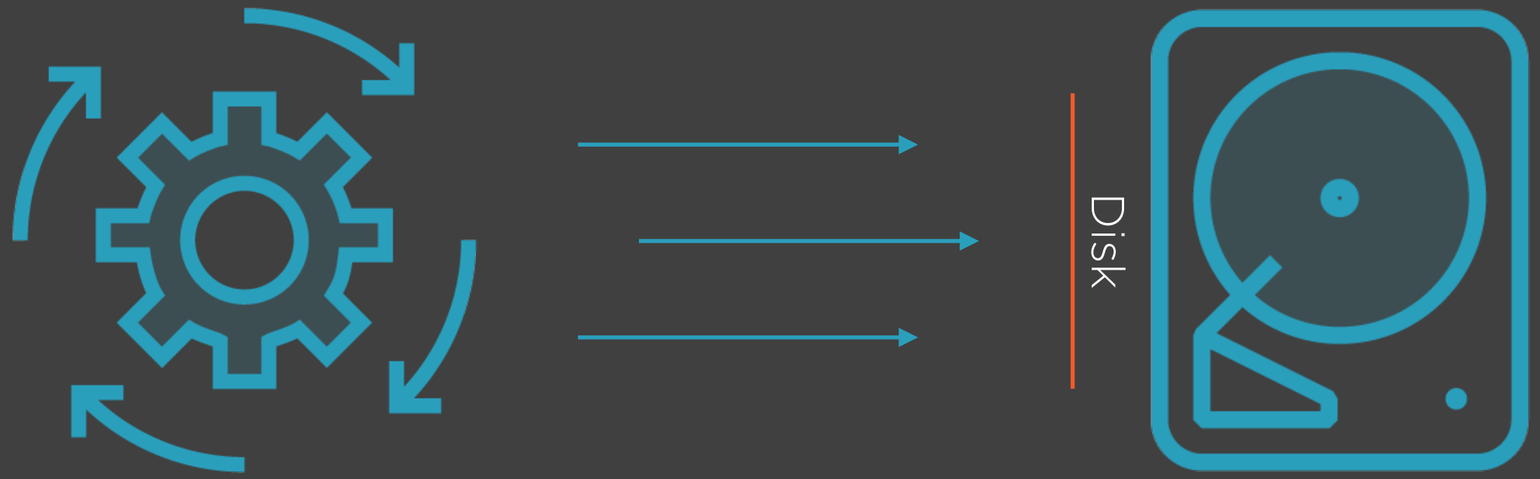
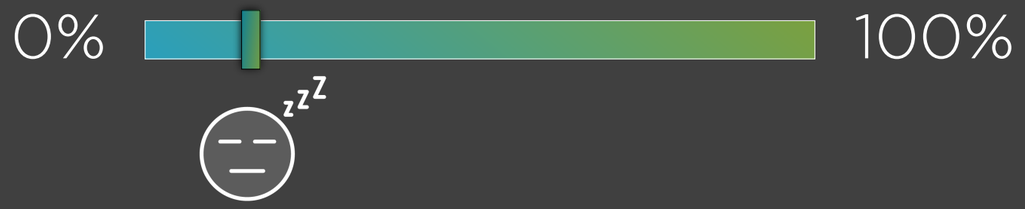
Isolating Performance Issues: A Filesystem Congestion

Uriah Levy
SOFTWARE ENGINEER
@iamuriahl www.medium.com/@iamuriahl

An I/O-bound Application





Filesystems Are Fast

- Logging
- Copying files from one location to another
- Downloading files from the internet

Overview

Understanding Filesystem read/write mechanics in Java

Identifying a filesystem bottleneck while reading/writing files using thread dumps

Identifying “Thread Stagnation”

