · File filter · Conversations

Filter changed files

- src/main
  - core-api/java/com/mysql/cj
    - Query.java
    - Session.java
  - core-impl/java/com/mysql/cj
    - AbstractQuery.java
    - CancelQueryTaskImpl.java
    - CoreSession.java
    - NativeSession.java
    - ServerPreparedQuery.java

17 src/main/core-impl/java/com/mysql/cj/AbstractQuery.java

```
129 133                 throw cause;
130 134             }
    135 +       } finally {
    136 +           this.cancelTimeoutMutex.unlock();
131 137         }
132 138     }
133 139
134 140     public void resetCancelledState() {
135     -       synchronized (this.cancelTimeoutMutex) {
    141 +       this.cancelTimeoutMutex.lock();
    142 +       try {
136 143             this.cancelStatus = CancelStatus.NOT_CANCELED;
    144 +       } finally {
    145 +           this.cancelTimeoutMutex.unlock();
137 146         }
138 147     }
```

Mysql - mysql-connector-j

Merged — refactor:(loom) replace the usages of synchronized with ReentrantLock #2635

Changes from all commits · File filter · Conversations

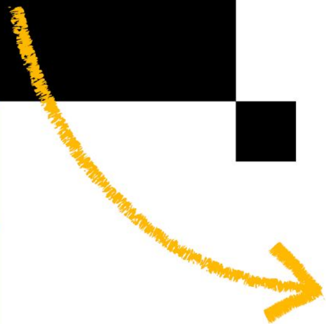
Filter changed files

- config/forbidden-apis
  - forbidden-apis.txt
- pgjdbc/src
  - main/java/org/postgresql
    - Driver.java
    - core
      - QueryExecutorBase.java
      - v3
        - CopyOperationImpl.java
        - QueryExecutorImpl.java
    - ds
      - PGPoolingDataSource.java
    - hostchooser

75 pgjdbc/src/main/java/org/postgresql/Driver.java

```
82  84      private @Nullable Properties defaultProperties;
83  85
84      -   private synchronized Properties getDefaultProperties() throws IOException {
85      -     if (defaultProperties != null) {
86      -       return defaultProperties;
87      -     }
    86  +   private final ResourceLock lock = new ResourceLock();
88  87
89      -     // Make sure we load properties with the maximum possible privileges.
90      -     try {
91      -       defaultProperties =
92      -           doPrivileged(new PrivilegedExceptionAction<Properties>() {
93      -             public Properties run() throws IOException {
94      -               return loadDefaultProperties();
95      -             }
96      -           });
97      -     } catch (PrivilegedActionException e) {
98      -       Exception ex = e.getException();
99      -       if (ex instanceof IOException) {
100     -         throw (IOException) ex;
```

Postgres - pgjdbcd



```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```

Google Developer Groups

Incheon/Songdo

Conclusion

오해 하면 안되는 것

Continuation 구현
Structured Concurrency 전혀 다른 얘기

Spring MVC의 Thread Per Request 모델 동일
1개 요청을 **1**개의 task(virtual thread)로 처리하면서 발생하는 **blocking**을 기다리는건 동일

300ms API를 **3**번 호출하면**? ->** 당연히 **900ms**
API를 동시호출해서 처리하고 싶으면**?** 하던데로 **CompletableFuture**

기존의 **Platform Thread** 방식이 대체되는것 **X**

결론 - 개인적인 견해

DB 커넥션 관련 이슈만 해결되면 **Spring MVC + Virtual Thread** 사용

Virtual Thread와 **Webflux**의 처리량 차이로 인해 이슈가 생길정도라면**?**

Webflux 부터 도입**X ->** 스케일아웃**,** 샤딩**,** 캐싱**, CQRS,** 업무 프로세스 재설계 등**..**

그래도 부족하면 **Rust? Elixir?**

```
Text(
  'Simple Statement or URL',
  style: TextStyle(
    color: Colors.yellow[200],
  ),
),
```

devfest

```
s.star,
r: Colors.yellow[500],

Text('23'),
```

Q & A