Beyond top(1)

Command-Line Monitoring on the JVM

Colin Jones @trptcolin 8th Light

command-line tooling

introspection & serviceability on the JVM

war stories

A long time ago in a startup far, far away...

Things are going pretty well 😏

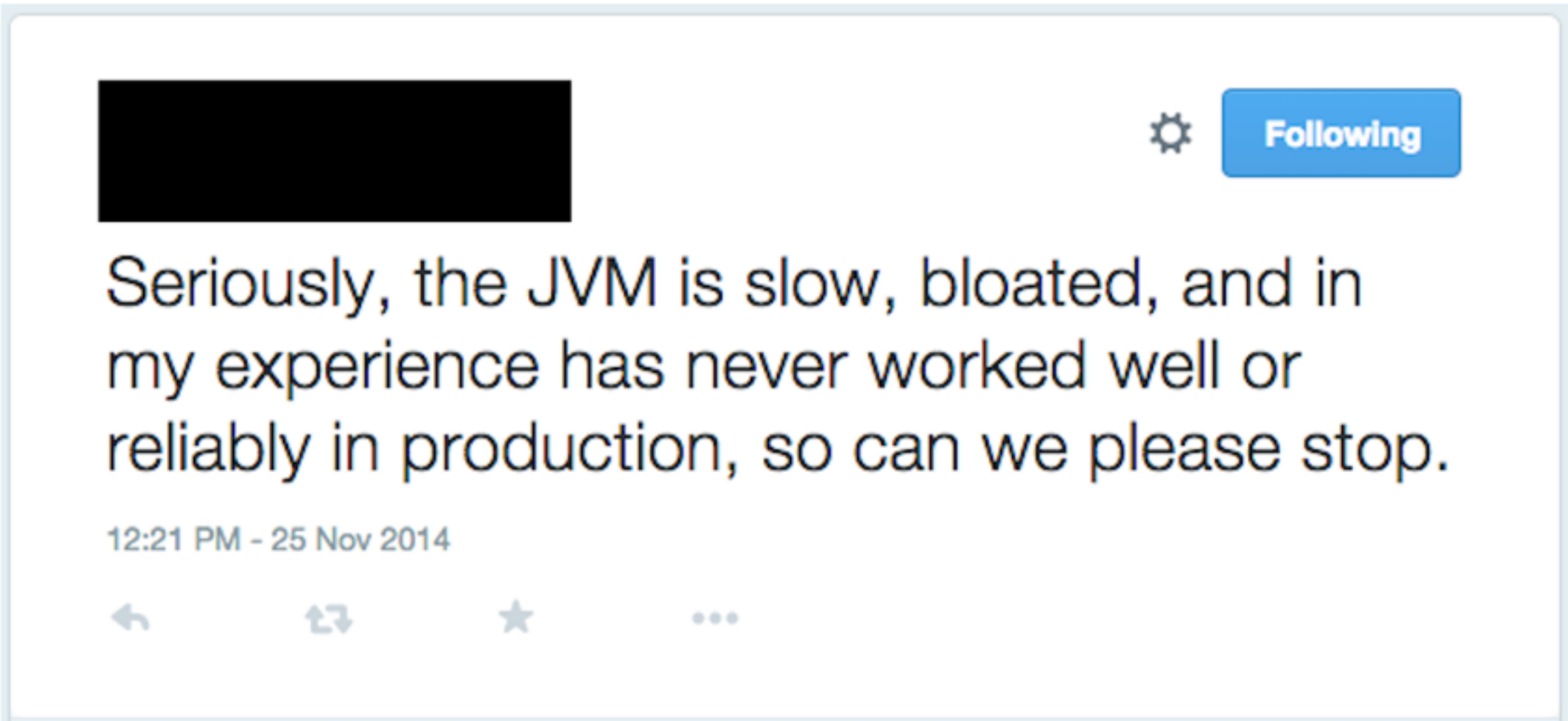
But strange things

are afoot 

the server sometimes
gets really slow

the team has to manually restart the application server

Pain, frustration, anger



Just the facts

sometimes,
things get slow

all requests seem
to be affected

the JVM stays up

restart the JVM and
everything is fine

What could it be?

