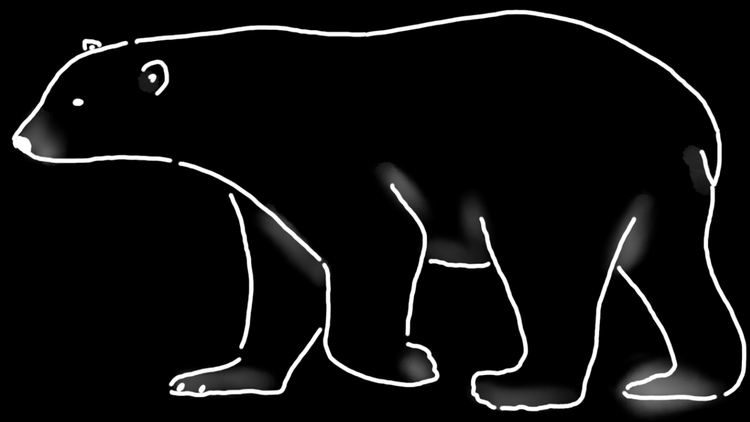
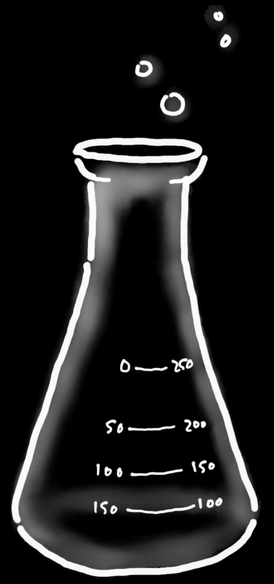
tradeoffs, bad science, and polar bears:

the world of java optimisation





Holly Cummins
IBM
@holly_cummins

why optimise?

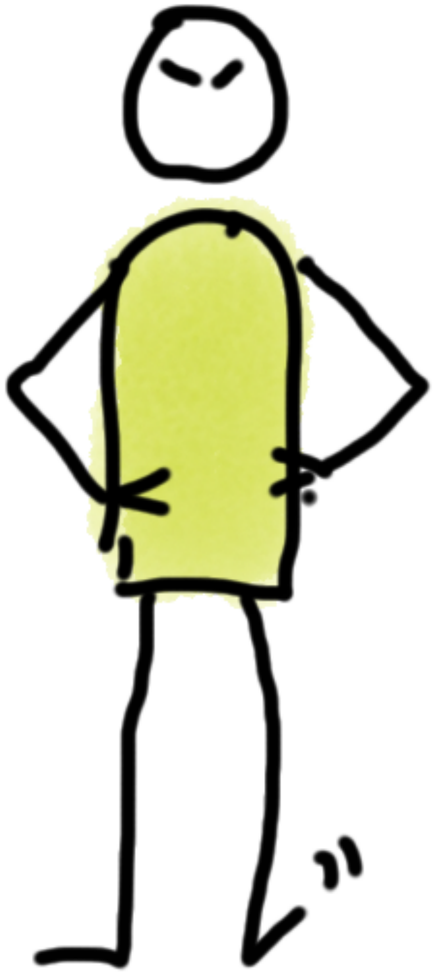
why optimise?

0.5s extra search page time

why optimise?

0.5s extra search page time

20% drop in traffic

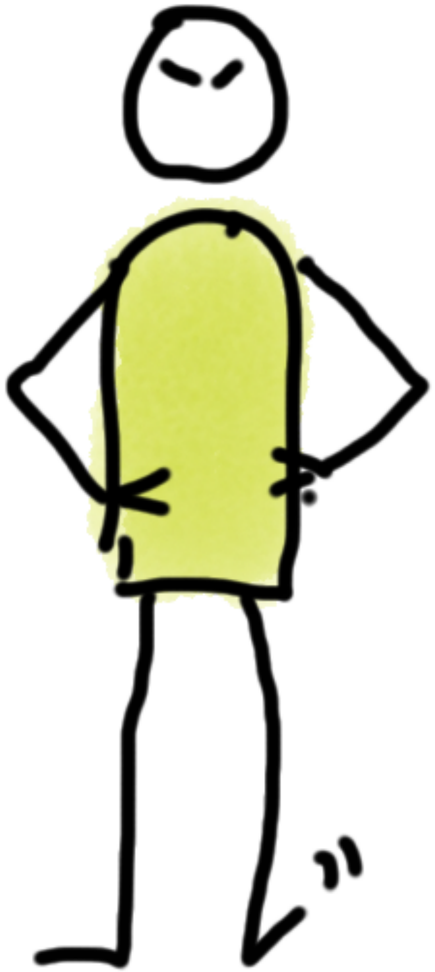


why optimise?

0.5s extra search page time

20% drop in traffic

100 ms latency on page load



why optimise?

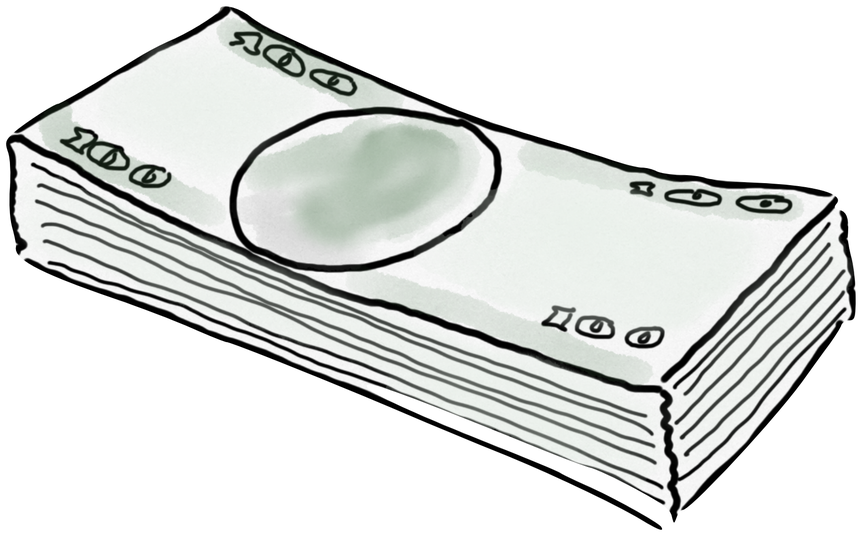
0.5s extra search page time → 20% drop in traffic

100 ms latency on page load → 7% lower conversion rate

why optimise?

0.5s extra search page time → 20% drop in traffic

100 ms latency on page load → 7% lower conversion rate

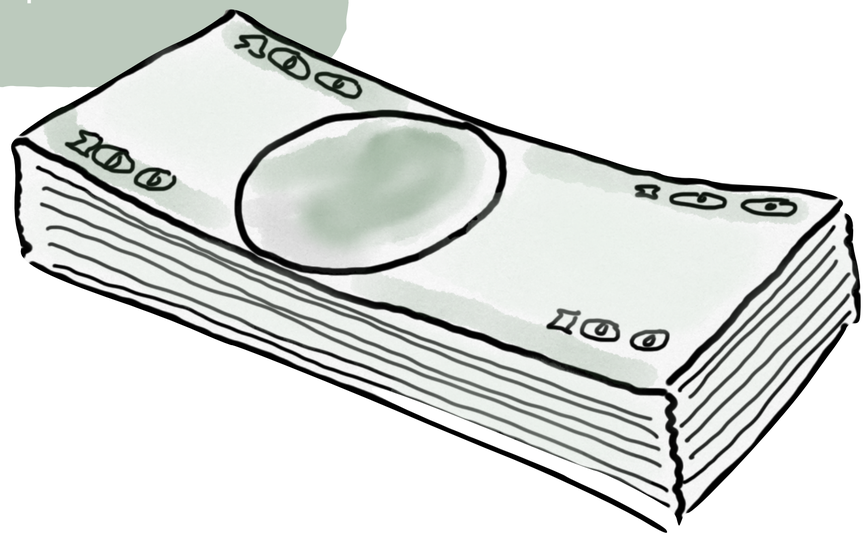


why optimise?

0.5s extra search page time → 20% drop in traffic

100 ms latency on page load → 7% lower conversion rate

10 ms delay in trading platform

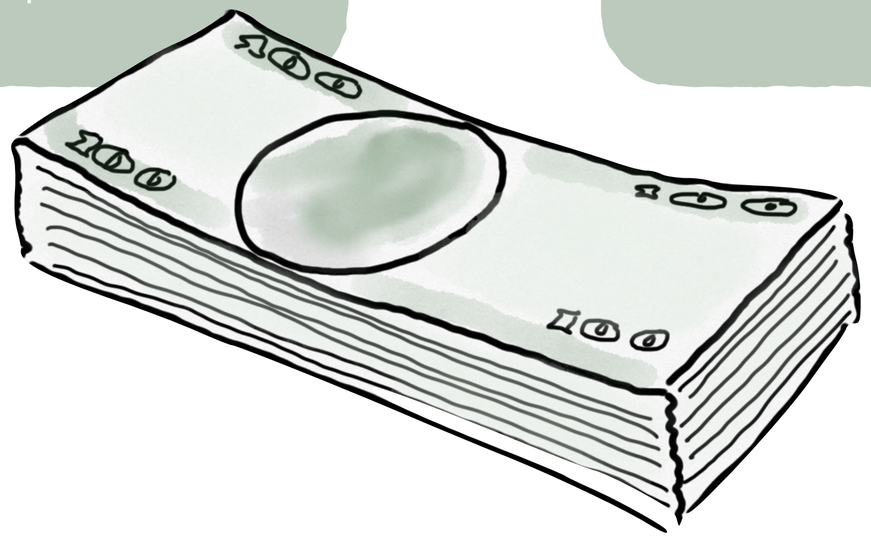


why optimise?

0.5s extra search page time → 20% drop in traffic

100 ms latency on page load → 7% lower conversion rate

10 ms delay in trading platform → 10% drop in revenue



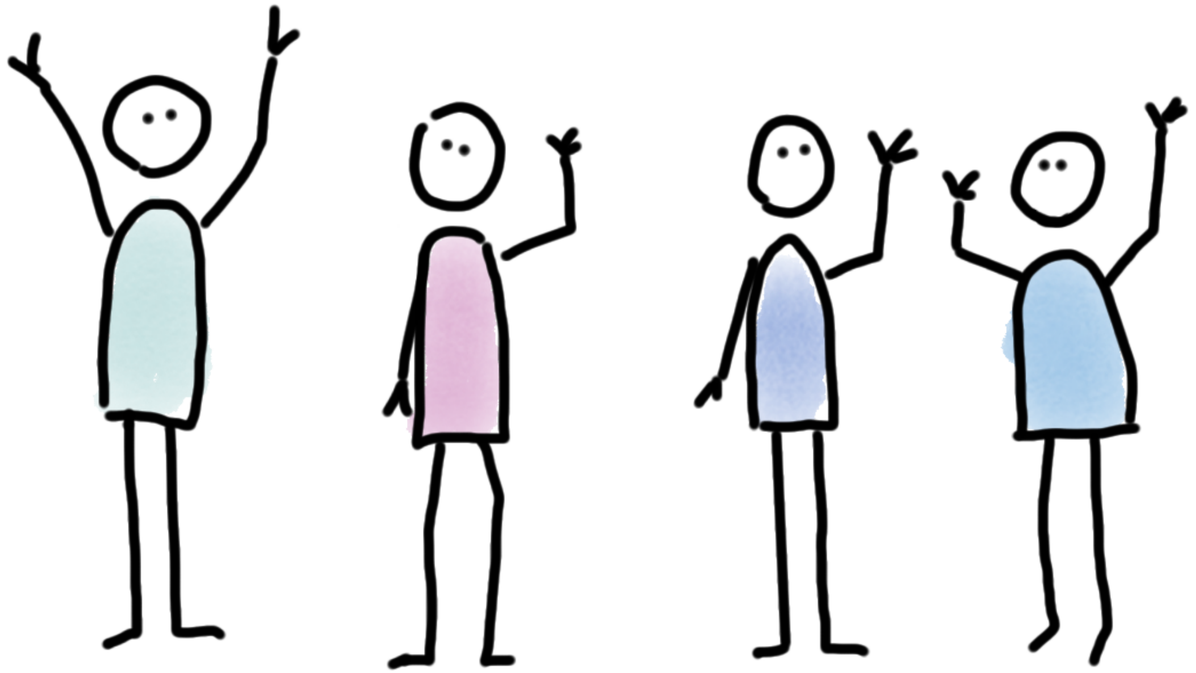
why optimise?

what **is** optimising?

“make it go faster”

for whom?
when?
doing what?

design thinking



performance can be:

performance can be:

throughput

performance can be:

throughput

transactions
per second

performance can be:

throughput

transactions per second

latency

performance can be:

throughput

transactions per second

latency

start-up time

performance can be:

throughput

transactions per second

latency

response time

start-up time

performance can be:

throughput

latency

transactions per second

response time

start-up time

ramp-up time

performance can be:

throughput

latency

capacity

transactions per second

response time

start-up time

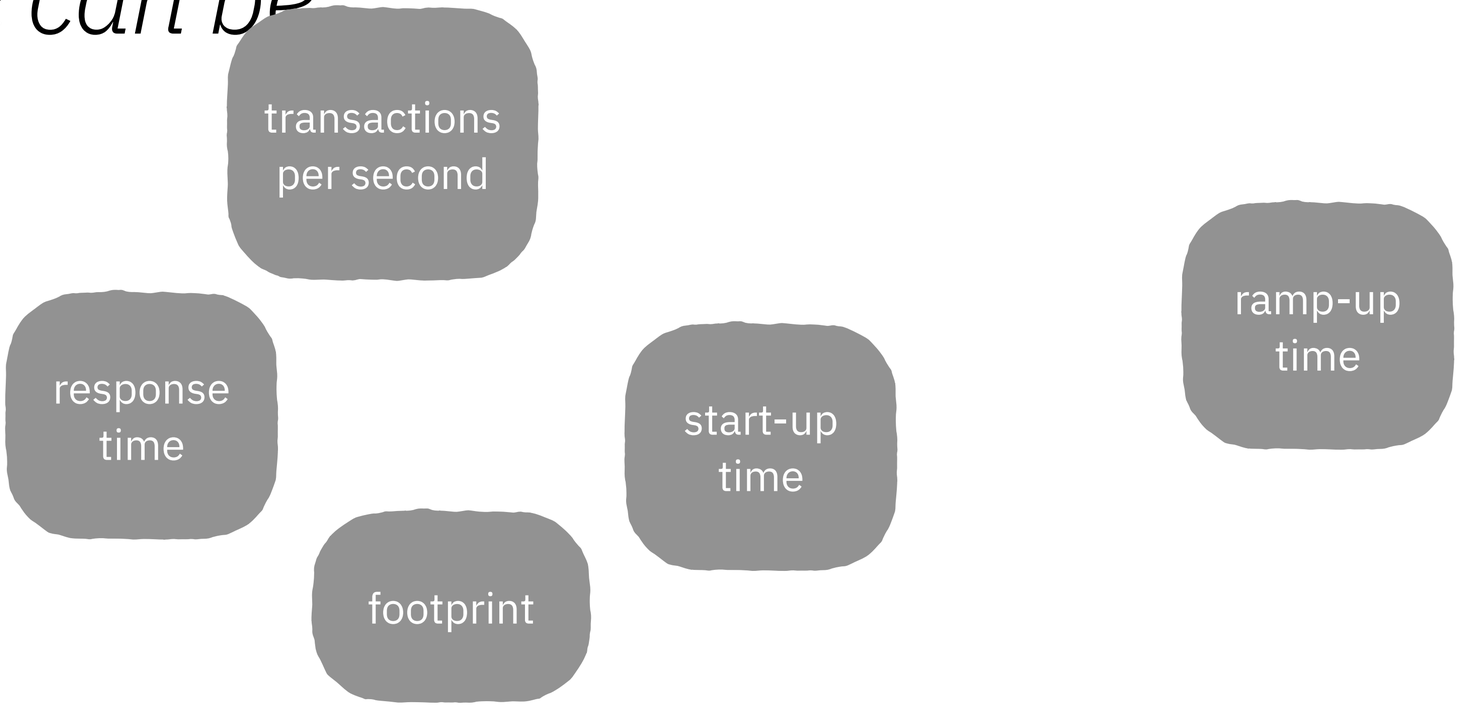
ramp-up time

performance can be:

throughput

latency

capacity

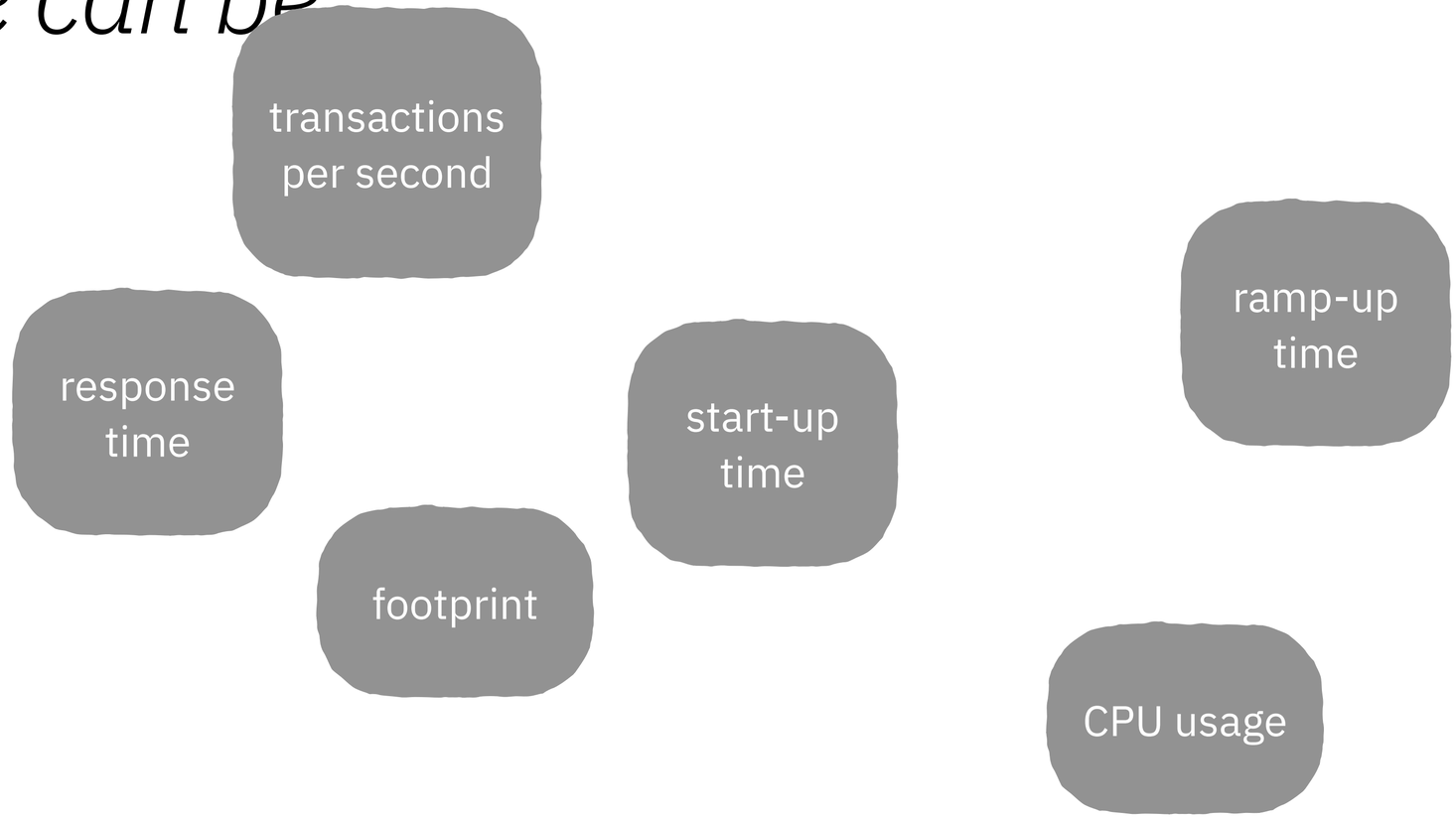


performance can be:

throughput

latency

capacity



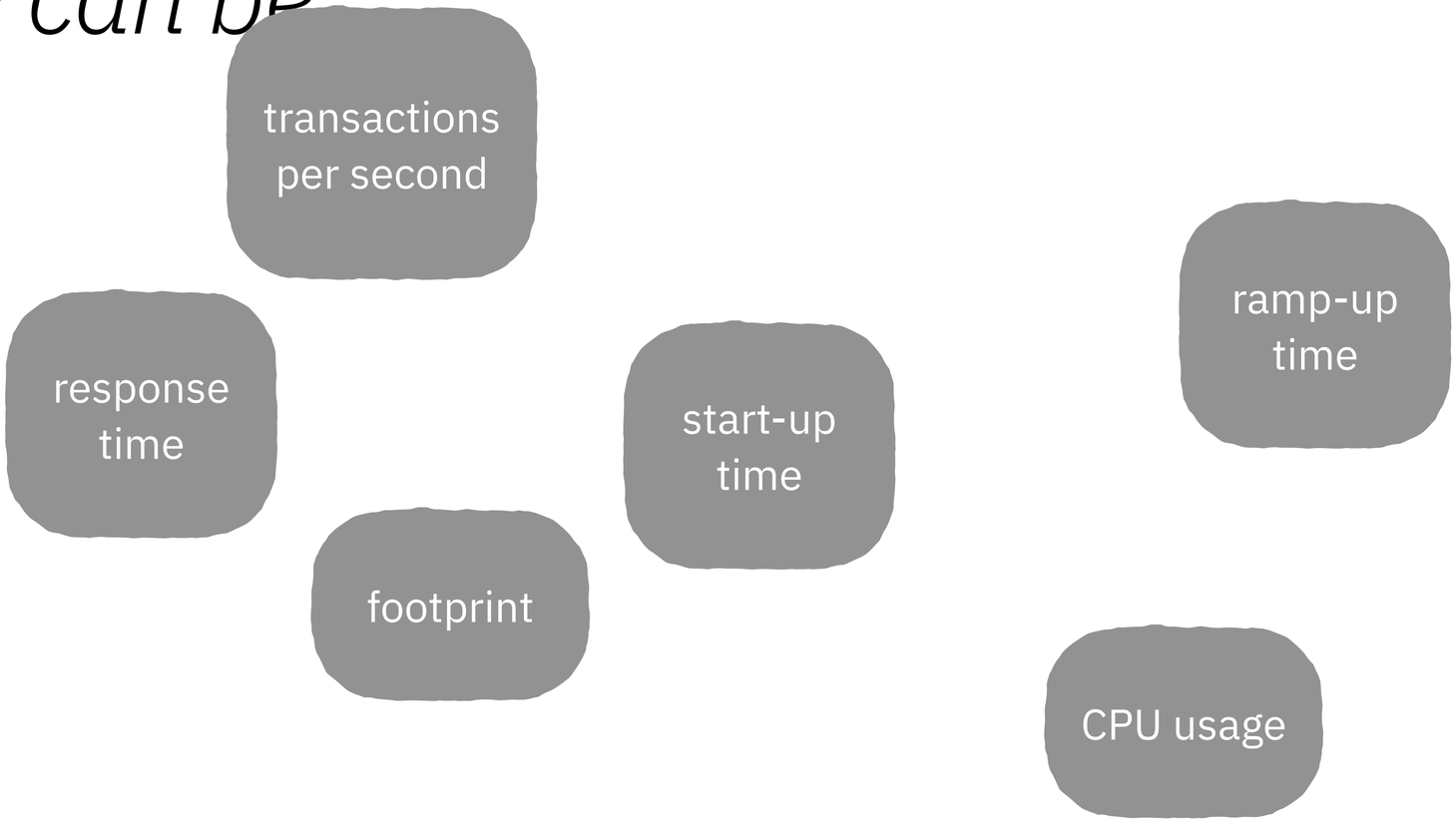
performance can be:

throughput

latency

capacity

utilisation



performance can be:

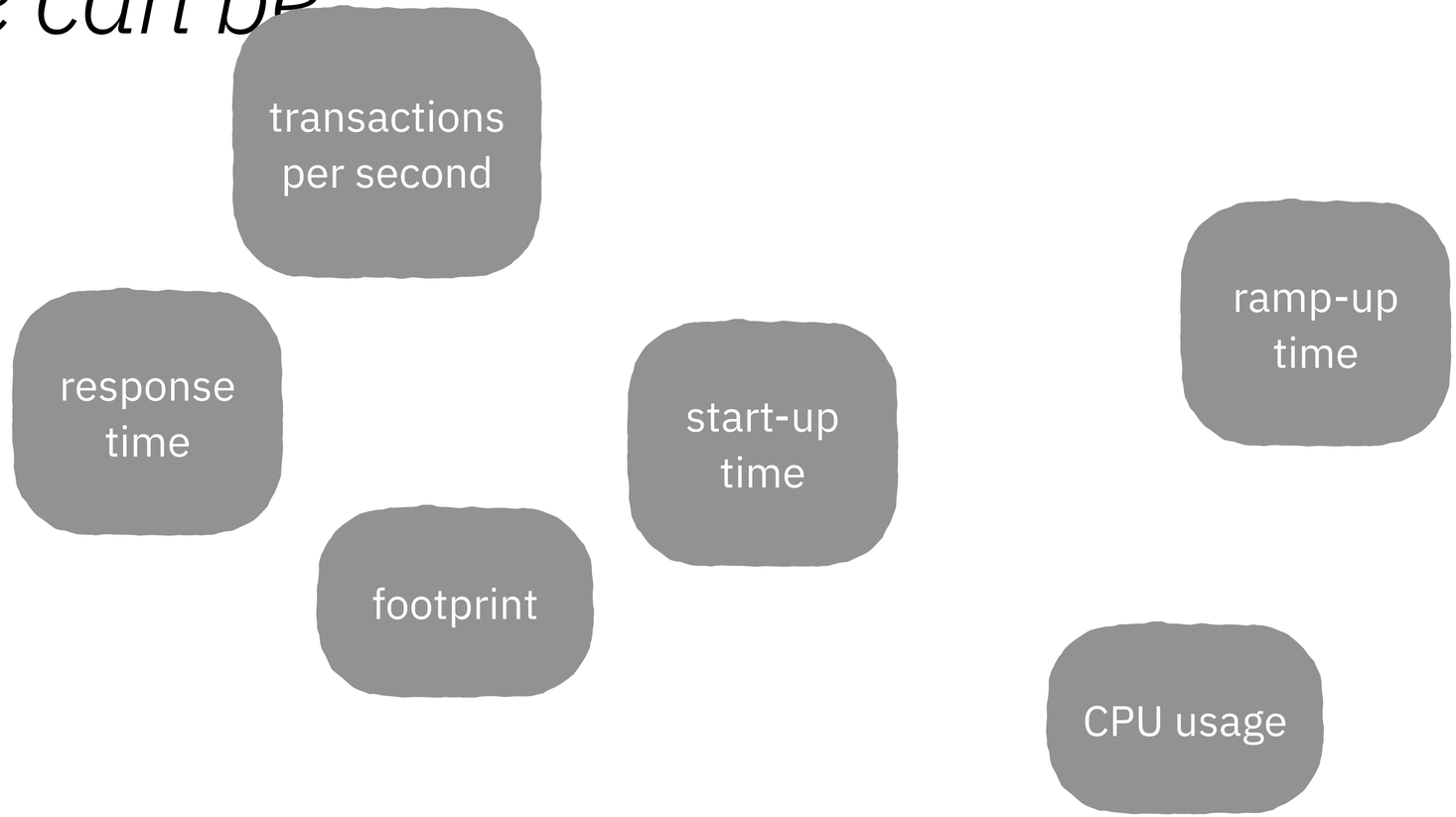
throughput

latency

capacity

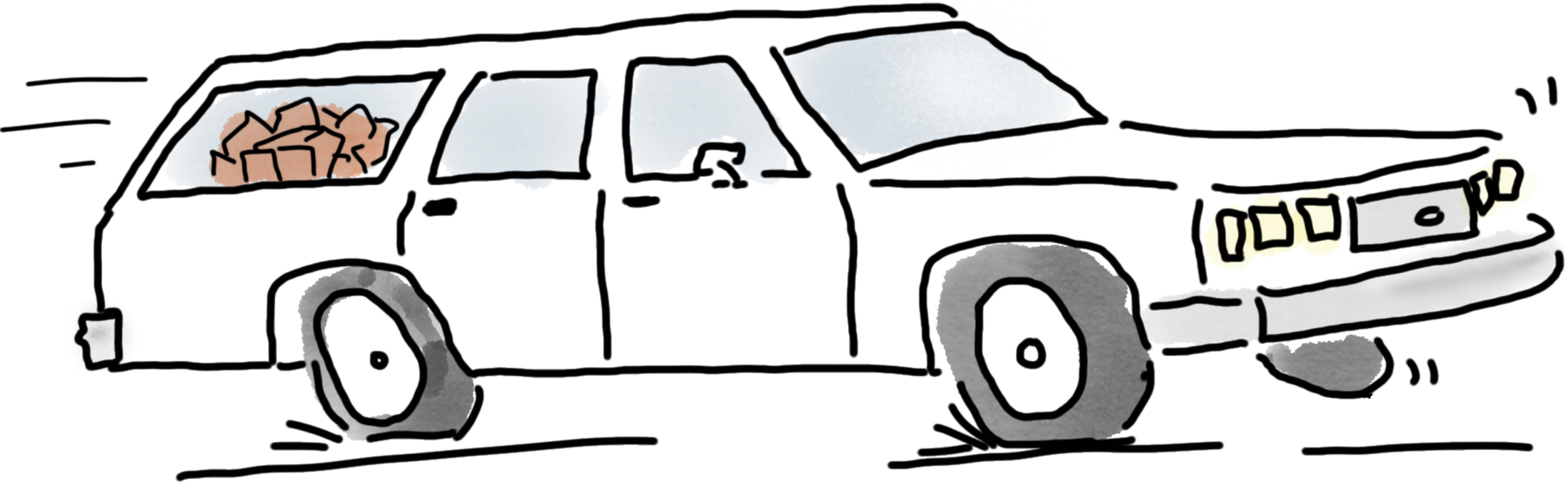
utilisation

...



