5

Speed Matters

An End-to-End Case Study

> The dangers of a slow web site: frustrated users, negative brand perception, increased operating expenses, and loss of revenue
> — *Steve Souders (2009)*

> An engineer that improves server performance by 10 msec (that's 1/30 of the speed that our eyes blink) more than pays for his (or her) fully-loaded annual costs. Every millisecond counts
> — *Kohavi, Deng, Frasca, Walker, Xu and Pohlmann (2013)*

> Fast is my favorite Feature
> — *Google shirt circa 2009*

Why you care: *We begin with an end-to-end example of the design (with explicit assumptions), execution, and interpretation of an experiment to assess the importance of speed. Many examples of experiments focus on the User Interface (UI) because it is easy to show examples, but there are many breakthroughs on the back-end side, and as multiple companies discovered: speed matters a lot! Of course, faster is better, but how important is it to improve performance by a tenth of a second? Should you have a person focused on performance? Maybe a team of five? The return-on-investment (ROI) of such efforts can be quantified by running a simple slowdown experiment. In 2017, every tenth of a second improvement for Bing was worth $18 million in incremental annual revenue, enough to fund a sizable team. Based on these results and multiple replications at several companies through the years, we recommend using latency as a guardrail metric.*

How important is product performance? Where in the product is reducing latency important? Controlled experiments provide clear answers to these questions through a simple yet powerful technique: slowdown experiments. By slowing down the product, we can assess the impact of increased latency on

key metrics, including declines in revenue and user satisfaction metrics. Under mild assumptions that we can verify, we can show that an improvement to performance that is, reducing latency, improves these metrics: revenue and satisfaction increase.

At Amazon, a 100 msec slowdown experiment decreased sales by 1% (Linden 2006, 10). A rare joint talk by speakers from Bing and Google (Schurman and Brutlag 2009) showed the significant impact of performance on key metrics, including distinct queries, revenue, clicks, satisfaction, and time-to-click. A 2012 detailed study at Bing (Kohavi et al.2013) showed that every 100 msec speedup improves revenue by 0.6%. In 2015, as Bing's performance improved, there were questions about whether there is still value to performance improvements when the server was returning results in under a second at the 95th percentile. A follow-on study was conducted and while the impact on revenue was somewhat reduced, Bing's revenue improved so much that each millisecond in improved performance was worth more than in the past: every four milliseconds improvement funded an engineer for a year!

Multiple performance-related results were shared in *Why Performance Matters* (Wagner 2019), showing improvements to conversions and user engagement, although many of the results are not from controlled experiments, so their results are confounded with other changes.

One decision you might face is whether to use a third-party product for personalization or optimization. Some of these products require that you insert a JavaScript snippet at the top of the HTML page. These are blocking snippets that slow the page significantly because they require a roundtrip to the snippet provider and transfer the JavaScript, which is typically tens of kilobytes (Schrijvers 2017, Optimizely 2018b). Putting the snippet lower on the page results in page *flashing*. Based on latency experiment results, any increase in goal metrics might be offset by the cost of the latency increase. Thus, we recommend using server-side personalization and optimization whenever possible, that is, have the server side do the variant assignment (see Chapter 12) and generate the HTML code for that variant.

Our goal is to show how to measure the impact of performance on key metrics, not the specific techniques for improving performance. There are several excellent resources available for improving performance (Sullivan 2008, Souders 2007, 2009).

Another benefit of running this type of experiment is that you can generate a mapping from performance delta to the delta impact on key metrics to answer such questions as:

- What is the immediate revenue impact of performance improvements?
- Is there a long-term impact (e.g., reduced churn) from performance improvements?

- What is the impact to a metric X? It is often the case that the initial implementation of a new feature is inefficient. If the A/B test shows a degradation to metric X, would speeding the implementation be enough to address the degradation? In many cases, a new feature slightly slows down the web site or application, so there is a tradeoff to make, and this mapping helps.
- Where are the performance improvements more critical? For example, increased latency for elements that users must scroll to see (also known as "below the fold") may be less critical. Similarly, right-pane elements have been found to be less critical.

To conduct a controlled experiment, you want to isolate latency as the *only* factor changed. It is very hard to improve performance or else developers would have already made those changes, so we resort to a simple technique: slowing down the web site or product. By slowing the Treatment relative to Control it is easy to measure the impact on any metric, but you need to make some assumptions.

Key Assumption: Local Linear Approximation

The key assumption for slowdown experiments is that the metric (e.g., revenue) graph vs. performance is well approximated by a linear line around the point matching today's performance. This is a first-order Taylor-series approximation, or linear approximation.

