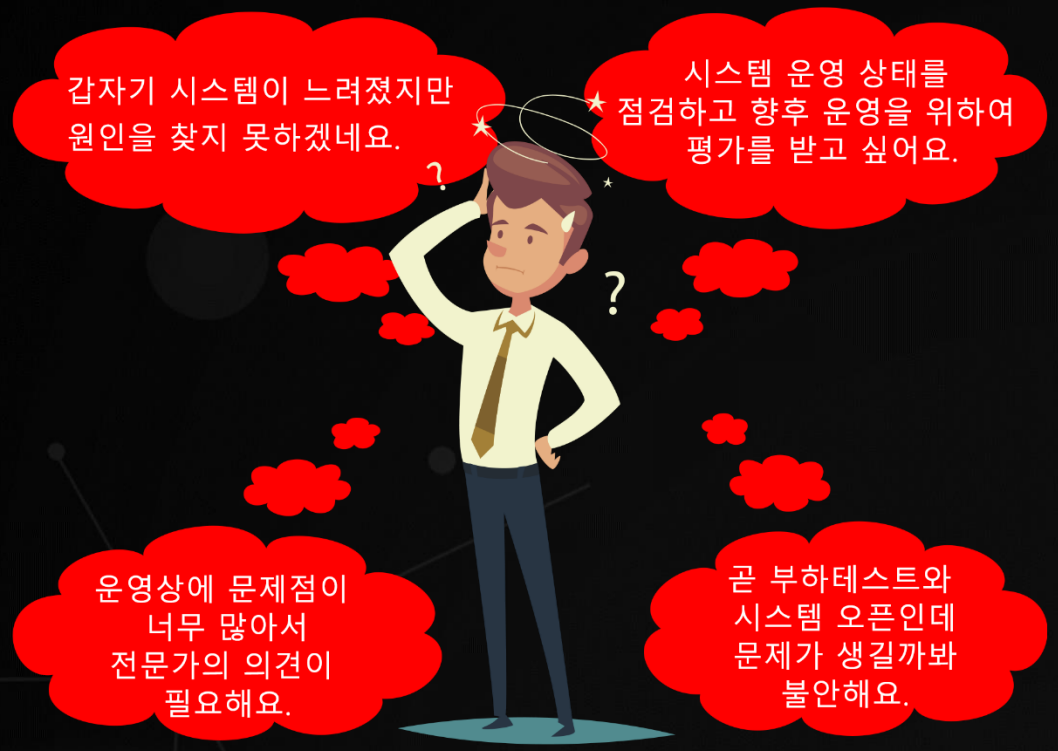
openmaru
APM
OPENMARU Quick Service

시스템은 점점 늘어나고 복잡해 지는 데....



- 시스템에 대한 모니터링은 잘 되고 있지만 애플리케이션은 블랙박스화 되어 내부 상태를 파악하지 못함
- 장애나 성능 이슈의 90% 이상이 애플리케이션이 원인

웹시스템 진단 서비스

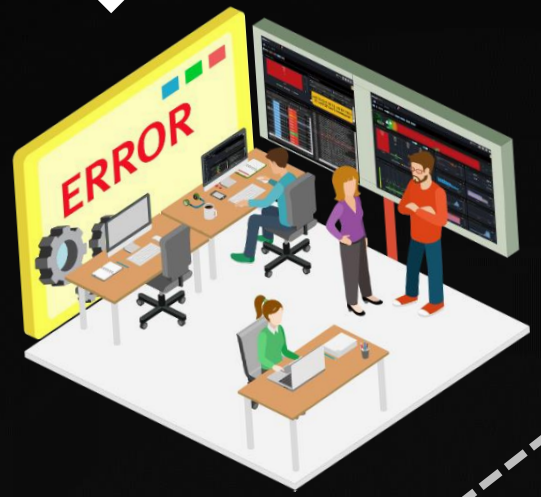
"바로 지금 신청하세요."

Opennaru Service Desk
고객께서 Quick Service를 통해
전달해 주신
화면과 Thread Dump 및
메타 정보를 접수합니다.
자동으로 케이스를 생성합니다.

Opennaru Quick Service 란?