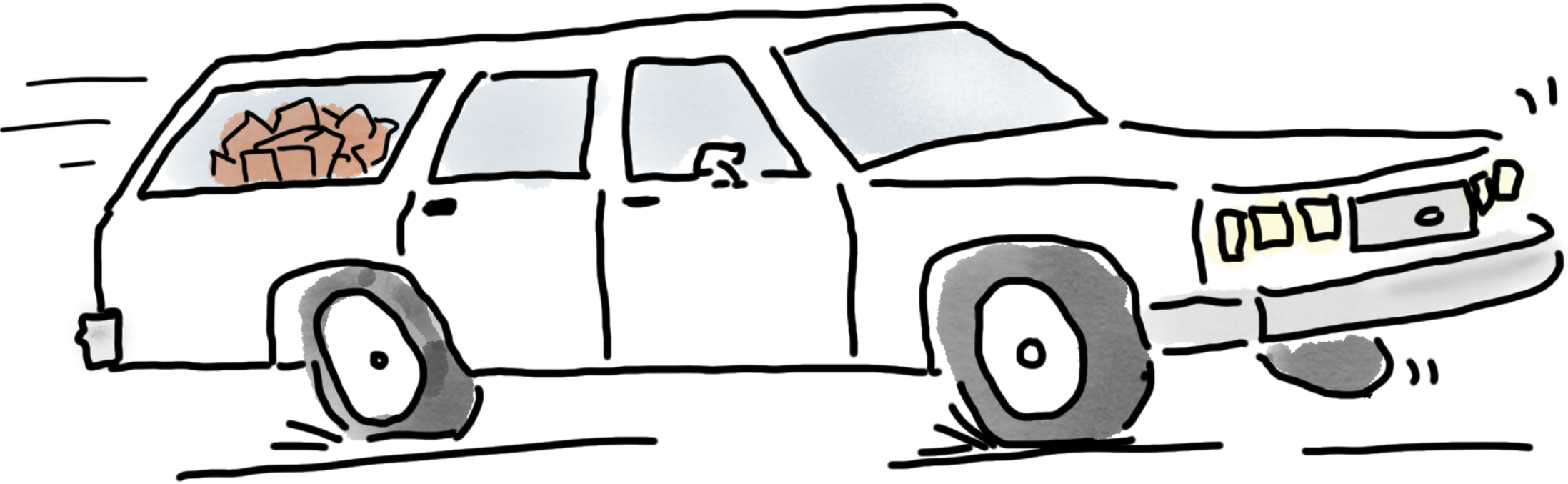
Never underestimate the bandwidth [throughput] of a station wagon full of tapes hurtling down the highway.

–Andrew Tanenbaum, 1981



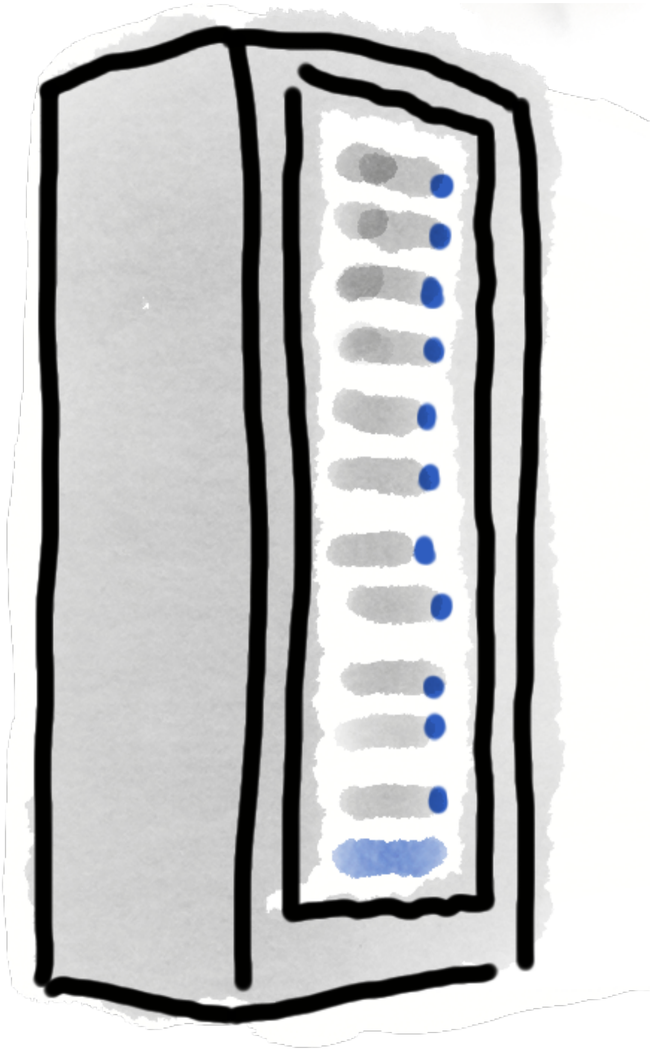
Never underestimate the bandwidth [throughput] of a station wagon full of tapes hurtling down the highway.

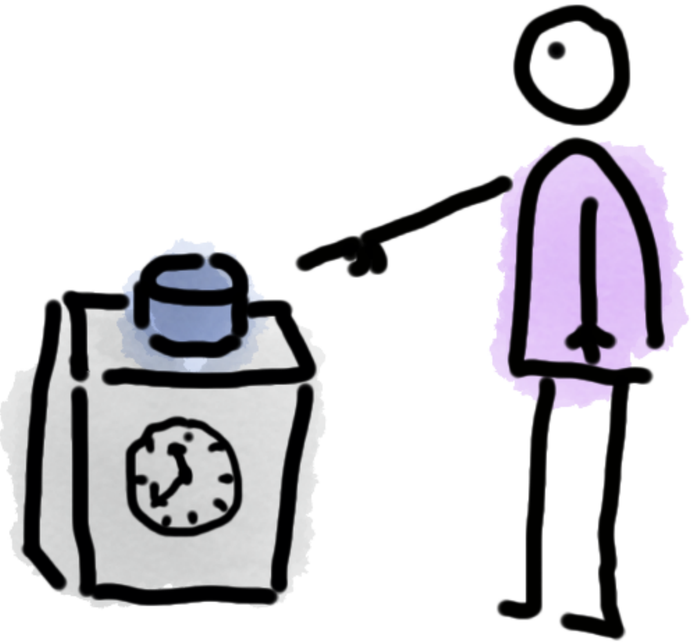
–Andrew Tanenbaum, 1981

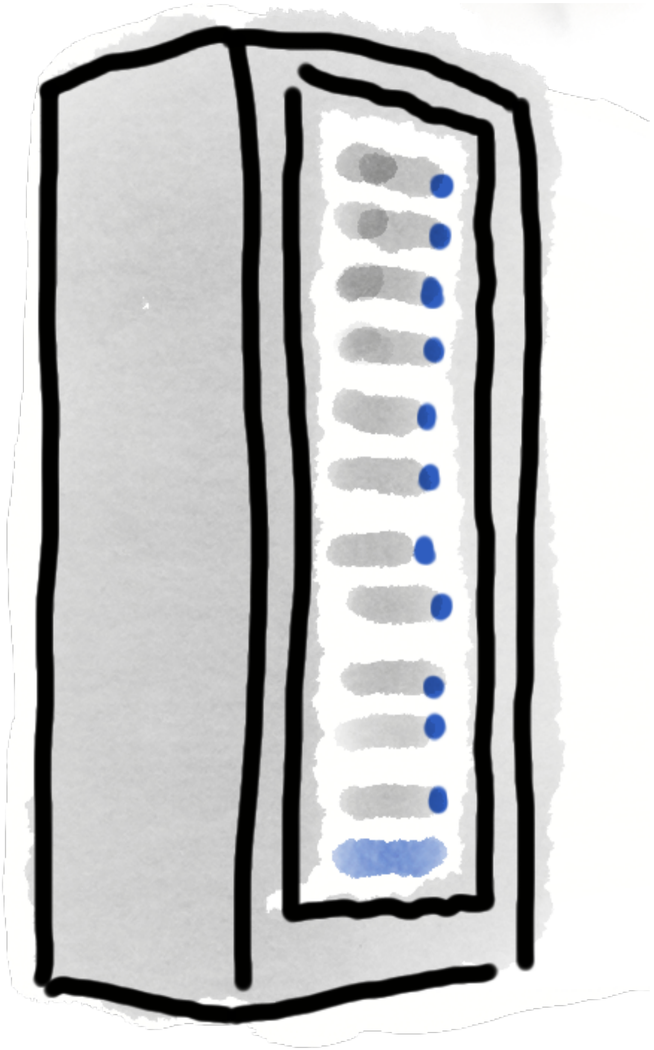


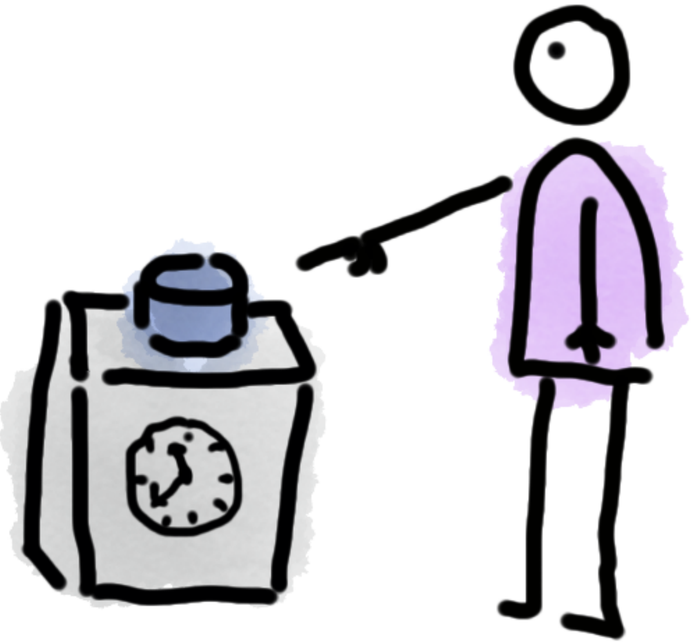
but the latency is terrible ...

