a talk for ApacheCon Europe 2008 by Jeremy Quinn

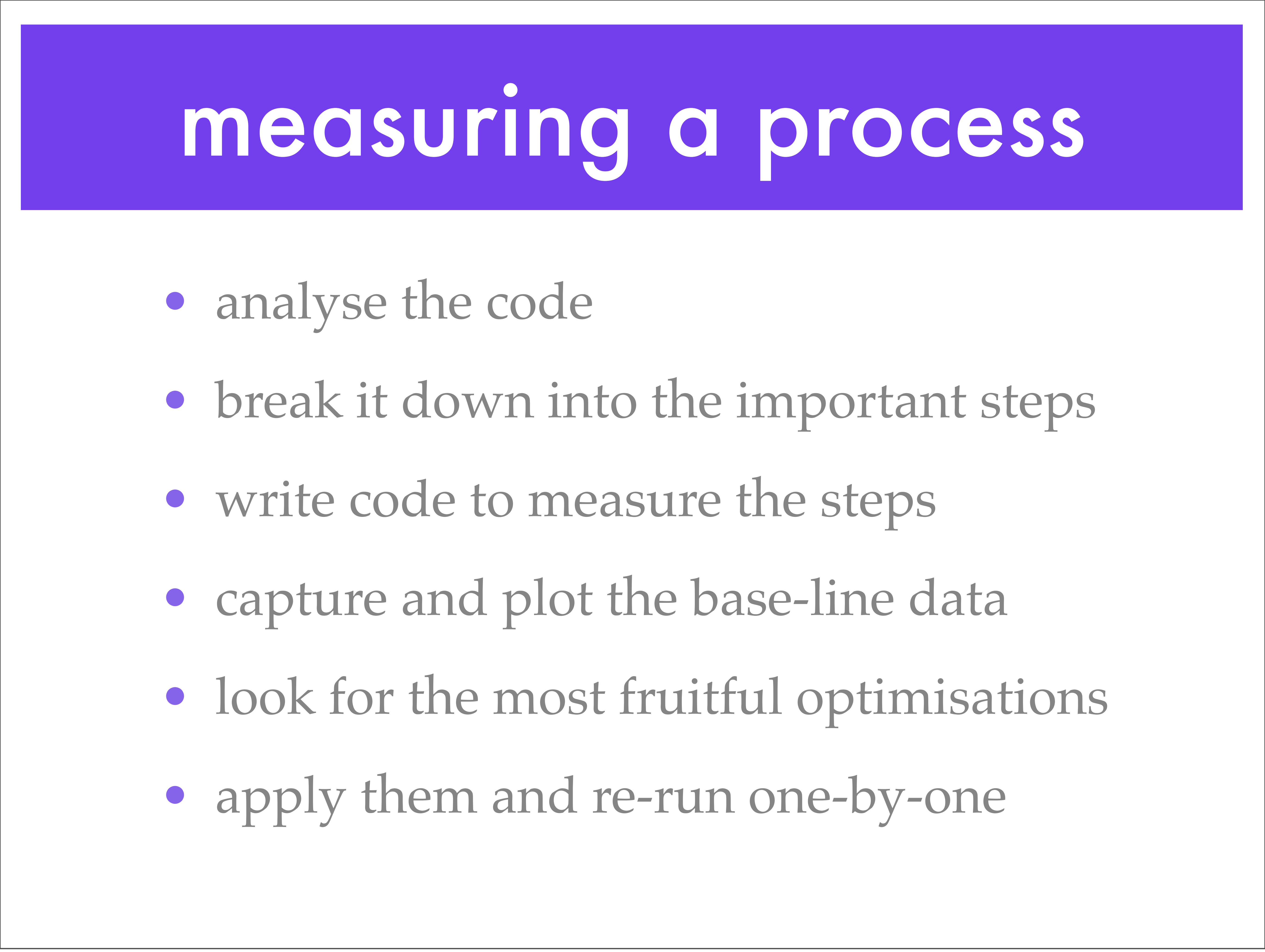
Break My Site

practical stress testing and tuning

photo credit: Môsieur J

This is designed as a beginner’s talk.
I am the beginner.

I will present two case studies:
1) measuring an expensive scheduled process, using custom code
2) measuring a complex web page built by Apache Cocoon, using Apache JMeter



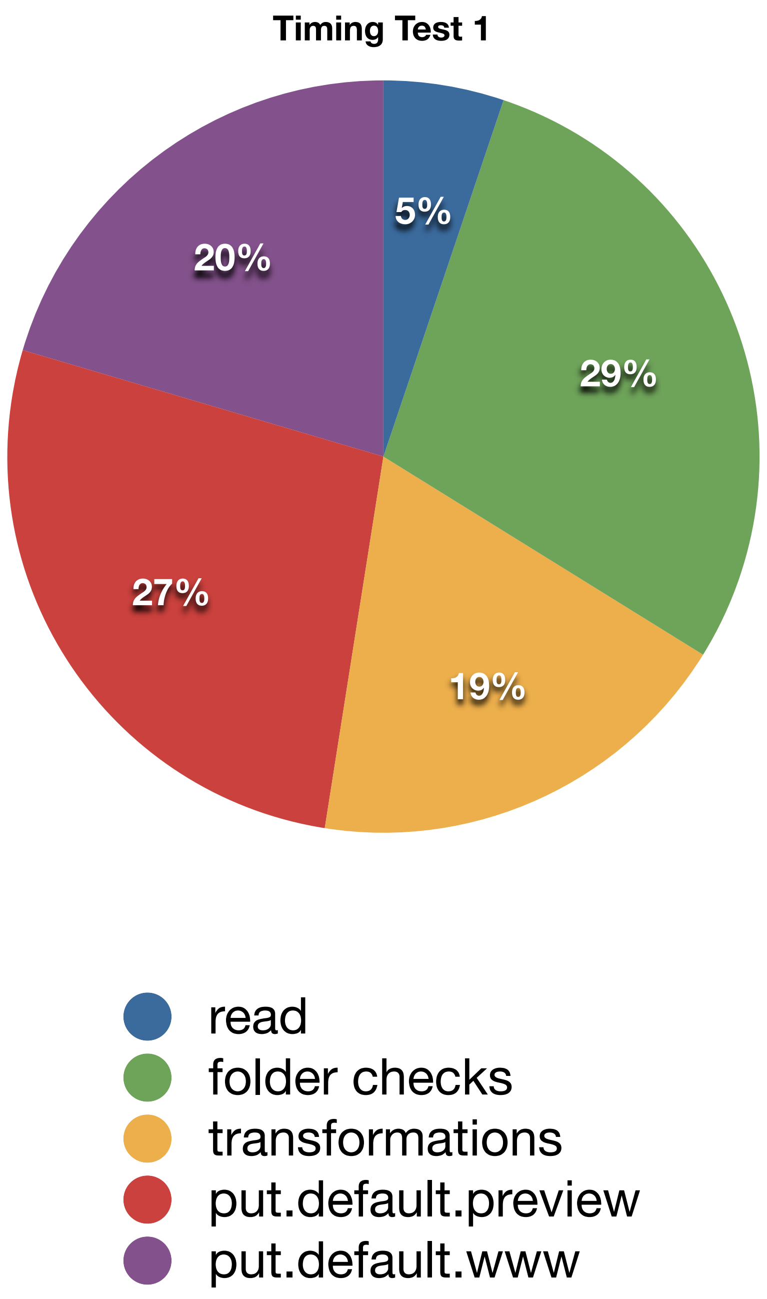
measuring a process

- analyse the code
- break it down into the important steps
- write code to measure the steps
- capture and plot the base-line data
- look for the most fruitful optimisations
- apply them and re-run one-by-one

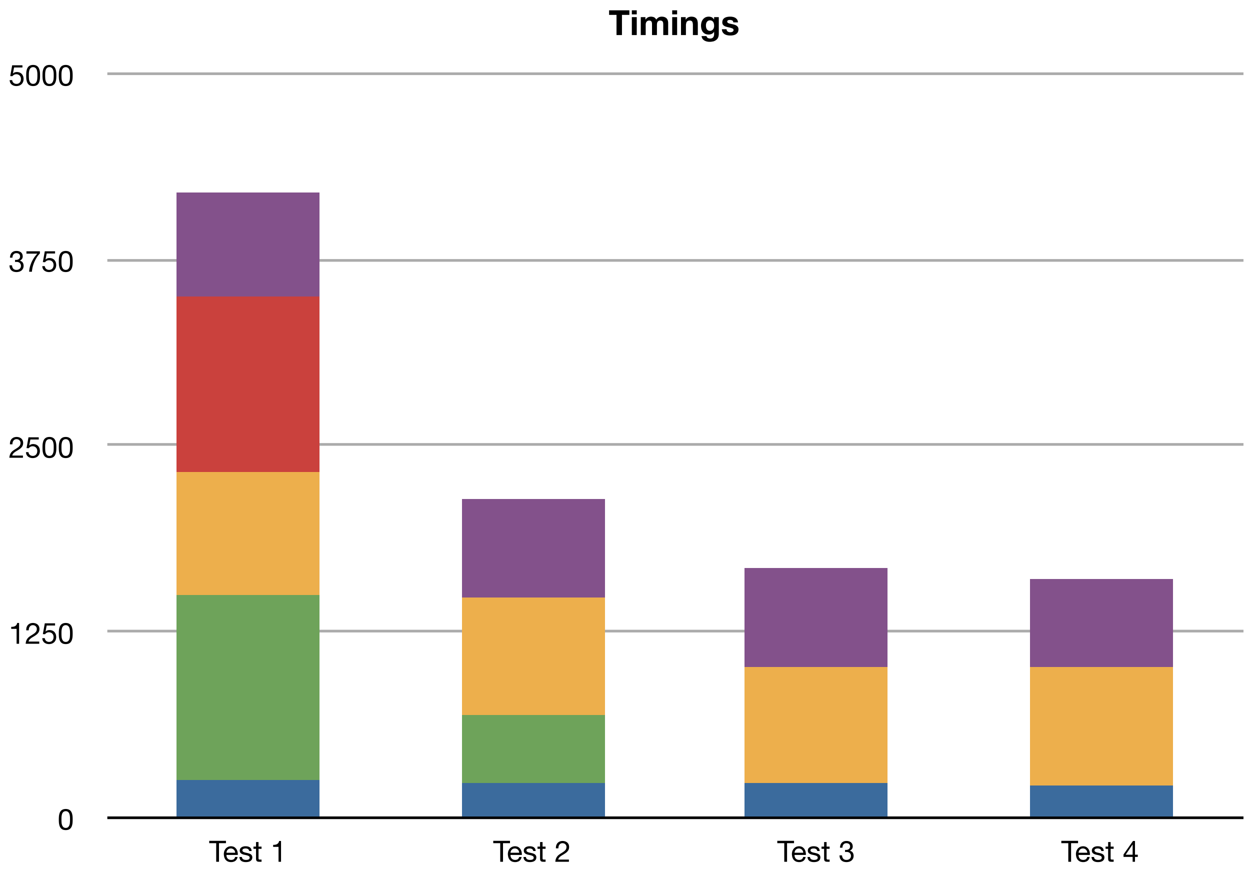
The first case study is a complicated automated import process that runs every few hours on 1000’s of assets

designing the tests is a bit recursive
you need to start by getting a baseline measurement, for comparison
look for places where optimisations will have the greatest effect
work out how to simulate the optimisations
also work out how long a real implementation would take to write
apply the simulation to the code
re-run the tests

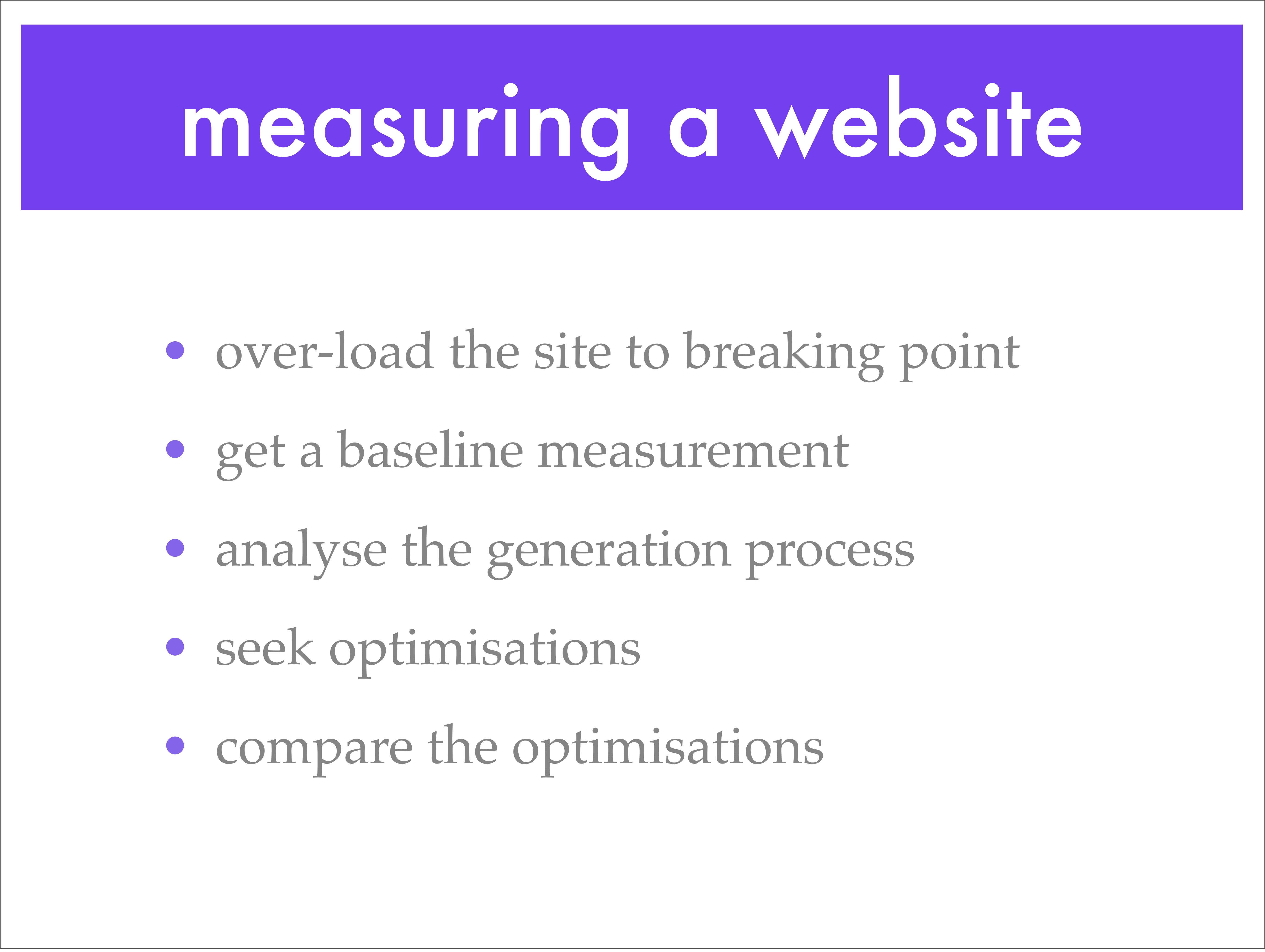
test results



- **Test 1**: the original state
- **Test 2**: write optimisation
- **Test 3**: + folder optimisation
- **Test 4**: + indexer optimisation