Demo

More facts!

what application
code was running

constant full GCs

what’s in the heap

What could it be? 🤔

```
11 (defn longest [xs ys] (if (> (count xs) (count ys)) xs ys))
12
13 (def lcs
14   (memoize
15    (fn [[x & xs] [y & ys]]
16      (cond
17       (or (= x nil) (= y nil) ) nil
18       (= x y) (cons x (lcs xs ys))
19       :else (longest (lcs (cons x xs) ys) (lcs xs (cons y ys)))))))
20
21 (defn closest-match [search-text names]
22   (let [measure-match-goodness (comp count (partial lcs search-text))]
23     (first (sort-by measure-match-goodness names))))
24
25 (defn users-routes [db]
26   (compojure/routes
27     (GET "/" [] {:status 200 :body "{}"})
28     (GET "/autocomplete" [q :as request]
29          (let [matches (find-all-like db q)
30                matching-names (map :name matches)
31                username (closest-match q matching-names)]
32            {:body {:username username}})))
```

```
11 (defn longest [xs ys] (if (> (count xs) (count ys)) xs ys))
12
13 (def lcs
14   (memoize
15    (fn [[x & xs] [y & ys]]
16      (cond
17       (or (= x nil) (= y nil) ) nil
18       (= x y) (cons x (lcs xs ys))
19       :else (longest (lcs (cons x xs) ys) (lcs xs (cons y ys)))))))
20
21 (defn closest-match [search-text names]
22   (let [measure-match-goodness (comp count (partial lcs search-text))]
23     (first (sort-by measure-match-goodness names))))
24
25 (defn users-routes [db]
26   (compojure/routes
27     (GET "/" [] {:status 200 :body "{}"})
28     (GET "/autocomplete" [q :as request]
29          (let [matches (find-all-like db q)
30                matching-names (map :name matches)
31                username (closest-match q matching-names)]
32            {:body {:username username}})))
```

Mystery solved! 🙌

Now “just” fix it 😂

idea 1:

eliminate the leak

idea 2:
eliminate the cache
altogether?

idea 3:
delete the feature

idea 4:
re-think the problem

So we're good,

for now... 

Lessons

“it’s slow” could mean lots of things

“high CPU” could mean lots of things

collecting data is crucial in a crisis

reproducing the issue
helps me sleep at night

The right tools for the job

vmstat

system-level:

CPU, memory, swapping, I/O

top

per-process:

CPU & memory

jps

what’s our PID?

jcmd

what **can't** it do?!

```
jcmd [PID] help
```

(JVM 6 users: see jinfo/jmap/jstack)

jstack

status of all threads (**right now-ish!**)