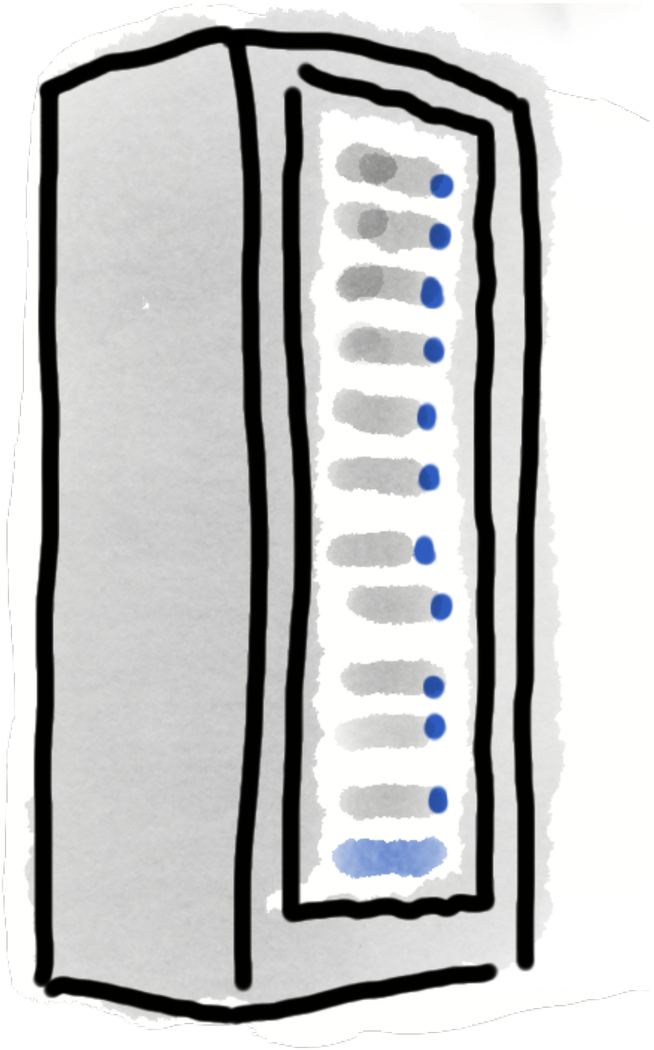
requirements change

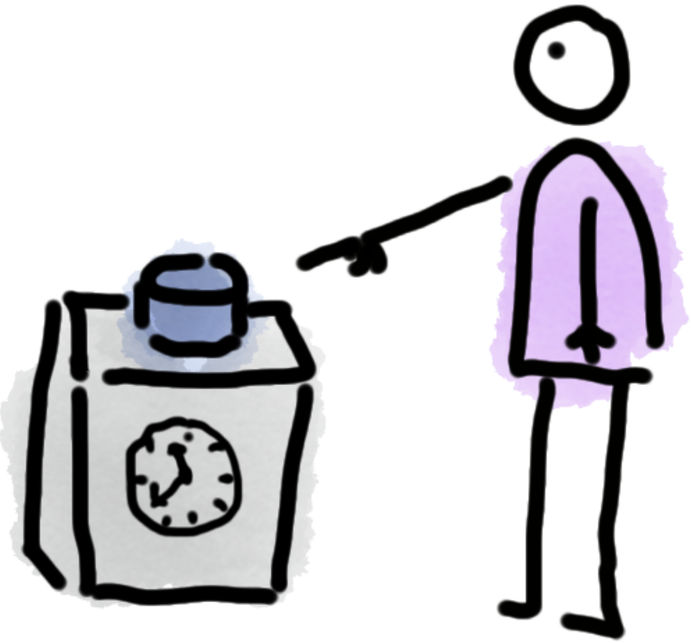


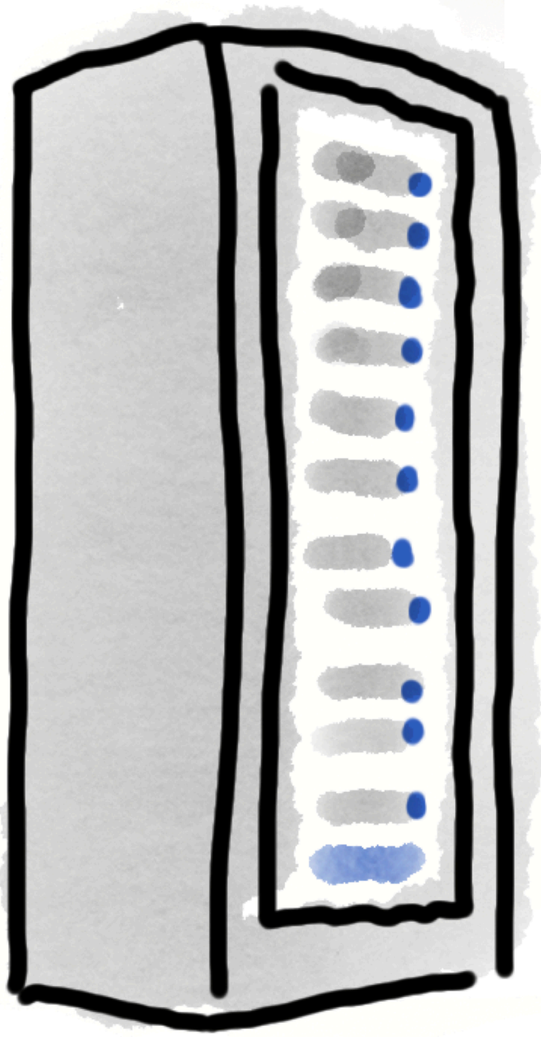


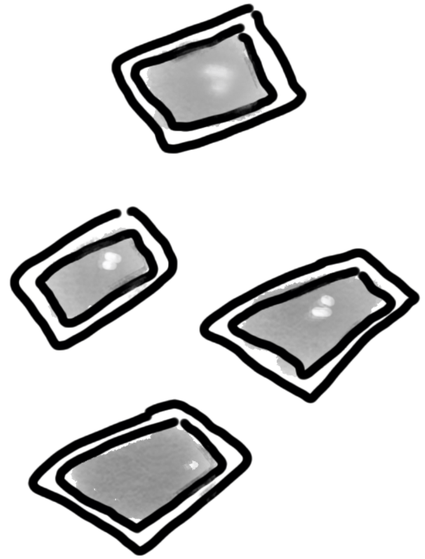


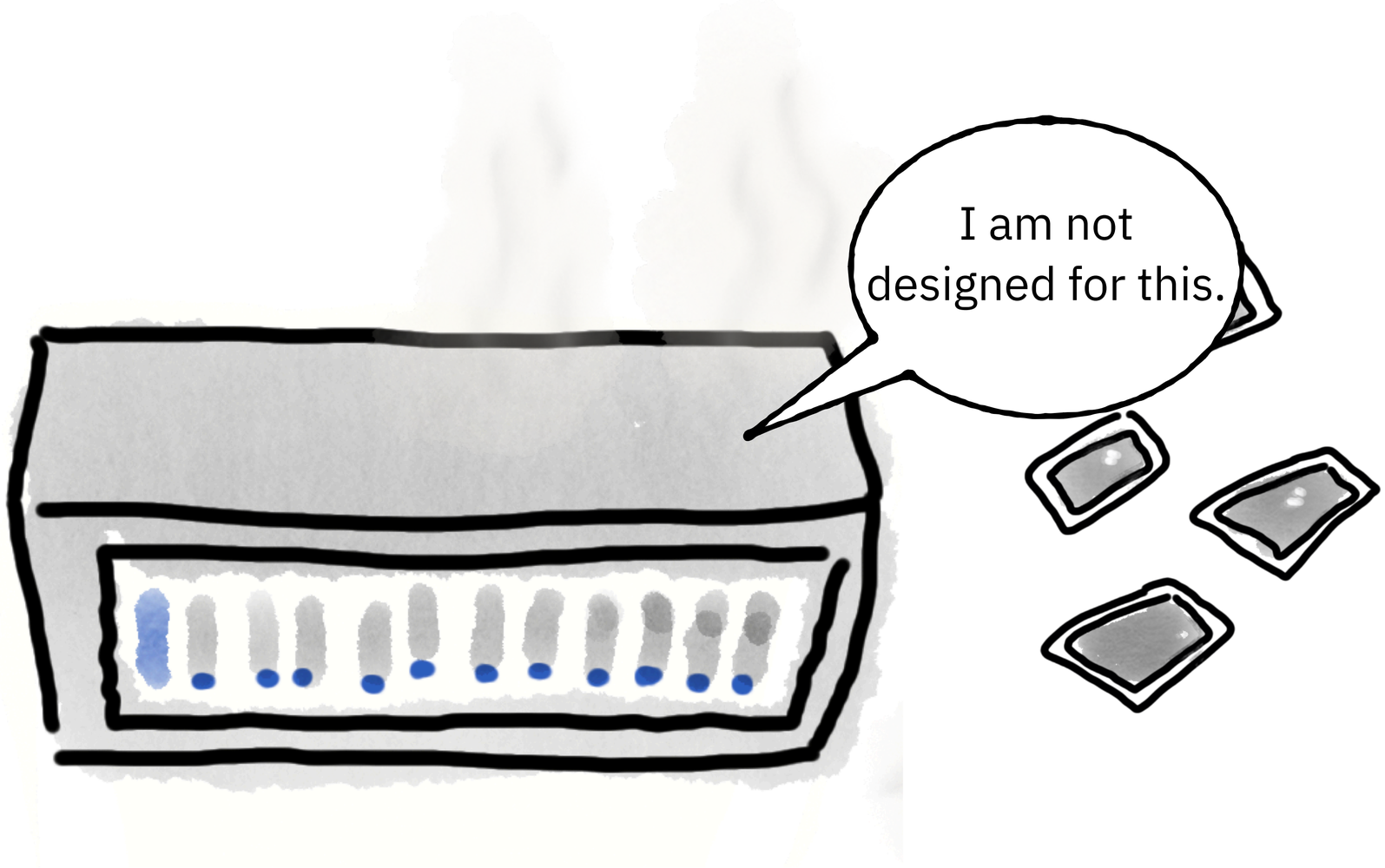






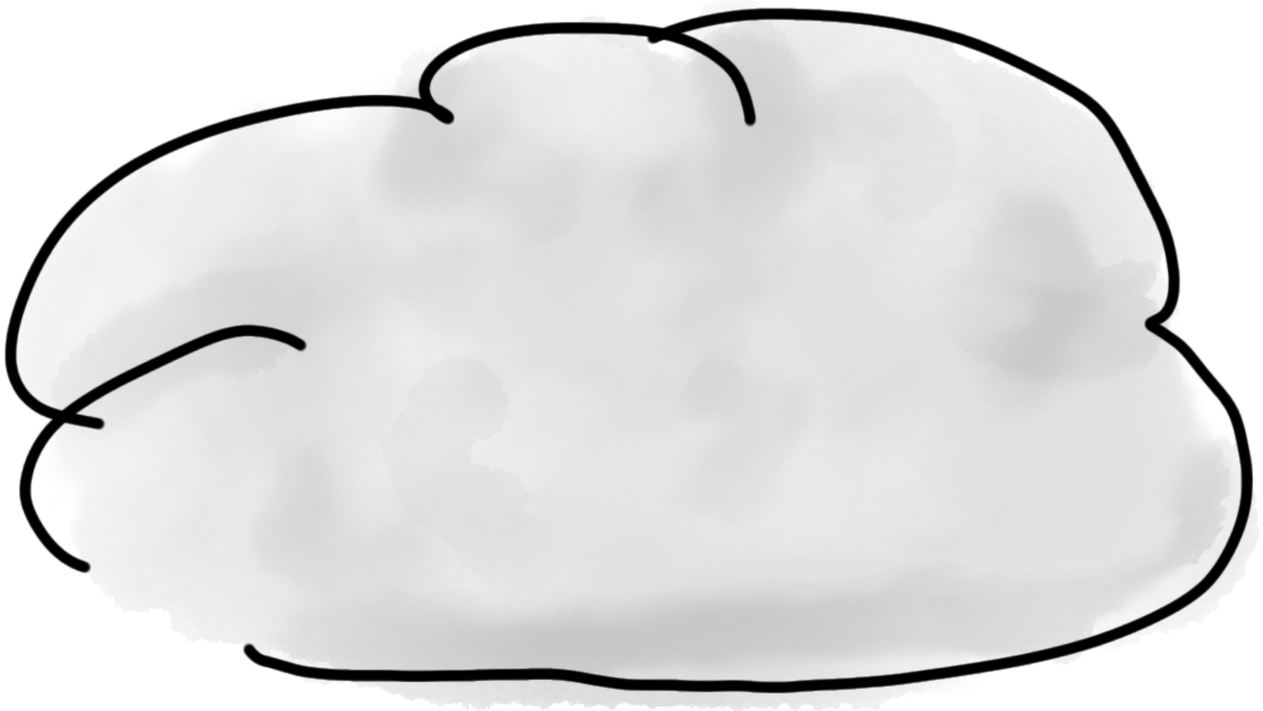






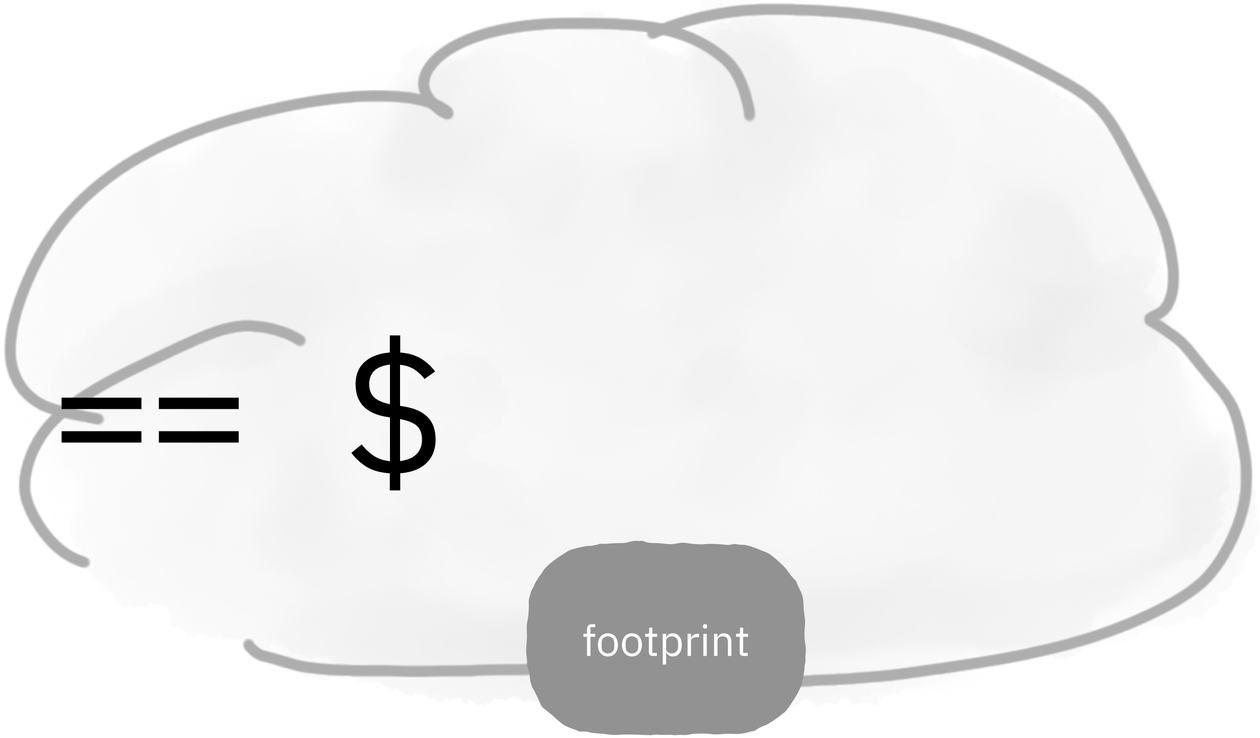
@holly_cummins

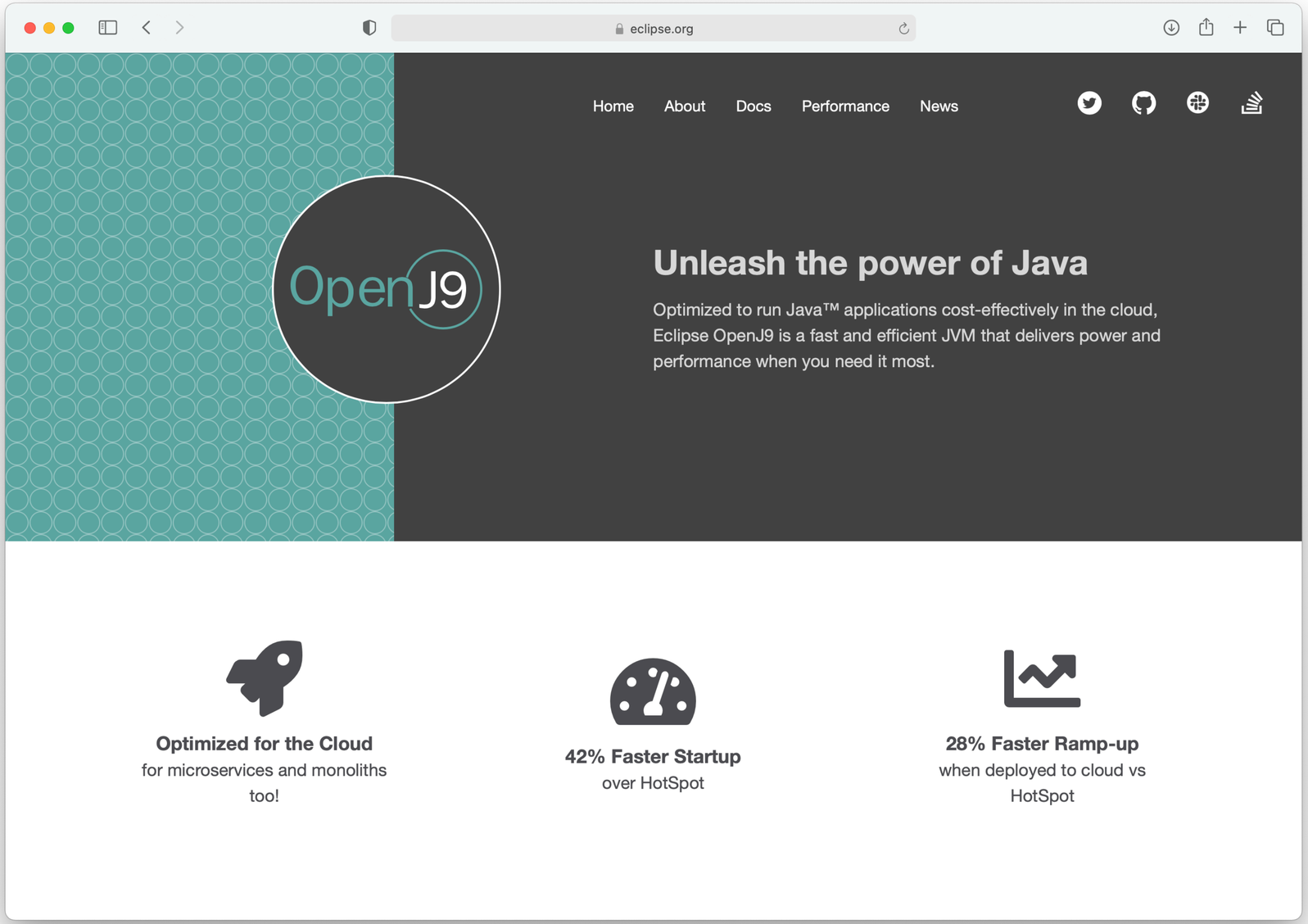
the world changes



-Xmx == $

-Xmx == $

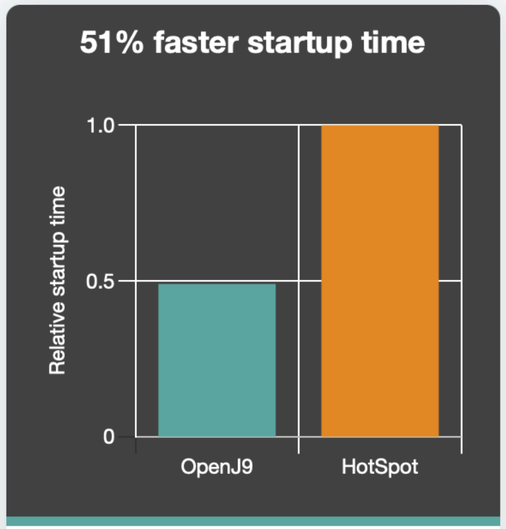


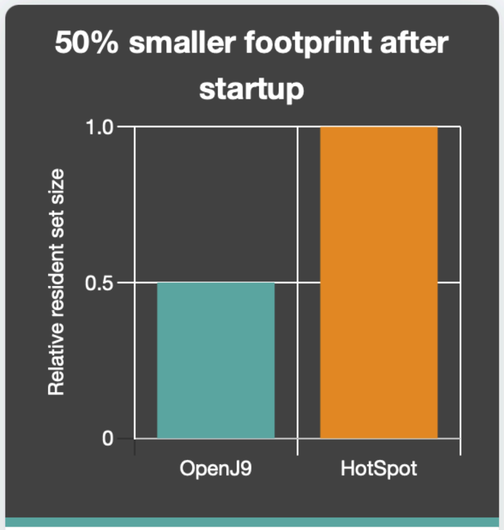

eclipse.org
Home
About
Docs
Performance
News
OpenJ9
Unleash the power of Java
Optimized to run Java™ applications cost-effectively in the cloud, Eclipse OpenJ9 is a fast and efficient JVM that delivers power and performance when you need it most.
Optimized for the Cloud
for microservices and monoliths too!
42% Faster Startup
over HotSpot
28% Faster Ramp-up
when deployed to cloud vs HotSpot

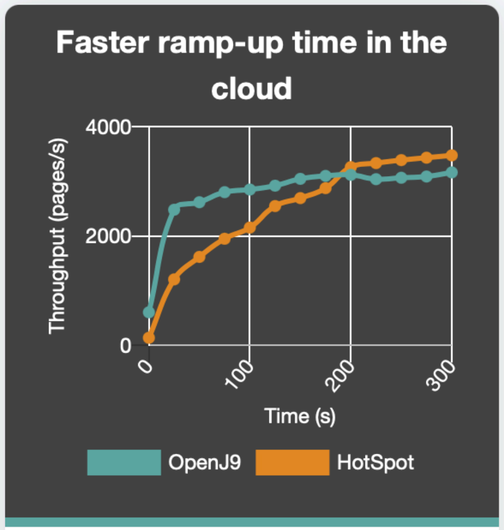
which performs better?

OpenJDK 11 performance with Eclipse OpenJ9

OpenJDK 11 with OpenJ9 significantly outperforms HotSpot on Liberty startup, ramp up, and footprint.

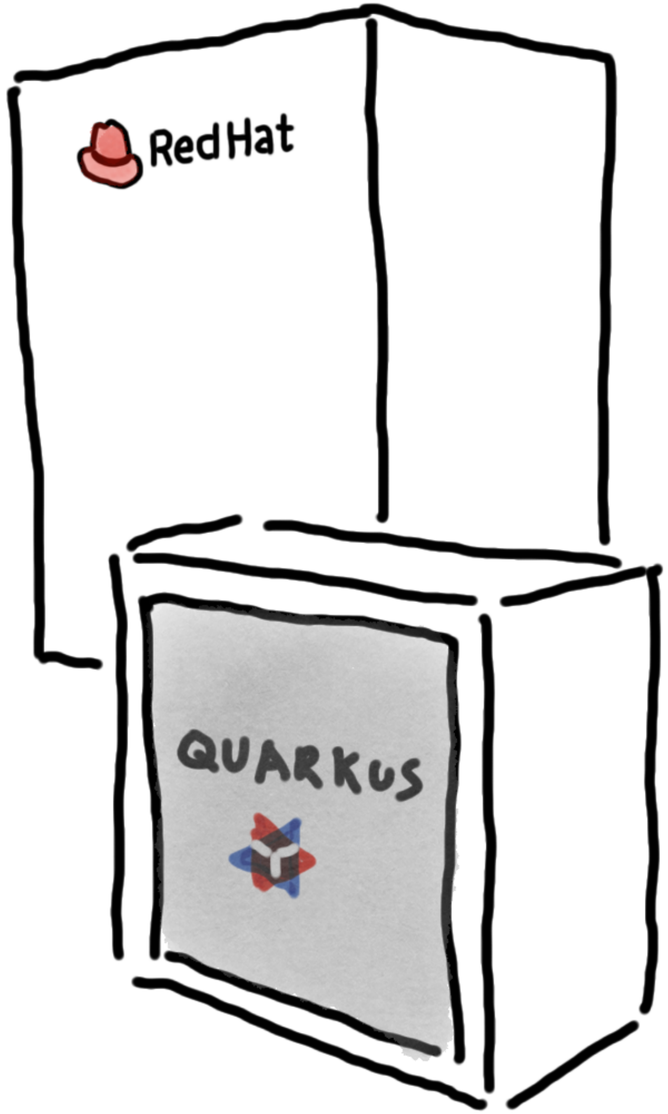






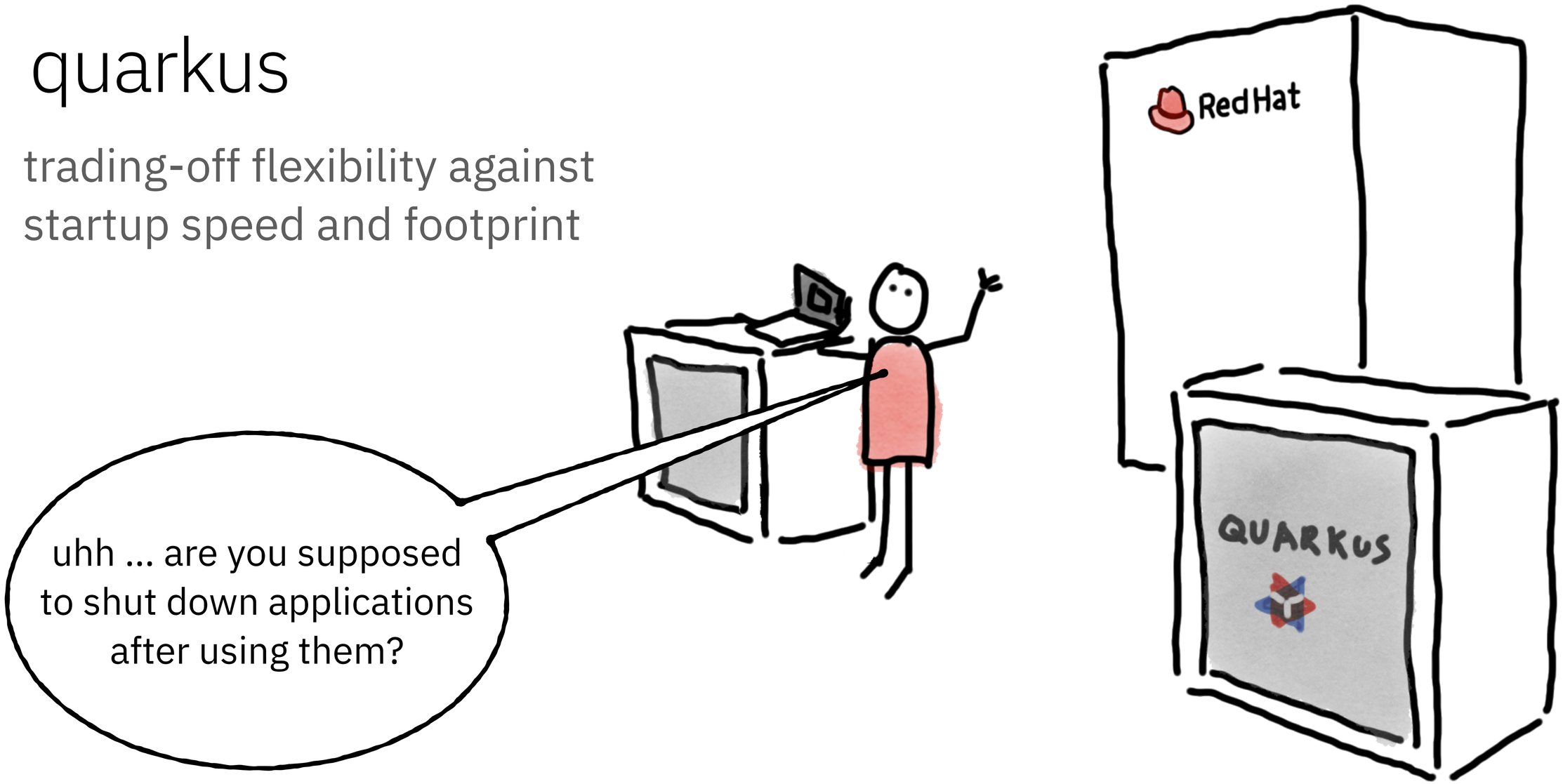
quarkus

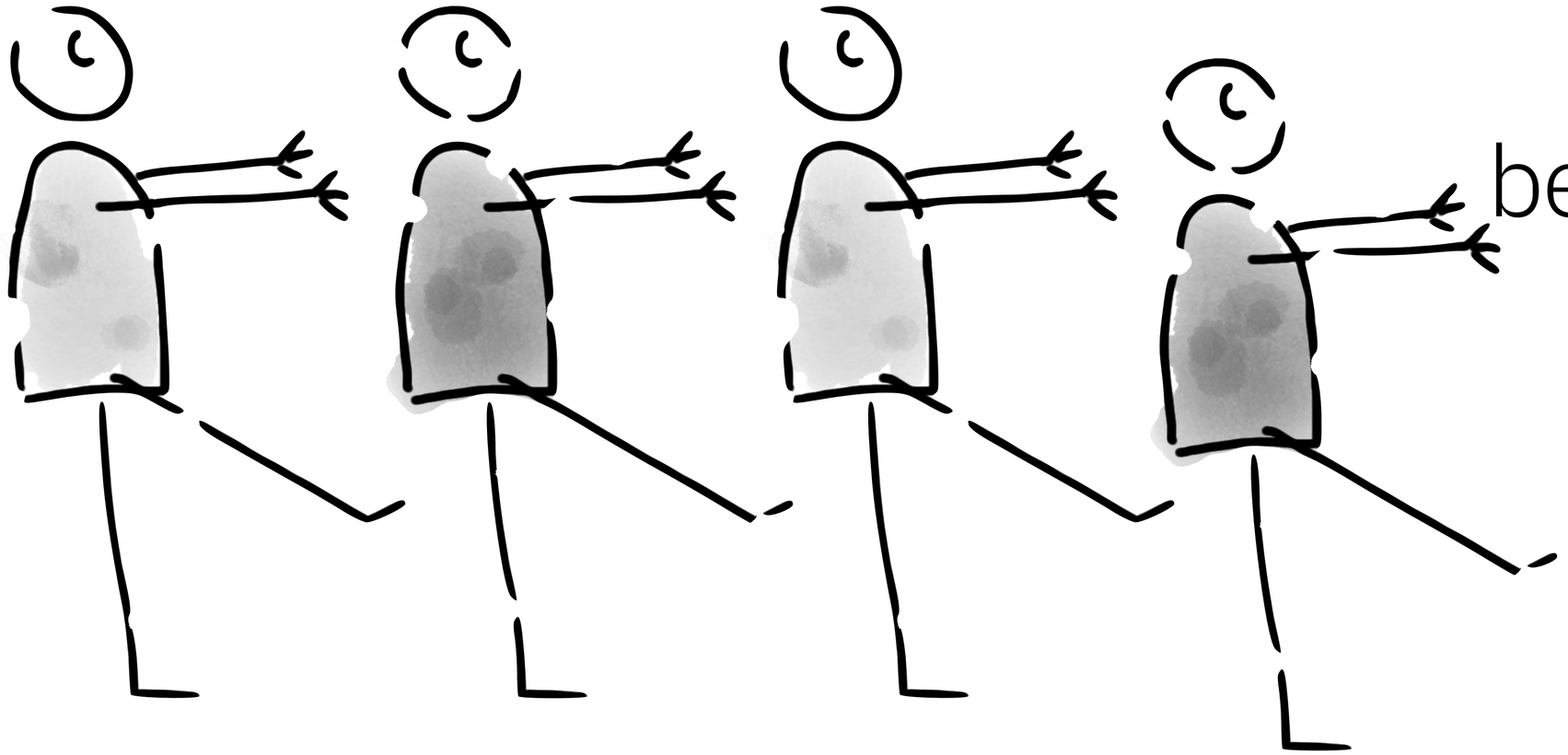
trading-off flexibility against
startup speed and footprint



quarkus

trading-off flexibility against
startup speed and footprint



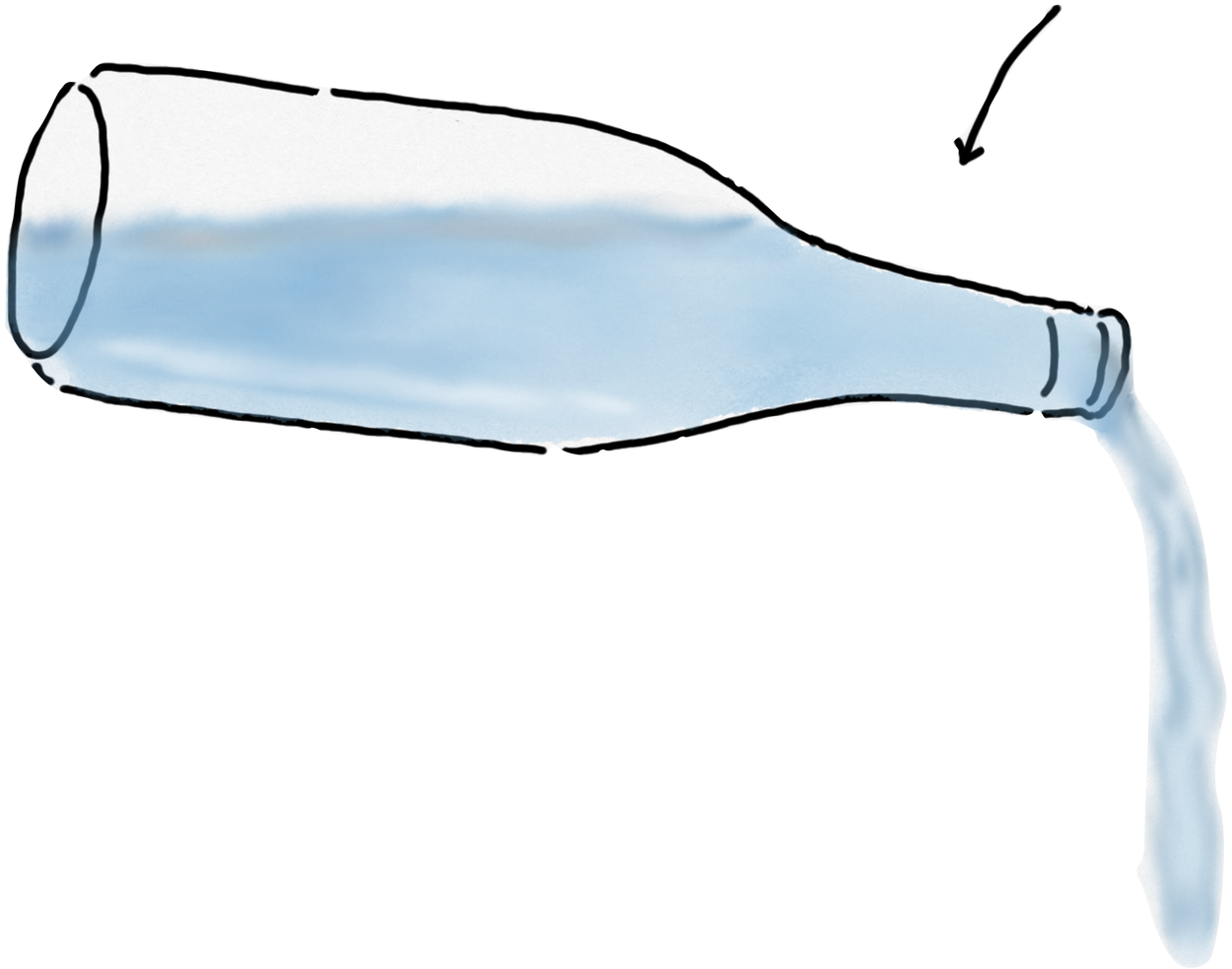


behaviour at idle

30% of VMs are zombies (antithesisgroup.com)

how to optimise?

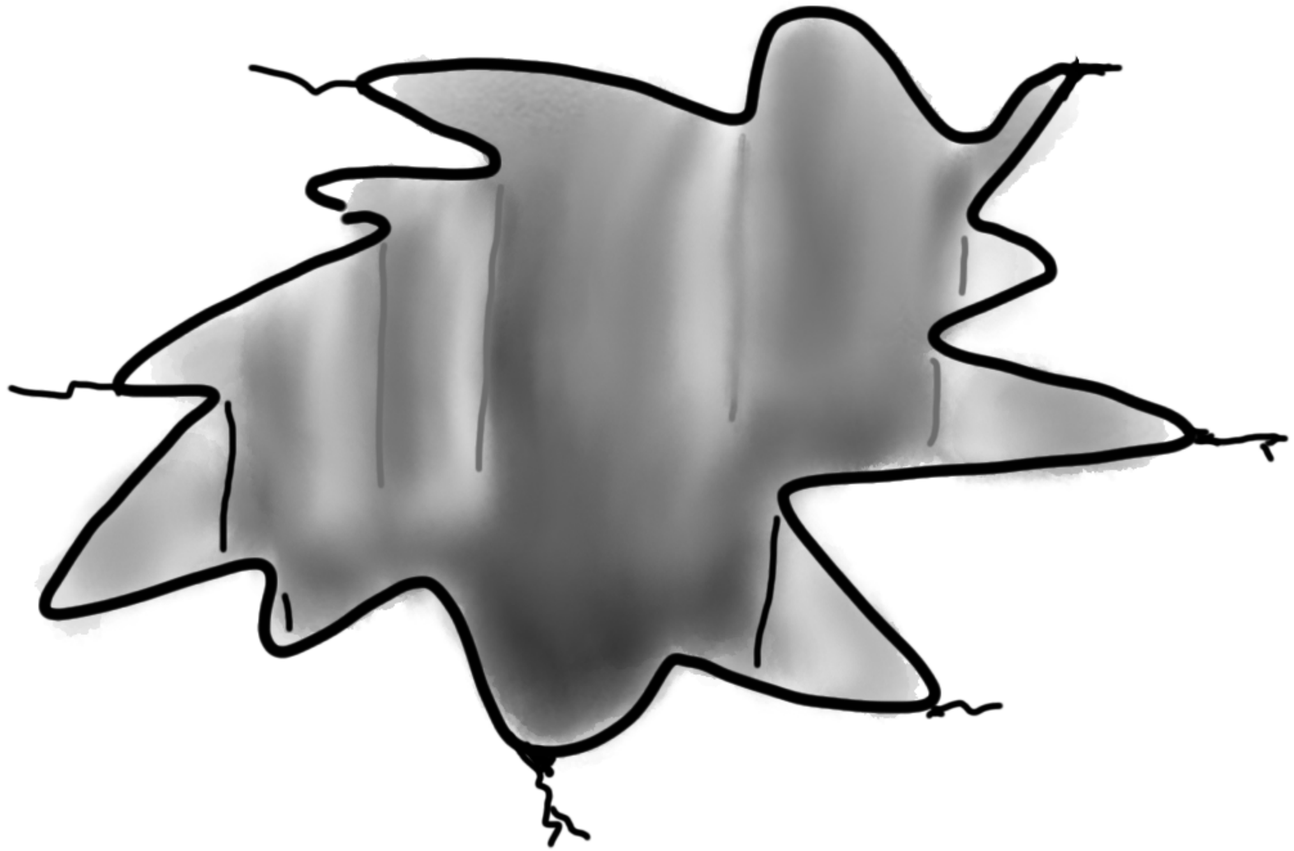
find the bottleneck.



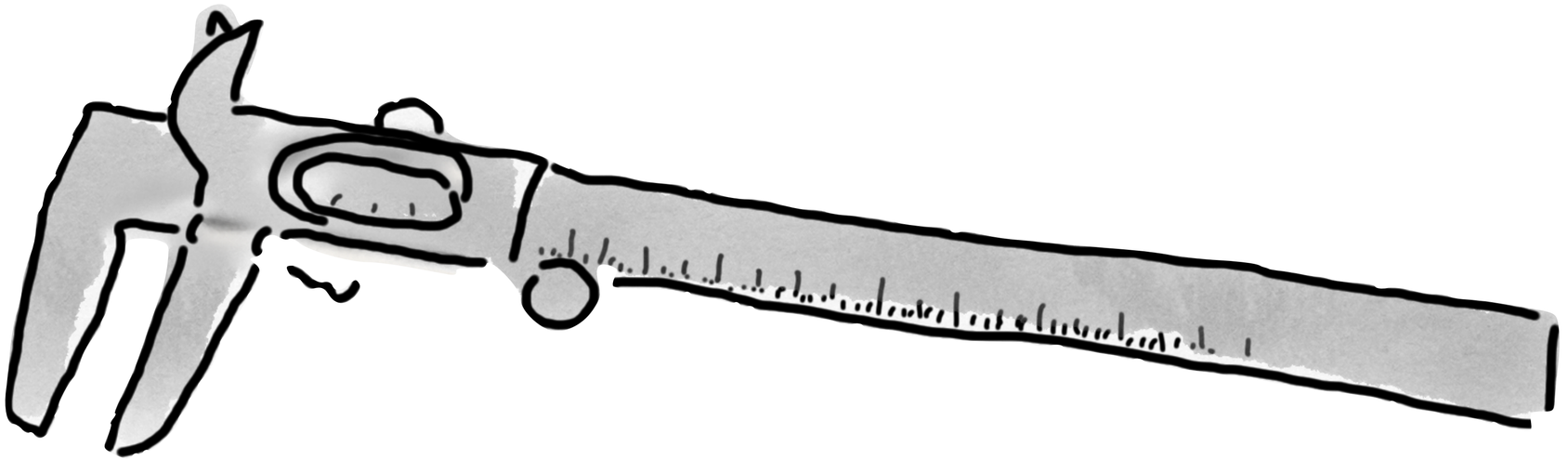
fix it.

pitfall 1

intuition



this is **not** the place for ideas



measure, don’t guess.