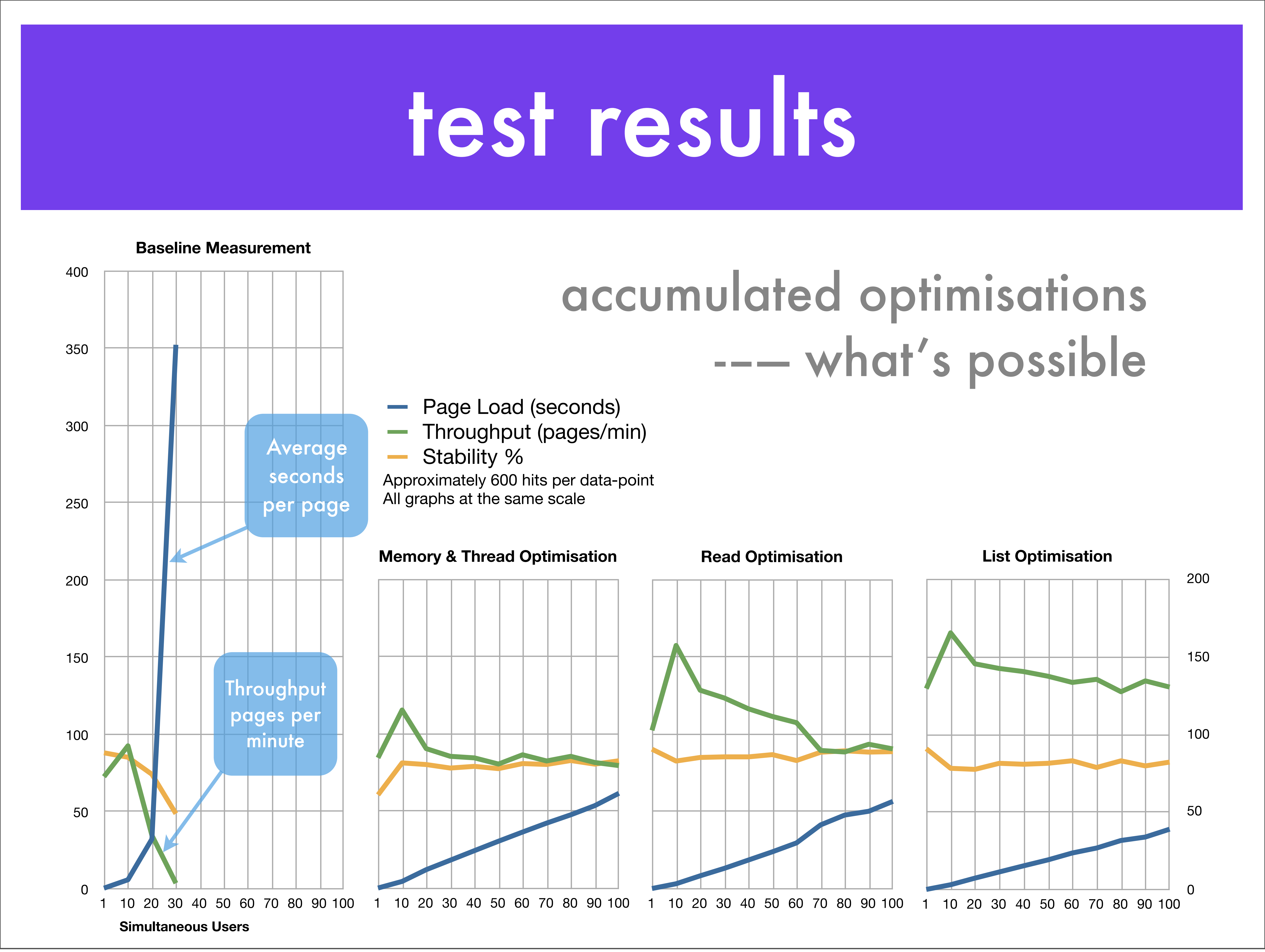
ran a series of four tests
[CLICK] after the first test we got a breakdown of timings for each step averaged over 600 runs
we looked for the low hanging fruit
[CLICK] we used the first run to design the subsequent tests
[CLICK] now we can compare the accumulation of optimisations
it was a recursive process that we could complete quickly because we only simulated the changes, but because we had an idea how long each optimisation would take to really implement, we could decide which optimisations were cost-effective
Get the addendum to this talk, to see the code used to capture and transform the log files into something useful



measuring a website

- over-load the site to breaking point
- get a baseline measurement
- analyse the generation process
- seek optimisations
- compare the optimisations

The second case-study was a representational sample of a very big news website, developed using Apache Cocoon.
The page aggregated 20 collections of links to categorised documents, each requiring a query to a back-end document repo.



test results

accumulated optimisations
--- what’s possible

Average seconds per page

Throughput pages per minute

Approximately 600 hits per data-point
All graphs at the same scale

•We started off with a baseline measurement (shows why the site was stalling)
[CLICK] the server gets into trouble
[CLICK] totally non-linear relationship between users and speed
the logs showed we needed memory and thread optimisation
after several iterations of tuning we got this [CLICK] then [CLICK] then [CLICK]
• Shows that through a series of optimisations, a vast improvement can be made.
•You may be able to see how useless one person clicking in a browser is at testing.
•Disregard the absolute speed values, this was done on my laptop.
•The original server configuration being taken to only 30 users, with disastrous effect.
•Throughput drops to single figures, page load time climbs sky high, stability reduces by nearly half.
•From the logs I could see there were not enough threads or memory to cope.

purpose

what do I hope to achieve?

photo credit: Brian Uhreen

Load testing can be used to solve different sorts of problems.
If it's a website, you don't want it broken by popularity.
Find out how much traffic will make a site break,
so you can plan to cope.

comparing changes

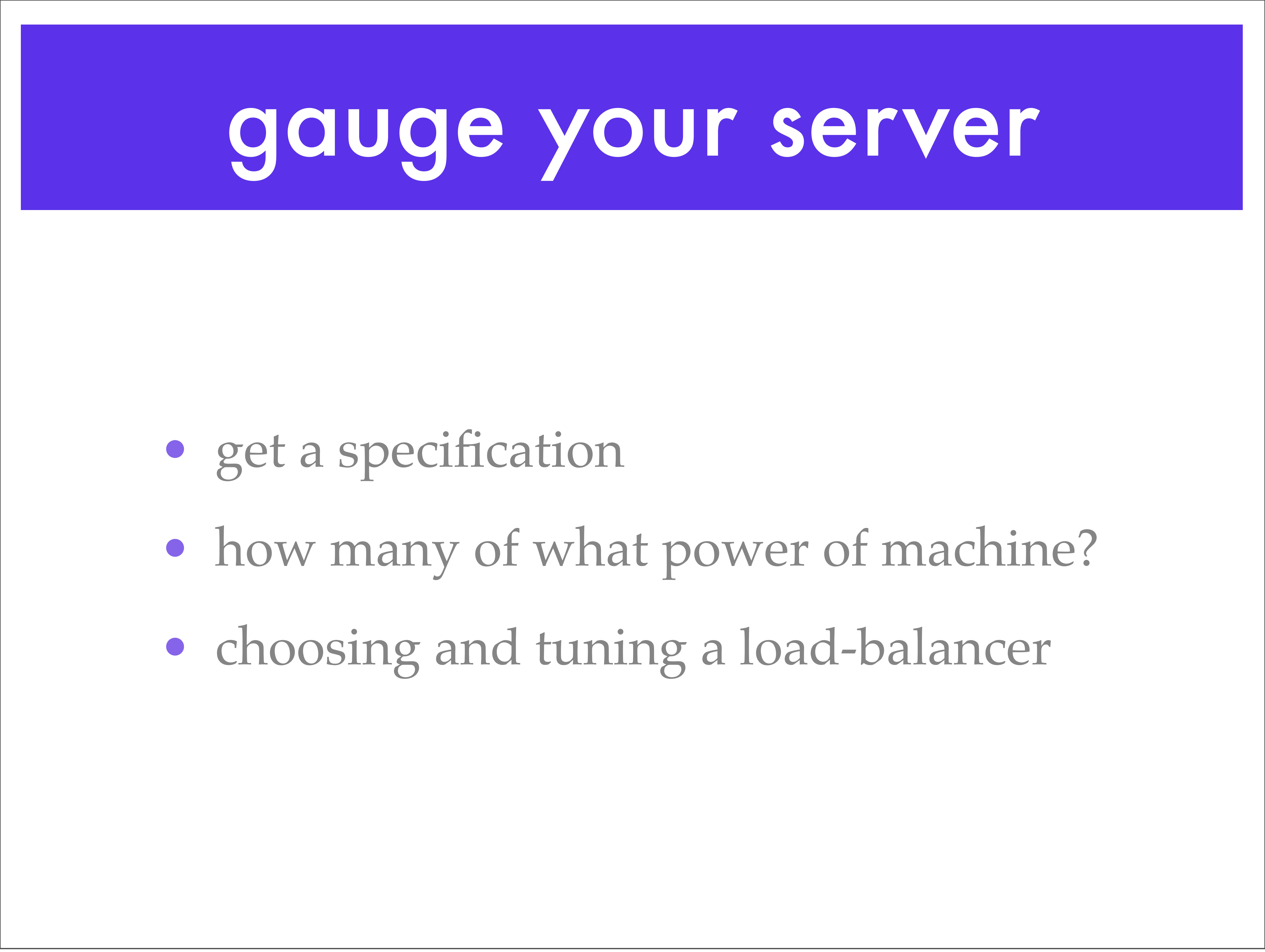
- code optimisations
- different implementations
- caching strategies
- different datasources
- different architectures

Load testing can be used during development to compare different sorts of changes.

tests

- speed of pages
- content of pages
- behaviour of webapps

JMeter can test the speed of webpage loads with different amounts of users
It also has tools for crawling and testing content within web pages.
It can also send parameters as POST requests etc. to help you test webapps.
This is useful to test if optimisations break a working site



gauge your server

- get a specification
- how many of what power of machine?
- choosing and tuning a load-balancer

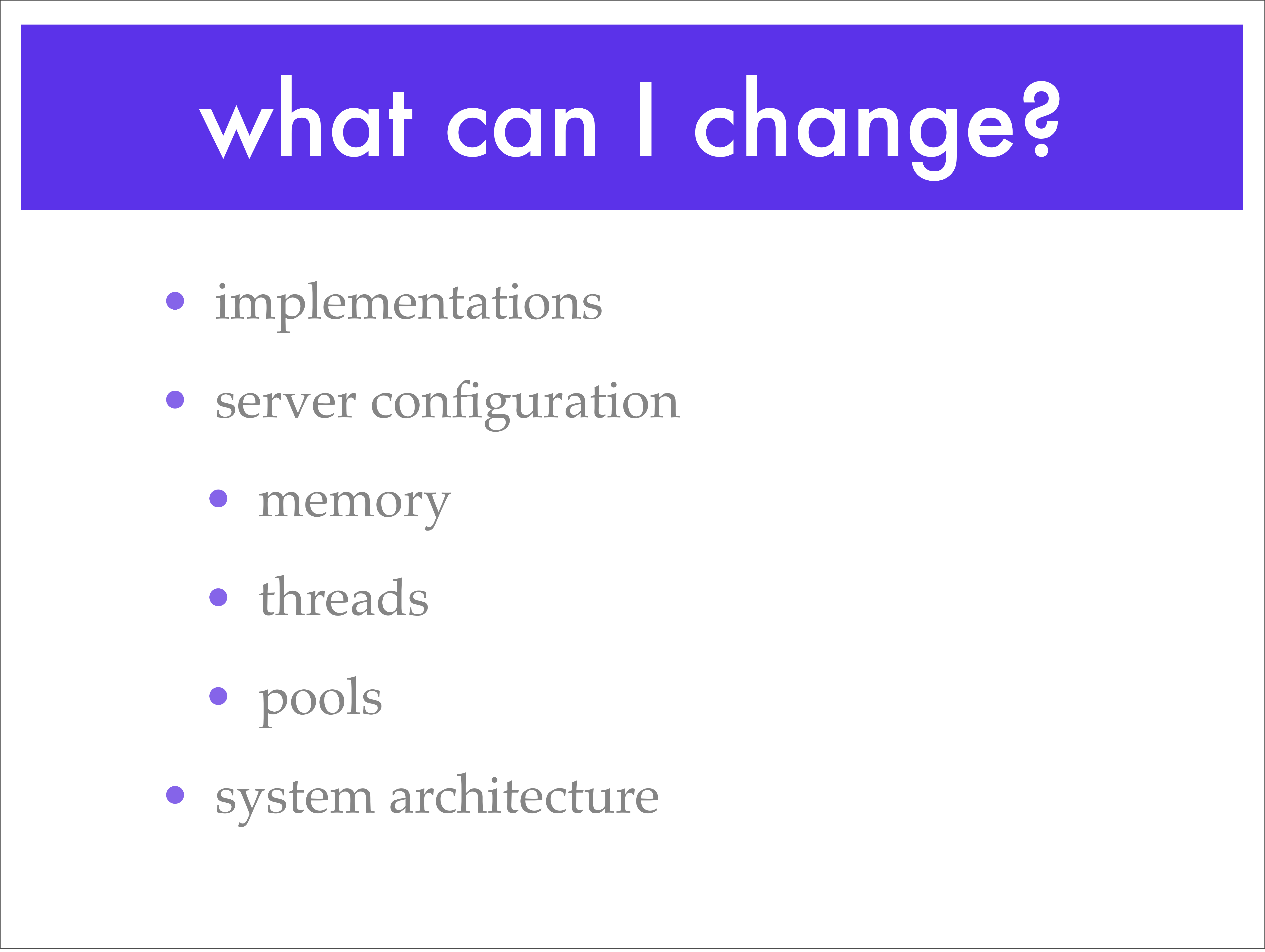
Before you start to work out what servers you need, you have to know what level of traffic it is intended to cope with.
This has to come from either the implementation you are replacing or from marketing.
Test the machines you plan to use, so you know what capacity they can handle.
Round-Robins with machines of different capacities may work really badly.

planning

what am I going to do?

photo credit: Paul Goyette

meaningful results are not guaranteed
through good planning you will hopefully get results from your tests that are useful to you



what can I change?