고객사

① 시스템 이슈발생

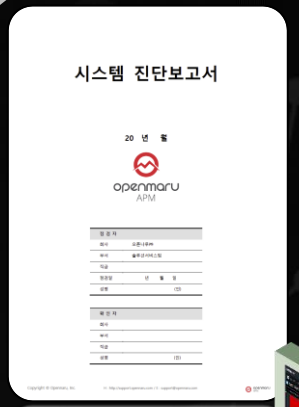


② 쓰레드 덤프와 모니터링 화면 전달

③ 오픈나루 서비스 데스크 접수

④ 장애 분석

⑤ 장애 분석 보고서 전달

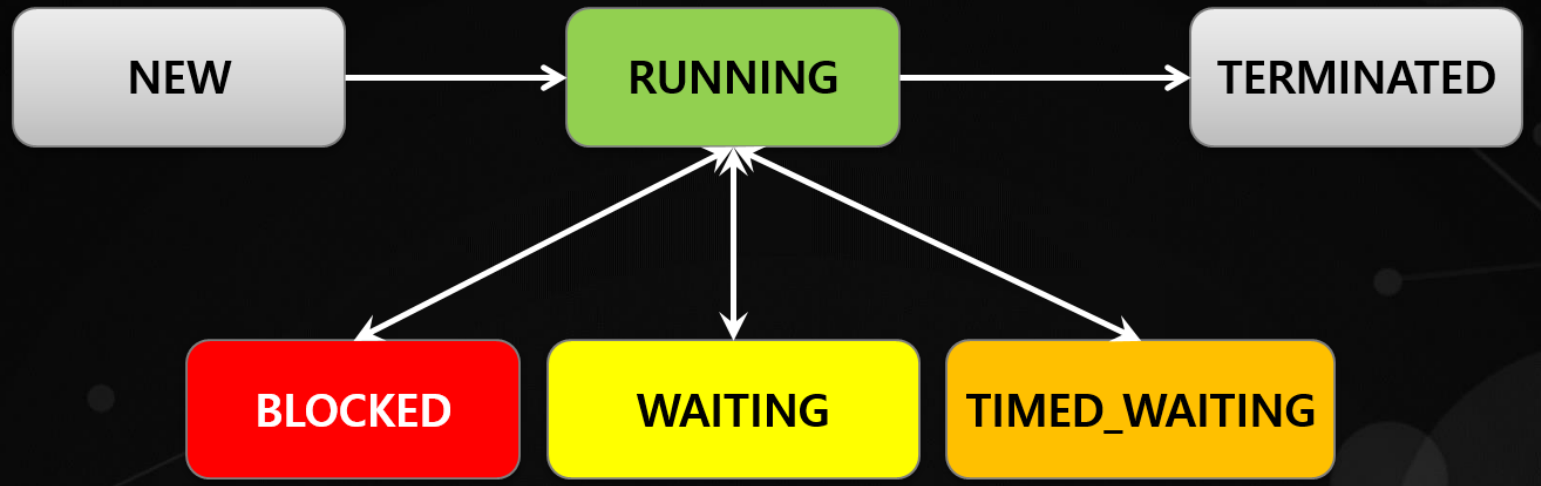


오픈나루

OPENMARU © 2021 | All Rights Reserved.

