Erik Ostermueller

Author of

Troubleshooting Java Performance

http://bit.ly/2017tjp

Author of Java performance certification exam at

http://c0d3r.org

java architect

lead performance engineer

FIS Global

eostermueller@gmail.com

@EOstermueller

erikostermueller.com

P.A.T.H Checklist

where to look for performance defects

- **P**ersistence
- **A**lien systems
- **t**hreads
- **h**eap

MTDp

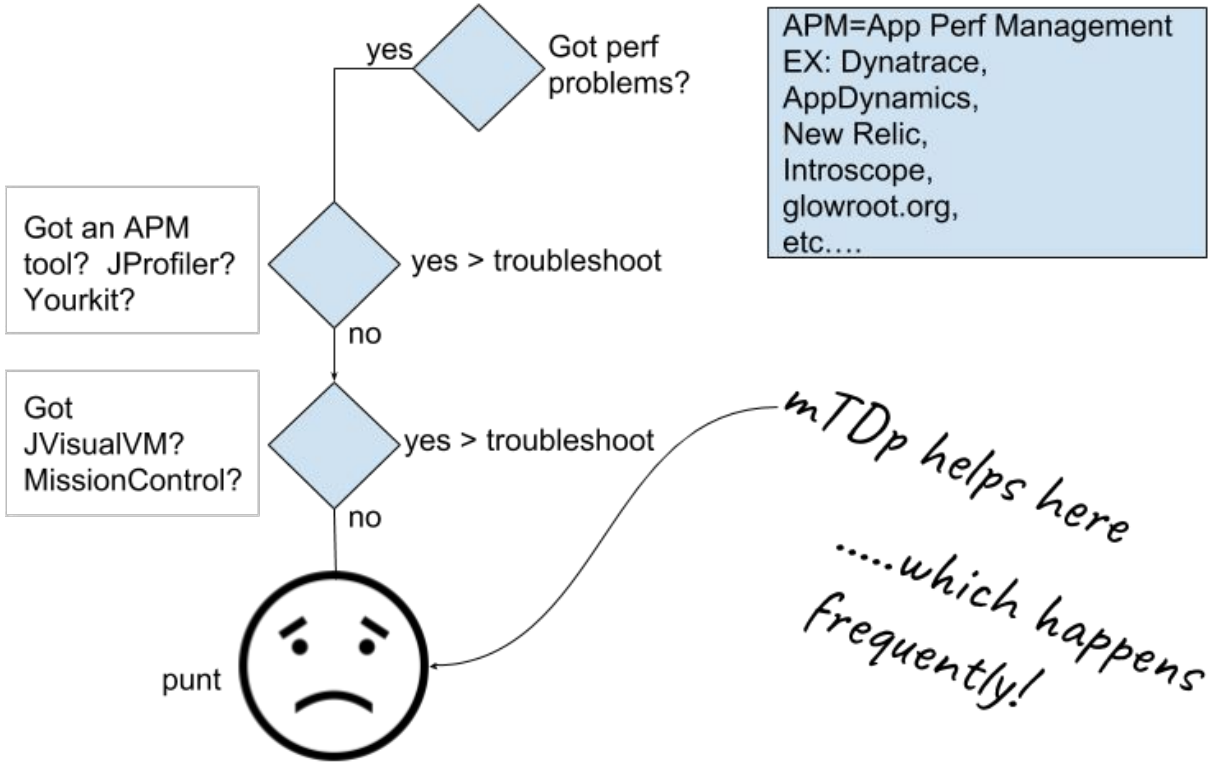
the Missing Visibility Tool

That will change everything

manual Thread Dump **profiling**

The problem: How can I Troubleshoot **any** running jvm?

Finally troubleshoot Any Running JVM



Plug it in now!

Don't make me restart my JVM

Don't make me install stuff

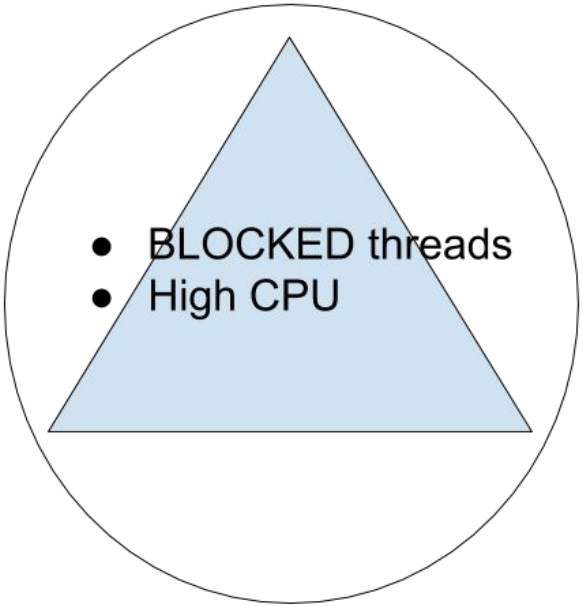
Why don't more people use mTDp?