- implementations
- server configuration
    - memory
    - threads
    - pools
- system architecture

Form a clear picture of what you will try to change
And what effect you expect to see

what will happen?

- build a mental model
- predict what you expect to happen
- learn to read the data
- test your ideas against reality

As your tests run, relate events in the server logs to how the graphs look

order of change

- only make one change at a time
- only make one change at a time
- only make one change at a time
- only make one change at a time
- in the most meaningful order
  - which depends on your purpose

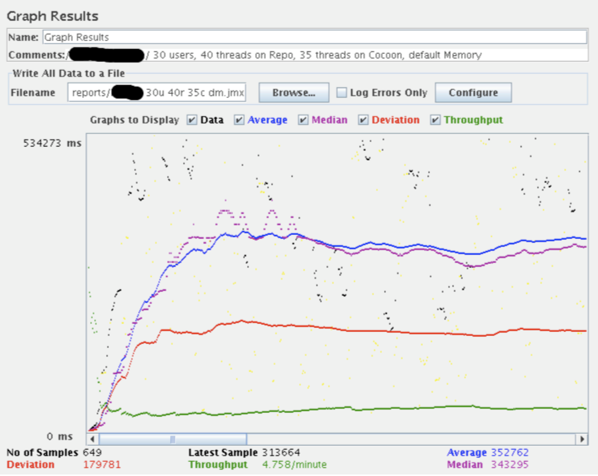
If you make more than one change at a time
you don’t know which change had what effect
which makes it more difficult to build a mental model of what is happening

I tested optimisations before playing with threads and memory
For each change, I ran the tests again

what tools?

on MacOSX I used :

- JMeter
- Terminal
- Console
- Activity Monitor
- Grab



Graph Results

Name: Graph Results

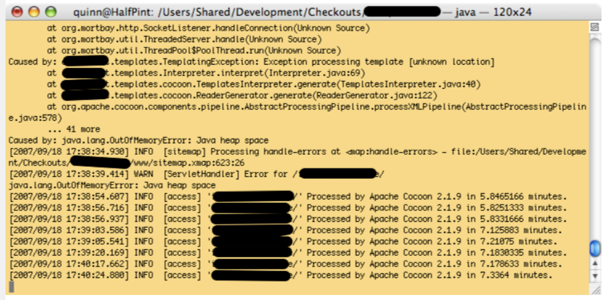
Comments: [redacted]/ 30 users, 40 threads on Repo, 35 threads on Cocoon, default Memory

Write All Data to a File

Filename: reports/[redacted]30u 40r 35c dm.jmx

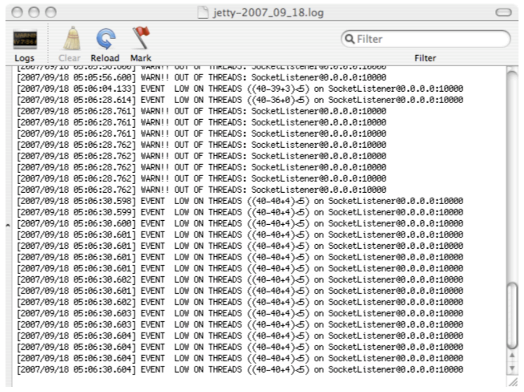
Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 649 | Latest Sample 313664 | Average 352762
Deviation 179781 | Throughput 4.758/minute | Median 343295



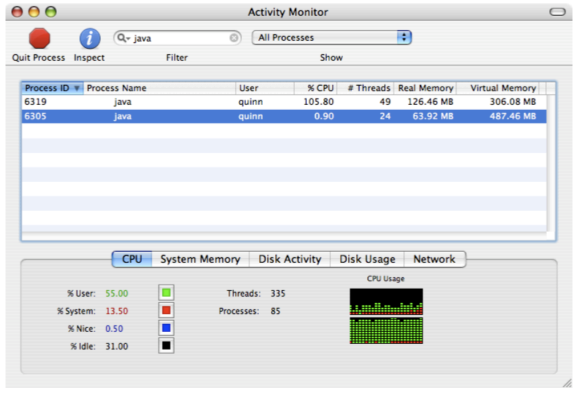
quinn@HalfPint: /Users/Shared/Development/Checkouts/[redacted]— java — 120x24

```
        at org.mortbay.http.SocketListener.handleConnection(Unknown Source)
        at org.mortbay.util.ThreadedServer.handle(Unknown Source)
        at org.mortbay.util.ThreadPool$PoolThread.run(Unknown Source)
Caused by: [redacted].templates.TemplatingException: Exception processing template [unknown location]
        at [redacted].templates.Interpreter.interpret(Interpreter.java:69)
        at [redacted].templates.cocoon.TemplatesInterpreter.generate(TemplatesInterpreter.java:40)
        at [redacted].templates.cocoon.ReaderGenerator.generate(ReaderGenerator.java:122)
        at org.apache.cocoon.components.pipeline.AbstractProcessingPipeline.processXMLPipeline(AbstractProcessingPipelin
e.java:578)
        ... 41 more
Caused by: java.lang.OutOfMemoryError: Java heap space
[2007/09/18 17:38:34.930] INFO  [sitemap] Processing handle-errors at <map:handle-errors> - file:/Users/Shared/Developme
nt/Checkouts/[redacted]/www/sitemap.xmap:623:26
[2007/09/18 17:38:39.414] WARN  [ServletHandler] Error for /[redacted]/
java.lang.OutOfMemoryError: Java heap space
[2007/09/18 17:38:54.607] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 5.8465166 minutes.
[2007/09/18 17:38:56.716] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 5.8251333 minutes.
[2007/09/18 17:38:56.937] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 5.8331666 minutes.
[2007/09/18 17:39:03.586] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.125883 minutes.
[2007/09/18 17:39:05.541] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.21075 minutes.
[2007/09/18 17:39:20.169] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.1830335 minutes.
[2007/09/18 17:40:17.662] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.178633 minutes.
[2007/09/18 17:40:24.880] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.3364 minutes.
```



jetty-2007_09_18.log

```
[2007/09/18 05:05:56.600] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:04.133] EVENT  LOW ON THREADS ((40-39+3)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.614] EVENT  LOW ON THREADS ((40-36+0)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.761] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.761] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.761] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.762] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.762] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.762] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.762] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:28.762] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.598] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.599] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.600] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.601] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.601] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.601] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.601] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.602] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.601] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.602] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.603] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.603] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.604] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.604] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.604] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
[2007/09/18 05:06:30.604] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
```



Activity Monitor

Filter: java

Show: All Processes

Process ID	Process Name	User	% CPU	# Threads	Real Memory	Virtual Memory
6319	java	quinn	105.80	49	126.46 MB	306.08 MB
6305	java	quinn	0.90	24	63.92 MB	487.46 MB

% User: 55.00
% System: 13.50
% Nice: 0.50
% Idle: 31.00

Threads: 335
Processes: 85

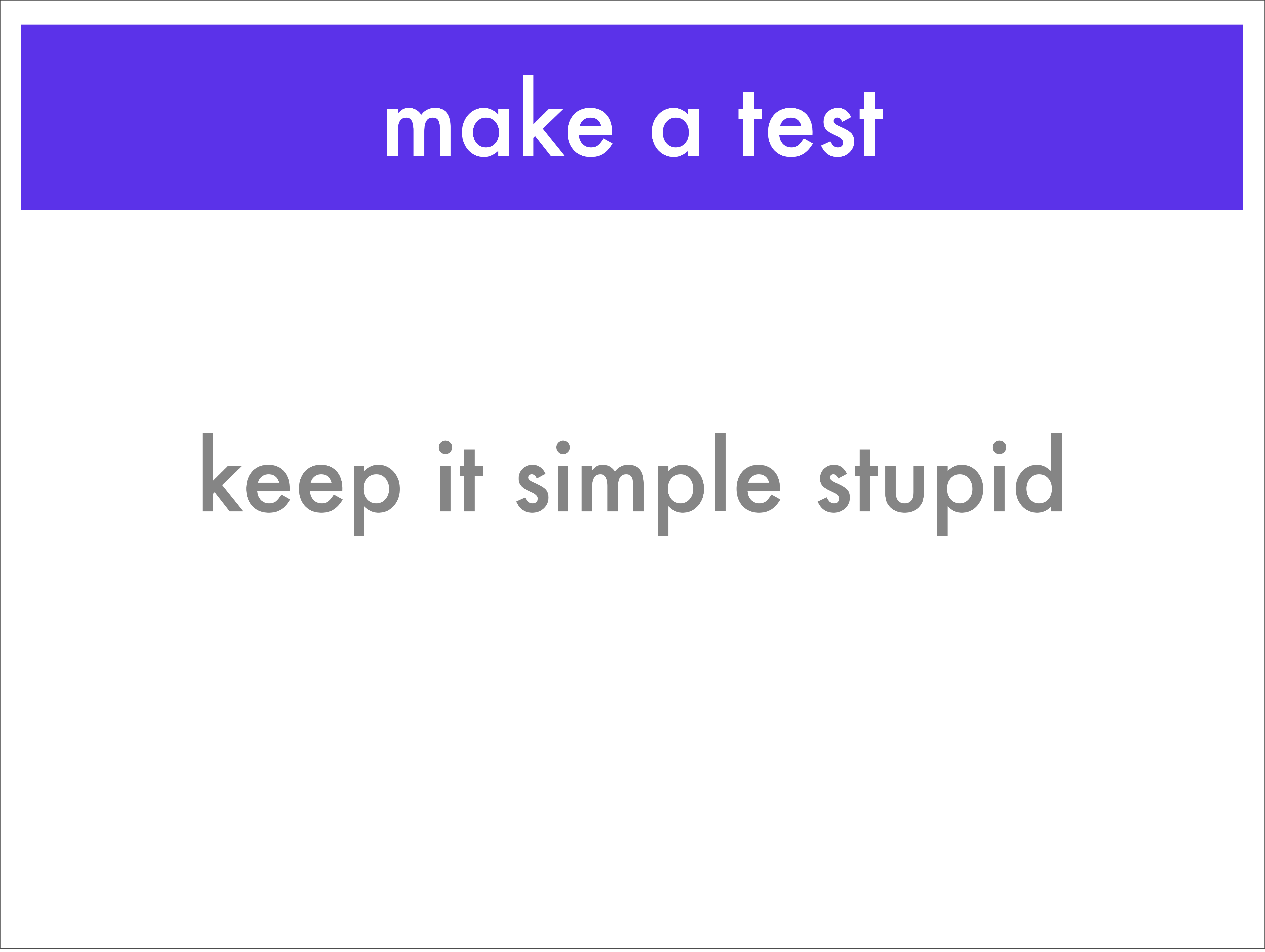
[CLICK] I found it was fine to run JMeter on the server for running low level tests. [CLICK]
It is better to run JMeter on another machine though.
For very high throughput testing, you can aggregate results from many JMeter clients.
[CLICK] I can watch Cocoon in its Terminal window. [CLICK]
[CLICK] I used the Console to filter and watch Jetty's logs. [CLICK]
[CLICK] In Activity Monitor I can track threads, memory and cpu usage.
[CLICK] Grab to capture screenshots.

setup

what do I do to get ready?

photo credit: Fabio Venni

Good preparation is everything
In diving I was taught : plan the dive, then dive the plan
this applies to good testing as well