Thread 상태

- java.lang.Thread.State

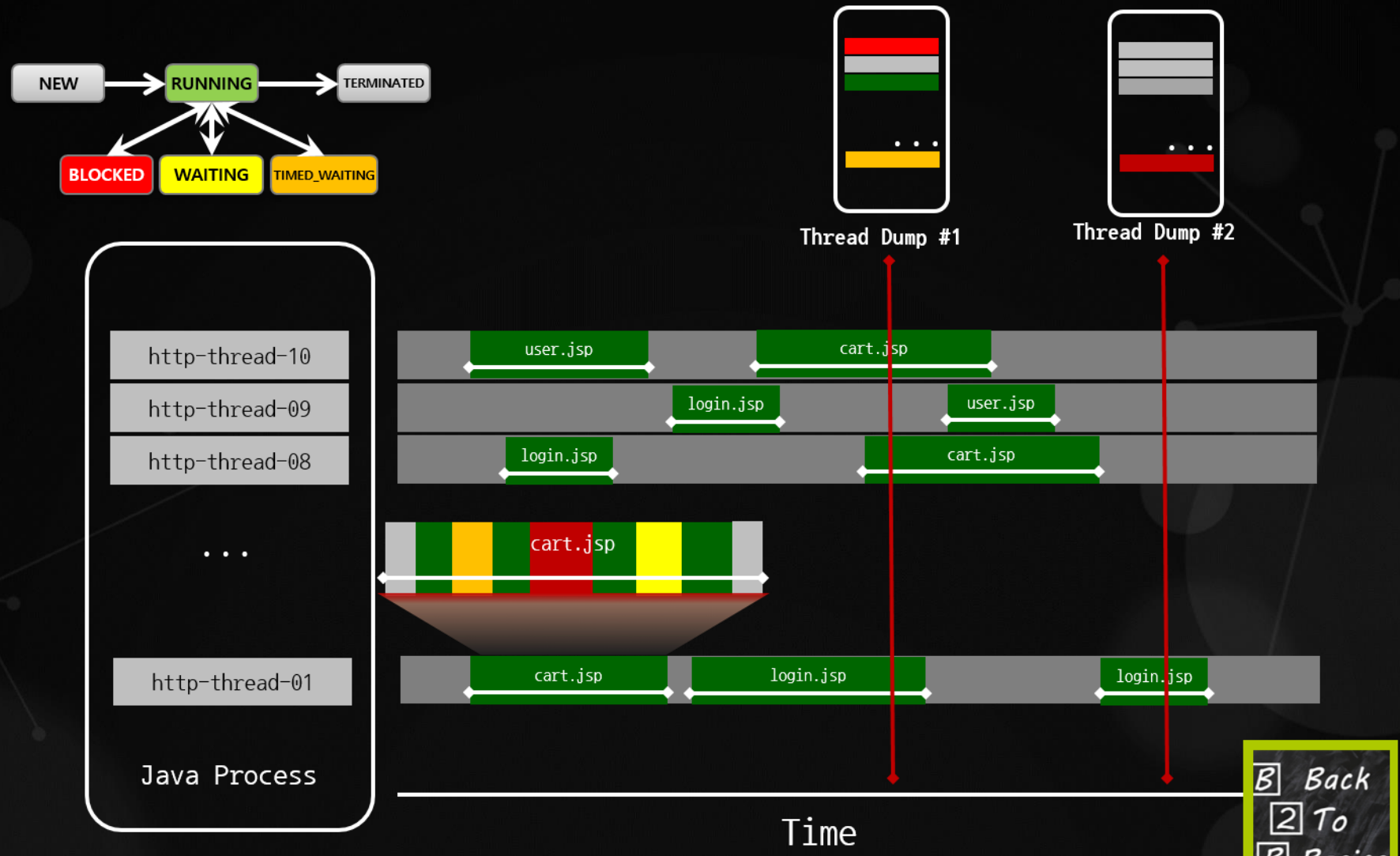


- NEW – 생성되어있지만 실행되지 않은 상태
- RUNNING – 작업 수행 중
- BLOCKED – 다른 스레드의 Lock 해제를 기다리는 중
- WAITING – wait(), join(), park() 메소드로 대기 중
- TIMED_WAITING – WAITING과 같지만, 정해진 시간만 대기
- TERMINATED – 종료됨

개념과 배경지식

OPENMARU © 2021 | All Rights Reserved.

Java Thread Dump의 이해



Back 2 To Basics

개념과 배경지식

OPENMARU © 2021 | All Rights Reserved.

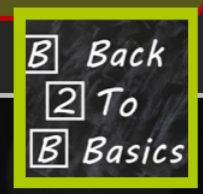
스레드 덤프의 정보 구성

- 스레드 덤프의 각 스레드의 정보 구성

스레드 이름 우선순위 스레드ID OS 스레드ID 스레드 상태

```
"http-thread-pool-threads - 250" prio=10 tid=0x00007fab9c192800 nid=0xdbc9 waiting on condition [0x00007faa6ce2c000]
   java.lang.Thread.State: WAITING (parking)
            at sun.misc.Unsafe.park(Native Method)
            - parking to wait for  <0x000000078b428170> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
            at java.util.concurrent.locks.LockSupport.park(LockSupport.java:186)
            at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2043)
            at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
            at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1068)
            at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1130)
            at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
            at java.lang.Thread.run(Thread.java:744)
            at org.jboss.threads.JBossThread.run(JBossThread.java:122)
```

스레드 호출 Stack



개념과 배경지식

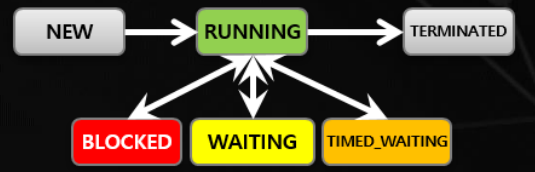
OPENMARU © 2021 | All Rights Reserved.