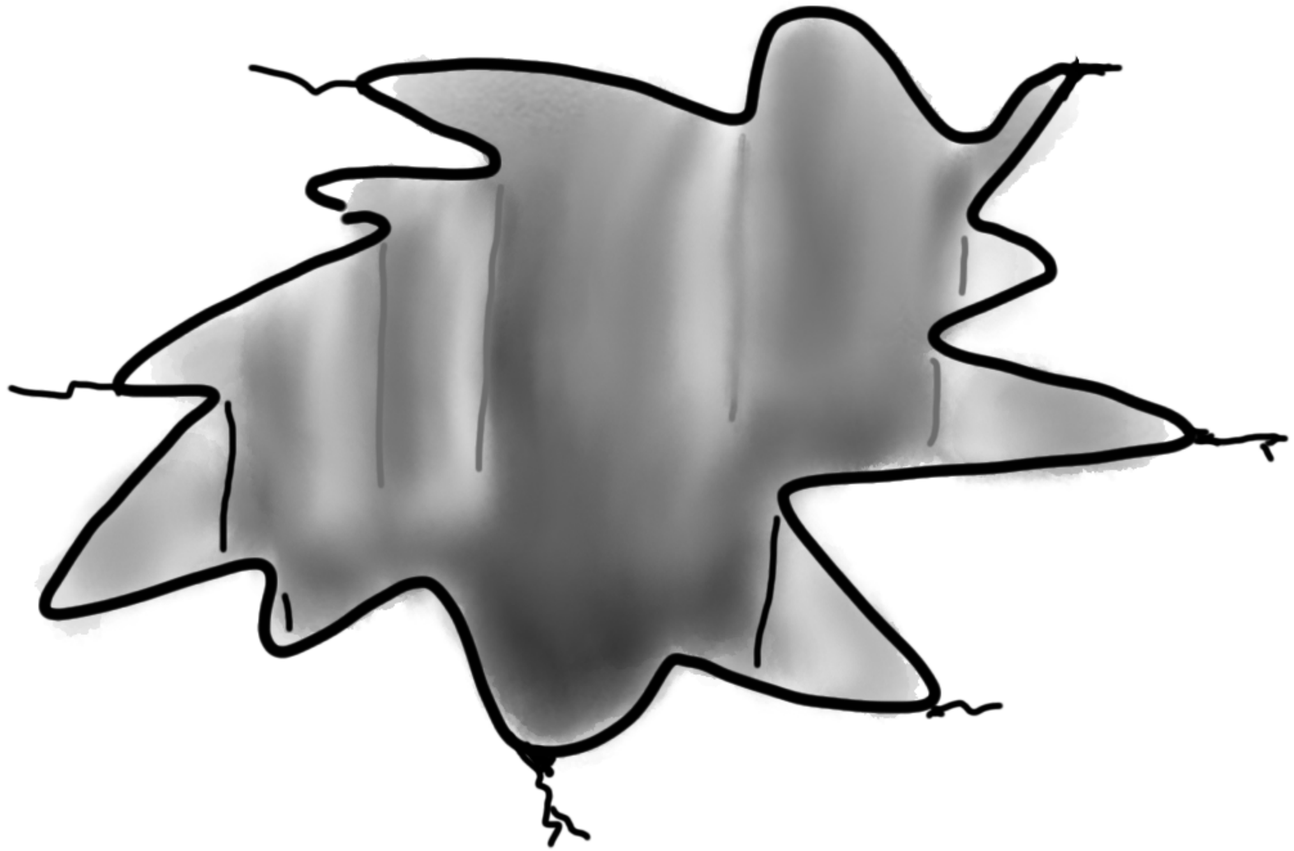
measure the **right** thing

measure the **right** thing

what do your users care about?

pitfall 2

numbers



leading indicators

leading indicators

lagging indicators

leading indicators

lagging indicators

we care about them

leading indicators

lagging indicators

we care about them
easy to measure

leading indicators

lagging indicators

we care about them
easy to measure
hard to change

leading indicators

easy to change

lagging indicators

we care about them
easy to measure
hard to change

leading indicators

predictive of a thing we care about

easy to change

lagging indicators

we care about them
easy to measure
hard to change

leading indicators

predictive of a thing we care about
hard to identify
easy to change

lagging indicators

we care about them
easy to measure
hard to change

leading indicators

predictive of a thing we care about
hard to identify
easy to change

lagging indicators

we care about them
easy to measure
hard to change

caution:

performance experiments for entertainment purposes only.

do not try these at home.

2007

Six Myths and Paradoxes of Garbage Collection

Holly Cummins
Kellogg College
University of Oxford

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering

April 17, 2007

bad-ish advice: “reduce time spent in garbage collection”

bad-ish advice: “reduce time spent in garbage collection”

actually, garbage collection can make your application go **faster**

2007

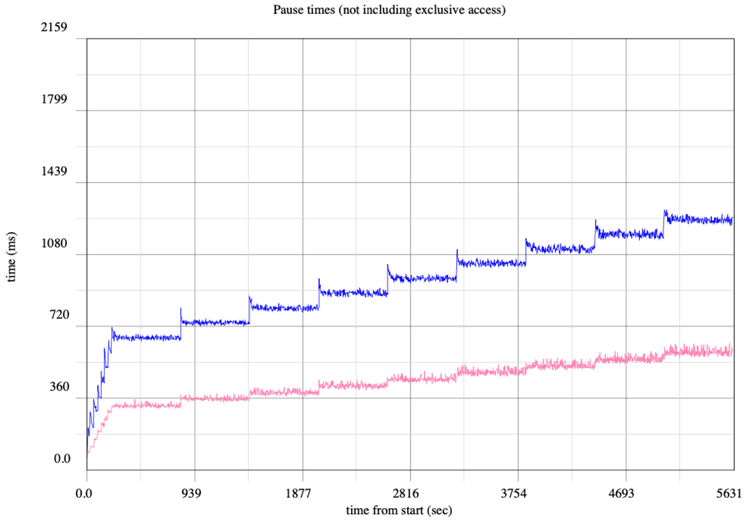


Figure 6.8: Pause times for the two SPECjbb2005 runs shown in figure 6.7. In one run, the collector was forced to compact every collection, resulting in significantly longer pause times. With forced compaction, there were fewer collections, so the total pause time is only 33% greater.

2007

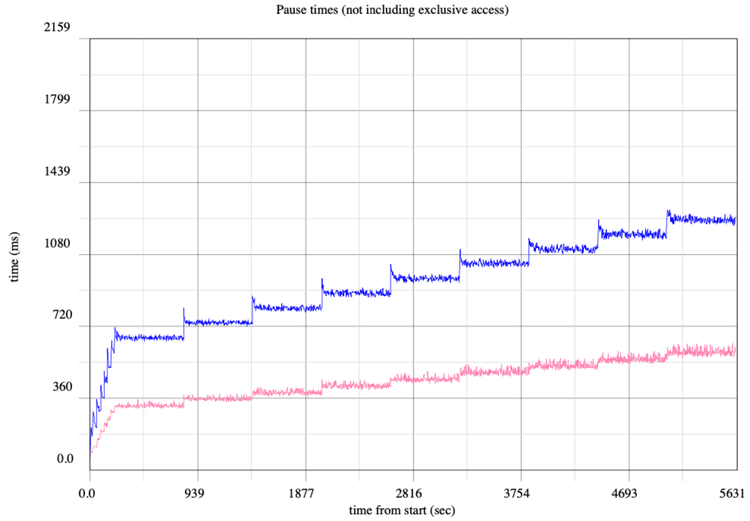


Figure 6.8: Pause times for the two SPECjbb2005 runs shown in figure 6.7. In one run, the collector was forced to compact every collection, resulting in significantly longer pause times. With forced compaction, there were fewer collections, so the total pause time is only 33% greater.

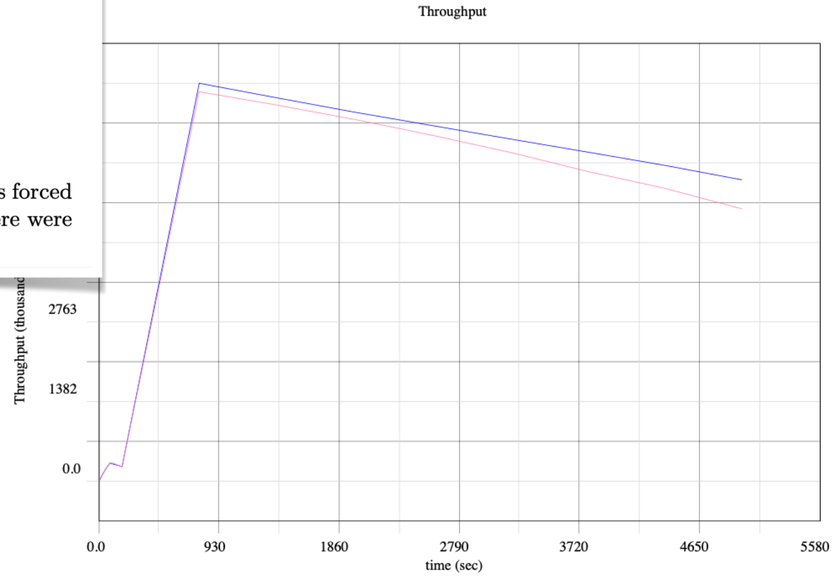
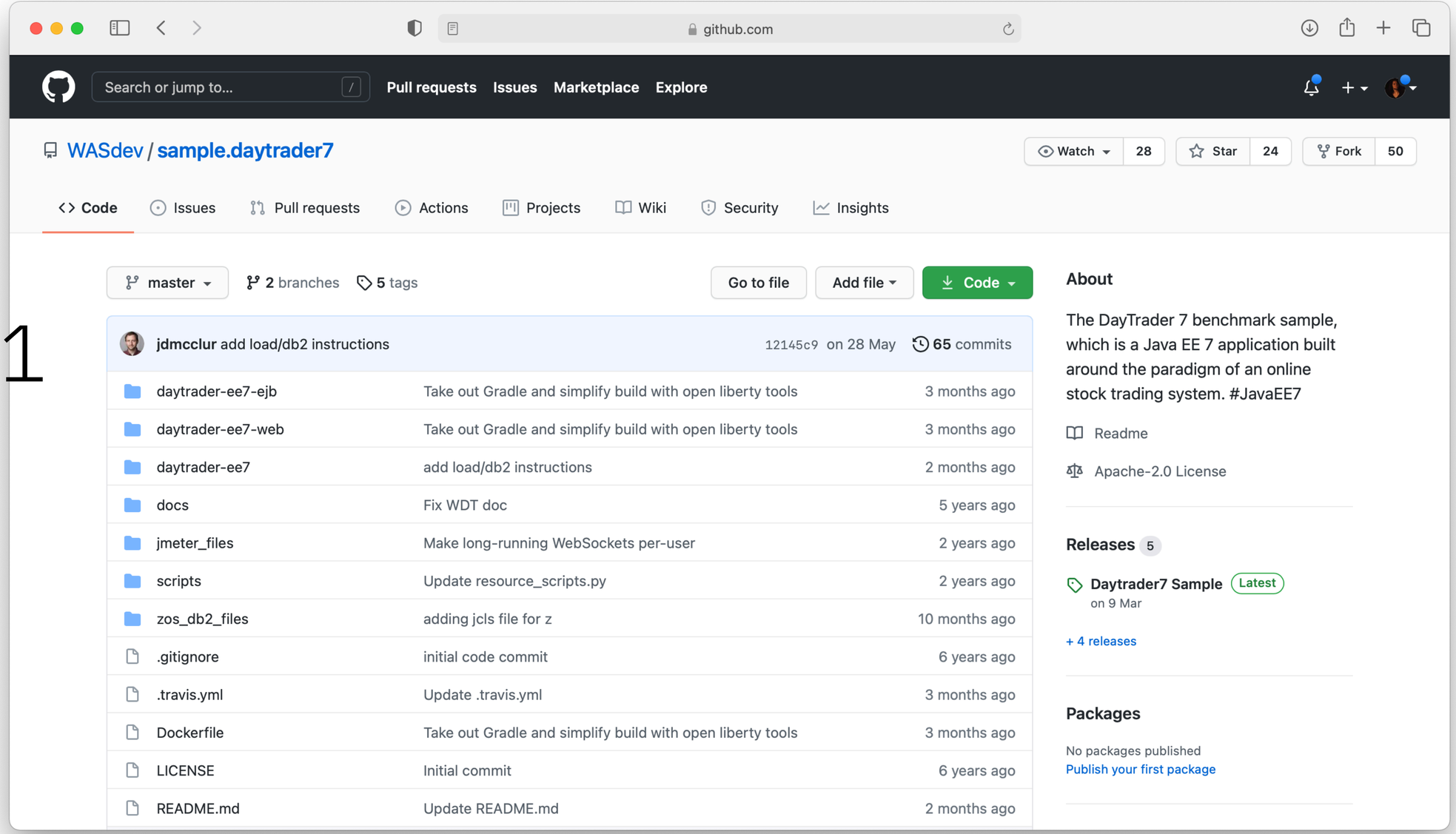
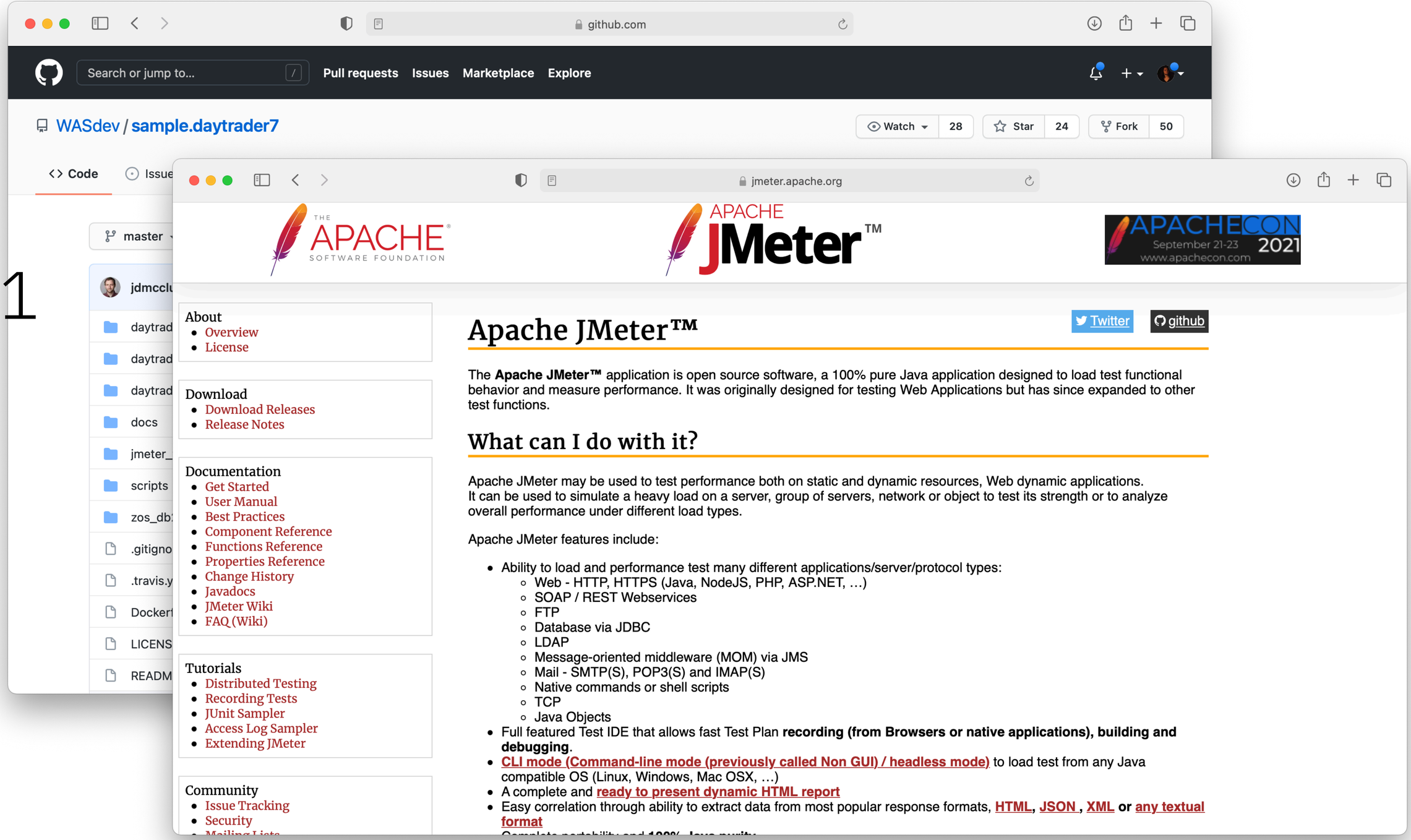


Figure 6.9: Reported throughput for the two SPECjbb2005 runs shown in figure 6.7. Forced compaction provides a mean throughput benefit of 5%.

2021


github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
WASdev / sample.daytrader7
Watch 28
Star 24
Fork 50
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
2 branches
5 tags
Go to file
Add file
Code
jdmcclur add load/db2 instructions
12145c9 on 28 May
65 commits
daytrader-ee7-ejb Take out Gradle and simplify build with open liberty tools 3 months ago
daytrader-ee7-web Take out Gradle and simplify build with open liberty tools 3 months ago
daytrader-ee7 add load/db2 instructions 2 months ago
docs Fix WDT doc 5 years ago
jmeter_files Make long-running WebSockets per-user 2 years ago
scripts Update resource_scripts.py 2 years ago
zos_db2_files adding jcls file for z 10 months ago
.gitignore initial code commit 6 years ago
.travis.yml Update .travis.yml 3 months ago
Dockerfile Take out Gradle and simplify build with open liberty tools 3 months ago
LICENSE Initial commit 6 years ago
README.md Update README.md 2 months ago
About
The DayTrader 7 benchmark sample, which is a Java EE 7 application built around the paradigm of an online stock trading system. #JavaEE7
Readme
Apache-2.0 License
Releases 5
Daytrader7 Sample Latest
on 9 Mar
+ 4 releases
Packages
No packages published
Publish your first package

2021


github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
WASdev / sample.daytrader7
Watch 28
Star 24
Fork 50
Code
Issue
master
jmeter.apache.org
THE APACHE SOFTWARE FOUNDATION
APACHE JMeter™
APACHECON September 21-23 2021 www.apachecon.com
About
Overview
License
Download
Download Releases
Release Notes
Documentation
Get Started
User Manual
Best Practices
Component Reference
Functions Reference
Properties Reference
Change History
Javadocs
JMeter Wiki
FAQ (Wiki)
Tutorials
Distributed Testing
Recording Tests
JUnit Sampler
Access Log Sampler
Extending JMeter
Community
Issue Tracking
Security
Apache JMeter™
Twitter
github
The Apache JMeter™ application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.
What can I do with it?
Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications.
It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.
Apache JMeter features include:
Ability to load and performance test many different applications/server/protocol types:
Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
SOAP / REST Webservices
FTP
Database via JDBC
LDAP
Message-oriented middleware (MOM) via JMS
Mail - SMTP(S), POP3(S) and IMAP(S)
Native commands or shell scripts
TCP
Java Objects
Full featured Test IDE that allows fast Test Plan recording (from Browsers or native applications), building and debugging.
CLI mode (Command-line mode (previously called Non GUI) / headless mode) to load test from any Java compatible OS (Linux, Windows, Mac OSX, …)
A complete and ready to present dynamic HTML report
Easy correlation through ability to extract data from most popular response formats, HTML, JSON , XML or any textual format

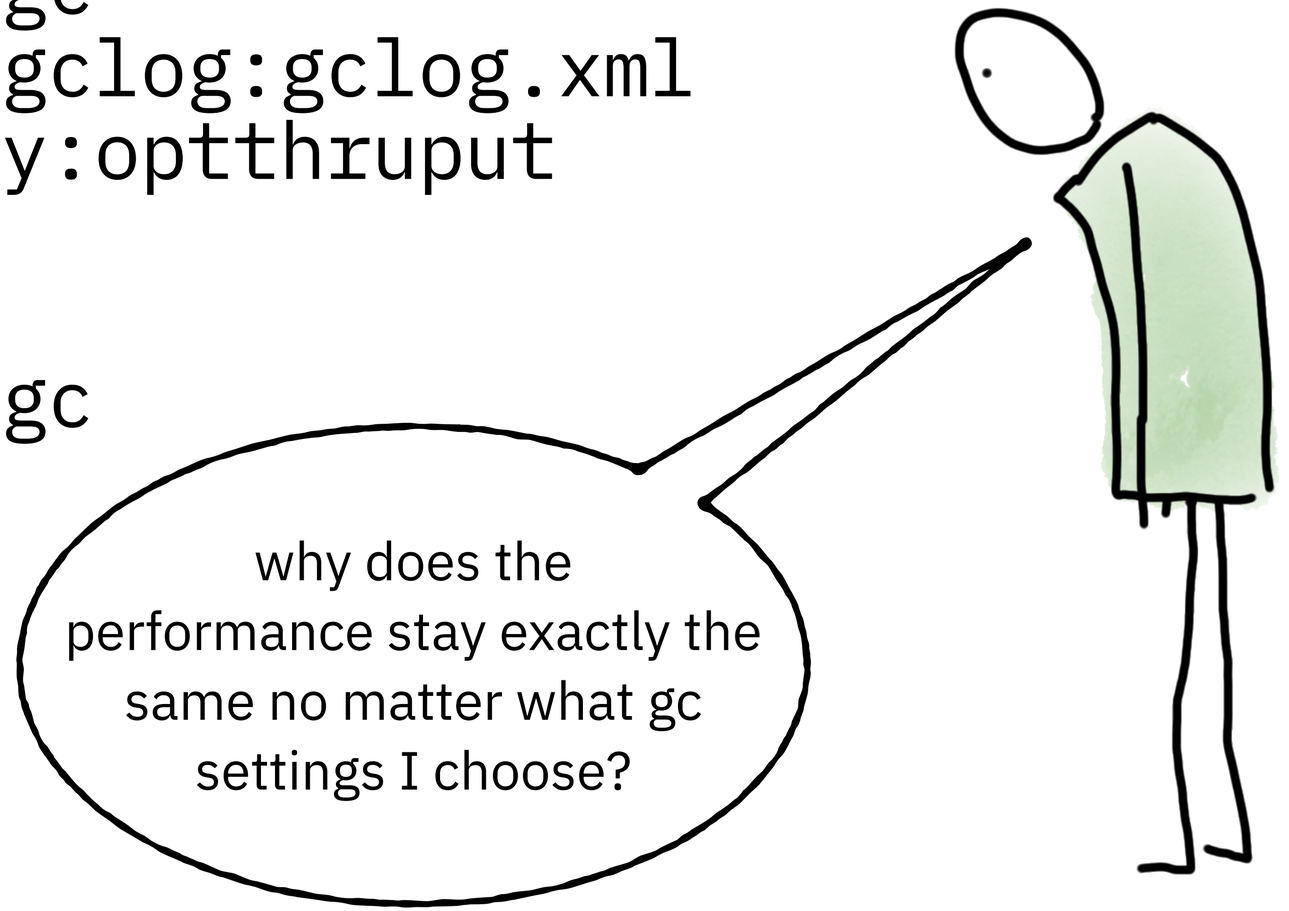
```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xcompactgc
```

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
-Xcompactgc
```

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
-Xmx110m
-Xms110m
-Xnocompactgc
```

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
-Xmx160m
-Xms160m
-Xnocompactgc
```

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
-Xmx300m
-Xms300m
-Xcompactgc
```



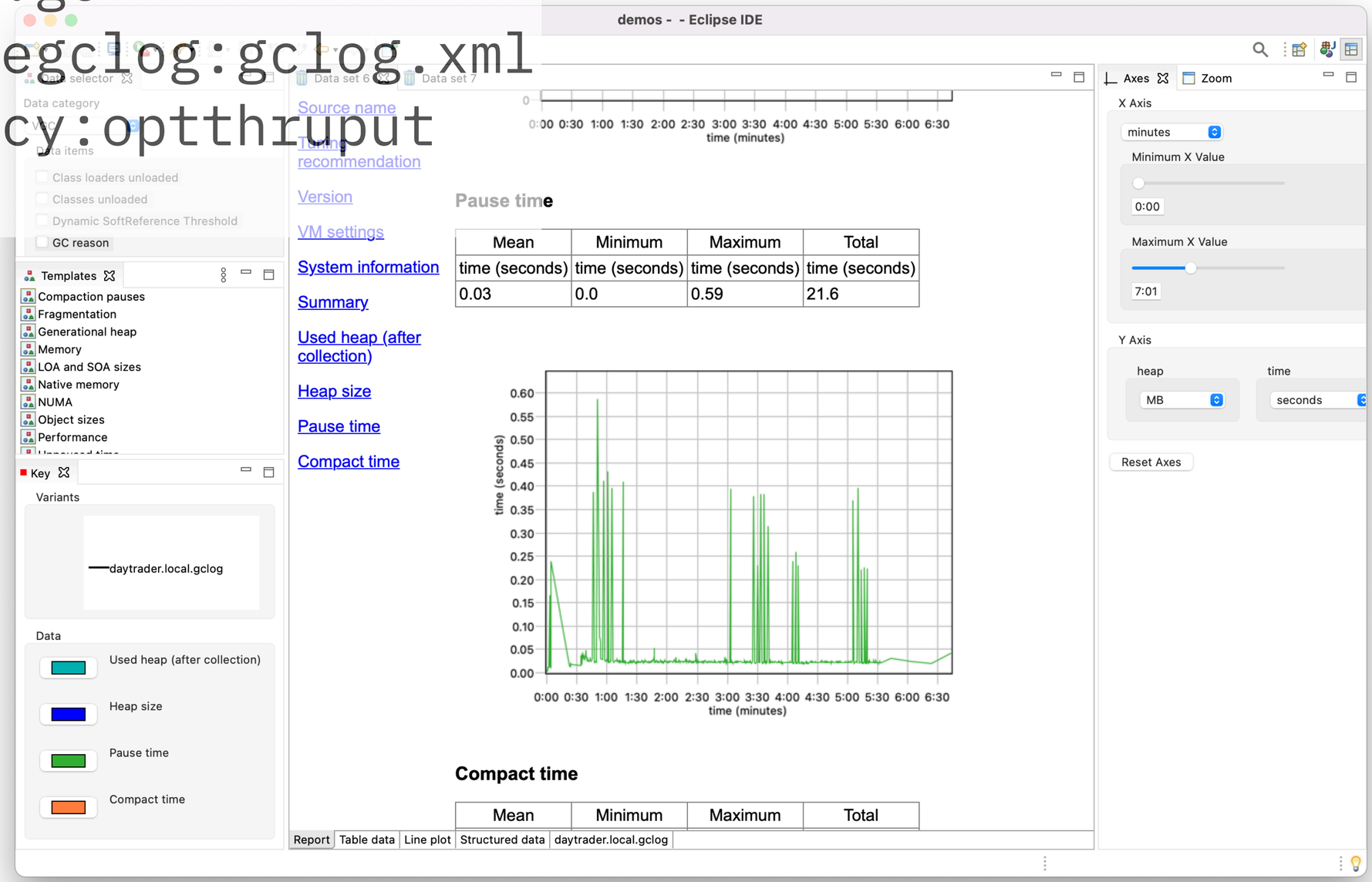
by the way, this is cheating.