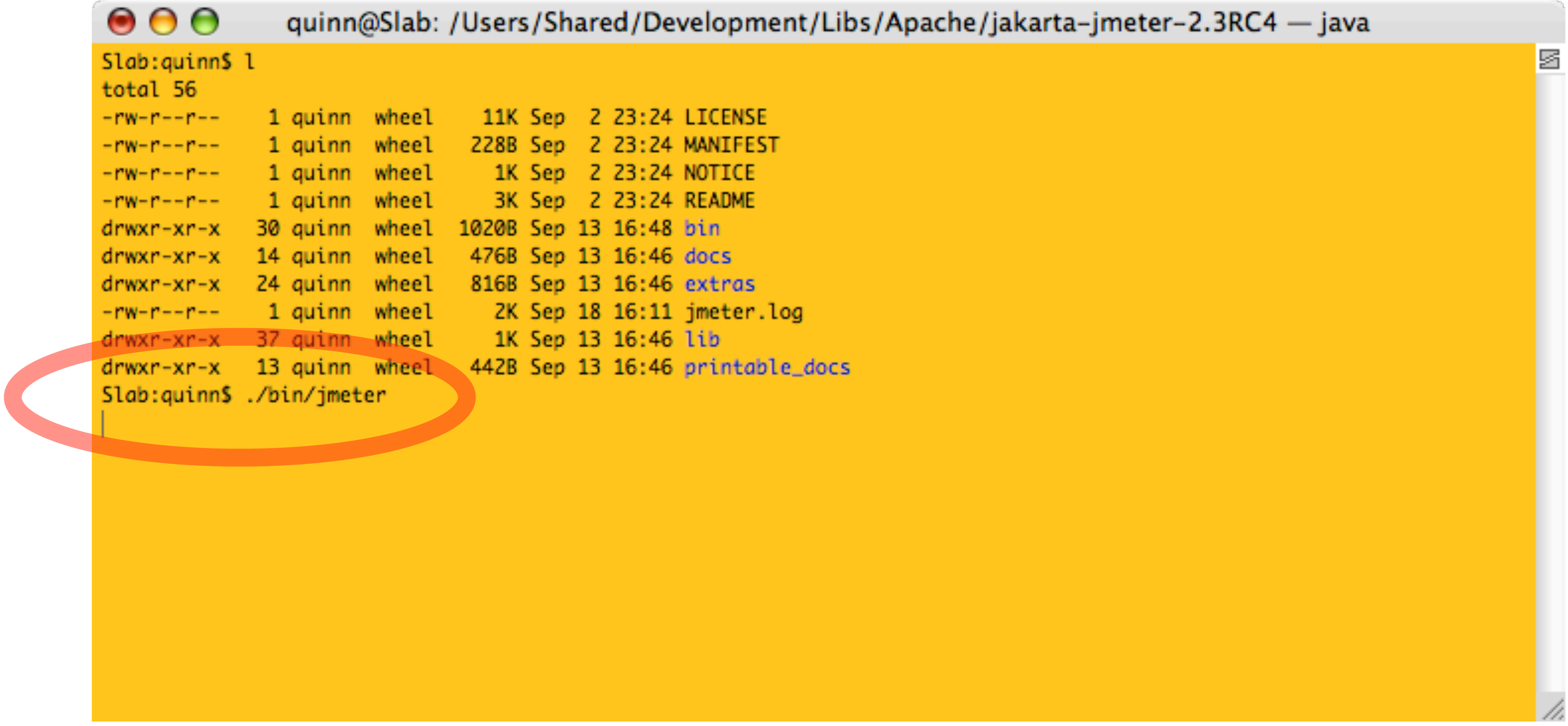


make a test

keep it simple stupid

unless you can make your tests completely reproducible, you are wasting your time

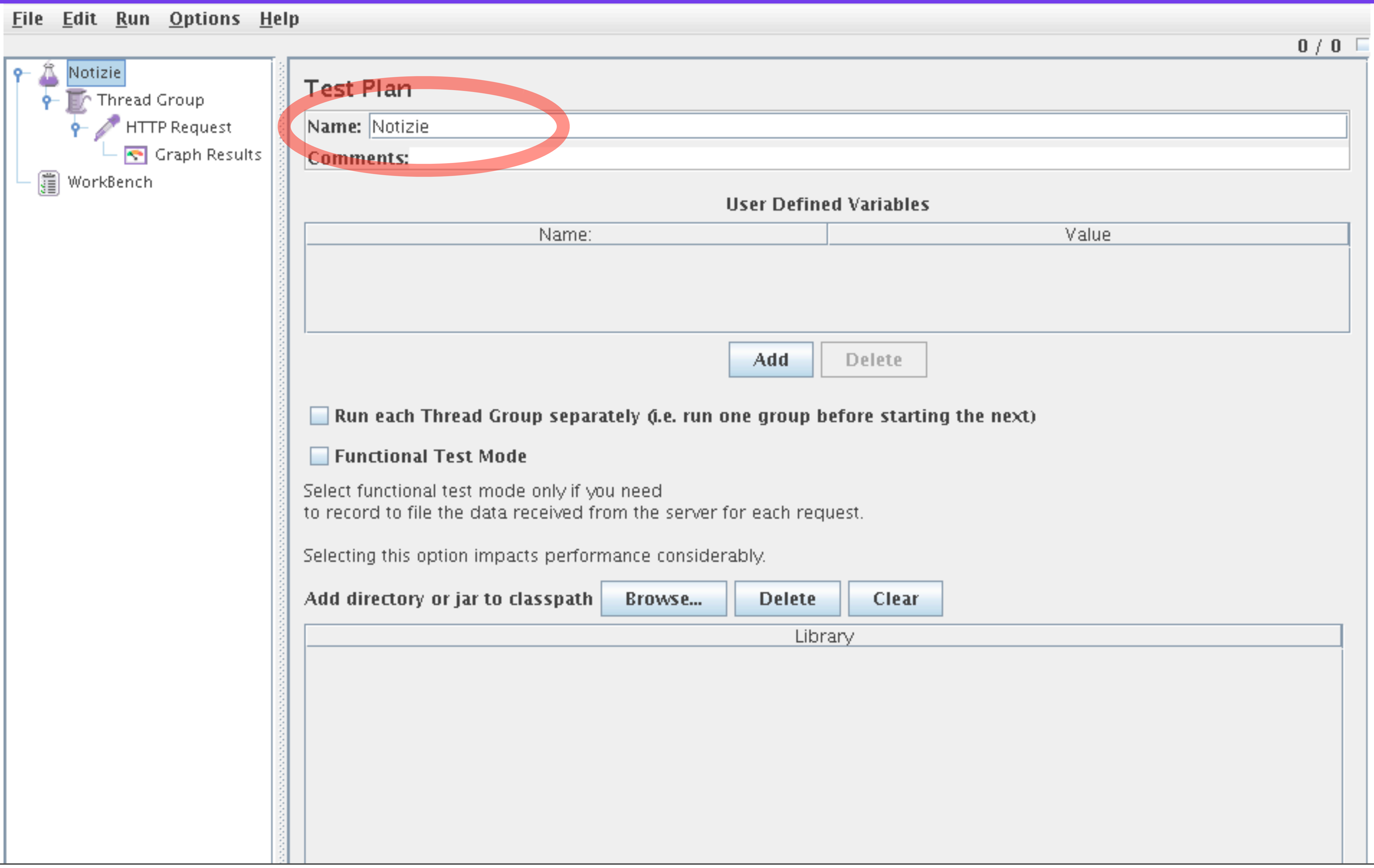
start JMeter



quinn@Slab: /Users/Shared/Development/Libs/Apache/jakarta-jmeter-2.3RC4 — java

```
Slab:quinn$ l
total 56
-rw-r--r--    1 quinn  wheel    11K Sep  2 23:24 LICENSE
-rw-r--r--    1 quinn  wheel   228B Sep  2 23:24 MANIFEST
-rw-r--r--    1 quinn  wheel    1K Sep  2 23:24 NOTICE
-rw-r--r--    1 quinn  wheel    3K Sep  2 23:24 README
drwxr-xr-x   30 quinn  wheel  1020B Sep 13 16:48 bin
drwxr-xr-x   14 quinn  wheel   476B Sep 13 16:46 docs
drwxr-xr-x   24 quinn  wheel   816B Sep 13 16:46 extras
-rw-r--r--    1 quinn  wheel     2K Sep 18 16:11 jmeter.log
drwxr-xr-x   37 quinn  wheel     1K Sep 13 16:46 lib
drwxr-xr-x   13 quinn  wheel   442B Sep 13 16:46 printable_docs
Slab:quinn$ ./bin/jmeter
```

start a plan



Test Plan

Name: Notizie

Comments:

User Defined Variables

Name:	Value

□ Run each Thread Group separately (i.e. run one group before starting the next)

□ Functional Test Mode

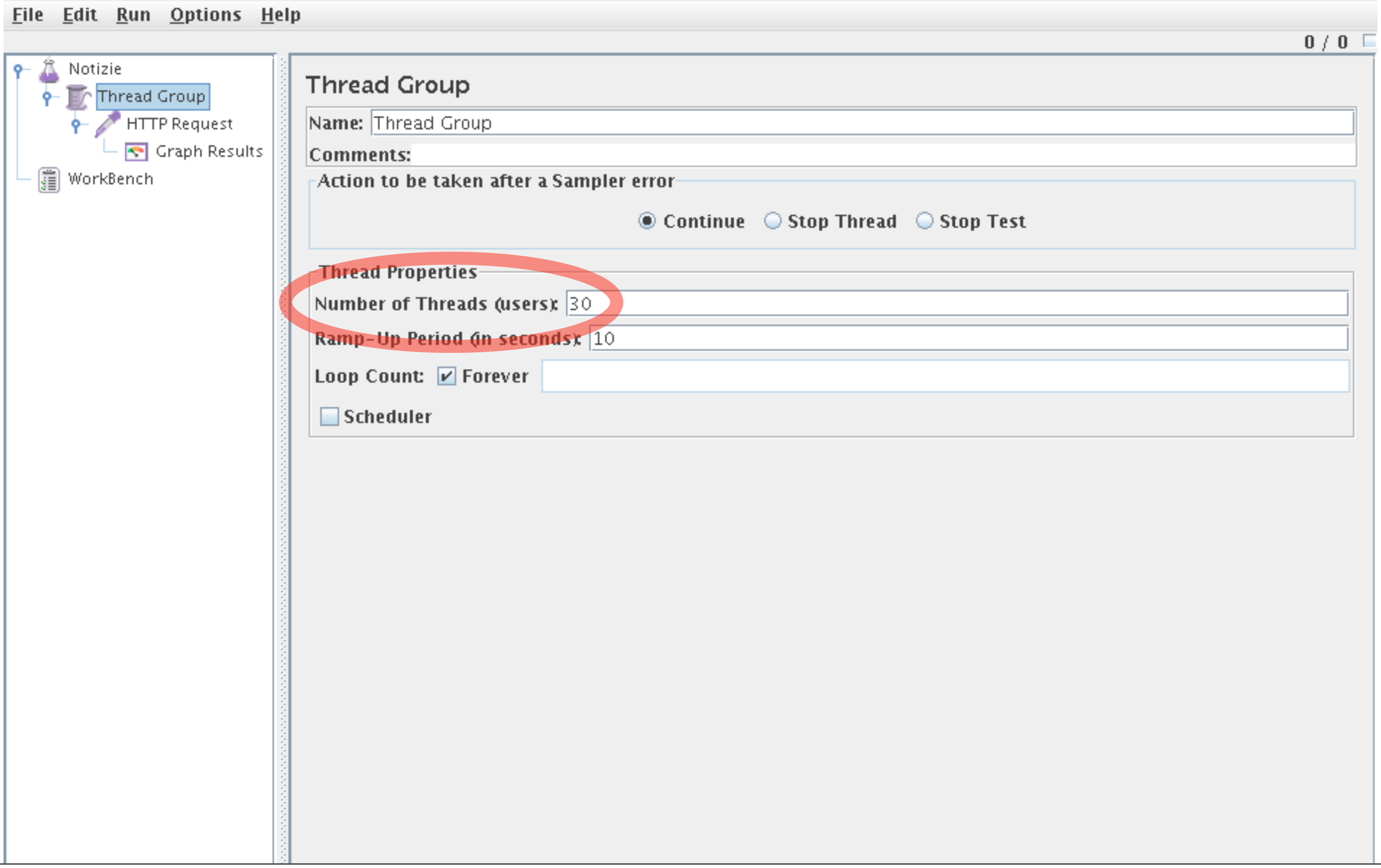
Select functional test mode only if you need
to record to file the data received from the server for each request.

Selecting this option impacts performance considerably.

