Lecture 25 — Load Testing

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

September 16, 2024

From Observation to Load Testing

We talked about observability, but we want to also talk about scalability.

Goal: scale number of users,

$$1 \rightarrow 100 \rightarrow 10000000.$$

"In the midnight hour, she cries 'more, more, more"' - Billy Idol

Not Hypothetical

CFO: Can this system handle 10$\times$ as many users as we have now?

My answer has to have numbers. Analysis and load testing.

UH, I WOULDN'T TAKE IT DOWN IF I WERE YOU.
IT'S A LOAD-BEARING POSTER.

Start With Why

Why are we doing this?

- A new system is being developed—limits & risks
- Workload increase; can we handle it?
- A sudden spike in workload is expected
- Uptime requirements: 99.99%
- Plan has checkbox “performance testing”

Reasons Why

Each implies a slightly different kind of testing.

- New system: average workload (plus buffer)
- Workload: what's our bottleneck?
- Spike: how to simulate it?
- Uptime: endurance!

(We'll ignore the last reason)

Stress Test?

Load testing is not the same as stress testing (to find the ultimate breaking strain...).

We're not going into it, but you can apply load testing strategies turned up to the max to get stress test results.

Making the Plan

Let’s make a test plan! Need answers to *who, what, where, when, & why*.

We already covered why and that tells us about what.

Who? We will!

When? Now!

Plan Signoff

How detailed the plan needs to be is going to depend on your organization, and the same applies for how many sign-offs (if any!) you need on the plan.

... let's not get sidetracked.

What Do We Need?

What will be tested?

How will it be tested?

How do we know if the test is passed?

What Workflows To Test

We cannot test everything; cost & effort are too high.

Do we already know what the rate-limiting steps are?

Maybe need to add observability first to identify those.

Excuse Me, I’m New Here

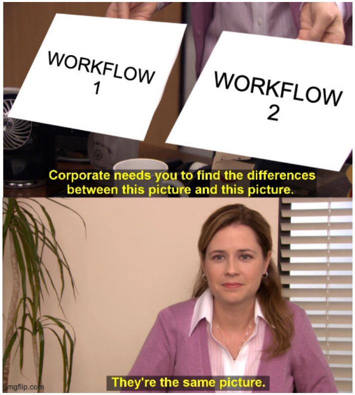
What if it’s a new system?

Critical is determined by product requirements:

- Computationally intensive work?
- User experience?
- External timing requirements?

Other Load Scenarios

If utilization is low it might not be clear what the bottlenecks are.
- Might need to guess and revise those later.


WORKFLOW 1
WORKFLOW 2
Corporate needs you to find the differences between this picture and this picture.
They're the same picture.
imgflip.com

If it's uptime requirements, we need endurance tests.
- We'll come back to those.

How to Test Them

Might need to provision additional (virtual) hardware.

Regular testing tools might not be correct; may need a script.

Dev and production aren’t the same, but how do they differ?

Maybe Don’t Do This

I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO I DO IT IN PRODUCTION
memegenerator.net

... Not quite what I mean.

Hardware Principle

Scalability testing $\neq$ QA testing.

Dev + QA on local computer (or CI infrastructure).
Your concerns: Is it right? Is it fast enough?

Fine, but it’s no way to test if it scales.

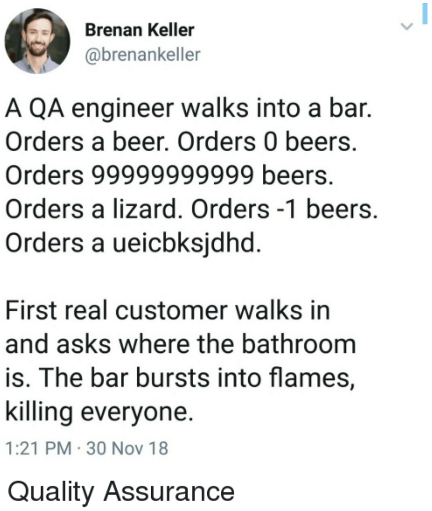
Hardware Principle

Test on prod-like machines: low-end systems have very different limiting factors.

Your laptop: limited by 16GB of RAM.
Prod server: 128GB of RAM.

So you might spend a great deal of time worrying about RAM usage and it just doesn’t matter.

Reality Principle



Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

1:21 PM · 30 Nov 18

Quality Assurance

Reality Principle

Use a “real” workload.
Maybe not actual live customer data, but try to come close.

Limited test data/scenarios $\Rightarrow$ inaccurate test results.

Your tests run summary reports occasionally…
your users might run them every hour.

Volume Principle

“More is the new more.”

Lighter workloads OK for regression testing.

To see how your system performs under pressure, you actually need to put it under pressure.

Can simulate pressure by limiting RAM or running something very CPU-intensive concurrently, but it’s just not the same.