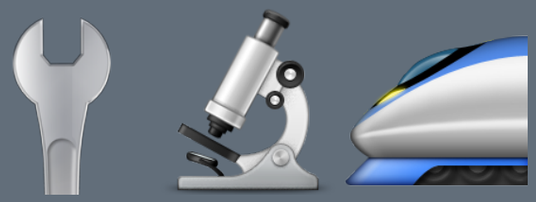
jstat

classloader

compiler

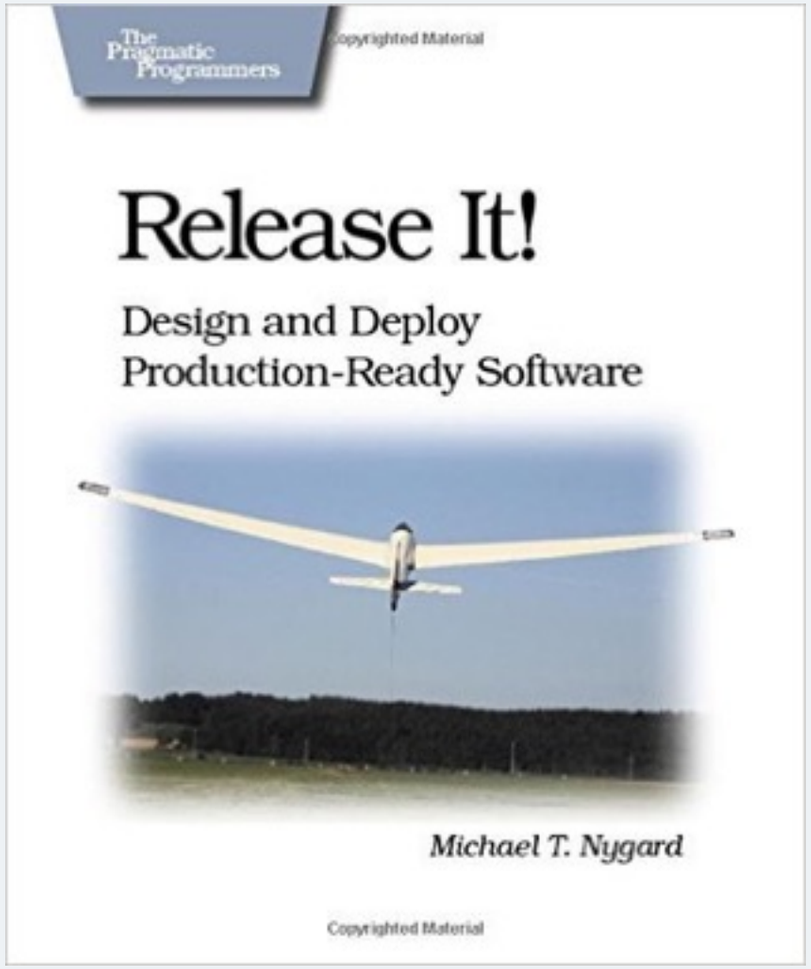
GC

Other “right tools for the job”

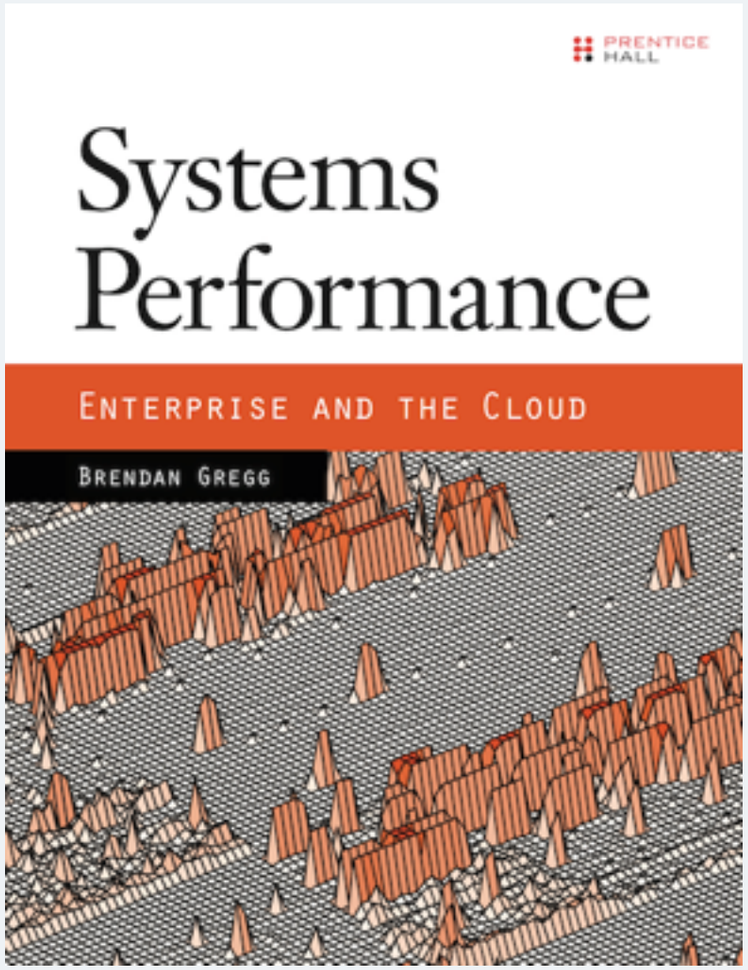


Learning more

Books!


The Pragmatic Programmers
Copyrighted Material
Release It!
Design and Deploy
Production-Ready Software
Michael T. Nygard
Copyrighted Material


O'REILLY
Java
Performance
The Definitive Guide
GETTING THE MOST OUT OF YOUR CODE
Scott Oaks


PRENTICE HALL
Systems
Performance
ENTERPRISE AND THE CLOUD
BRENDAN GREGG

operators are standing by!

Thank you!

Colin Jones @trptcolin

8th Light