(remember the ‘bad science’?)

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

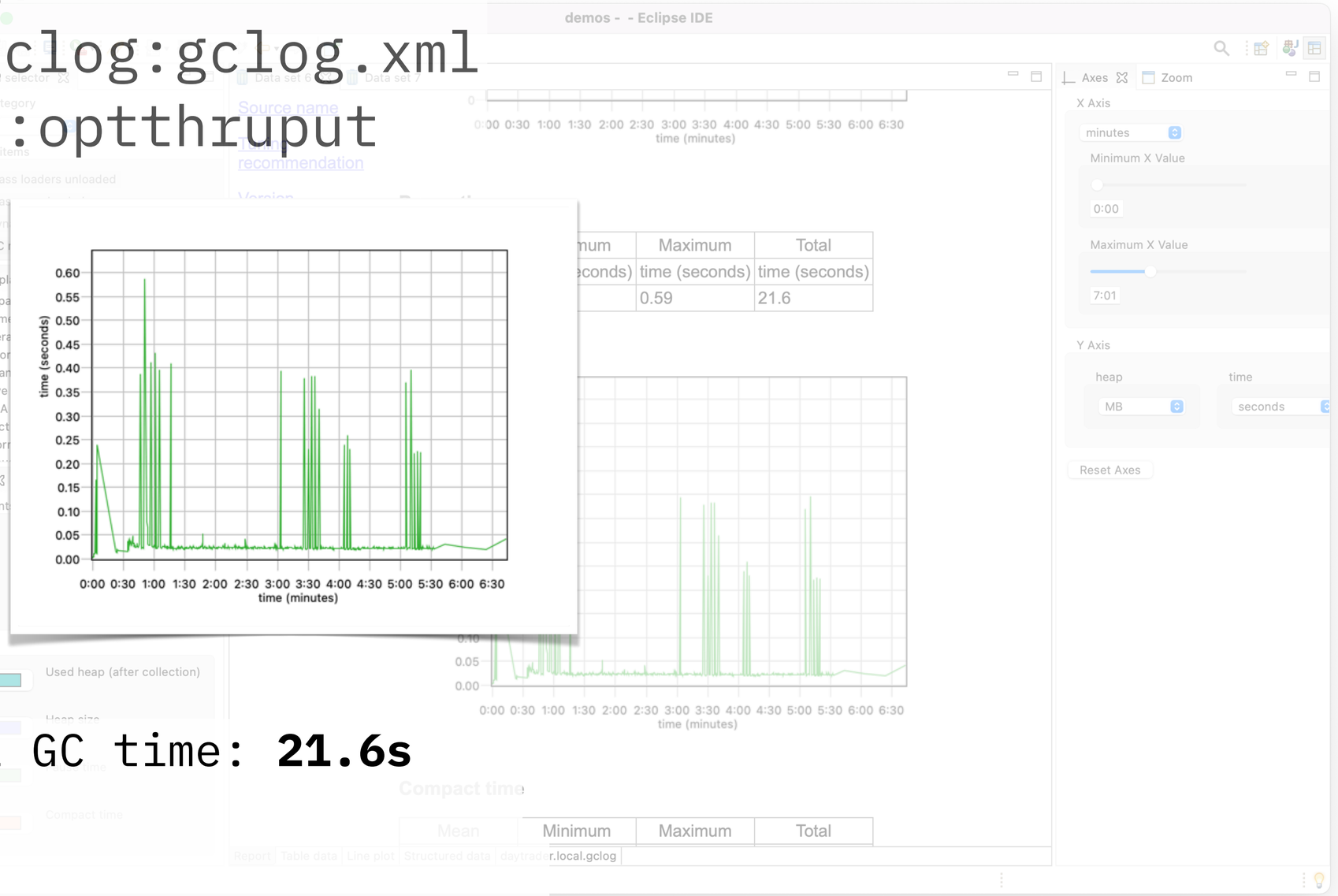
```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

tool: GCMV



tool: GCMV

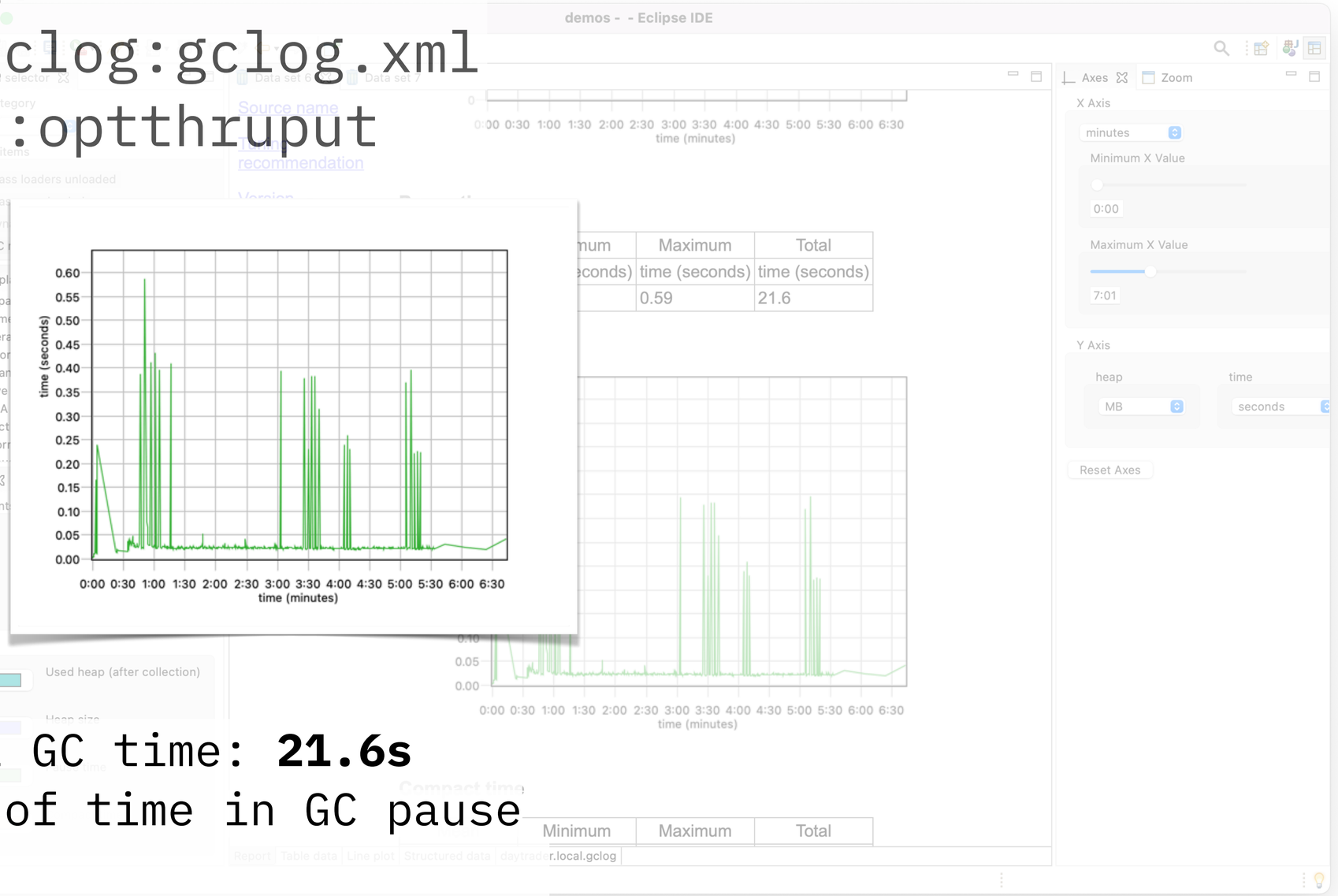
```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```



total GC time: **21.6s**

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

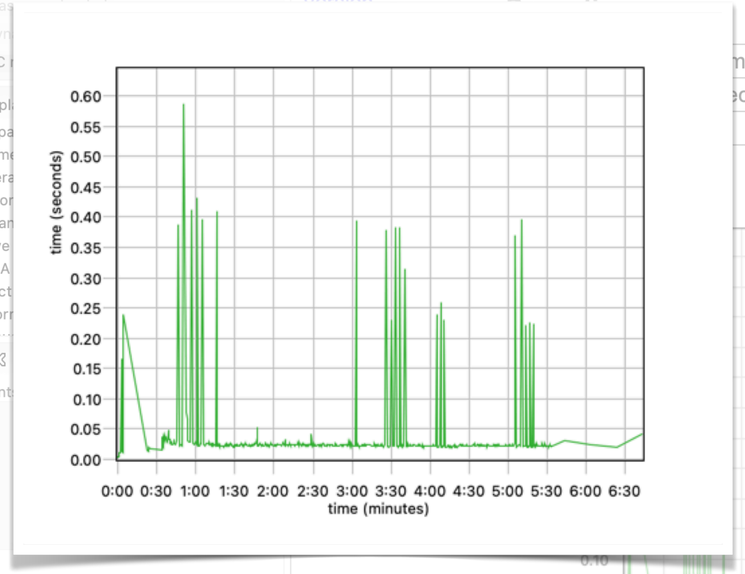


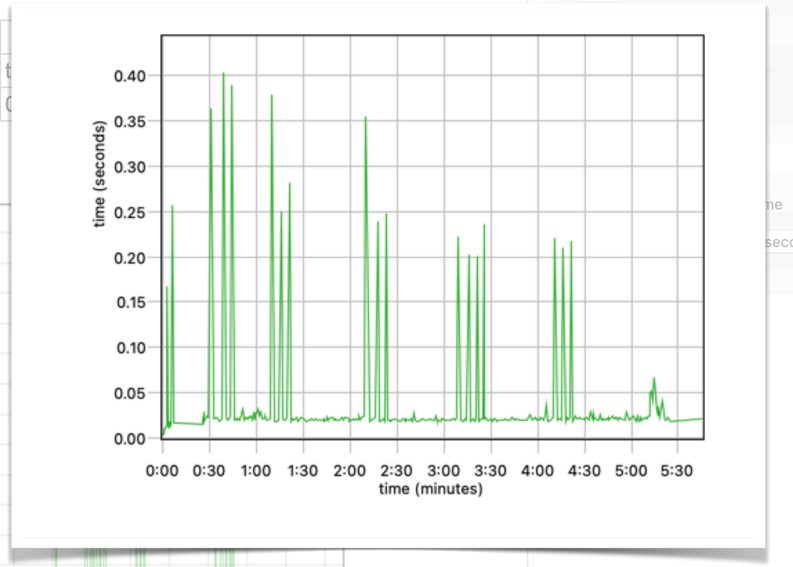
total GC time: **21.6s**

4.1% of time in GC pause

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```





total GC time: **21.6s**
4.1% of time in GC pause

total GC time: **12.0s**

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

total GC time: **21.6s**
4.1% of time in GC pause

total GC time: **12.0s**
3.6% of time in GC pause

tool: GCMV

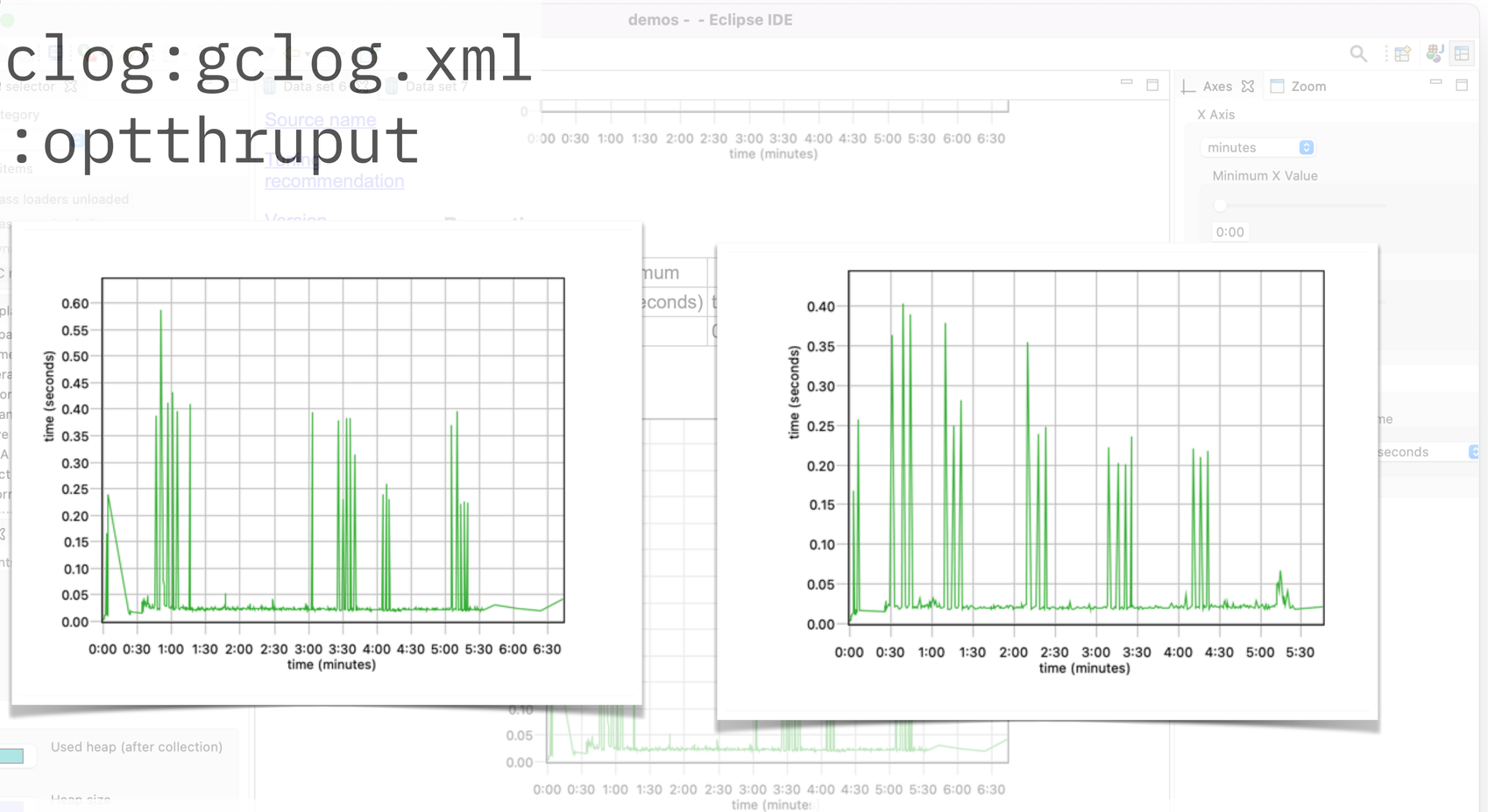
```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected

total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

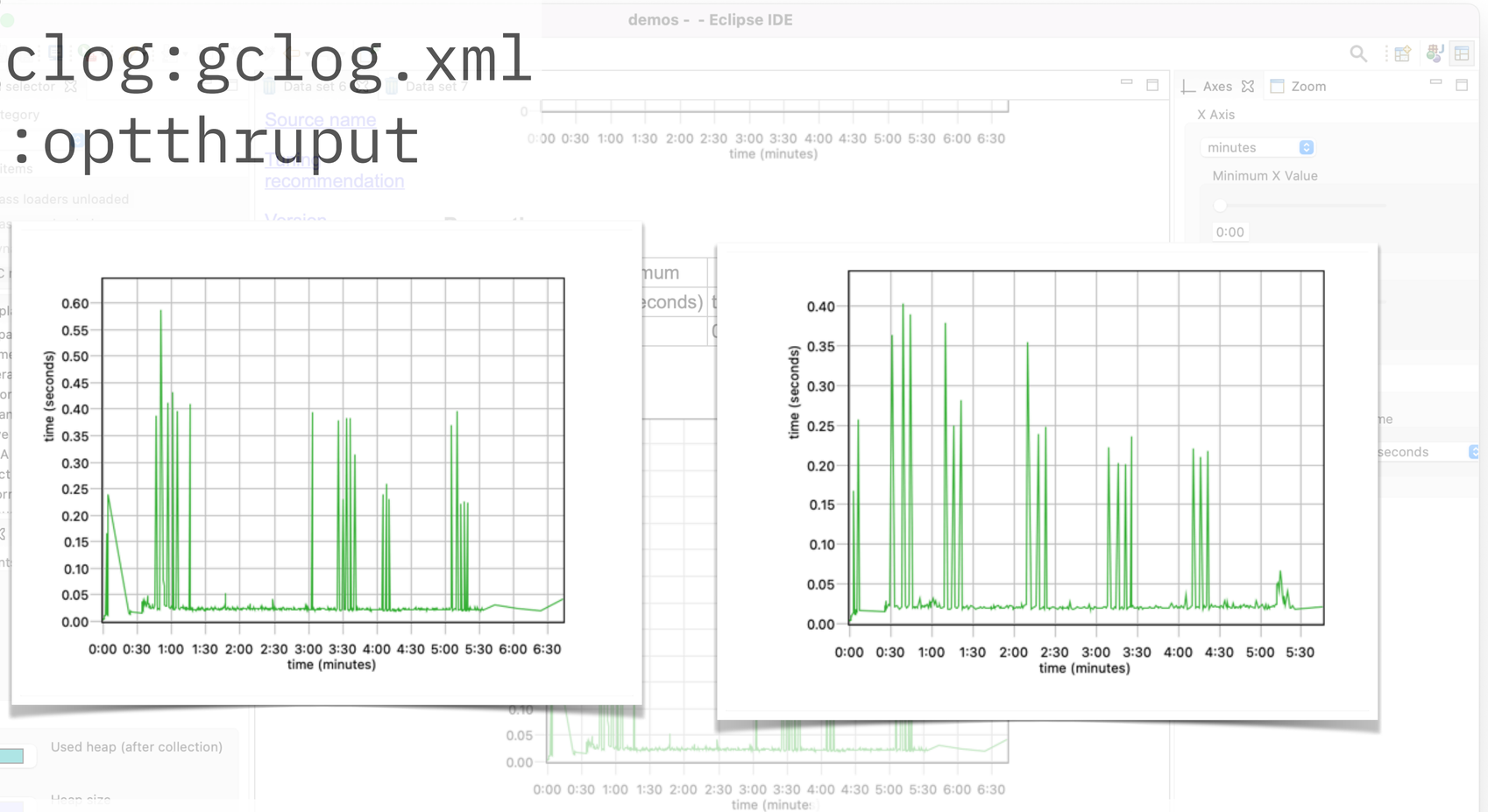


total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s

total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```

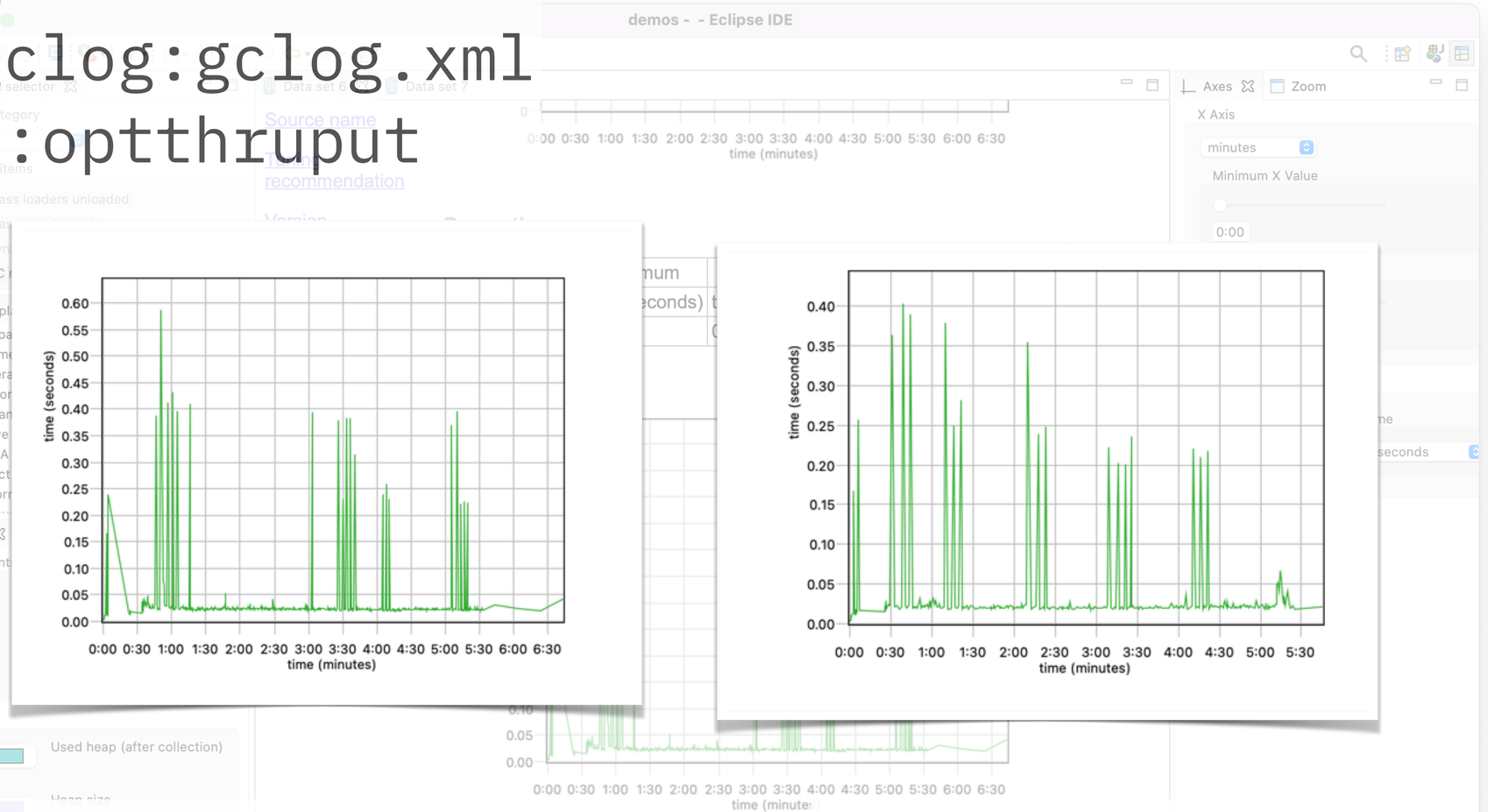


total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s

total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

tool: GCMV