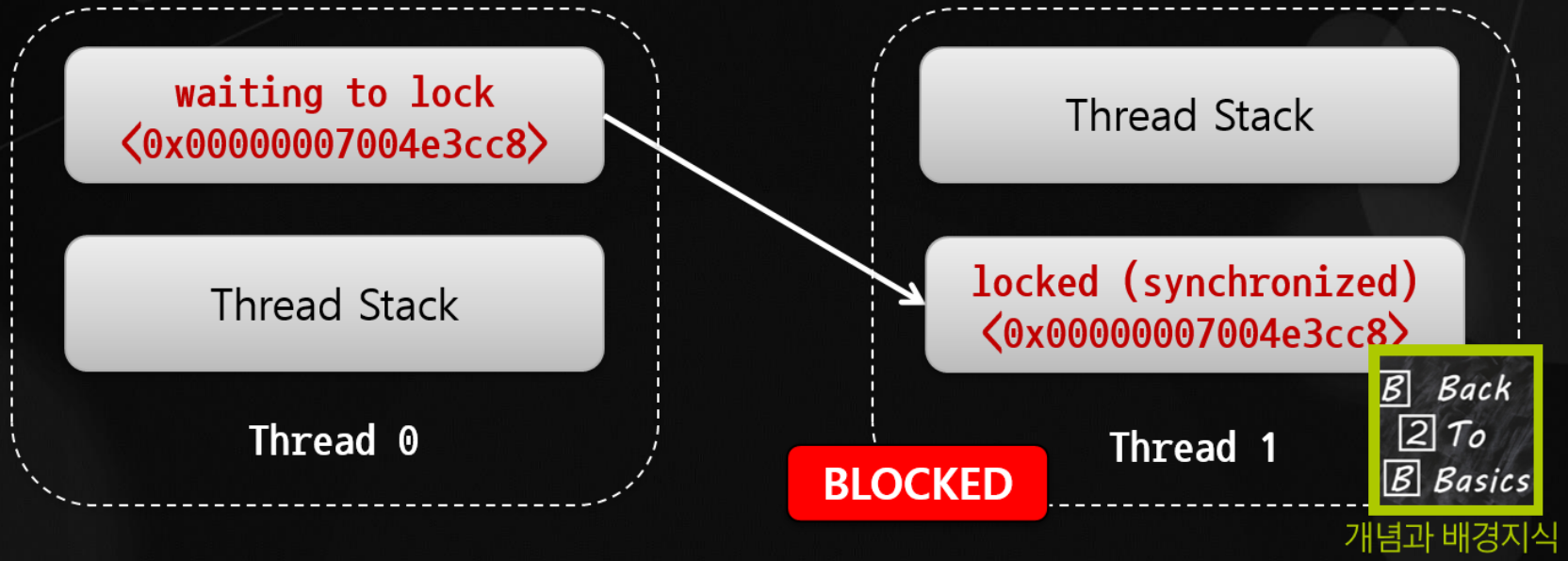
Thread Dump의 Blocked의 의미

- 여러 스레드에서 같은 변수나 메소드를 변경할 경우
- synchronized로 다른 스레드의 처리가 끝날 때 까지 제어가 넘어가지 않도록 보호

```
// 특정한 객체에 lock을 걸고자 할 때
synchronized(객체의 참조변수){
    // ...
}

// 메서드에 lock을 걸고자할 때
public synchronized void methodName(){
    // ...
}
```

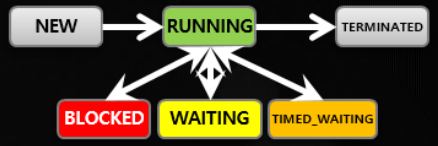


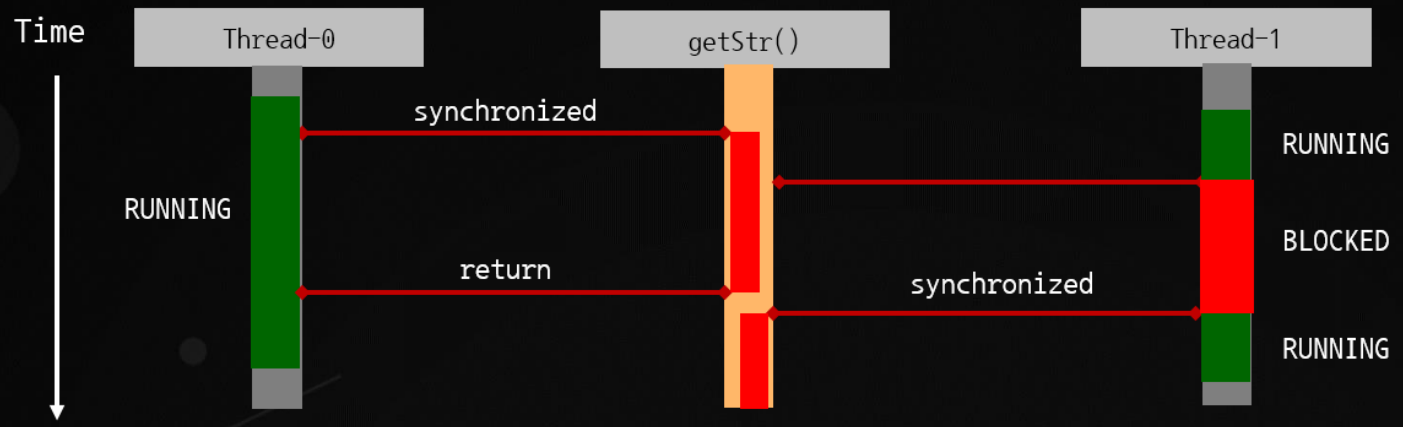


OPENMARU © 2021 | All Rights Reserved.