Examining the application’s resource-footprint

Filesystem Read/Write Mechanics in Java

Input/Output Streams

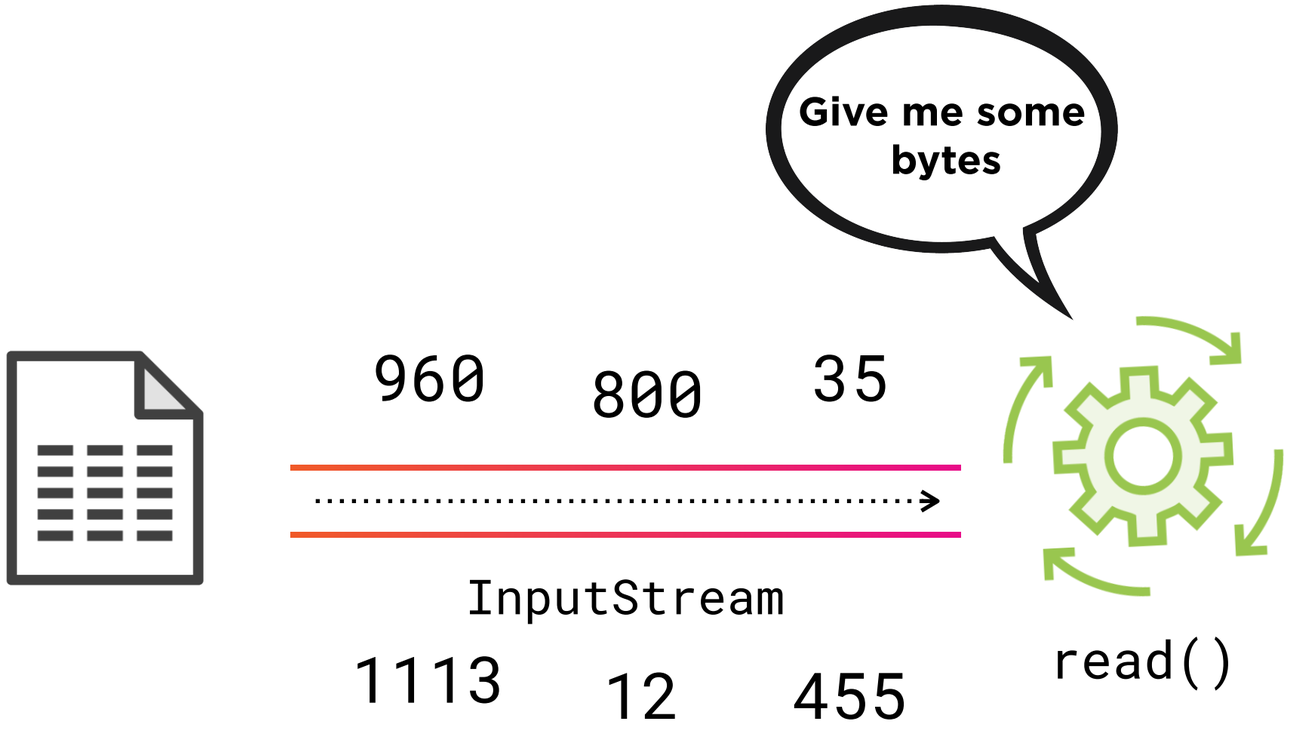
InputStream:

Reads data

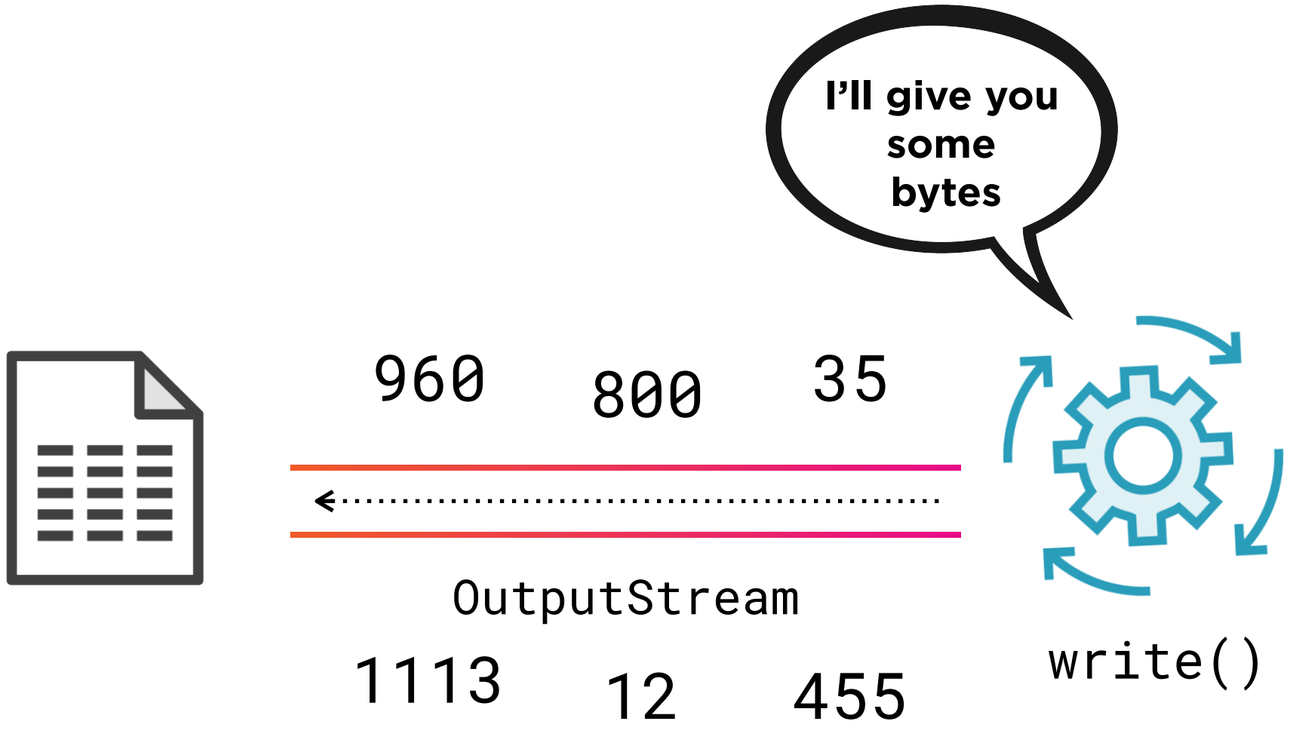
OutputStream:

Writes data

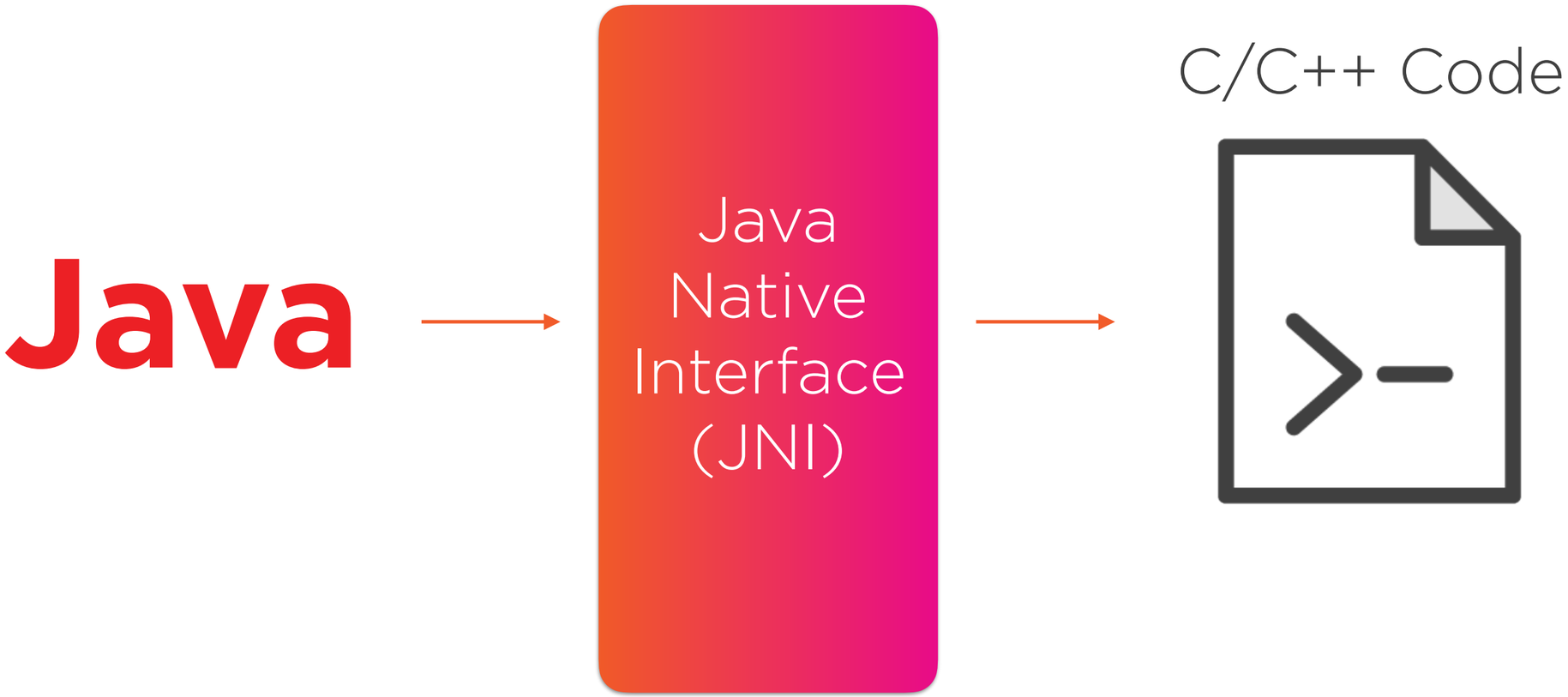
Reading from a Filesystem



Writing to a Filesystem



Low-level Action Delegation

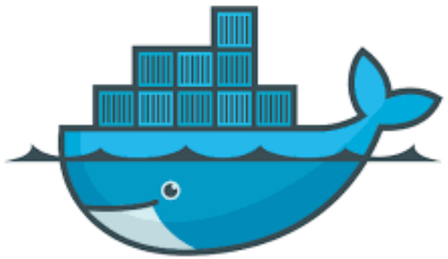


Remember Native Methods?

```
"pool-1-thread-1" #9 prio=5 os_prio=0 tid=0x00007f00001e9800 nid=0x6b15 runnable [0x00007effe9147000]
   java.lang.Thread.State: RUNNABLE
        at net.smacke.jaydio.DirectIoLib.pwrite(Native Method)
        at net.smacke.jaydio.DirectIoLib.pwrite(DirectIoLib.java:223)
        at net.smacke.jaydio.channel.DirectIoByteChannel.write(DirectIoByteChannel.java:93)
        at net.smacke.jaydio.channel.DirectIoByteChannel.write(DirectIoByteChannel.java:41)
        at net.smacke.jaydio.align.ByteChannelAligner.flush(ByteChannelAligner.java:321)
        at net.smacke.jaydio.align.ByteChannelAligner.truncate(ByteChannelAligner.java:292)
        at net.smacke.jaydio.align.ByteChannelAligner.close(ByteChannelAligner.java:88)
        at net.smacke.jaydio.DirectRandomAccessFile.close(DirectRandomAccessFile.java:97)
        at org.uriahl.ajtd.filesystem.DirectIoWriterCallable.call(FilesystemOperator.java:136)
```

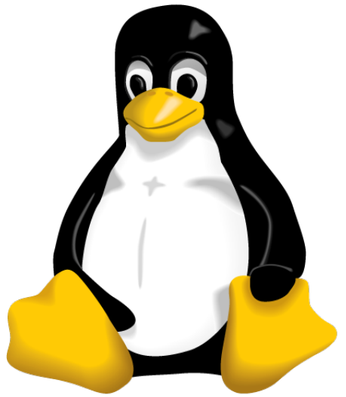
Native Method in the Stack

Docker / Linux cgroups



Docker

+



cgroups

```
docker run -it --device-read-bps /dev/sda:150kb ajtd-
ubuntu:latest /bin/bash
```

Spawns a container with **reads** limited to 150KB per-second

```
docker run -it --device-write-bps /dev/sda:150kb ajtd-
ubuntu:latest /bin/bash
```

Spawns a container with **writes** limited to 150KB per-second

The “blkio” Controller

```
> cat /sys/fs/cgroup/blkio/blkio.throttle.read_bps_device

8:0 153600
```

```
> cat /sys/fs/cgroup/blkio/blkio.throttle.write_bps_device

8:0 153600
```

> Demo

Follow the README.txt from the course assets