Reproducibility and Regression Testing

Science rule 1: results need to be reproducible!

There is likely some variance in runs during load testing due to the natural randomness of the OS, scheduler, luck, etc.

Endurance Tests—How Long?

You're familiar with the idea that endurance is different from peak workload.

Jacob Reinecker
@jsreinecker

Starting your day with an early morning run is a great way to make sure your day can't get any worse than it started

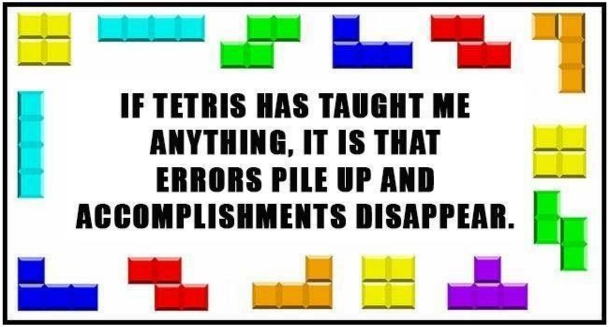
Let’s Go For A Run

Can I [JZ] run at 10 km/h for...

- 1 minute? Yes, of course!
- 30 minutes (5 km)? Yes!
- 60 minutes (10 km)? Yes (with difficulty).
- 4 hours (40 km)? No.

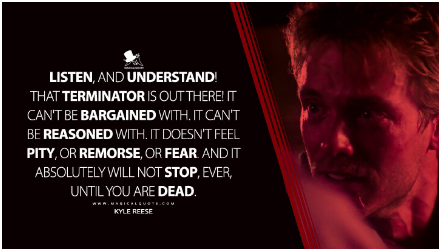
If we just observed my running for 15 minutes, we might conclude I could run at 10 km/h indefinitely. But that’s not true.

Cumulative Negative Effects


IF TETRIS HAS TAUGHT ME ANYTHING, IT IS THAT ERRORS PILE UP AND ACCOMPLISHMENTS DISAPPEAR.

If our sample period is 15 minutes, it's not long enough to reflect the cumulative negative effects that contribute to my slowing down and being forced to stop.

Wait, CPUs Don't Get Tired


LISTEN, AND UNDERSTAND! THAT TERMINATOR IS OUT THERE! IT CAN'T BE BARGAINED WITH. IT CAN'T BE REASONED WITH. IT DOESN'T FEEL PITY, OR REMORSE, OR FEAR. AND IT ABSOLUTELY WILL NOT STOP, EVER, UNTIL YOU ARE DEAD.
WWW.MAGICALQUOTE.COM
KYLE REESE

Their parts accumulate fatigue, but not at the rate of a runner's muscles.

Yes, it's still a valid analogy. Consider a program that has a data hoarding issue.

It’ll Fit

One outcome: `java.lang.OutOfMemoryError:  Java heap space`

Same for filling up disk, exhausting file handles, log length.

Another example: holiday code freeze as unplanned surprise endurance test.

Okay, But How Long?

Is 30 minutes of running the right amount? 3 hours?

Maybe it’s product requirements again: e-commerce on Black Friday?

What about the maintenance window?

No easy answers.

Evaluating Success

There are two kinds of answers load testing can give us:

Can the system handle a load of X?

What is the maximum load Y the current system can handle?

Answering question two is hard; sometimes question one is all we need.

Nuance: is Y a hard stop or just a degradation?

Observability

We may need to add observability to be able to decide if a test passes.

Examples: Is the total work completed in a time limit?
    Do items get completed in the time limit 99%+ of the time?

The raw results might need post-processing.

Endurance Requires Consistency

Success in an endurance test is not just whether it could handle *X* load.

It’s about whether the answer continued to be yes consistently.

Don’t forget to look at the trends.

Failed Load Test?

Like a software program, I can get better at running.
  Need to make changes, practice, improve.

Or as the internet might say: git gud.

Make Changes, Try Again

Given specific scenarios to improve, apply techniques from this course!

Re-run tests, re-evaluate.

Repeat until we pass the scenario!

Git Gud or Git Rekt?

There is a point at which I can make no more improvements in my running.

Eliud Kipchoge ran a marathon in 2:01:09.

I’m not going to catch up to him no matter how much I train!

Limits

Maybe the software we have is not the right one: redesign or replace.

Alternatively: change expectations!
  Example: What if we don’t bill all customers on the 1st of the month?

Think outside the constraints—i.e. those given in the initial problem statement.

Constant Vigilance

Load testing gives a picture at a given moment.
    Need to re-test to ensure we catch slowdowns early.

Software tends towards increased complexity over time (slower).
    So it takes meaningful effort to stay on top of it.

Real-life example: `https://arewefastyet.com` tracks regressions/improvements in Firefox performance.