Java Thread Dump의 BLOCKED의 의미

- **waiting for monitor entry**
  - 오브젝트에 대한 락의 해제를 기다리고 있는 상태



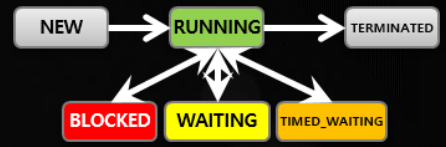


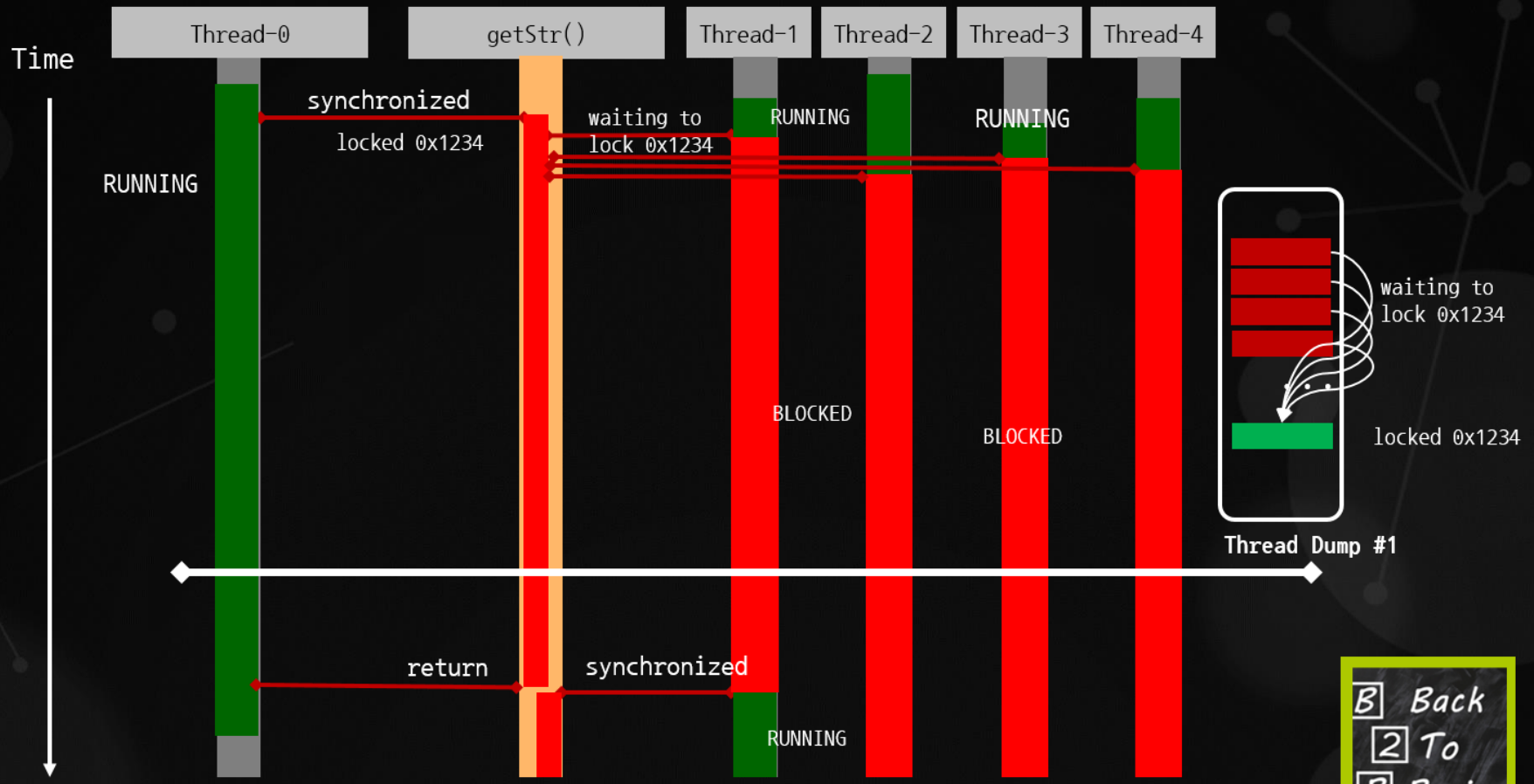

```
"Thread-1" prio=10 tid=0x00007f69fc0b8800 nid=0x7358 waiting for monitor entry [0x00007f6a0050a000]
   java.lang.Thread.State: BLOCKED (on object monitor)
          at sample.SampleMain.getStr(SampleMain.java:23)
          - waiting to lock <0x000000077acacf60> (a java.lang.Class for sample.SampleMain)
          at sample.SampleMain.run(SampleMain.java:17)
          at java.lang.Thread.run(Thread.java:679)
"Thread-0" prio=10 tid=0x00007f69fc0b6800 nid=0x7357 waiting on condition [0x00007f6a0060b000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
          at java.lang.Thread.sleep(Native Method)
          at sample.SampleMain.getStr(SampleMain.java:23)
          - locked <0x000000077acacf60> (a java.lang.Class for sample.SampleMain)
          at sample.SampleMain.run(SampleMain.java:17)
          at java.lang.Thread.run(Thread.java:679)
```

Back 2 To Basics
개념과 배경지식

OPENMARU © 2021 | All Rights Reserved.