```
-verbose:gc
-Xverbosegclog:gclog.xml
-Xgcpolicy:optthruput
```



total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s

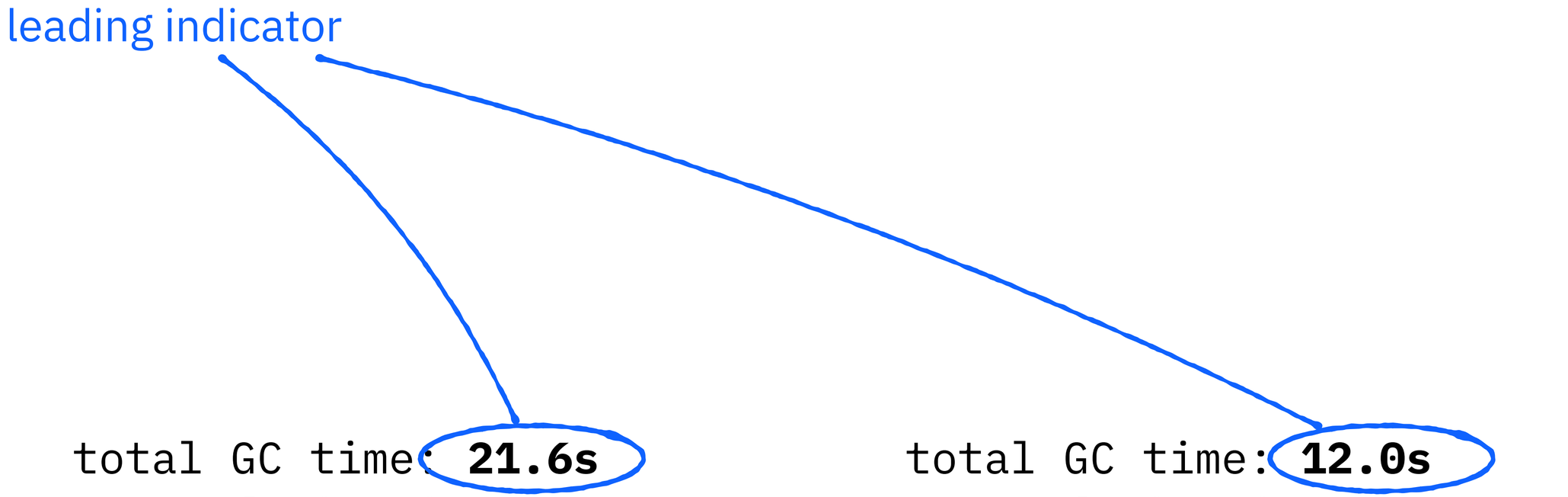
total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

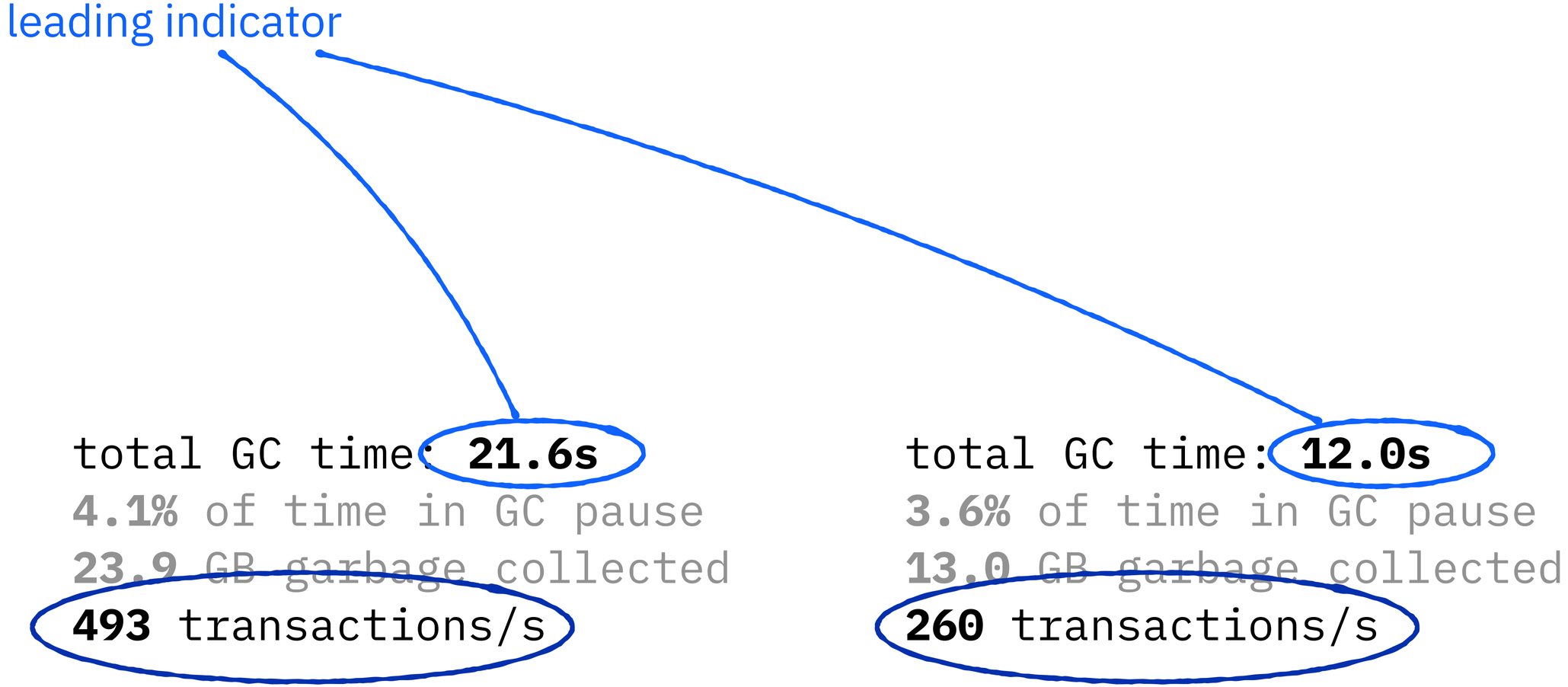
total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s

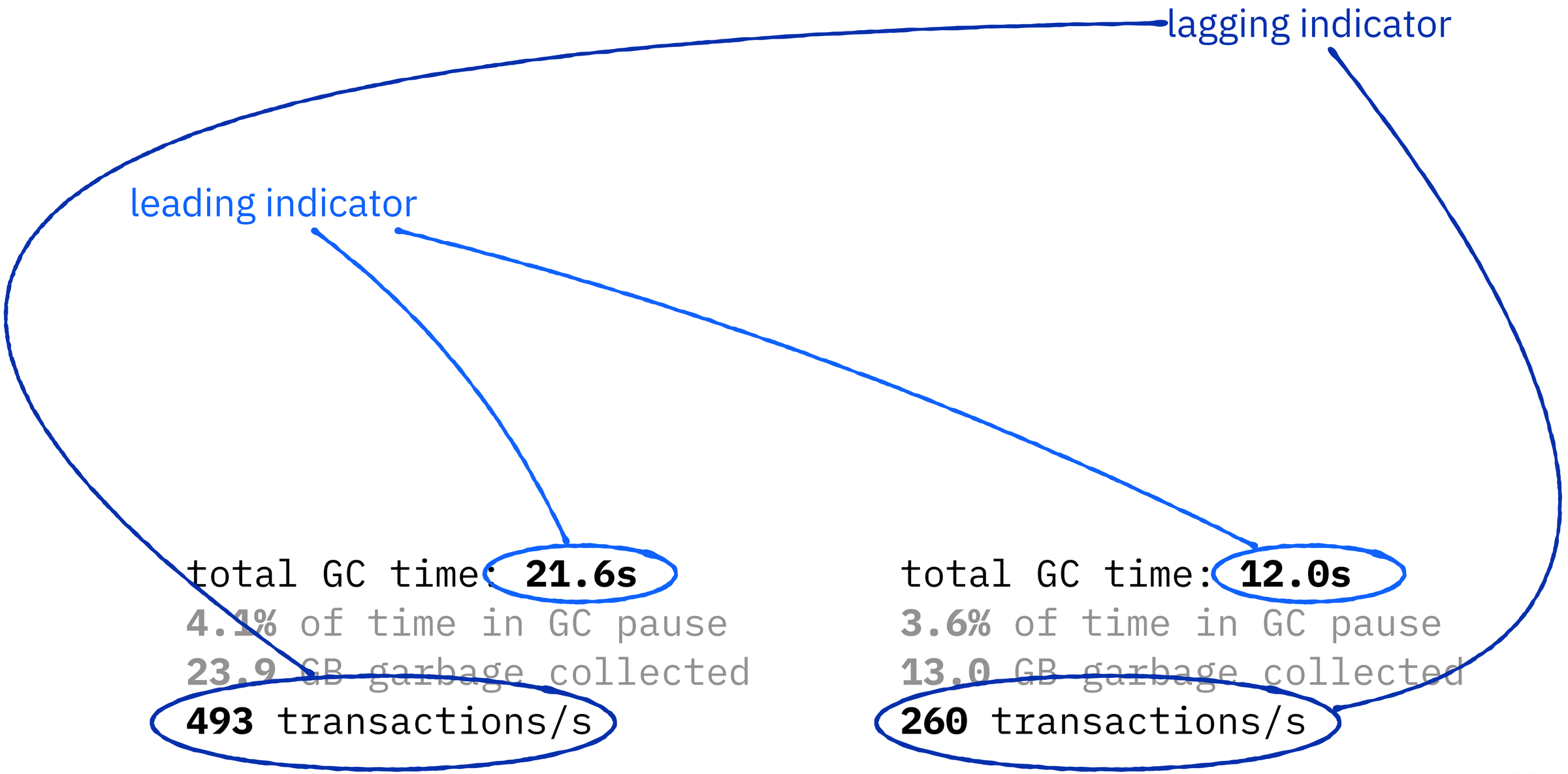
total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

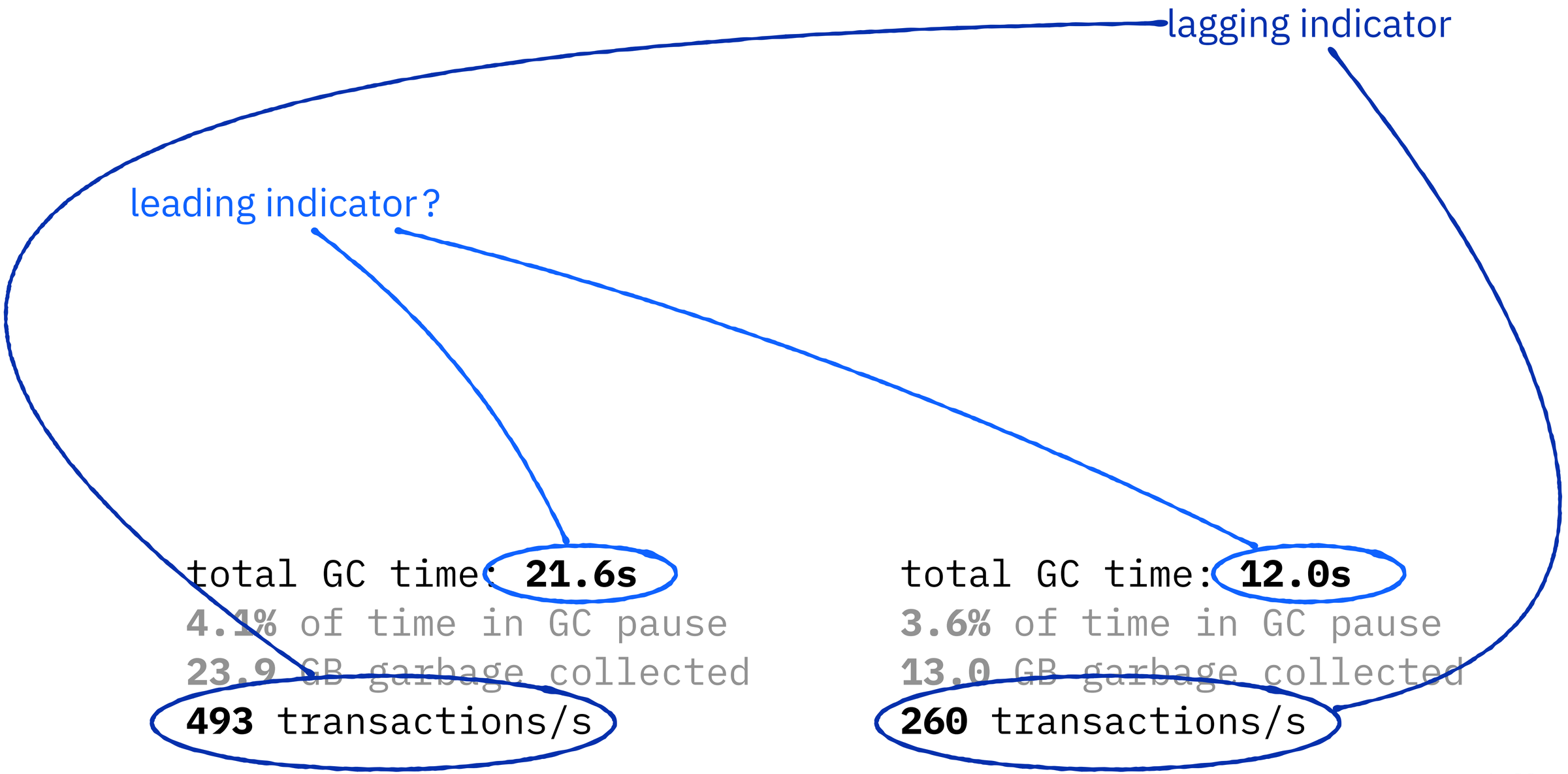
total GC time: **21.6s**
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s

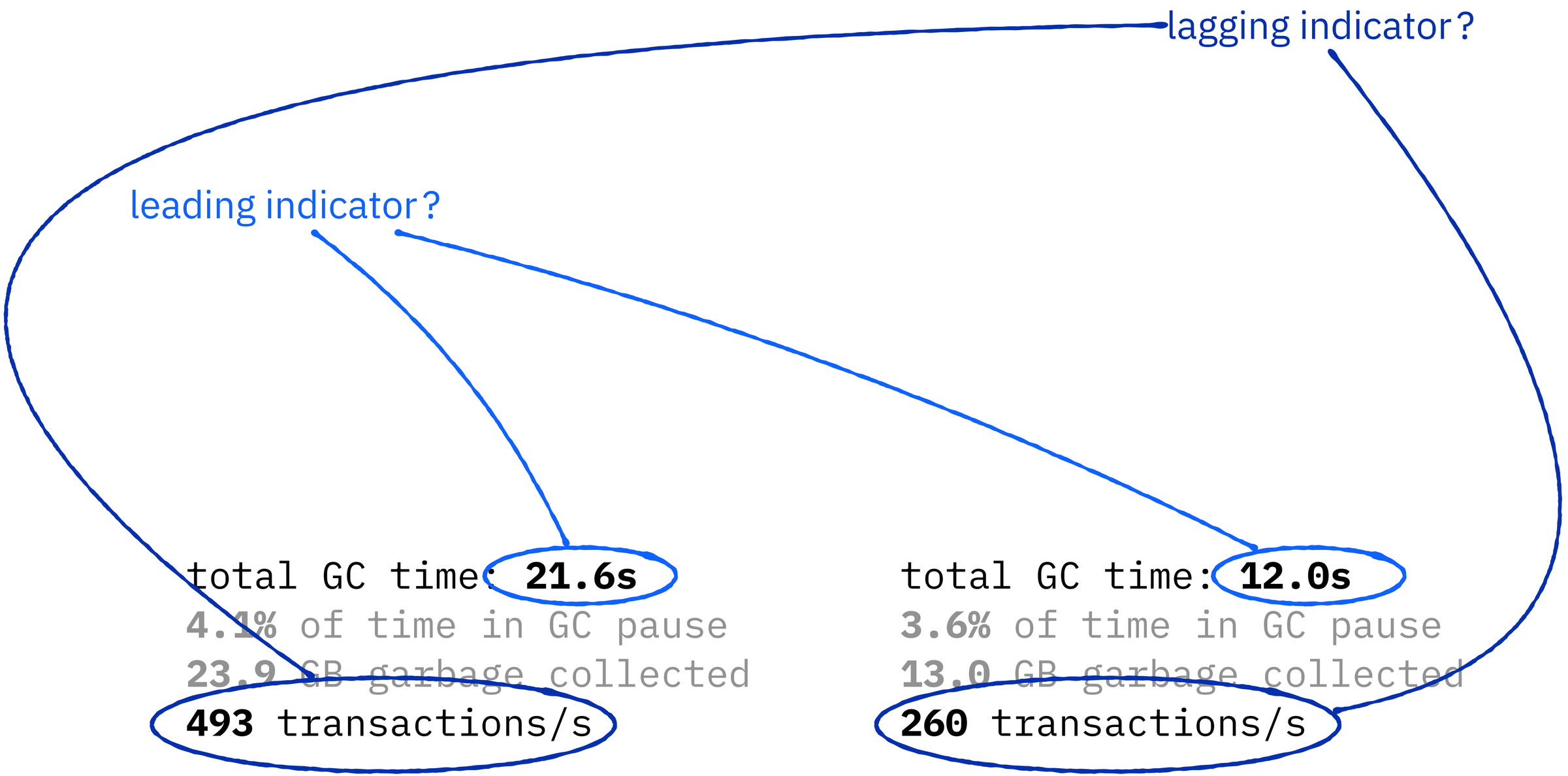
total GC time: **12.0s**
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s


leading indicator
total GC time: 21.6s
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s
total GC time: 12.0s
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s


leading indicator
total GC time: 21.6s
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s
total GC time: 12.0s
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

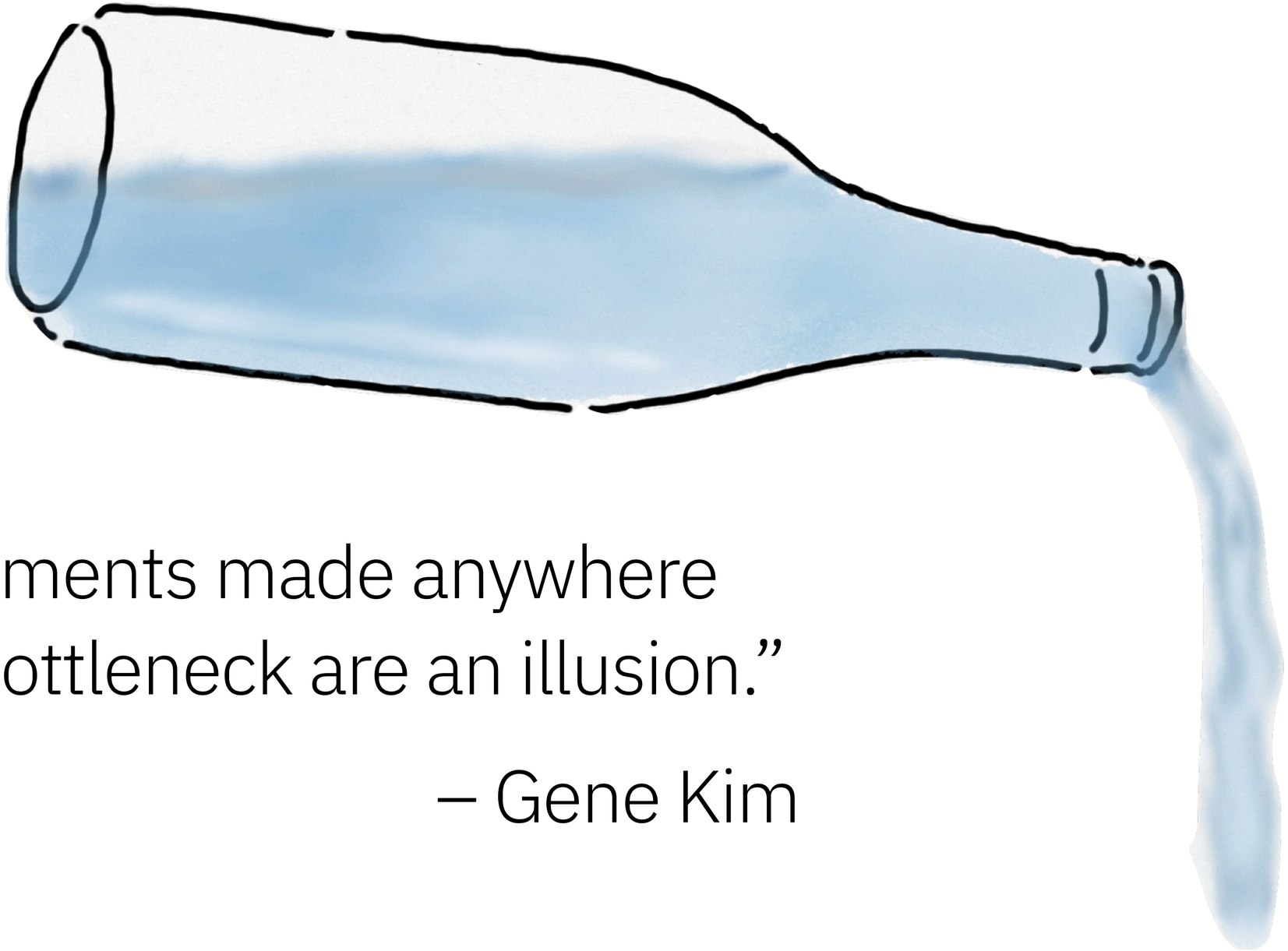

lagging indicator
leading indicator
total GC time: 21.6s
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s
total GC time: 12.0s
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s


lagging indicator
leading indicator?
total GC time: 21.6s
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s
total GC time: 12.0s
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s


lagging indicator?
leading indicator?
total GC time: 21.6s
4.1% of time in GC pause
23.9 GB garbage collected
493 transactions/s
total GC time: 12.0s
3.6% of time in GC pause
13.0 GB garbage collected
260 transactions/s

so wait, what changed to make the app faster?

running jmeter on the same machine as the app gives a big speedup!



“Any improvements made anywhere besides the bottleneck are an illusion.”

– Gene Kim

time kills all performance advice

(even mine)

the takeaways:

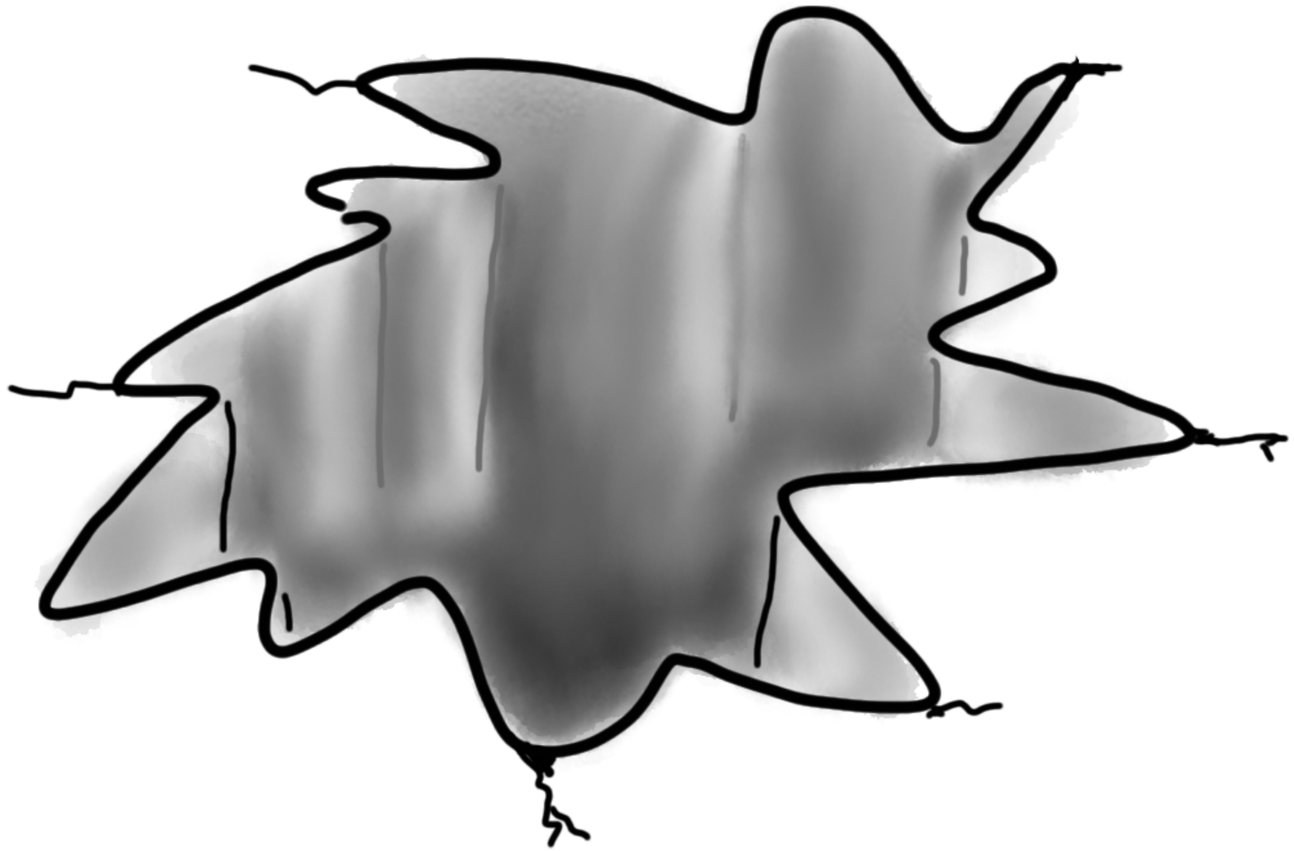
gc can improve performance by rearranging the heap

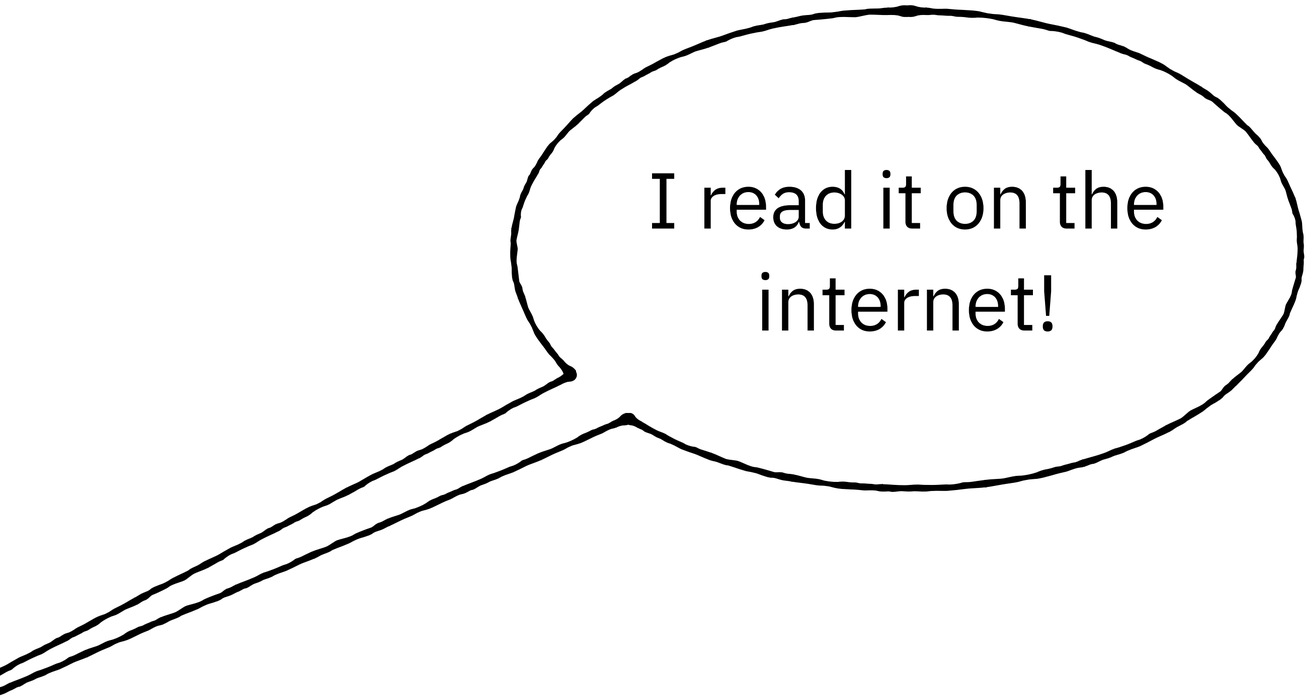
find the bottleneck

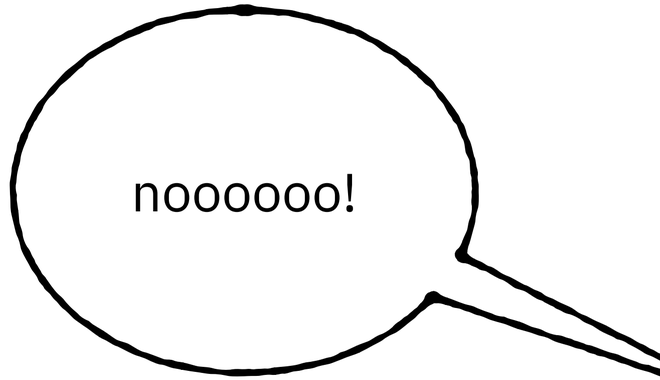
validate advice independently

pitfall 3

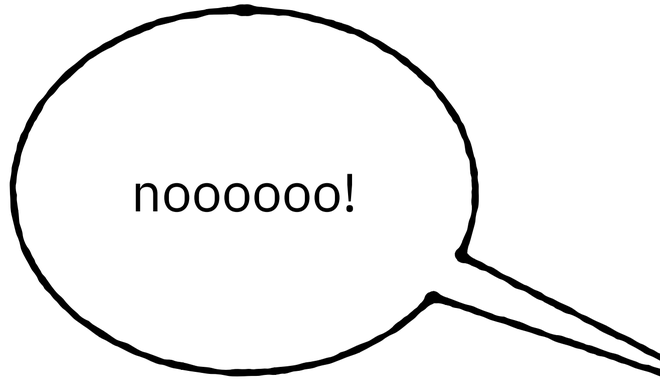
advice



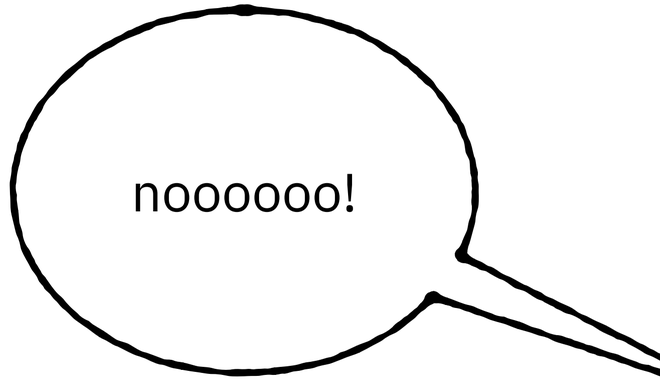

I read it on the internet!



“make one big method because method dispatching is slow”

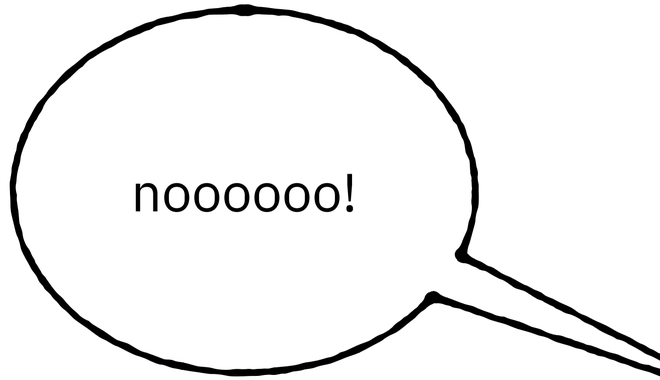


“re-use your objects to help the garbage collector”

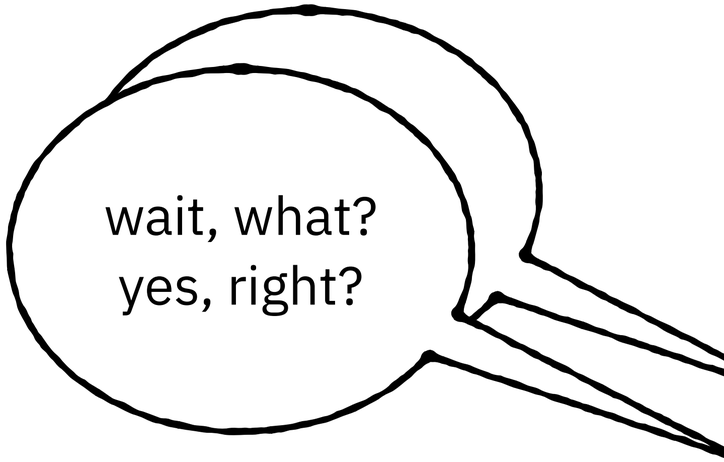


“to tune your JVM, use this command-line:”