- mTDp is hiding in a messy closet of under-documented perf techniques (*Building Better Applications* / 1994)
- Perhaps mTDp is mathematically misconstrued?

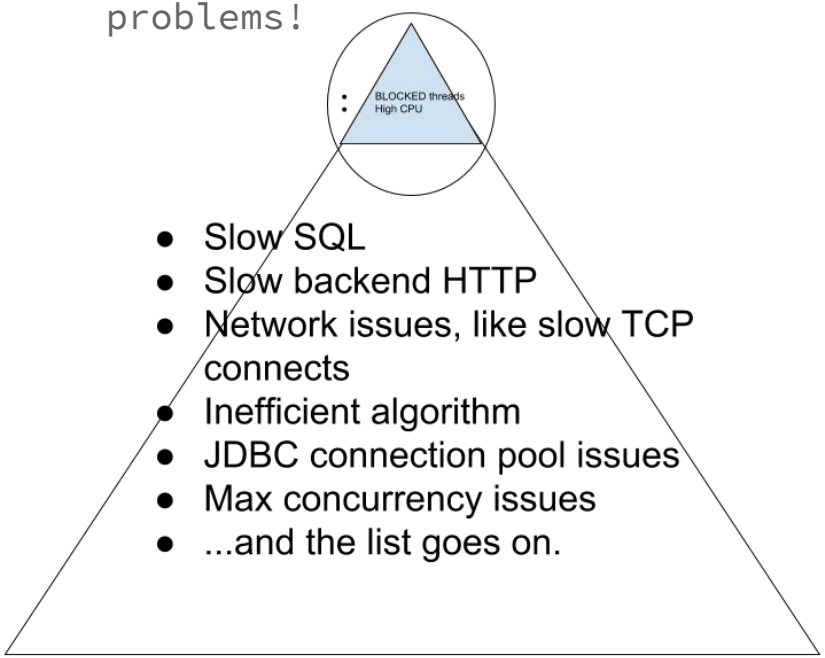
  http://ostermueller.blogspot.com/2017/04/the-math-behind-manual-thread-dump.html

Thread Dumps: Old school and New school

Without mTDp, thread dumps diagnose these:



With mTDp we can solve many more problems!



60 seconds on Thread Dump the basic

Jstack is low overhead

I use it regularly in **prod** and test

```
#~/jpt_threads: jcmd
6817 org.h2.tools.Server -tcp -web -baseDir ./data
8342 sun.tools.jcmd.JCmd
6839 warProject/target/performanceGolf.war
8341 com.jpt.MyThreadStarter <<<< 8341 is the PID we just started
```

Capture a Thread DUMP

```
#~/jpt_threads: jstack 8341 > myThreadDump.txt
```

Thread dump example: Jstack <myPid> or "kill -3"

```java
public class MyThreadStarter {
  public static void main( String[] args ) {
    new MyThread("jpt-first" ).start();
    new MyThread("jpt-second" ).start();
    new MyThread("jpt-third" ).start();
    }
}
class MyThread extends Thread {
  public MyThread(String name) {
        setName(name);
  }
  private void mySleep() {
        try { Thread.sleep(60000); } catch(Exception e) {}
  }
  public void run() {
          mySleep();
  }
}
```

```
java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at com.jpt.MyThread.mySleep(MyThreadStarter.java:18)
   at com.jpt.MyThread.run(MyThreadStarter.java:21)
```

```
java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at com.jpt.MyThread.mySleep(MyThreadStarter.java:18)
   at com.jpt.MyThread.run(MyThreadStarter.java:21)
```

```
java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at com.jpt.MyThread.mySleep(MyThreadStarter.java:18)
   at com.jpt.MyThread.run(MyThreadStarter.java:21)
```

Landmarks in a Stack Trace (my terms)

- Current – running on CPU when jstack was invoked
- Trigger – your code that leads to ‘Current’
- Entry – The protocol (HTTP, JMS) that started thread

```
1                          java.lang.Thread.State: TIMED_WAITING (sleepin
2   Current    -->              at java.lang.Thread.sleep(Native Method)
3   Trigger    -->              at com.jpt.MyThread.mySleep(MyThreadStar
4   Entry      -->              at com.jpt.MyThread.run(MyThreadStarter.
```