Add directory or jar to classpath

Library

threads



Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

● Continue ○ Stop Thread ○ Stop Test

Thread Properties

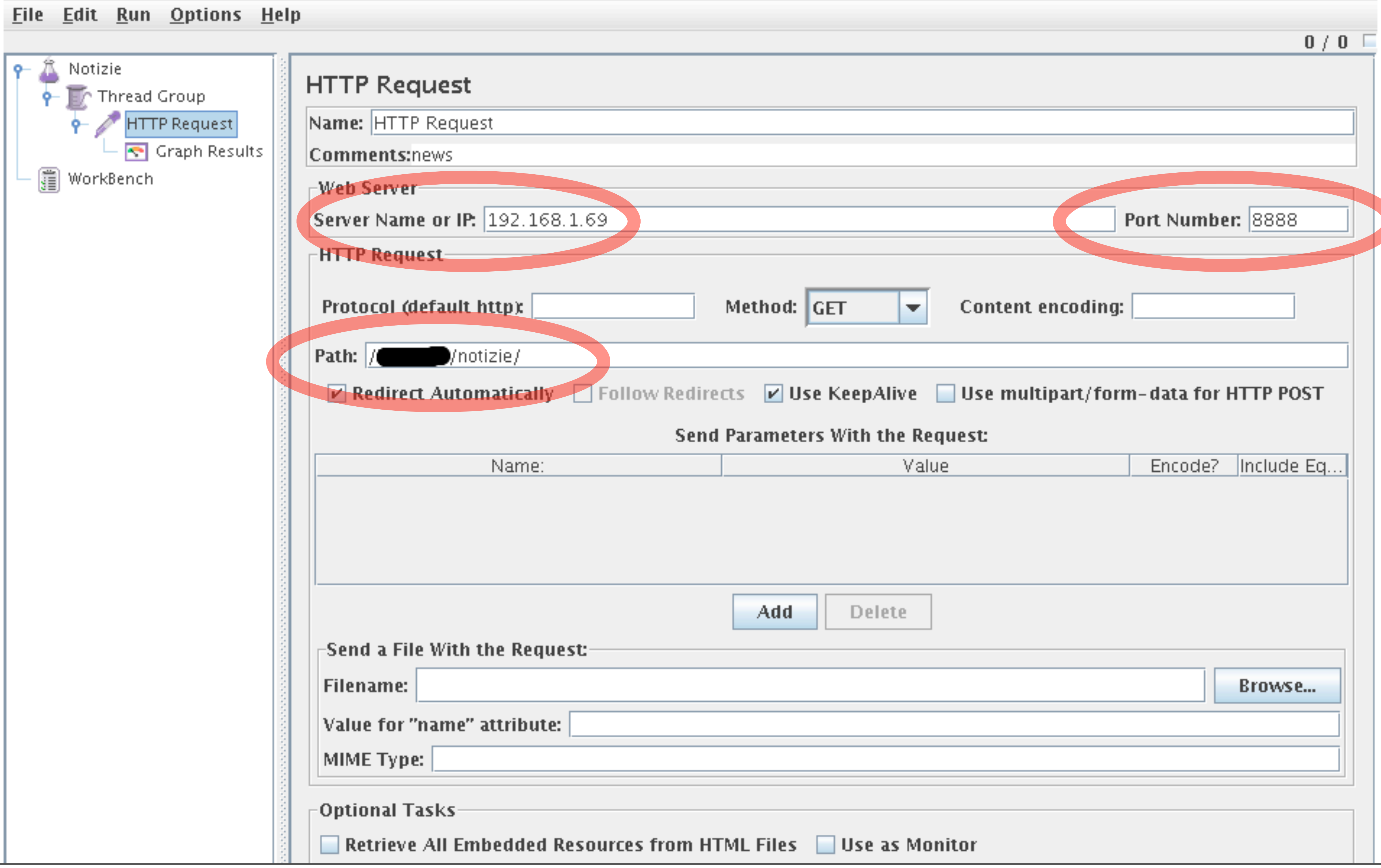
Number of Threads (users): 30

Ramp-Up Period (in seconds): 10

Loop Count: ☑ Forever

□ Scheduler

request



HTTP Request

Name: HTTP Request

Comments: news

Web Server

Server Name or IP: 192.168.1.69

Port Number: 8888

HTTP Request

Protocol (default http):

Method: GET

Content encoding:

Path: /[redacted]/notizie/

☑ Redirect Automatically □ Follow Redirects ☑ Use KeepAlive □ Use multipart/form-data for HTTP POST

Send Parameters With the Request:

Name:	Value	Encode?	Include Eq...

Send a File With the Request:

Filename:

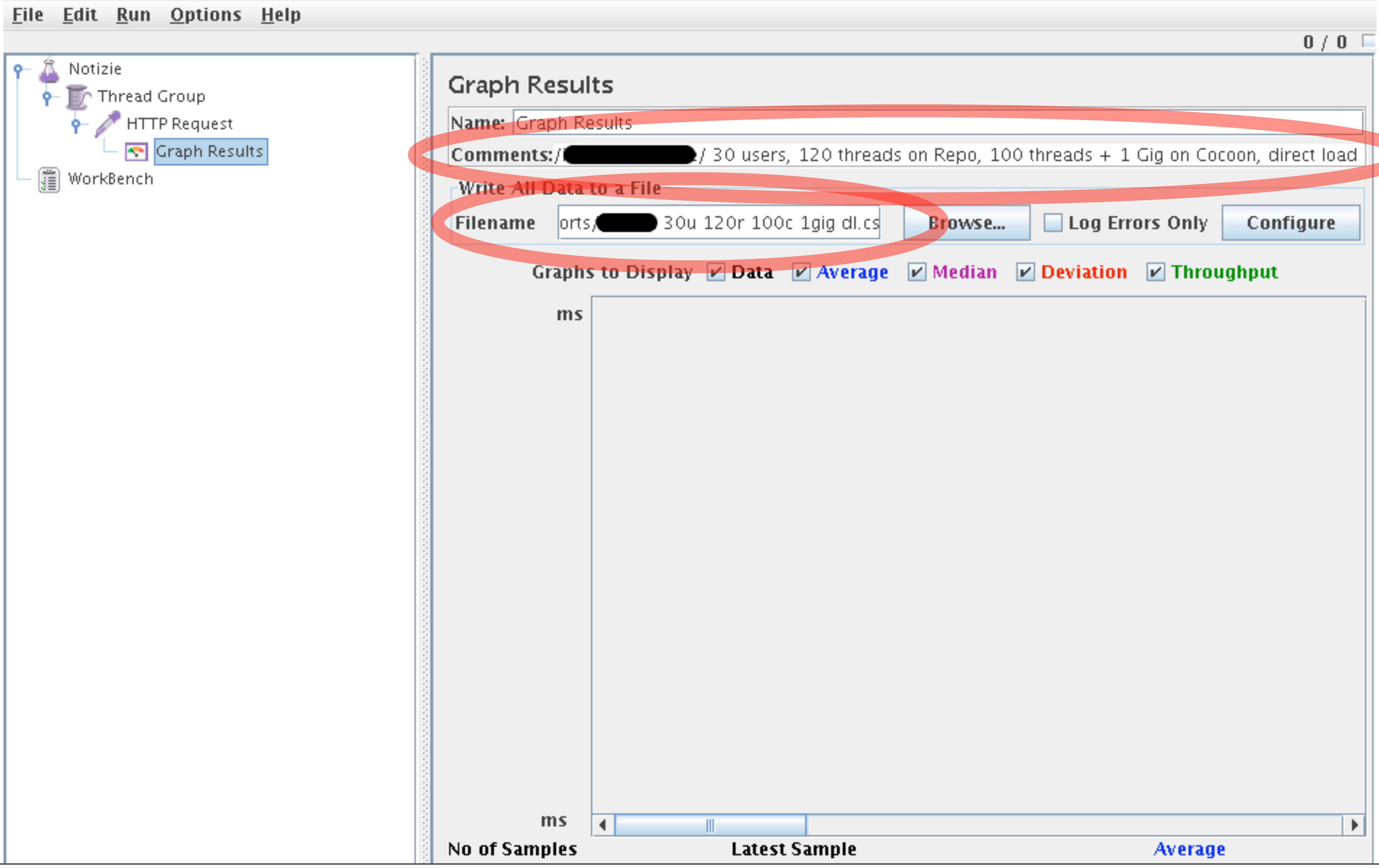
Value for "name" attribute:

MIME Type:

Optional Tasks

□ Retrieve All Embedded Resources from HTML Files □ Use as Monitor

recording



Graph Results

Name: Graph Results

Comments: /[redacted]/ 30 users, 120 threads on Repo, 100 threads + 1 Gig on Cocoon, direct load

Write All Data to a File

Filename orts/[redacted] 30u 120r 100c 1gig dl.cs

record enough detail about what you are changing for each comparative test, so that you can :

a) know which test you performed

b) can reproduce it accurately

check everything

- content of datafeeds
- behaviour of pipelines / urls
- outcome of transformations

Check every stage,
so you know everything is working properly
and set up as you expected,
before you start testing.
Otherwise you might not be testing what you think you are.

caching

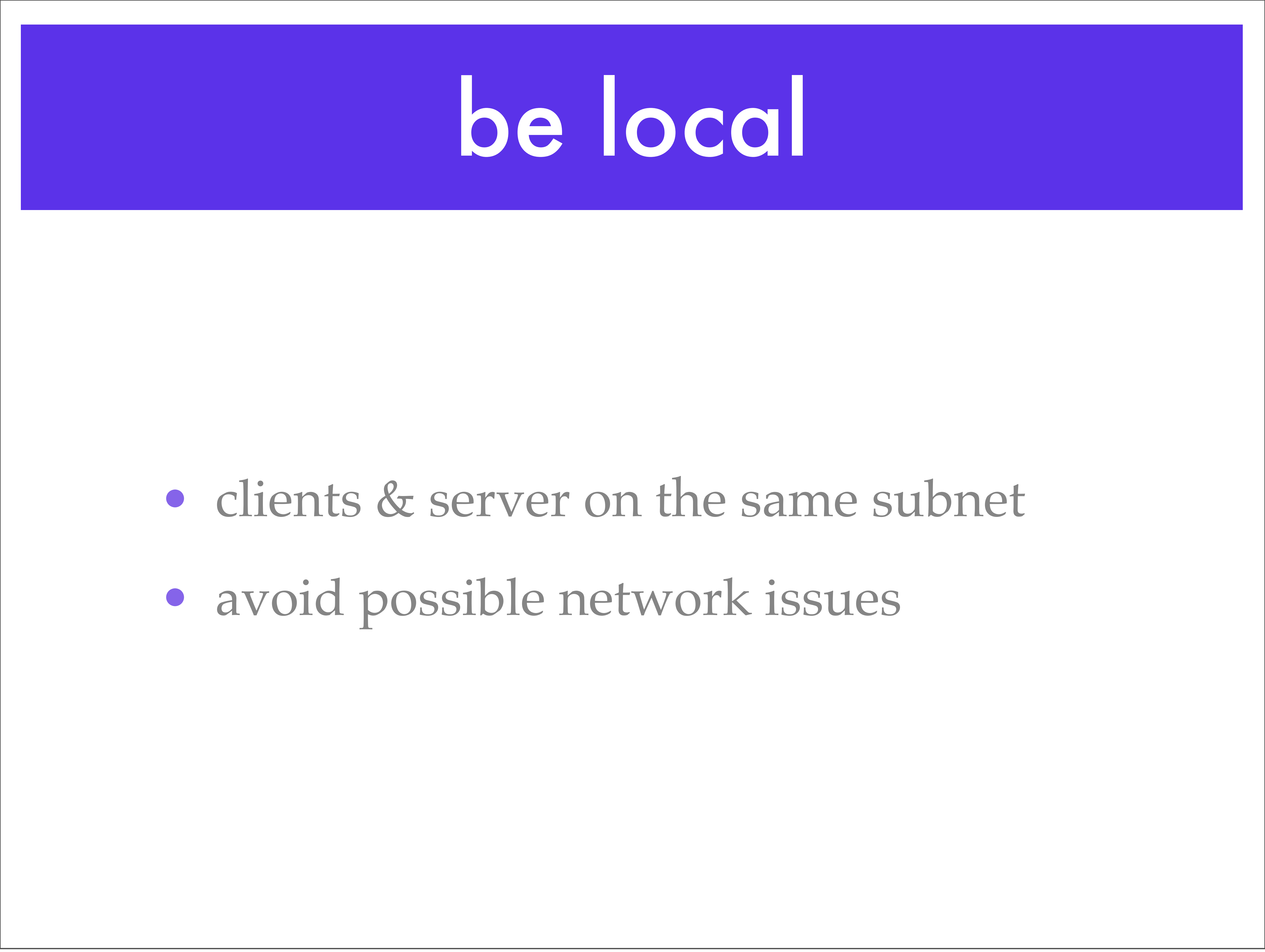
- repository?
- datasource?
- pipeline?

Make sure you know the state of all the caching,
so it matches what you want to test.

warm-up

- hit your test page(s) with a browser
- last chance for a visual check
- get your server up to speed

Warm-up your server.
Hit the pages you want to test in a browser to make sure they work.



be local

- clients & server on the same subnet
- avoid possible network issues

Be on the server's local subnet.

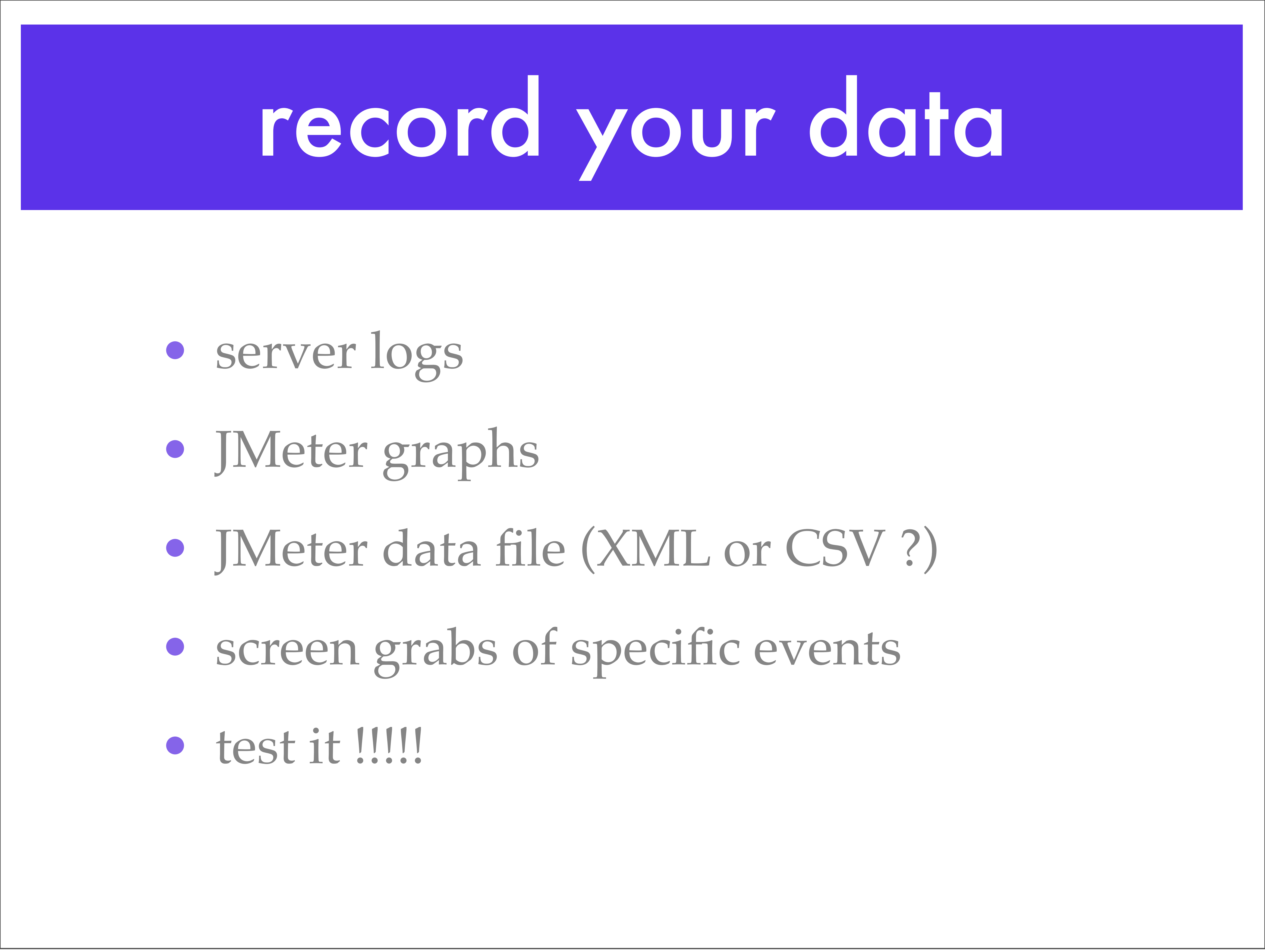
enough clients?

