Performance vs. Scalability

Alex Podelko

After attending Sergey Chernyshev's Size & Speed presentation at NY Web Performance Meetup, and reading Scalability: it's the question that drives us by Robert David Graham and Scalability vs. Performance: it isn't a battle by Theo Schlossnagle, I would like to share my understanding. While I agree in general with everything said, I would rather word it differently. The topic became loaded, so accents are important. Robert, for example, states that *"performance" and "scalability" are orthogonal problems*. Well, no, in my opinion, they are not. They are different, but correlated notions -- even leaving aside that performance and scalability are somewhat vague terms.

Components of Response Time

If we speak about web systems now, it looks like we can roughly separate two main components in response time (which *is* the main performance metric):

- Back-end (server-side) time, and
- Front-end (network and client-side time).

(There are subtleties and grey areas, but I'll ignore them here.) Front-end time, the subject of Web Performance Optimization (WPO), doesn't relate to scalability in so as far as it is not involving server processing (again ignoring subtleties).

The proportion of front-end time vs. back-end time may be any at all. According to Steve Souders, the founder of WPO, "80-90% of the end-user response time is spent on the front-end." But even for major web sites, the back-end time for requests involving database processing (such as submitting orders or querying order status) may be more noticeable. And there are plenty of corporate web applications where front-end time is a rather small share of the total. Of course, the starting point of any performance troubleshooting is to find where time is spent. And there is absolutely no sense in optimizing parts where time is not spent.

However, there is one important "but". While front-end time is supposed to be constant (another simplification, but again ignoring subtleties), the back-end (server) time depends on load. The heavier the load, the larger the server time may be. And at some point it may skyrocket, making the system practically unusable. So, thinking about what you need to optimize, you need to check where time is spent under maximal load. Disregarding downtime and user experience, the way to do it is load testing.

Scalability

And here we get to scalability. The front-end performance indeed doesn't matter here and is independent of scalability. But the back-end performance *is* directly related to scalability. The relationship, of course, may be non-linear and quite sophisticated – but it does exist.

To illustrate, let's consider one simple, but still typical, example. The back-end processing takes X ms, the time is mainly spent in CPU, and we don't have any other bottlenecks. In this case the server response time would be mainly CPU processing time – and every request would take X ms of CPU time (if we don't have parallelism here). As soon as we take most of available CPU time, server response time would skyrocket (that situation may be modeled using queuing theory). So there is a load when the system becomes practically unusable – and the question is just when we get to this load. (Of course, we may get to problems sooner if we have any scalability problem inside the system or run out of another kind of resource.)

Generally speaking, you can increase scalability either by optimizing server processing (using less resources for each transaction) or by providing more resources. Of course, if your architecture allows using these additional resources then mainly scalability boils down to the ability to parallelize your processing (and is often limited by what you can't fully parallelize – like a centralized database).

Load Testing

We do have two parts of response time – front-end and back-end – which behaves differently and may need different approaches and tools to optimize. But the end user experience is the sum of these two parts, where the back-end time is a function of load. You can't say much about your end-to-end performance and its back-end part until you check it under load – and load testing is the safe way to do so.

Historically, performance engineering concentrated on back-end - where main performance and scalability issues were – and practically ignored front-end (which indeed was usually pretty straightforward then). Several sub-disciplines were formed including performance analysis, capacity planning, and load testing. Later, when sophistication of front-end skyrocketed, a whole new discipline was established by Steve Souders and quickly grew around Velocity and Web Performance meetups. Unfortunately, it practically dismissed performance engineering developments of the last 40 years (maybe even more - the Computer Measurement Group (CMG) was founded in 1975). While front-end Web Performance Optimization definitely has its own specific focus, I'd still expect to see a holistic approach to performance engineering, taking in account all aspects of performance and scalability end-to-end.

Steve Souders says that since most of the end-user response time is spent on the front-end, that it is where you need to spend time optimizing it – because back-end is usually well optimized for major web sites. And the final part here (which is not always clearly spelled out by Steve) is very important and ignoring it is very dangerous: If you didn't spent time optimizing back-end, there is absolutely no reason to believe that it would be fast and scalable. Using

well-tested solutions helps up to some level of load, but then the only way to know is to test it. And if you care about your customers, it is better to do load testing.

Of course, if you are talking about response time and you see that 90% is spent on front-end, you definitely shouldn't spend time optimizing back-end. It is the cornerstone of performance optimization in general: Before optimizing, first find where time is spent. It goes beyond front-end vs. back-end: You need to figure out where exactly time is spent. Even if front-end takes 90%, working on something that takes a few ms of this time won't help you either. What concerns me is that if it takes the "80-90%" without proper context, it may become a reason to dismiss a need for any back-end activities.

Summary

You should clearly understand what you test/measure with load testing. It is about server-side performance and scalability. In most cases it is not about front-end (or end-to-end performance) – while there are tools/ways allowing to incorporate them into load testing, it is not the mainstream and needs to be done with care.