MTDp
A Description

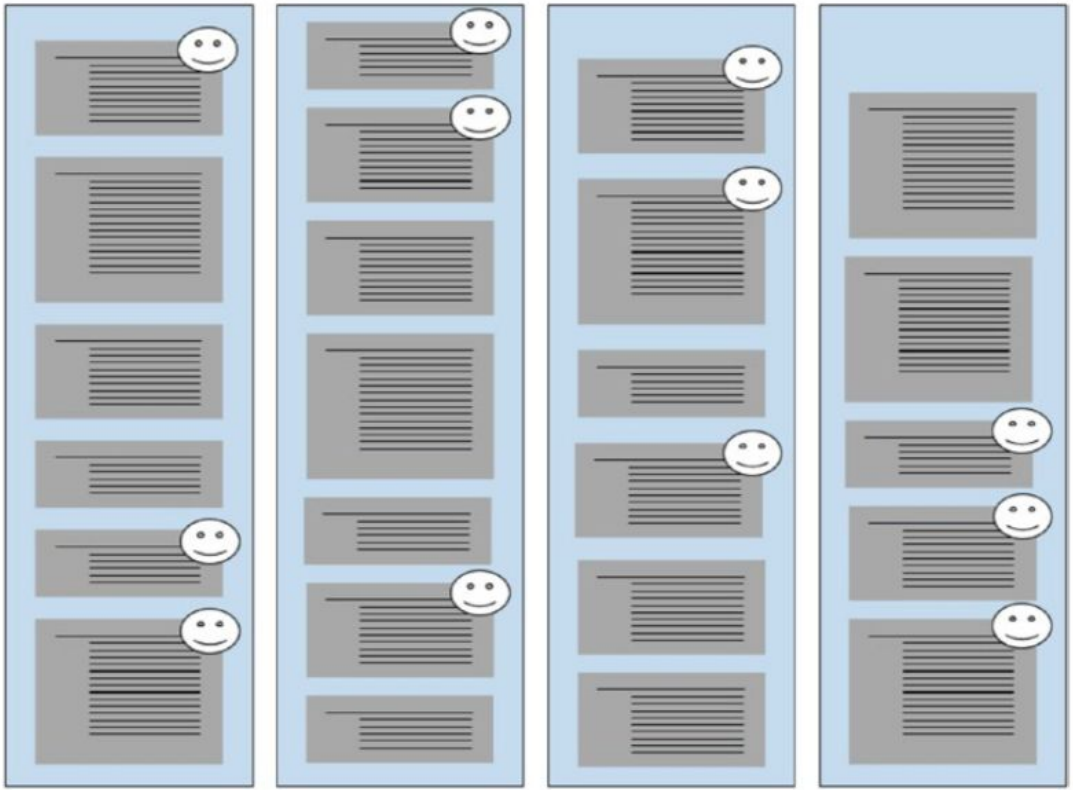
Step 1

Take four or so thread dumps
…as load is applied
…with a few seconds between each dump.

Step 2

If something that you could fix …shows up in the dumps …for two or more threads that are under load, it is worth fixing.

A Visual of Four Thread Dumps



MTDP Tips

Focus on Threads Under Load

(assume your code is the problem)

- Focus on all stack traces containing your package name!
- Focus on all stack traces started by your protocol.

  If an **HTTPS** system, look for threads with an “entry” marked with **HTTPS-ish** class names.

Names of "web container" threads

Container	Example of thread name
WebSphere	`WebContainer : 5`
Spring Boot / Tomcat	`http-nio-8080-exec-7`
Spring Boot / Jetty	`qtp266500815-40`
Wildfly Servlet 11.0	`default task-127`

Common ways to mess up mTDp

- Make judgements based on just one thread dump
  FIX → take 4 or more

- Make judgements based on system with low-load or no-load
  FIX → check for moderate CPU use

Common ways to mess up mTDp (part 2)

- Don’t lose focus of the threads under load.
    - If an **HTTPS** system, look for threads with an “entry” marked with **HTTPS-ish** class names.
    - If a **JMS** system, look for threads with an “entry” marked with **JMS-ish** class names.
- Thread dumps don’t show GC/heap problems
  FIX → use jstat for a “plug-it-in-now” view of GC.

Reminder: GC issues don't show up in thread dumps

Use vi and Notepad++

Thread Dump analysis tools are great, but they limit you to the tip of the iceberg.

MTDP CAN DETECT PROBLEMS LARGE (>1S) AND SMALL (2-3MS)

MTDP
Demo

THE END
Erik Ostermueller

Author of

Troubleshooting Java Performance

http://bit.ly/2017tjp

Author of Java performance certification exam at

http://c0d3r.org

java architect

lead performance engineer

FIS Global

eostermueller@gmail.com

@EOstermueller

erikostermueller.com