Thread BLOCKED 상태가 길어지면 발생하는 상황





Back 2 To Basics
개념과 배경지식

- Confidential -
OPENMARU © 2021 | All Rights Reserved.

Application Performance Management

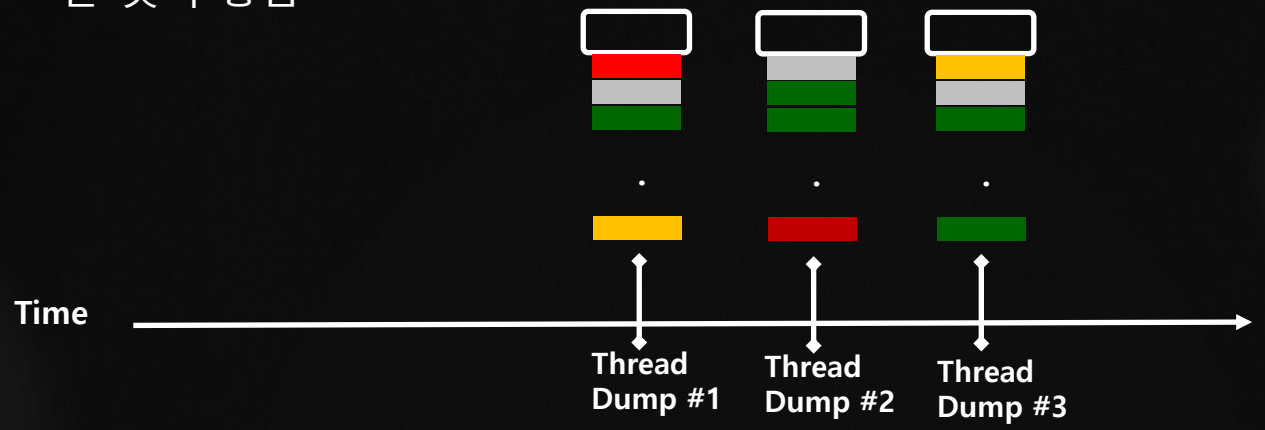
Thread Dump 분석 방법

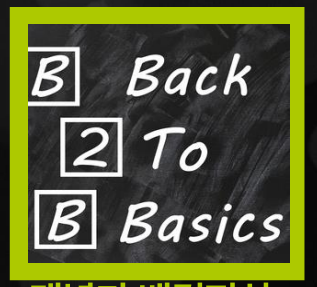
openmaru APM

스레드 덤프

- **스레드 덤프란?**
  - JVM 내부에서 현재 수행중인 스레드의 SnapShot을 텍스트로 출력
  - 서버 Hang, 서버 Slowdown등 서버에서 발생하는 문제를 해결하는 가장 유용한 도구
  - 약 3~5초간격으로 3~5회 연속해서 스레드 덤프를 받아야 분석이 가능하다

- **언제 받아야 하나?**
  - 운영 중인 서버에서도 수행가능(많은 시간이 걸리지 않는 작업임)
  - 서버 Hang, Slowdown시, 서버가 Crash되거나 OutOfMemory가 발생한 직후에 받는 것이 좋음





개념과 배경지식

OPENMARU © 2021 | All Rights Reserved.

스레드 덤프를 분석

openmaru

Take at least 3 times, with 3 to 5 seconds interval

진단 및 모니터링

Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
Thread 9

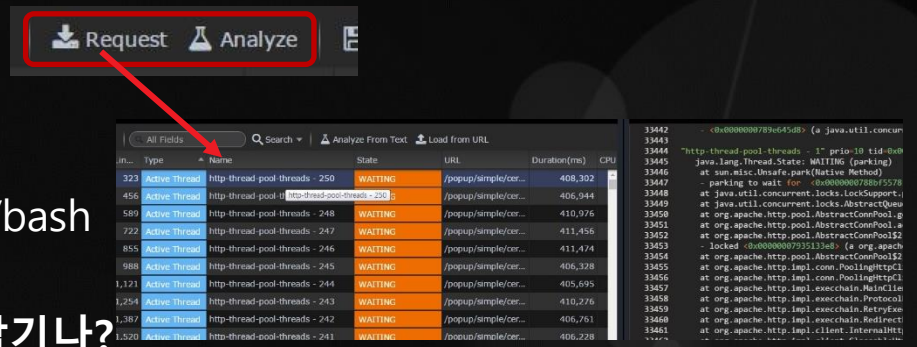
OPENMARU © 2021 | All Rights Reserved.