```
-server  -Xms1g -Xmx1g  -XX:PermSize=1g -XX:MaxPermSize=256m  -Xmn256m
-Xss64k  -XX:SurvivorRatio=30  -XX:+UseConcMarkSweepGC -XX:
+CMSParallelRemarkEnabled  -XX:+UseCMSInitiatingOccupancyOnly
-XX:CMSInitiatingOccupancyFraction=10  -XX:+ScavengeBeforeFullGC -XX:
+CMSScavengeBeforeRemark  -XX:+PrintGCDateStamps -verbose:gc -XX:
+PrintGCDetails -Dsun.net.inetaddr.ttl=5  -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=`date`.hprof   -Dcom.sun.management.jmxremote.port=5616
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false -server -Xms2g -Xmx2g
-XX:MaxPermSize=256m -XX:NewRatio=1 -XX:+UseConcMarkSweepGC
```



use `StringBuilder`, never concatenate strings with `+=`



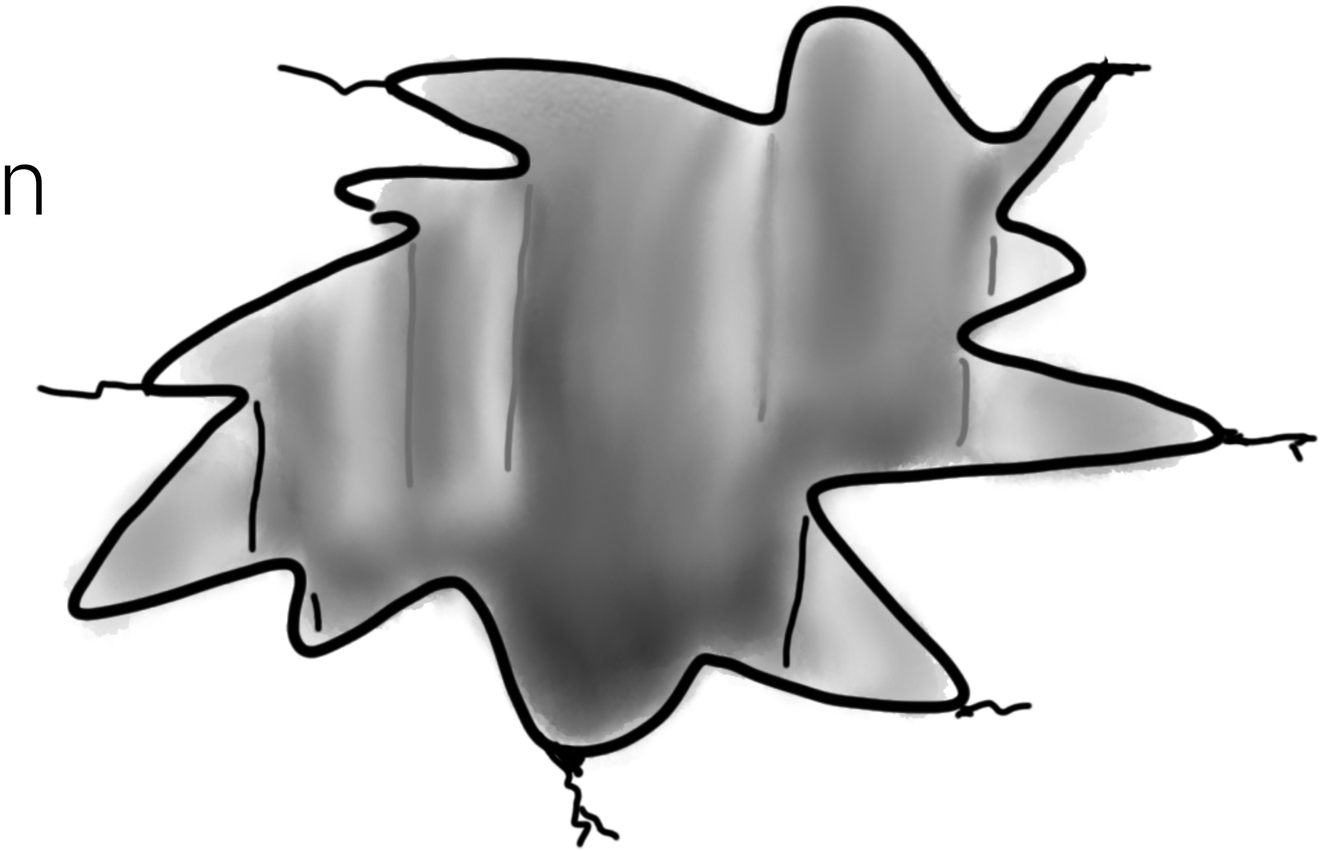
use `StringBuilder`, never concatenate strings with `+=`

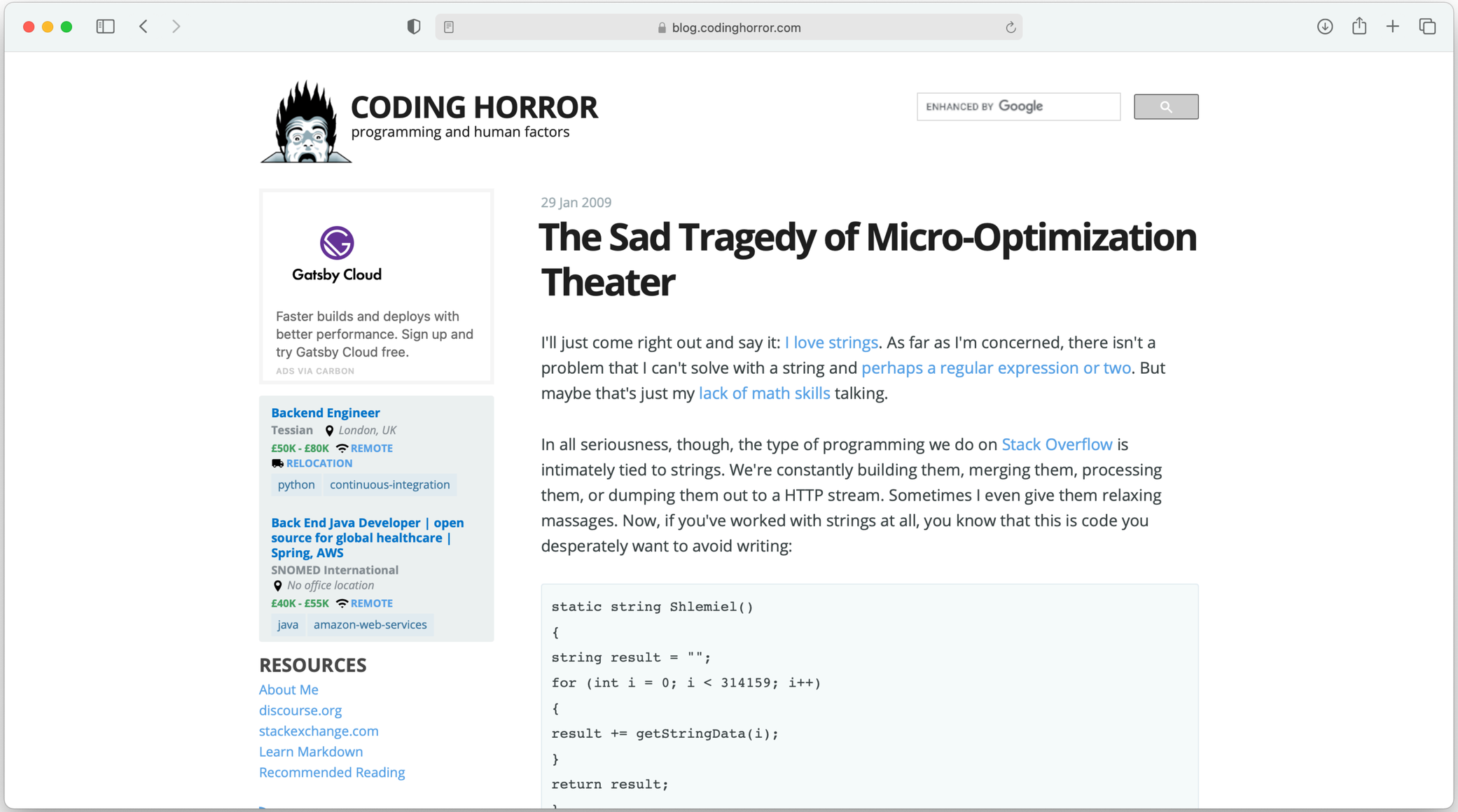
2 things ruin advice:

- context
- time

pitfall 4

micro-optimisation




blog.codinghorror.com
CODING HORROR
programming and human factors
ENHANCED BY Google
Gatsby Cloud
Faster builds and deploys with better performance. Sign up and try Gatsby Cloud free.
ADS VIA CARBON
Backend Engineer
Tessian London, UK
£50K - £80K REMOTE
RELOCATION
python continuous-integration
Back End Java Developer | open source for global healthcare | Spring, AWS
SNOMED International
No office location
£40K - £55K REMOTE
java amazon-web-services
RESOURCES
About Me
discourse.org
stackexchange.com
Learn Markdown
Recommended Reading
29 Jan 2009
The Sad Tragedy of Micro-Optimization Theater
I'll just come right out and say it: I love strings. As far as I'm concerned, there isn't a problem that I can't solve with a string and perhaps a regular expression or two. But maybe that's just my lack of math skills talking.
In all seriousness, though, the type of programming we do on Stack Overflow is intimately tied to strings. We're constantly building them, merging them, processing them, or dumping them out to a HTTP stream. Sometimes I even give them relaxing massages. Now, if you've worked with strings at all, you know that this is code you desperately want to avoid writing:
static string Shlemiel()
{
string result = "";
for (int i = 0; i < 314159; i++)
{
result += getStringData(i);
}
return result;

```
static string beSlow()
{
   string result = "";
   for (int i = 0; i < 314159; i++)
   {
    result += getStringData(i);
   }
   return result;
}
```

```
                                {
        String ret = "\n\tMarket Summary at: " + getSummaryDate() + "\n\t\t        TSIA:" + getTSIA() + "\n\t\t
openTSIA:" + getOpenTSIA()
                + "\n\t\t        gain:" + getGainPercent() + "\n\t\t      volume:" + getVolume();

        if ((getTopGainers() == null) || (getTopLosers() == null)) {
            return ret;
        }
        ret += "\n\t\t   Current Top Gainers:";
        Iterator<QuoteDataBean> it = getTopGainers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        ret += "\n\t\t   Current Top Losers:";
        it = getTopLosers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        return ret;
    }
```

```
                                {
        String ret = "\n\tMarket Summary at: " + getSummaryDate() + "\n\t\t        TSIA:" + getTSIA() + "\n\t\t
openTSIA:" + getOpenTSIA()
                + "\n\t\t        gain:" + getGainPercent() + "\n\t\t      volume:" + getVolume();

        if ((getTopGainers() == null) || (getTopLosers() == null)) {
            return ret;
        }
        ret += "\n\t\t   Current Top Gainers:";
        Iterator<QuoteDataBean> it = getTopGainers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        ret += "\n\t\t   Current Top Losers:";
        it = getTopLosers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        return ret;
    }
```

```
                                {
        String ret = "\n\tMarket Summary at: " + getSummaryDate() + "\n\t\t        TSIA:" + getTSIA() + "\n\t\t
openTSIA:" + getOpenTSIA()
                + "\n\t\t        gain:" + getGainPercent() + "\n\t\t      volume:" + getVolume();

        if ((getTopGainers() == null) || (getTopLosers() == null)) {
            return ret;
        }
        ret += "\n\t\t   Current Top Gainers:";
        Iterator<QuoteDataBean> it = getTopGainers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        ret += "\n\t\t   Current Top Losers:";
        it = getTopLosers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        return ret;
    }
```

this **never** gets called

```
@Override
    public String toString() {
        String ret = "\n\tMarket Summary at: " + getSummaryDate() + "\n\t\t        TSIA:" + getTSIA() + "\n\t\t
openTSIA:" + getOpenTSIA()
                + "\n\t\t        gain:" + getGainPercent() + "\n\t\t      volume:" + getVolume();

        if ((getTopGainers() == null) || (getTopLosers() == null)) {
            return ret;
        }
        ret += "\n\t\t   Current Top Gainers:";
        Iterator<QuoteDataBean> it = getTopGainers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        ret += "\n\t\t   Current Top Losers:";
        it = getTopLosers().iterator();
        while (it.hasNext()) {
            QuoteDataBean quoteData = it.next();
            ret += ("\n\t\t\t" + quoteData.toString());
        }
        return ret;
    }
```

let’s make travel energy-efficient?

every little helps?

every little helps?

every optimisation is another optimisation you **aren’t** doing

our platforms help

```
static string beSlow()
{
   string result = "";
   for (int i = 0; i < 314159; i++)
   {
    result += getStringData(i);
   }
   return result;
}
```

```
static string beSlow()
{
   string result = “";
   result += getStringData(1);
   result += getStringData(2);
   result += getStringData(3);

   return result;
}
```

```
static string beSlow()
{
   string result = “";
   result += getStringData(1);
   result += getStringData(2);
   result += getStringData(3);