- creating many threads
- creates its own overhead

Have enough clients
to perform the level of testing you need.

quit processes

- everything that is unnecessary
- periodic processes skew your results



record your data

- server logs
- JMeter graphs
- JMeter data file (XML or CSV ?)
- screen grabs of specific events
- test it !!!!!

Prepare properly,
to make sure all of your data will be recorded.
Grab relevant sections of logs.
Grab graphs after a run.
Choose what data to write to a file.

running tests

time for a cup of tea?

photo credit: thespeak

No, you have lots to do.

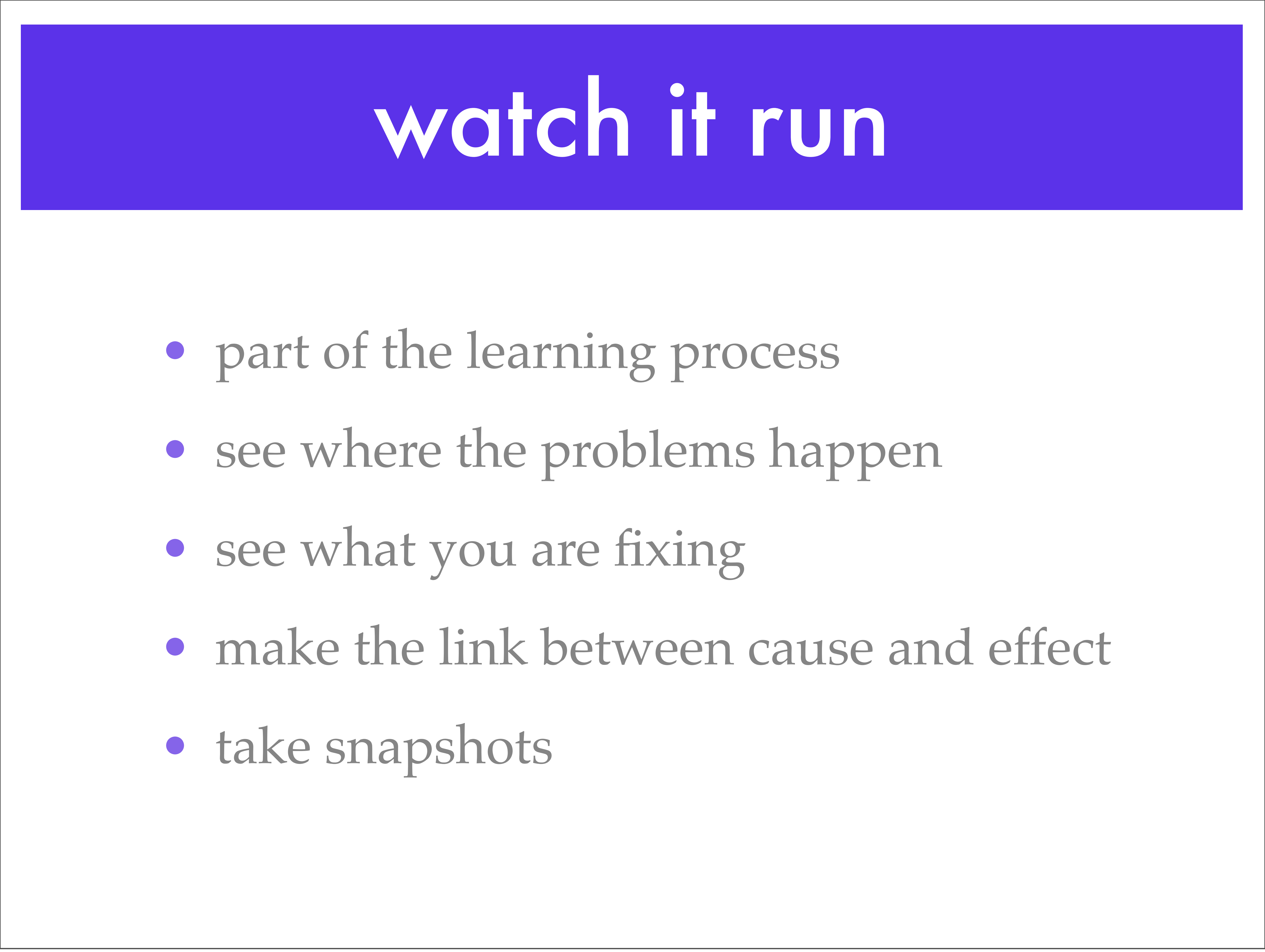
run-time data

- clear log windows on each run
- keep it organised
- use a file or folder naming convention

Record the real-time data you need.

I clear the log windows for each run
so after the run is finished,
I can easily scroll back looking for exceptions etc.

Use naming conventions for multiple runs.

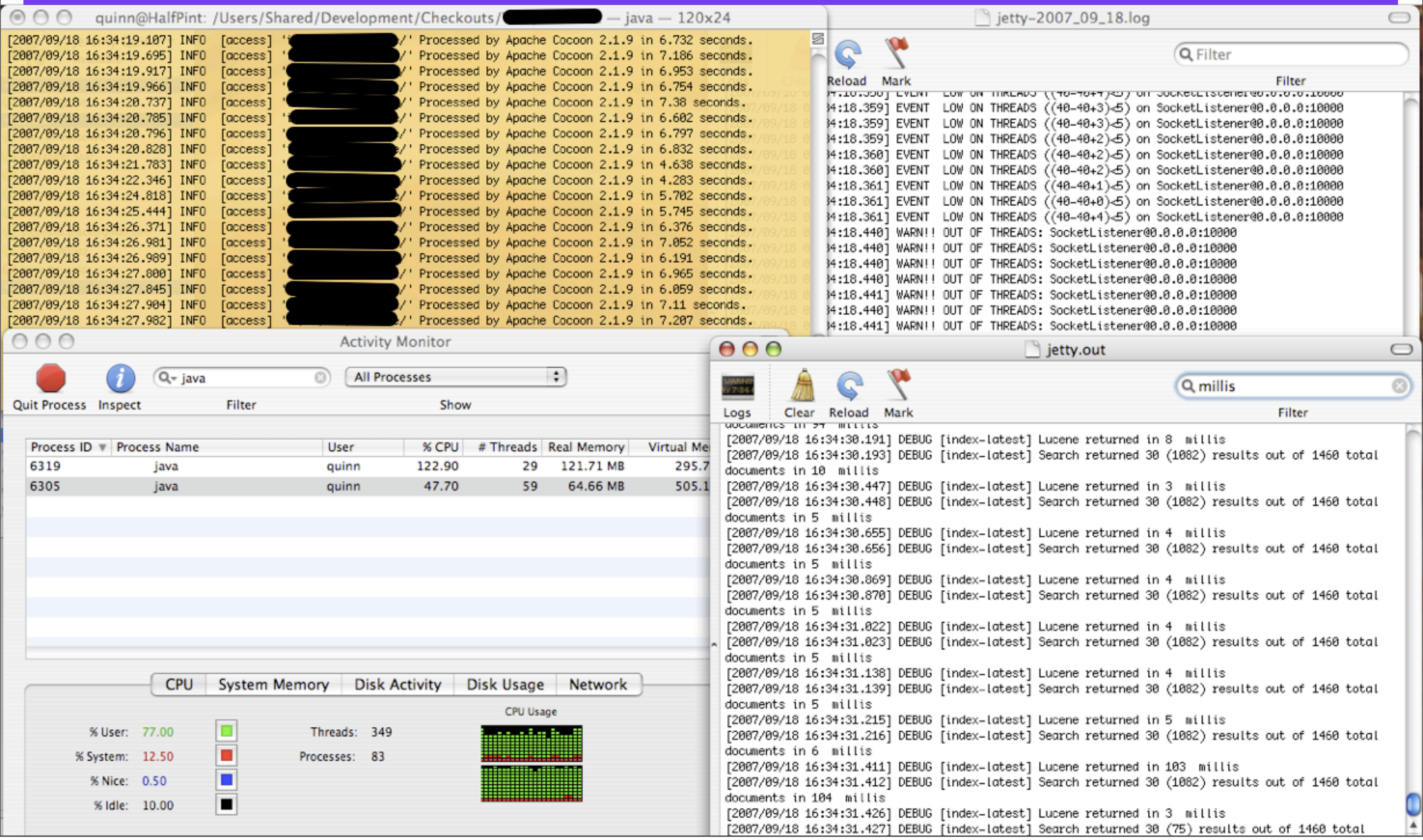


watch it run

- part of the learning process
- see where the problems happen
- see what you are fixing
- make the link between cause and effect
- take snapshots

Watch the tools as the tests run.
Learn to read the graphs.

my screen



quinn@HalfPint: /Users/Shared/Development/Checkouts/[redacted] — java — 120x24

```
[2007/09/18 16:34:19.107] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.732 seconds.
[2007/09/18 16:34:19.695] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.186 seconds.
[2007/09/18 16:34:19.917] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.953 seconds.
[2007/09/18 16:34:19.966] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.754 seconds.
[2007/09/18 16:34:20.737] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.38 seconds.
[2007/09/18 16:34:20.785] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.602 seconds.
[2007/09/18 16:34:20.796] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.797 seconds.
[2007/09/18 16:34:20.828] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.832 seconds.
[2007/09/18 16:34:21.783] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 4.638 seconds.
[2007/09/18 16:34:22.346] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 4.283 seconds.
[2007/09/18 16:34:24.818] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 5.702 seconds.
[2007/09/18 16:34:25.444] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 5.745 seconds.
[2007/09/18 16:34:26.371] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.376 seconds.
[2007/09/18 16:34:26.981] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.052 seconds.
[2007/09/18 16:34:26.989] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.191 seconds.
[2007/09/18 16:34:27.800] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.965 seconds.
[2007/09/18 16:34:27.845] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 6.059 seconds.
[2007/09/18 16:34:27.904] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.11 seconds.
[2007/09/18 16:34:27.982] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 7.207 seconds.
```

jetty-2007_09_18.log

```
4:18.359] EVENT  LOW ON THREADS ((40-40+3)<5) on SocketListener@0.0.0.0:10000
4:18.359] EVENT  LOW ON THREADS ((40-40+3)<5) on SocketListener@0.0.0.0:10000
4:18.359] EVENT  LOW ON THREADS ((40-40+2)<5) on SocketListener@0.0.0.0:10000
4:18.360] EVENT  LOW ON THREADS ((40-40+2)<5) on SocketListener@0.0.0.0:10000
4:18.360] EVENT  LOW ON THREADS ((40-40+2)<5) on SocketListener@0.0.0.0:10000
4:18.361] EVENT  LOW ON THREADS ((40-40+1)<5) on SocketListener@0.0.0.0:10000
4:18.361] EVENT  LOW ON THREADS ((40-40+0)<5) on SocketListener@0.0.0.0:10000
4:18.361] EVENT  LOW ON THREADS ((40-40+4)<5) on SocketListener@0.0.0.0:10000
4:18.440] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.440] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.440] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.440] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.441] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.440] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
4:18.441] WARN!! OUT OF THREADS: SocketListener@0.0.0.0:10000
```

Activity Monitor

Process ID	Process Name	User	% CPU	# Threads	Real Memory	Virtual Me
6319	java	quinn	122.90	29	121.71 MB	295.7
6305	java	quinn	47.70	59	64.66 MB	505.1

% User: 77.00 🟩
% System: 12.50 🟥
% Nice: 0.50 🟦
% Idle: 10.00 ■

Threads: 349
Processes: 83

jetty.out

```
[2007/09/18 16:34:30.191] DEBUG [index-latest] Lucene returned in 8  millis
[2007/09/18 16:34:30.193] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 10  millis
[2007/09/18 16:34:30.447] DEBUG [index-latest] Lucene returned in 3  millis
[2007/09/18 16:34:30.448] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 5  millis
[2007/09/18 16:34:30.655] DEBUG [index-latest] Lucene returned in 4  millis
[2007/09/18 16:34:30.656] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 5  millis
[2007/09/18 16:34:30.869] DEBUG [index-latest] Lucene returned in 4  millis
[2007/09/18 16:34:30.870] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 5  millis
[2007/09/18 16:34:31.022] DEBUG [index-latest] Lucene returned in 4  millis
[2007/09/18 16:34:31.023] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 5  millis
[2007/09/18 16:34:31.138] DEBUG [index-latest] Lucene returned in 4  millis
[2007/09/18 16:34:31.139] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 5  millis
[2007/09/18 16:34:31.215] DEBUG [index-latest] Lucene returned in 5  millis
[2007/09/18 16:34:31.216] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 6  millis
[2007/09/18 16:34:31.411] DEBUG [index-latest] Lucene returned in 103  millis
[2007/09/18 16:34:31.412] DEBUG [index-latest] Search returned 30 (1082) results out of 1460 total
documents in 104  millis
[2007/09/18 16:34:31.426] DEBUG [index-latest] Lucene returned in 3  millis
[2007/09/18 16:34:31.427] DEBUG [index-latest] Search returned 30 (75) results out of 1460 total
```

I can follow errors and timings in Cocoon’s output.
Errors and query times changing for the repo, in Jetty’s logs.
Thread, memory and cpu usage in Activity Monitor.