to be able to follow along on your environment

https://github.com/smacke/jaydio

Demo

Identifying “Thread Stagnation”

Debunking Thread Stagnation using ‘strace’

Examining the application’s resource-footprint

https://linux.die.net/man/1/strace

* Follow the README file from the Module 4 assets to deploy our strace-equipped docker container locally

Debugging Chain Pseudo-Snippet

```
First, invoke fs-operator:
  *while fs-operator is running:
    -Take some thread dumps
    -Attach to the JVM process with ‘strace’
```

Invoke ‘fs-operator.jar’

```
java -jar fs-operator.jar read 1 | ./nid-translator.pl >
thread-dumps-w-decimal-nids.out &
```

Translate ‘nid’ from Hex to Decimal

```
1  #!/usr/bin/perl -w
2
3  while (<>) {
4      if (/nid=(0x[[:xdigit:]]+)/) {
5          $lwp = hex($1);
6          s/nid=/lwp=$lwp nid=/;
7      }
8      print;
9  }
```

Credit: https://stackoverflow.com/questions/222108/getting-the-java-thread-id-and-stack-trace-of-run-away-java-thread

Capture 10 Thread Dumps

```
1  #!/bin/bash
2
3  a=1
4  numberOfKills=10
5  pid=$(ps aux| grep [j]ava| awk '{print $2}')
6
7  echo "Sending kill -3 at PID:" $pid
8
9  while [ $a -le $numberOfKills ]; do
10          kill -3 $pid
11          sleep 1
12          a=`expr $a + 1`
13 done
```

Attach with ‘strace’

Get the PID programatically

Specify the output filename(s) prefix

```
ps aux| grep [j]ava| awk '{print $2}'| xargs strace -o
fs-operator -tt -T -ff -p
```

Indicate system time in microseconds

Indicate total call time

Trace child processes

The Entire Debugging Chain

```
1  #!/bin/bash
2
3  java -jar fs-operator.jar read 1 | ./nid-translator.pl
4  > thread-dumps-w-decimal-nids.out &
5
6  sh dump-threads.sh &
7
8  ps aux| grep [j]ava| awk '{print $2}'| xargs strace -o
9  fs-operator -tt -T -ff -p
```