   return result;
}
```

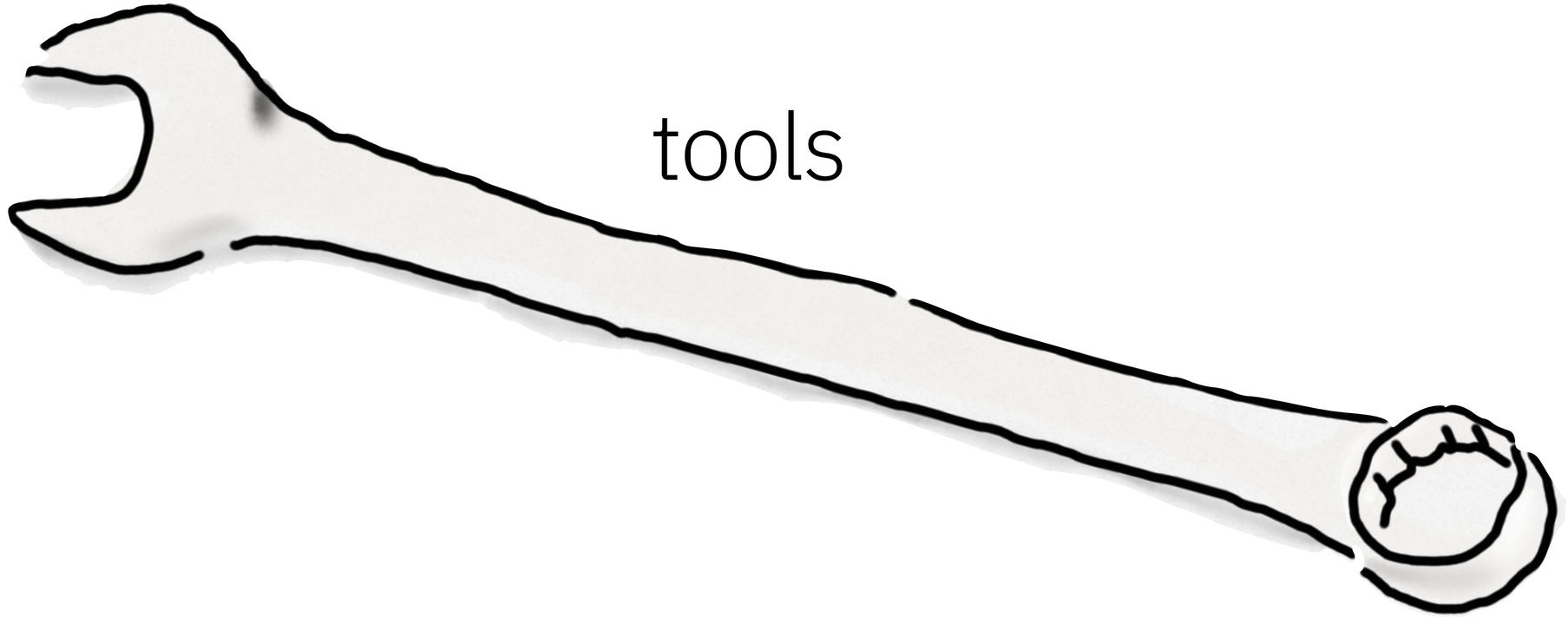
this is **fine**

the JVM writers have **far** more time for optimising than you do

clean, typical, code runs best

ok, but how to optimise?

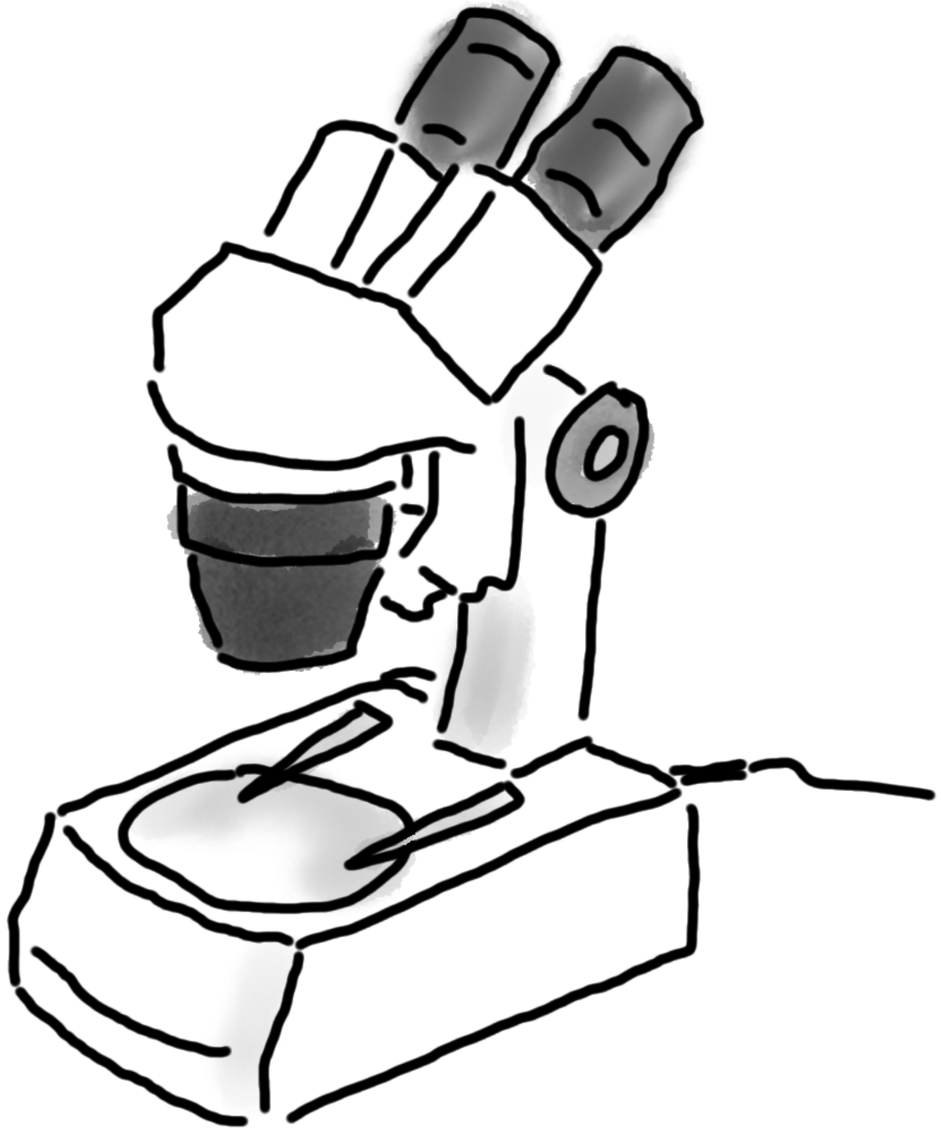
tools



“What you can optimize is limited to what you can observe.”

-Susie Xia, Netflix

observability



method profiler

GC analysis

heap analysis

APM

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

VisualVM

GC analysis

heap analysis

APM

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

VisualVM

Mission Control

GC analysis

heap analysis

APM

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

VisualVM

GC analysis

Mission Control

heap analysis

IBM Health Center
(for OpenJ9)

APM

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

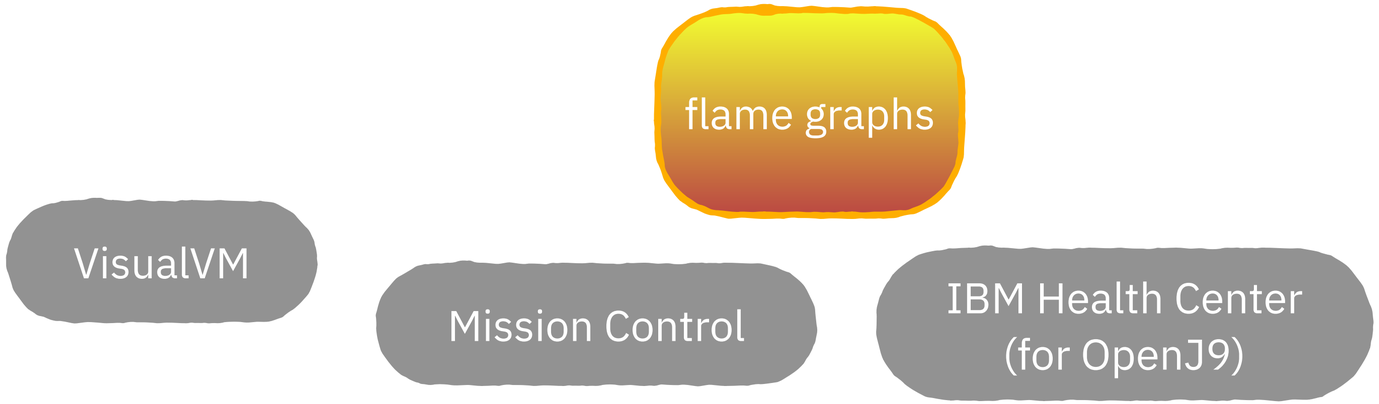
method profiler

GC analysis

heap analysis

APM

distributed tracing



* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

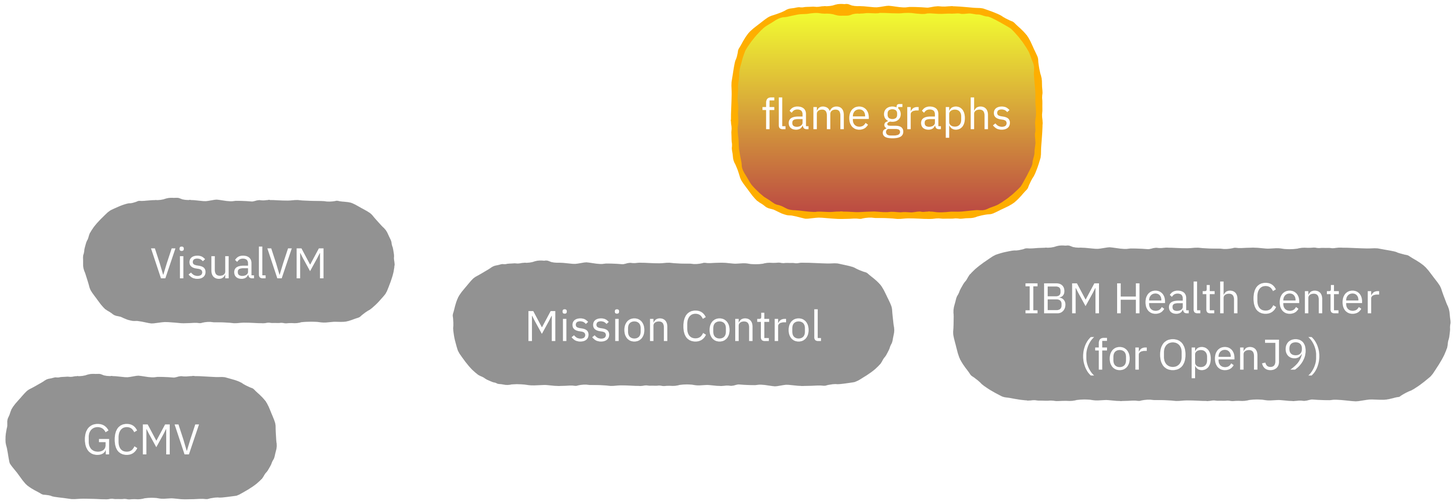
method profiler

GC analysis

heap analysis

APM

distributed tracing



* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

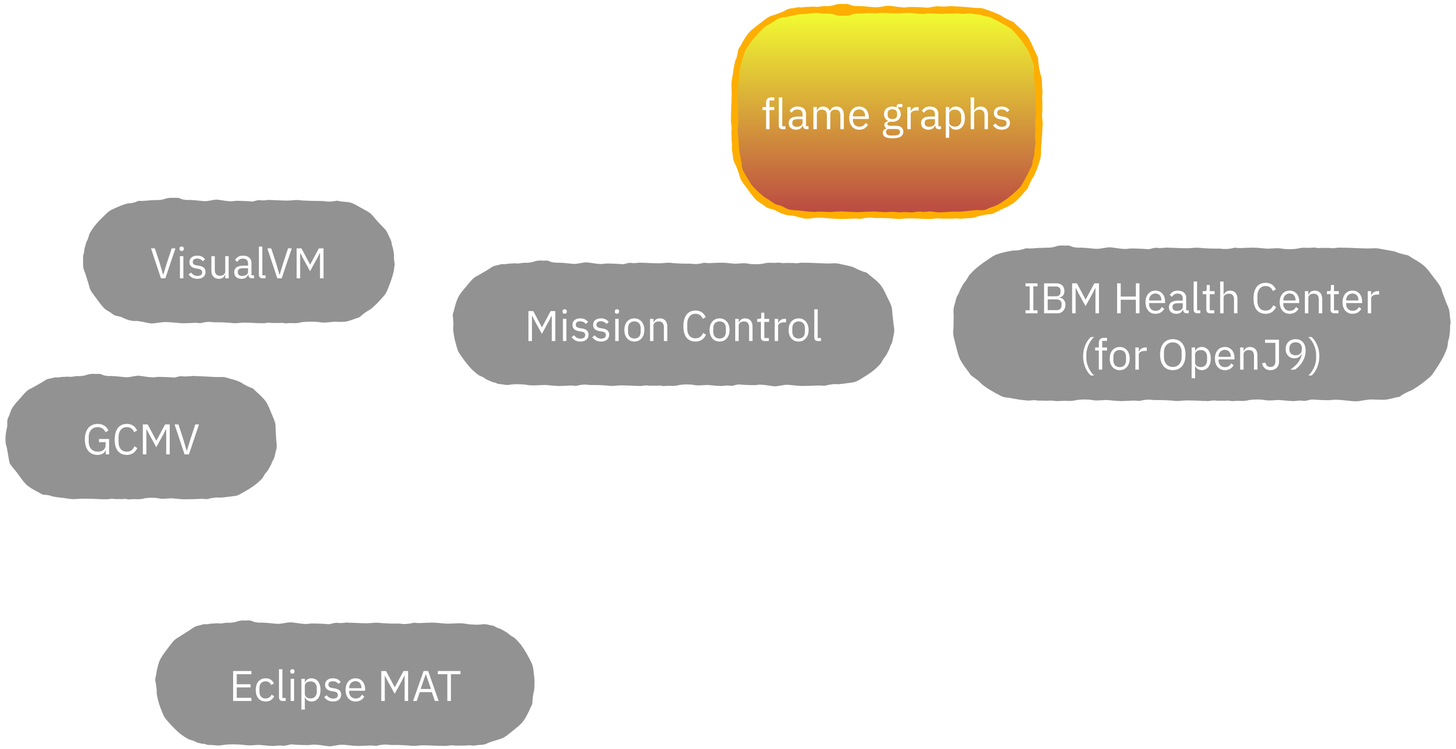
method profiler

GC analysis

heap analysis

APM

distributed tracing



* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

flame graphs

VisualVM

Mission Control

IBM Health Center
(for OpenJ9)

GC analysis

GCMV

heap analysis

Eclipse MAT

APM

GlowRoot

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

flame graphs

VisualVM

Mission Control

IBM Health Center
(for OpenJ9)

GC analysis

GCMV

heap analysis

Eclipse MAT

APM

GlowRoot

New Relic*

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

flame graphs

VisualVM

Mission Control

IBM Health Center
(for OpenJ9)

GC analysis

GCMV

heap analysis

Eclipse MAT

APM

GlowRoot

New Relic*

AppDynamics*

distributed tracing

* not free
this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

flame graphs

VisualVM

Mission Control

IBM Health Center (for OpenJ9)

GC analysis

GCMV

heap analysis

Eclipse MAT

APM

GlowRoot

New Relic*

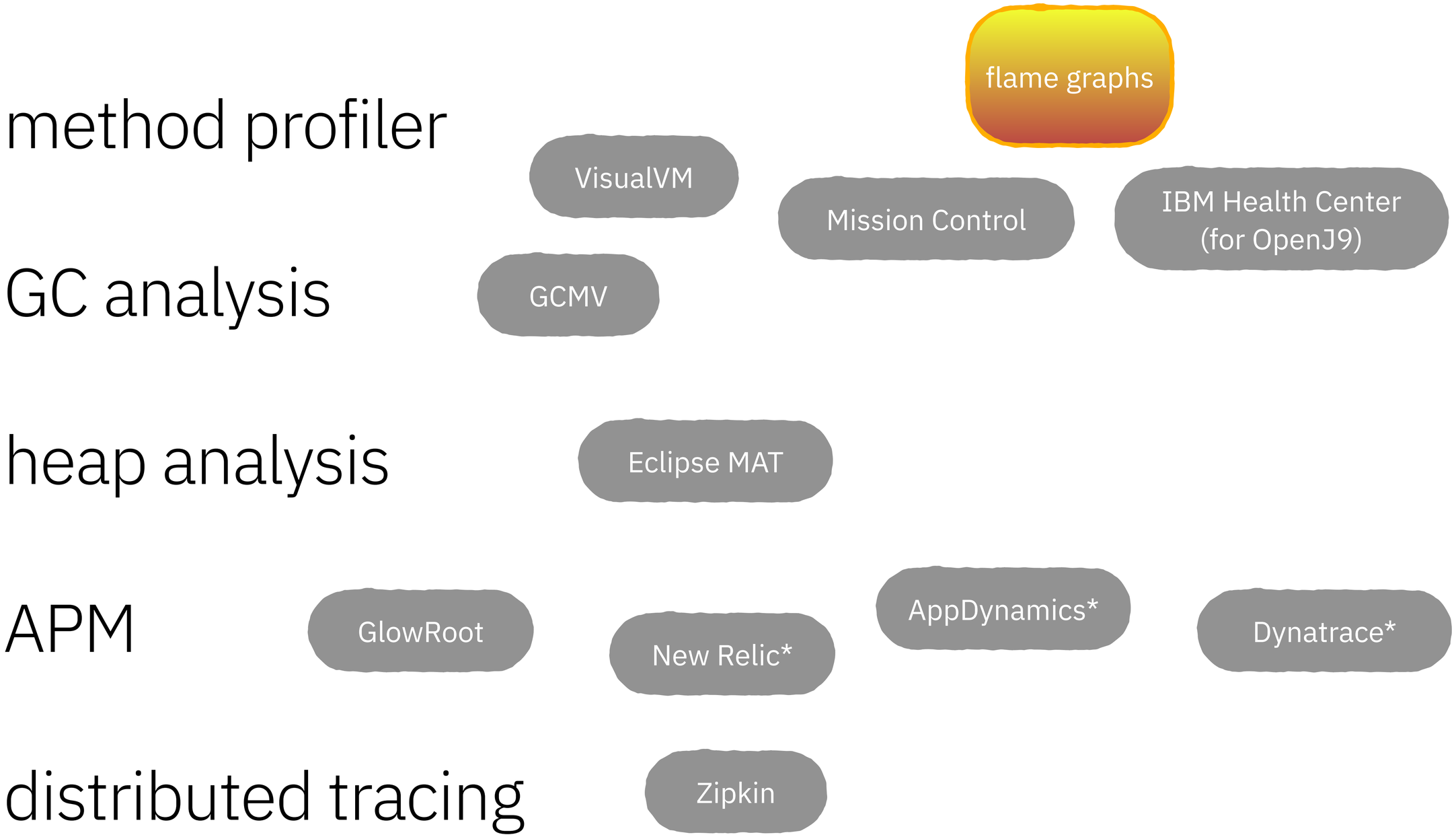
AppDynamics*

Dynatrace*

distributed tracing

* not free

this is an incomplete list, because there are a lot of tools out there, and many cost money



* not free

this is an incomplete list, because there are a lot of tools out there, and many cost money

method profiler

flame graphs

VisualVM

Mission Control

IBM Health Center (for OpenJ9)

GC analysis

GCMV

heap analysis

Eclipse MAT

APM

GlowRoot

New Relic*

AppDynamics*

Dynatrace*

distributed tracing

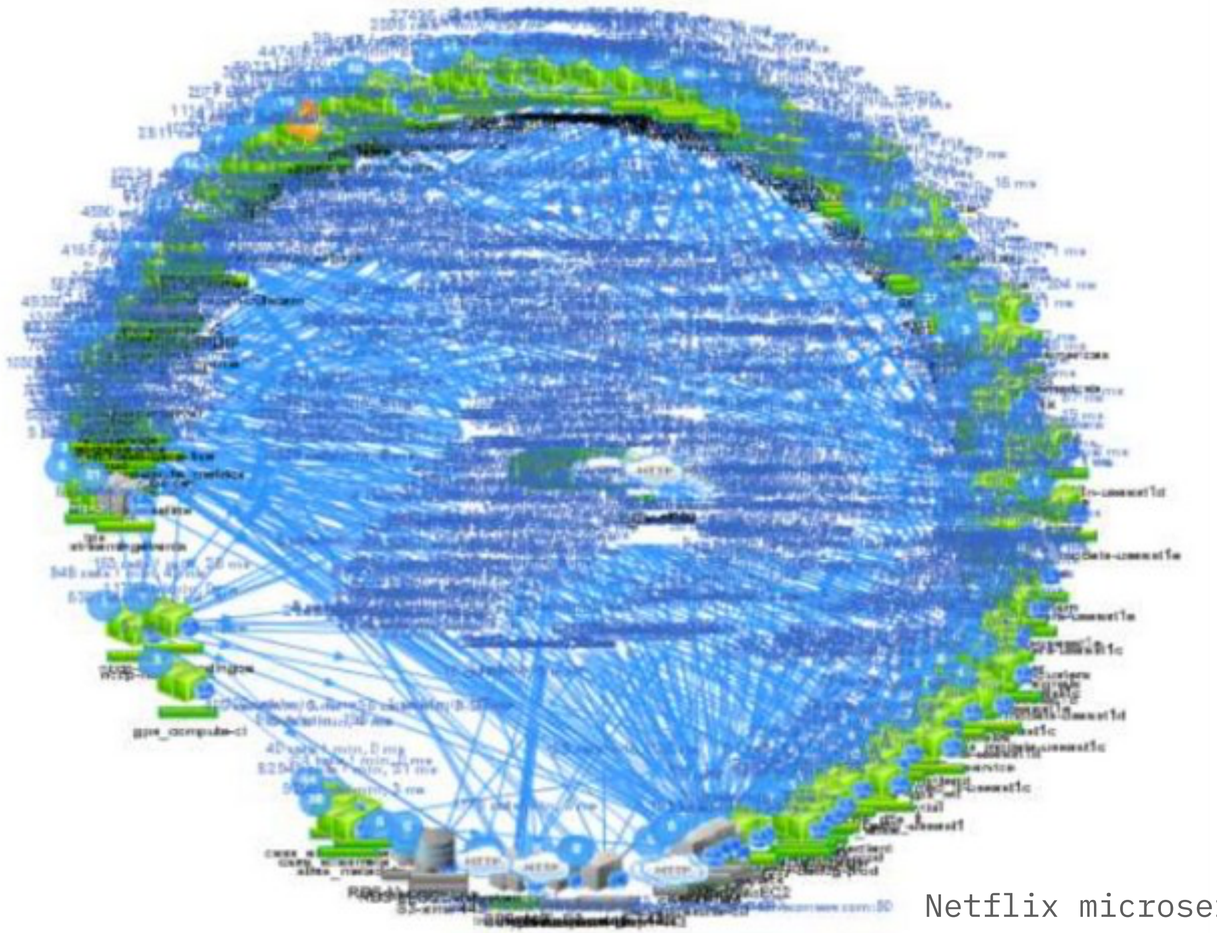
Zipkin

Jaeger

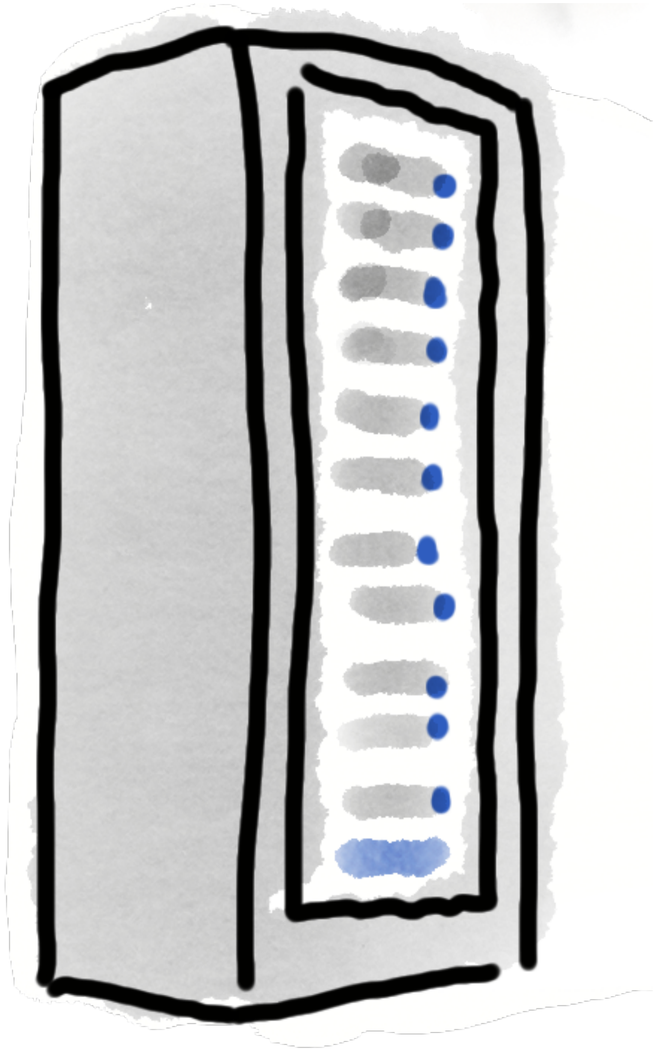
* not free

this is an incomplete list, because there are a lot of tools out there, and many cost money

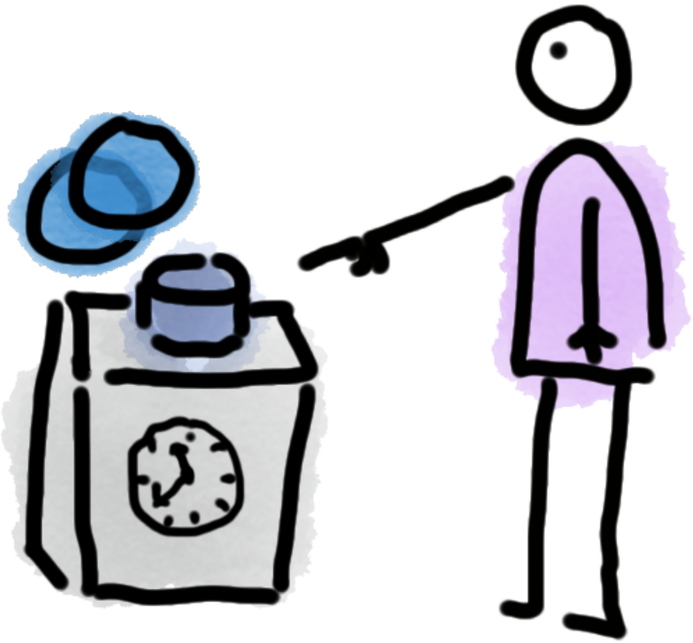
optimising a micro-service: is that micro-optimising?



Netflix microservice architecture



you may need to know the whole system context to know what to optimise



“Nines don’t matter if your users aren’t happy.”

– Charity Majors

don’t forget the edges

queueing theory helps us understand where the disasters happen

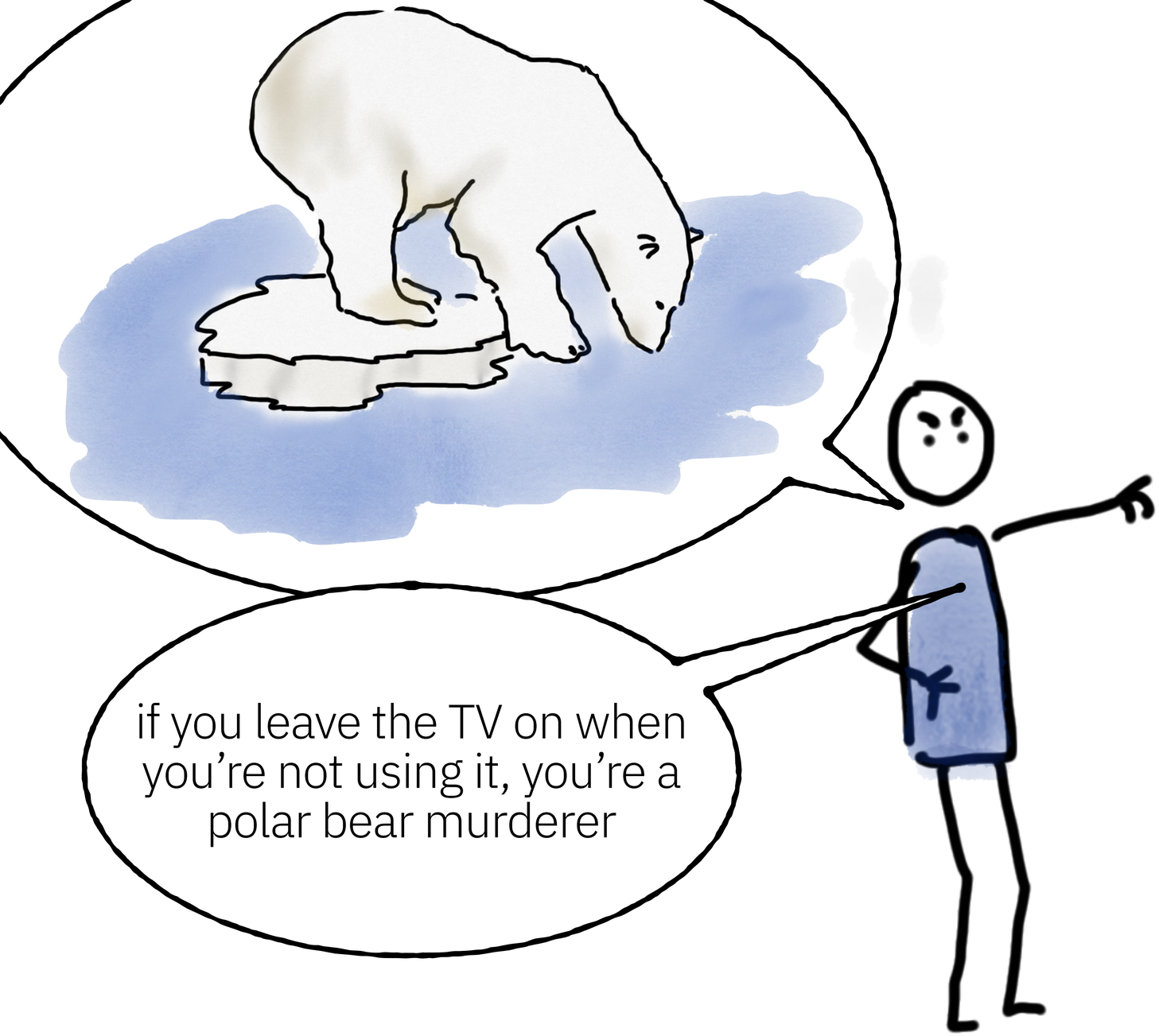
“When it comes to IT performance, amateurs look at averages. Professionals look at distributions.”

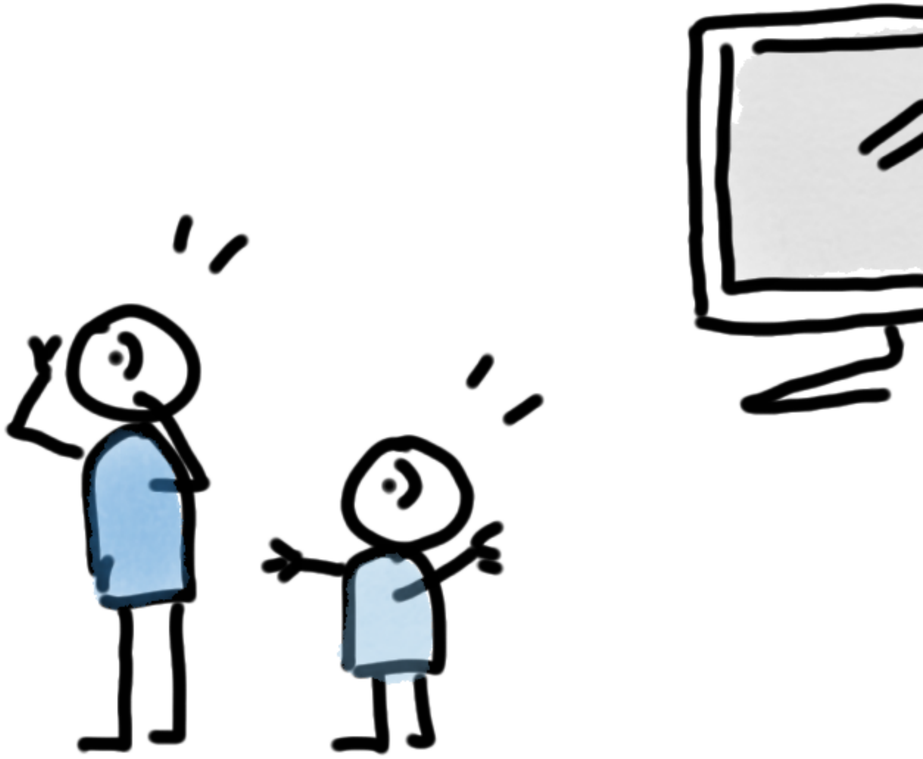
– Avishai Ish-Shalom

slow performance can turn into big cloud bills

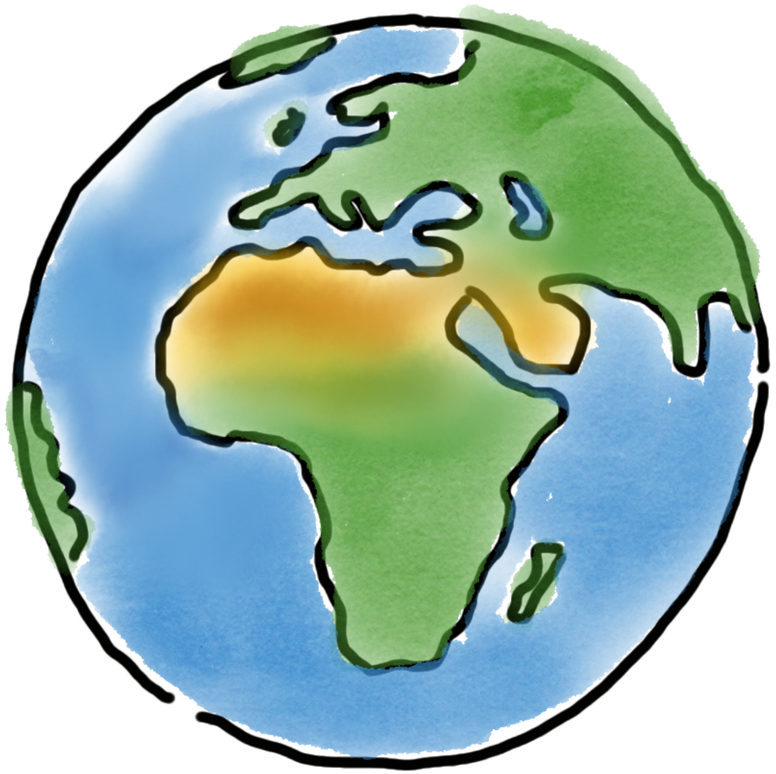
make cloud costs visible to engineers

ok, but you promised bears


if you leave the TV on when you're not using it, you're a polar bear murderer

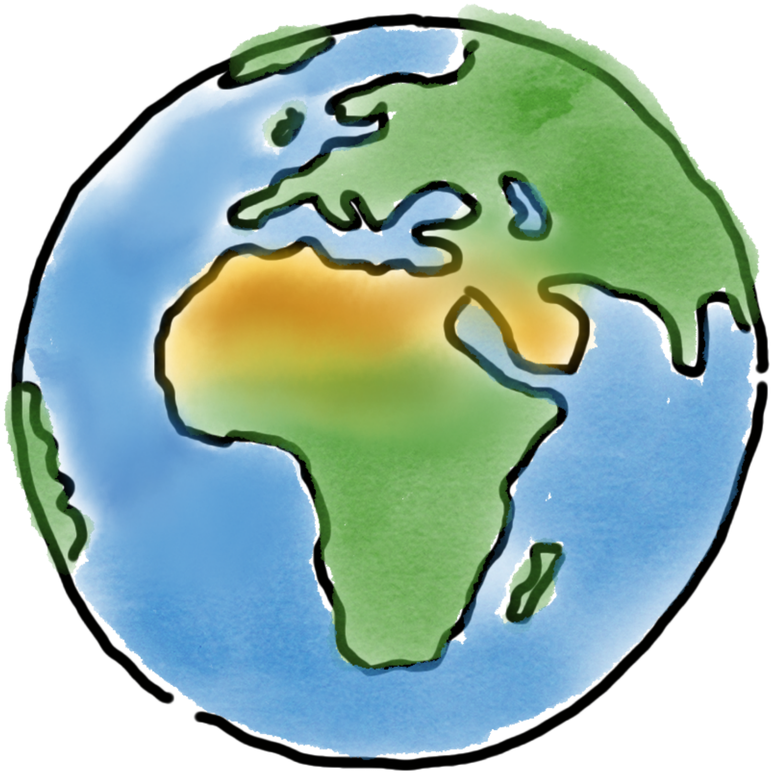


there is a moral imperative to avoid waste

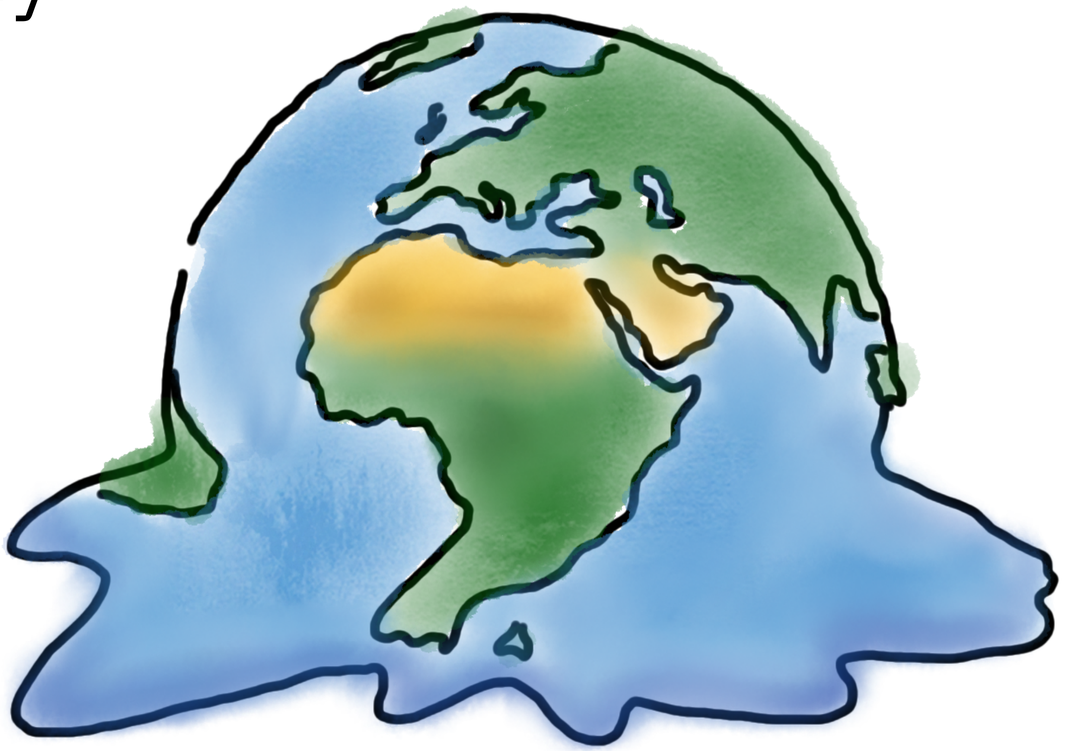


there is a moral imperative to avoid waste

electricity
hardware



data centres use 1-2% of the world's electricity



fewer devices

longer lifetime

higher efficiency

fewer devices

longer lifetime

fewer devices

higher efficiency

lower footprint

longer lifetime

fewer devices

higher efficiency

lower footprint

more multitenancy

longer lifetime

fewer devices

higher efficiency

lower footprint

more multitenancy

longer lifetime

optimise for longevity

fewer devices

higher efficiency

lower footprint

more multitenancy

longer lifetime

optimise for longevity

the end of planned obsolescence?

sooo ...

you **can** optimise, and it can be fun

measure, don't guess

only optimise what matters

now for questions!