스레드 덤프 받는 방법은?

- **Linux / UNIX 시스템에서 스레드 덤프**
  - kill -3 <*PID*>
  - 일반적을 nohup 명령을 이용하여 시작하기 때문에 nohup.out파일에 stdout 로그가 남게 됨
  - 윈도우에서는 Ctrl + Break
  - **KHAN [apm] 화면에서 버튼 클릭(장애시 자동 생성)**

진단 및 모니터링

- **Docker와 Openshift 같은 환경**
  - **스레드 덤프을 어떻게 받나?**
    - docker exec -it c4343003c1 /bin/bash
      ~# kill -3 <PID>
  - **내용을 어떻게 복사? nohup 로그는 남기나?**
    - docker cp c4343003c1:/file/path/nohup.out /host/path/target
  - KHAN [apm]은 Openshift, Docker를 지원 → 버튼 클릭 & 분석 UI 제공

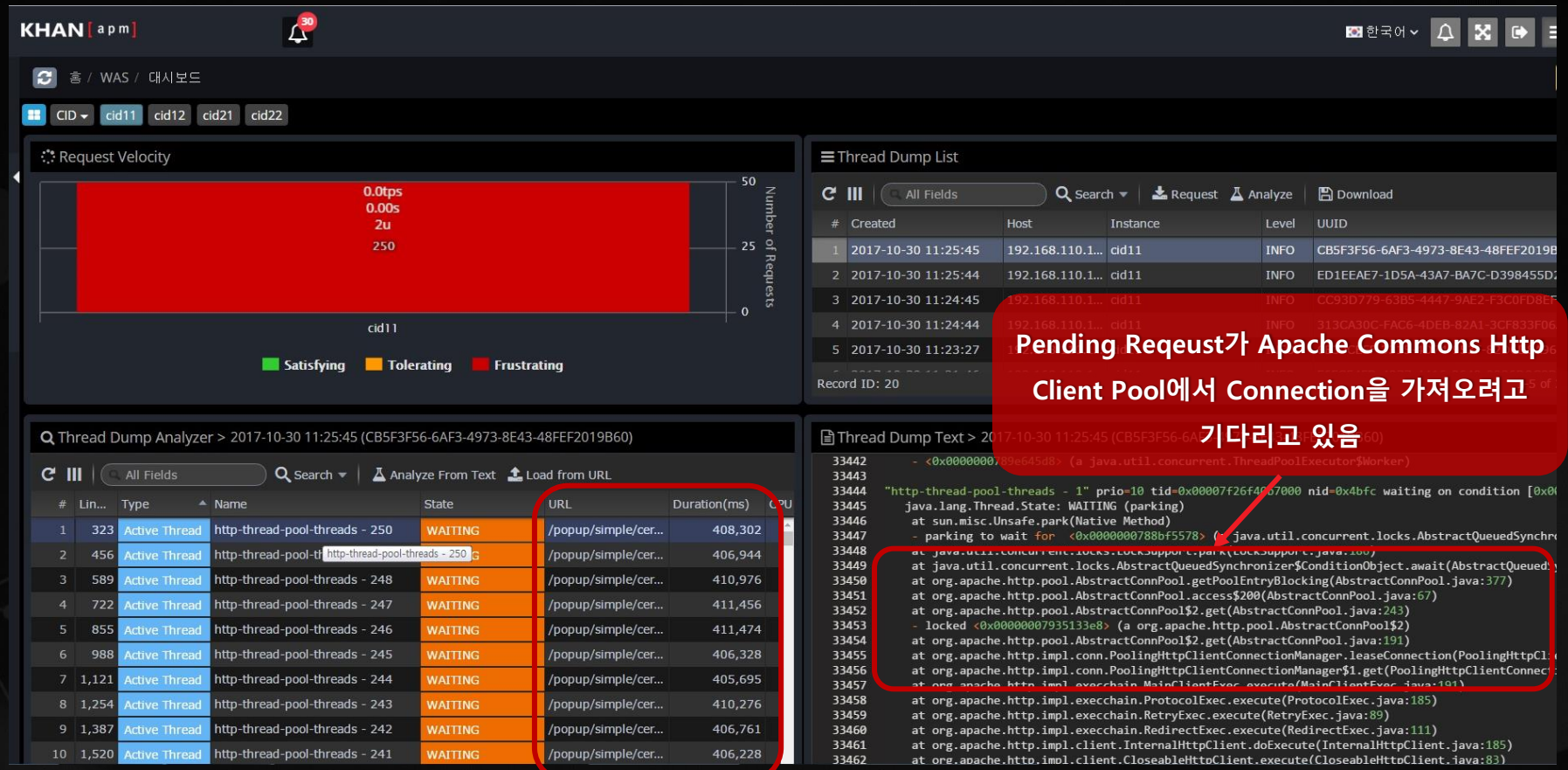


OPENMARU © 2021 | All Rights Reserved.