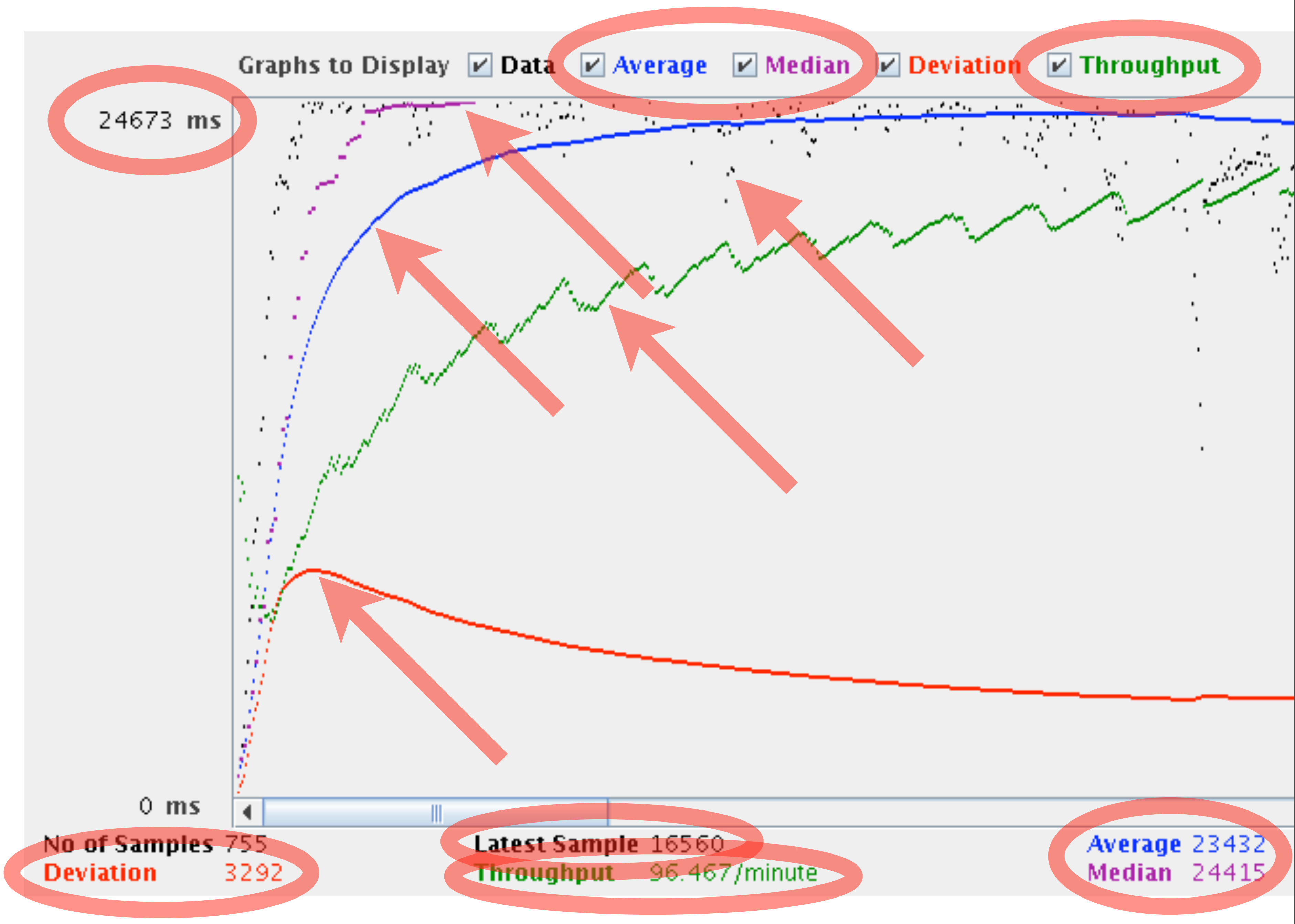
interpreting the results

what does it all mean?

photo credit: woneffe

reading the graph

- load speed
- throughput
- deviation
- hits



Graphs to Display ☑ Data ☑ Average ☑ Median ☑ Deviation ☑ Throughput

No of Samples 755
Deviation 3292

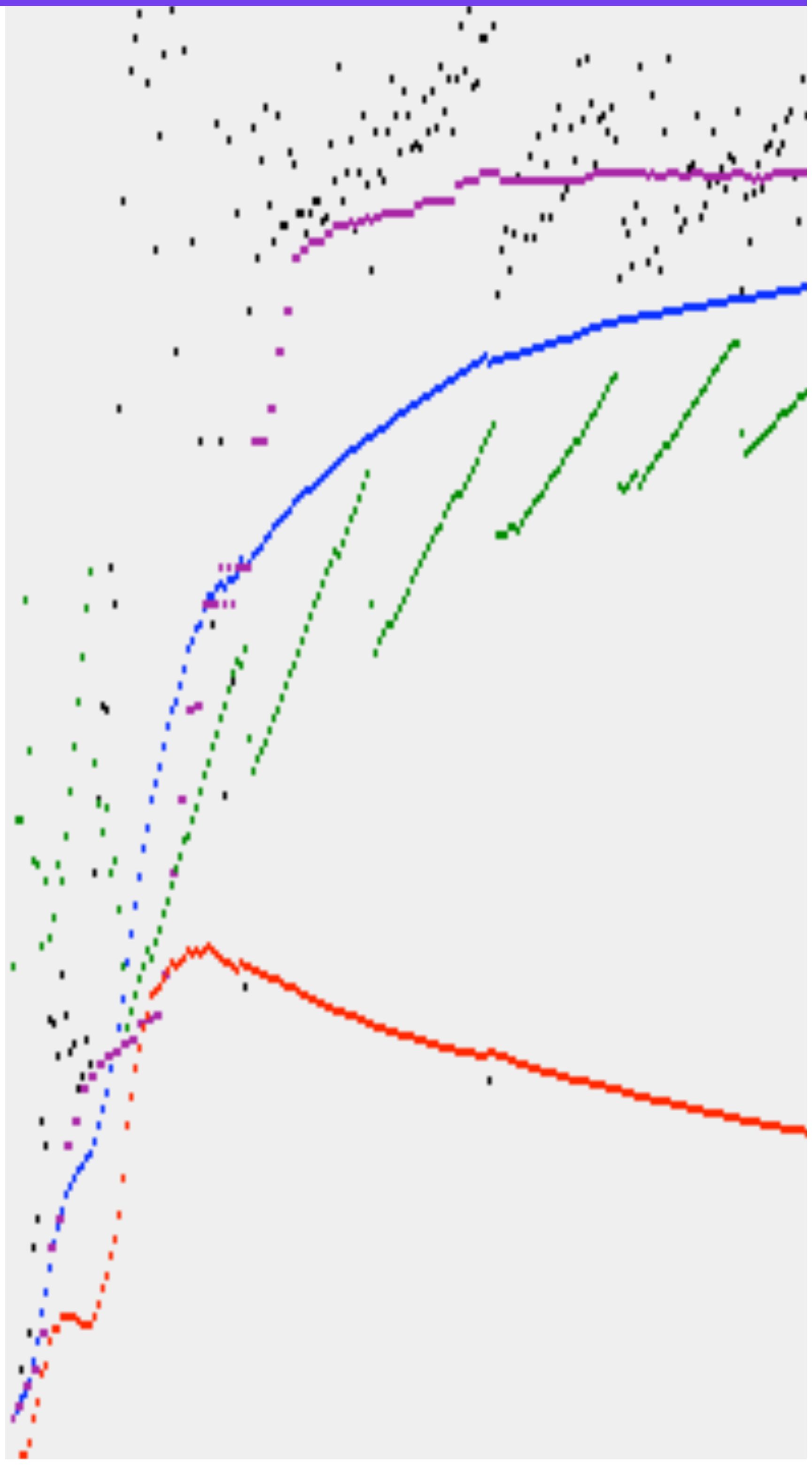
Latest Sample 16560
Throughput 96.467/minute

Average 23432
Median 24415

[CLICK] Page load speeds in milliseconds.
[CLICK] Throughput in pages per second.
[CLICK] The deviation (variability) of the load speed in milliseconds
[CLICK] The black dots are individual times
[CLICK] Sometimes JMeter draws data off the graph
[CLICK] Unless you are testing on real hardware, these speeds have no meaning in isolation but are useful for comparison

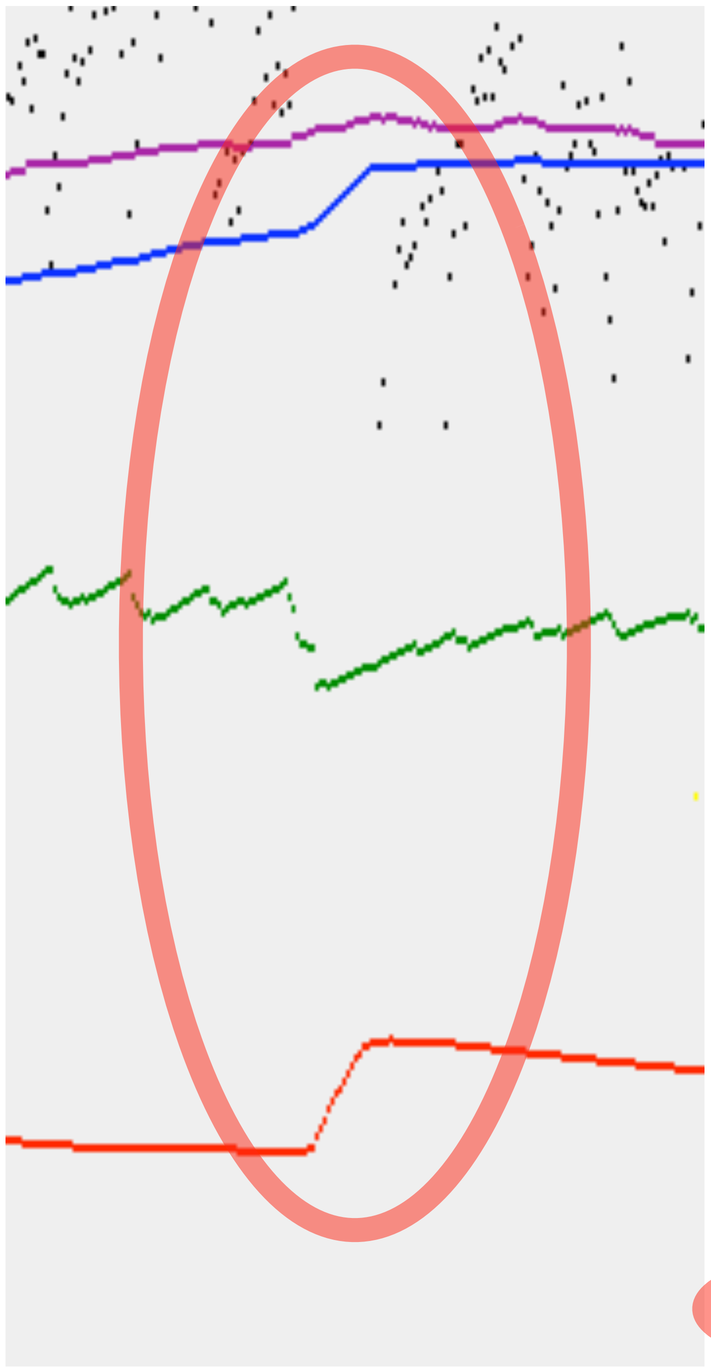
chaotic start

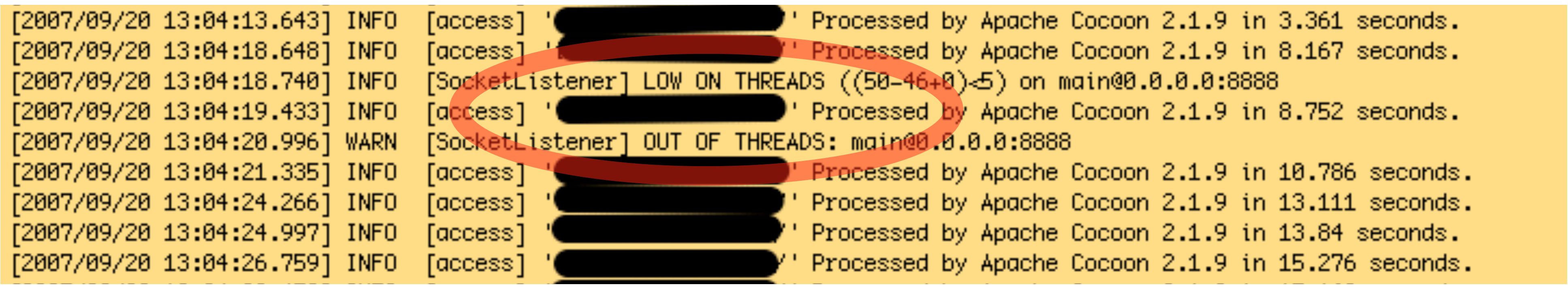
- test for long enough
  - JMeter ramps threads
  - JVMs warm up
  - Servlet adds threads