Server가 느려 지는 상황에서 Dashboard

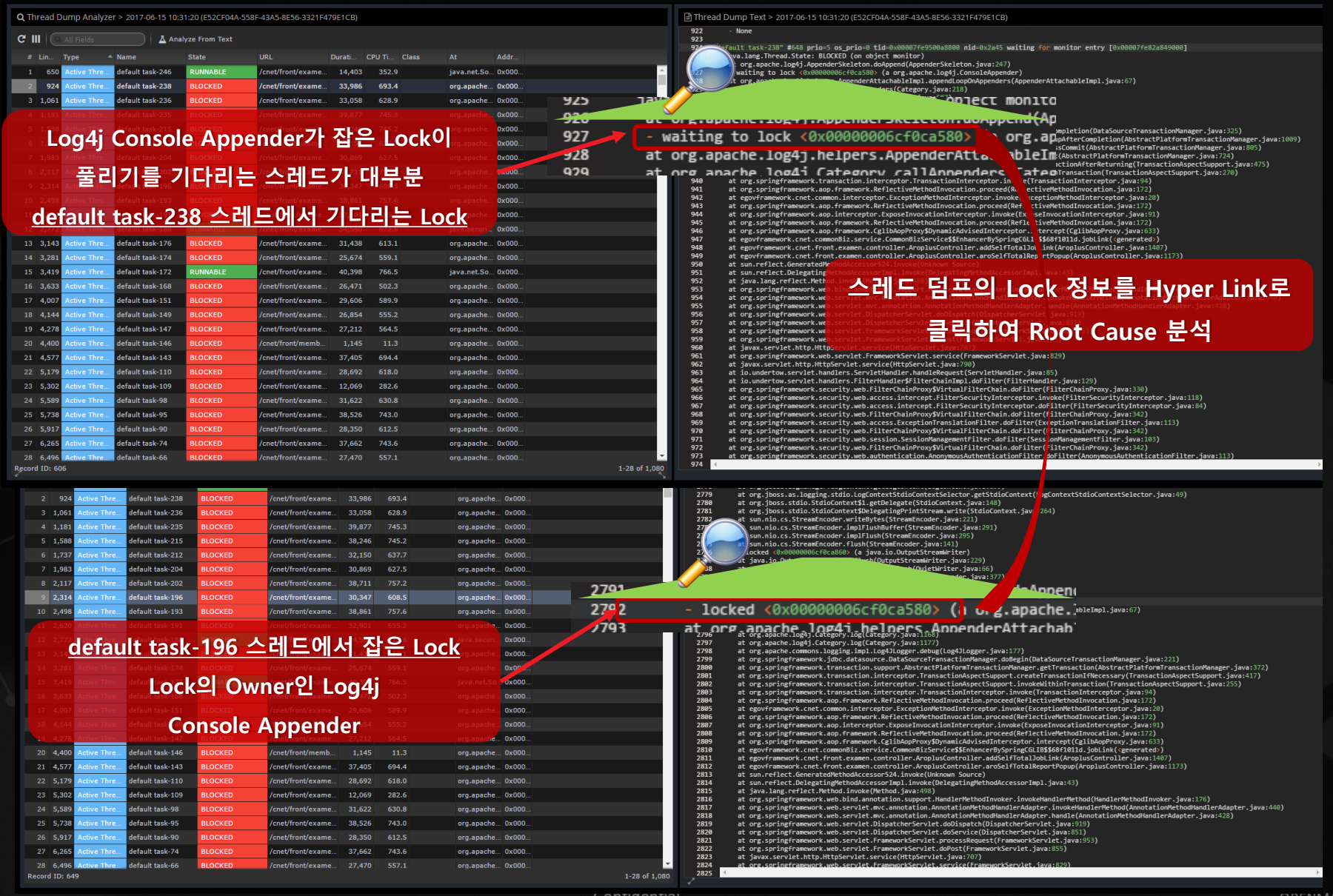
OPENMARU © 2021 | All Rights Reserved.

KHAN [apm] 스레드 덤프 분석 예시



OPENMARU © 2021 | All Rights Reserved.

KHAN [apm] 스레드 덤프 분석 예시



OPENMARU © 2021 | All Rights Reserved.

openmaru