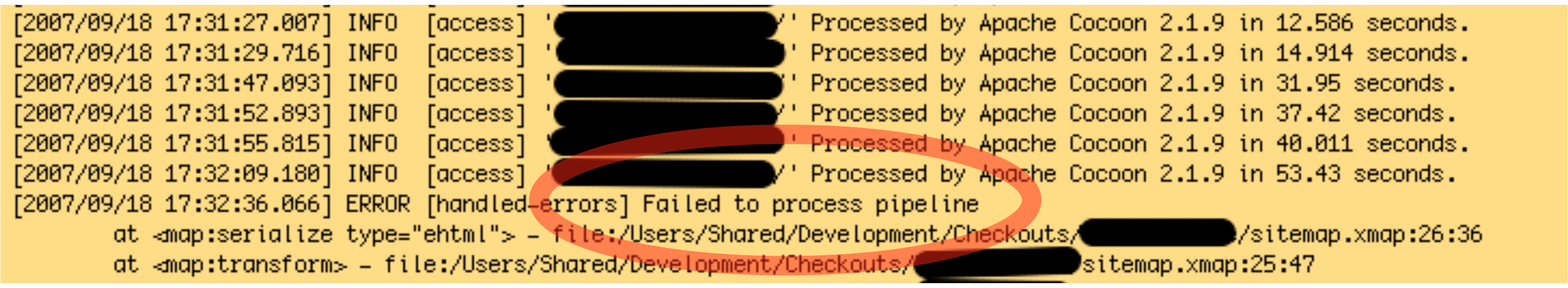
Your tests need to run for long enough for you to get meaningful results.
I ran a minimum of 600 hits, per test

shi‘appen

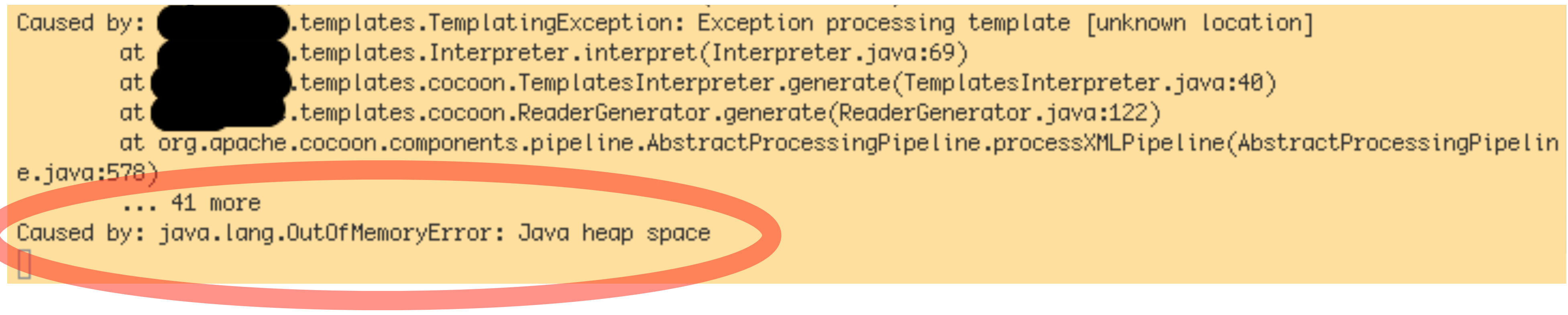




```
[2007/09/20 13:04:13.643] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 3.361 seconds.
[2007/09/20 13:04:18.648] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 8.167 seconds.
[2007/09/20 13:04:18.740] INFO  [SocketListener] LOW ON THREADS ((50-46+0)<5) on main@0.0.0.0:8888
[2007/09/20 13:04:19.433] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 8.752 seconds.
[2007/09/20 13:04:20.996] WARN  [SocketListener] OUT OF THREADS: main@0.0.0.0:8888
[2007/09/20 13:04:21.335] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 10.786 seconds.
[2007/09/20 13:04:24.266] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 13.111 seconds.
[2007/09/20 13:04:24.997] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 13.84 seconds.
[2007/09/20 13:04:26.759] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 15.276 seconds.
```



```
[2007/09/18 17:31:27.007] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 12.586 seconds.
[2007/09/18 17:31:29.716] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 14.914 seconds.
[2007/09/18 17:31:47.093] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 31.95 seconds.
[2007/09/18 17:31:52.893] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 37.42 seconds.
[2007/09/18 17:31:55.815] INFO  [access] '[redacted]' Processed by Apache Cocoon 2.1.9 in 40.011 seconds.
[2007/09/18 17:32:09.180] INFO  [access] '[redacted]/' Processed by Apache Cocoon 2.1.9 in 53.43 seconds.
[2007/09/18 17:32:36.066] ERROR [handled-errors] Failed to process pipeline
        at <map:serialize type="ehtml"> - file:/Users/Shared/Development/Checkouts/[redacted]/sitemap.xmap:26:36
        at <map:transform> - file:/Users/Shared/Development/Checkouts/[redacted]sitemap.xmap:25:47
```



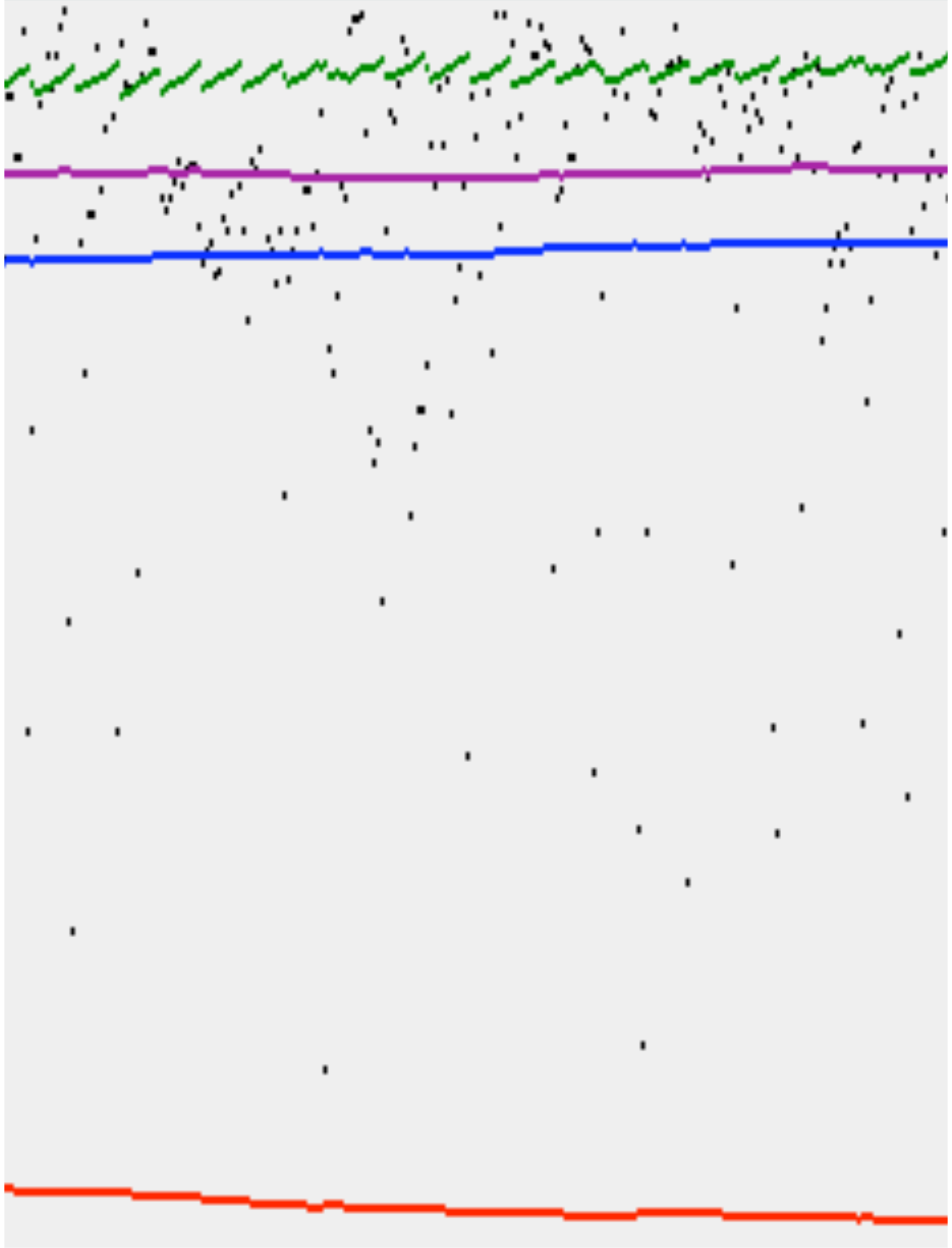
```
Caused by: [redacted].templates.TemplatingException: Exception processing template [unknown location]
        at [redacted].templates.Interpreter.interpret(Interpreter.java:69)
        at [redacted].templates.cocoon.TemplatesInterpreter.generate(TemplatesInterpreter.java:40)
        at [redacted].templates.cocoon.ReaderGenerator.generate(ReaderGenerator.java:122)
        at org.apache.cocoon.components.pipeline.AbstractProcessingPipeline.processXMLPipeline(AbstractProcessingPipelin
e.java:578)
        ... 41 more
Caused by: java.lang.OutOfMemoryError: Java heap space
```

[CLICK] Learn to relate problems on the server to its effect on the graph.

You may see the effects of:
[CLICK] exceptions [CLICK]
[CLICK] garbage collection [CLICK]

good signs

- speed flattening
- throughput flattening
- deviation dropping
- no exceptions ;)

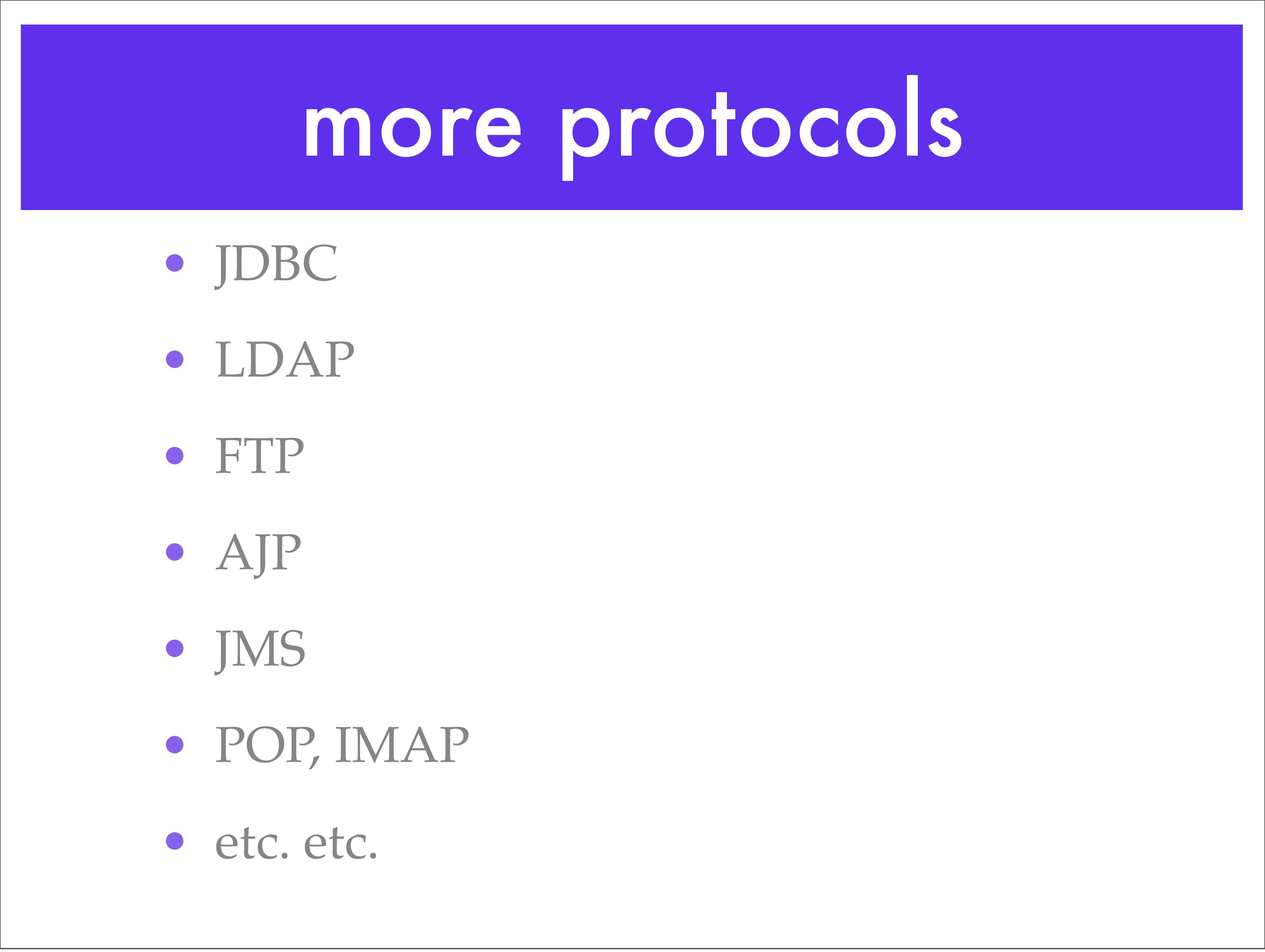


When the values on the graph begin to flatten out,
it shows that the system has become stable at that load.

more tests

- authorise
- fill in data
- upload files
- wrap tests in logic
- etc. etc.

I used the simplest possible configuration in JMeter.
I was only hitting one url.
It does a lot more.



more protocols

- JDBC
- LDAP
- FTP
- AJP
- JMS
- POP, IMAP
- etc. etc.

You can apply tests to many different parts of your infrastructure.

conclusions

- design to cope
- test early, test often
- plan well
- pay attention
- capture everything
- maintain a model