Figure 5.1 shows a graph depicting a common relationship between time (performance) and a metric of interest (e.g., click-throughrate (CTR) or revenue-per-user). Typically, the faster the site, the better (higher in this example) the metric value.

When we slow down the Treatment, we move from the point where the vertical line intersects the graph to the right, and we can measure the change to the metric. The assumption we make is that if we were to move left of the vertical line (improving performance), the vertical delta on the left would be approximately the same as the one we measured on the right.

Is the assumption realistic? Two things make this a good assumption:

1. From our own experience as users, faster is better in search. It is hard to think about reasons why there would be discontinuities or dramatic changes in the graph, especially around today's performance point. If we delayed by three seconds, one can imagine a significant cliff, but for adding or subtracting a tenth of a second, this is less likely.

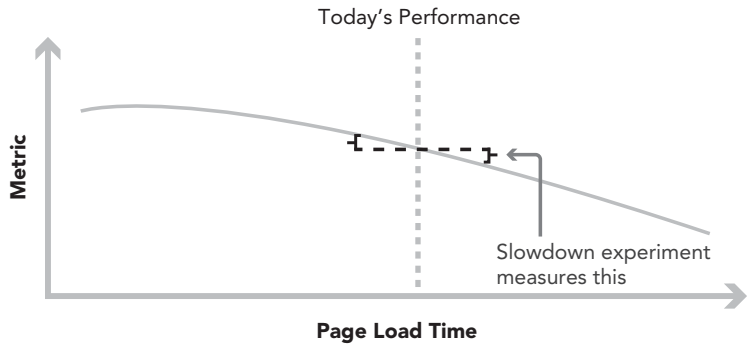

Figure 5.1 Typical relationship between performance (time) and a metric of interest

2. We can sample the graph at two points to see whether a linear approximation is reasonable. Specifically, Bing ran the slowdown experiment with a 100 msec and a 250 msec slowdown. The delta for the 250 msec experiment was ~2.5 times that of the 100 msec study (within the confidence intervals) for several key metrics, which supports the linearity assumption.

How to Measure Website Performance

Measuring website performance is not obvious. This section shares some of the complexity involved and some of the assumptions made. These significant details impact your experiment design. We go into details here to give a taste of real-life complexity for something that appears simple; feel free to skim this section.

To reliably measure latency, servers must be synchronized as requests are typically handled by different servers and we have seen clock skew between servers contribute to data quality issues (e.g., negative durations). It is very important that servers sync their clocks often. Our examples do not mix client and server times, because they can be in different time zones (see Chapter 13), and client clocks are usually less reliable, sometimes years off (e.g., when their batteries die).

Figure 5.2 shows a request for a highly optimized website, such as a search engine. The steps are as follows:

1. The user makes a request at time T0, say typing a query into the browser address bar or search box and hitting return or clicking on the magnifying glass.

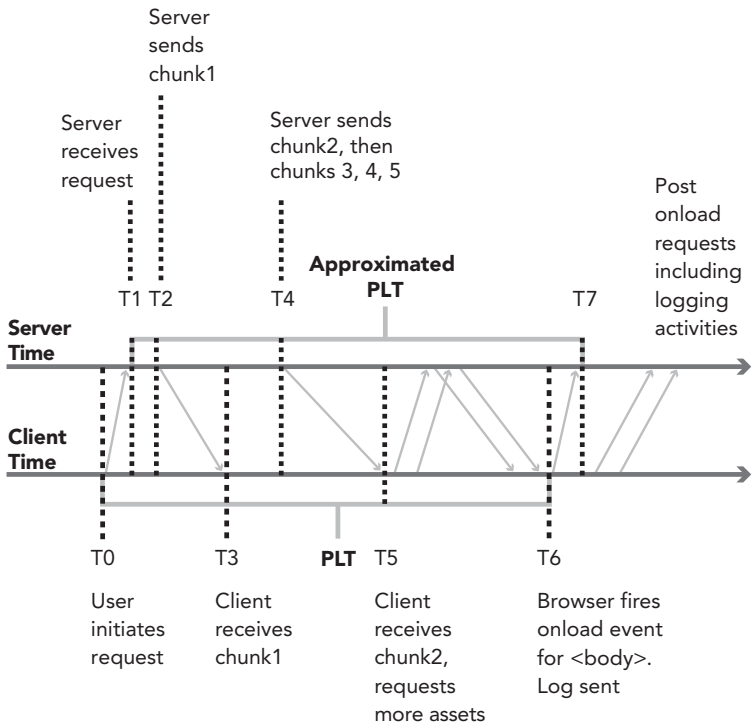

Figure 5.2 Measuring Page Load Time (PLT)

2. The request takes time to reach the server, and it arrives at time T1. T1−T0 seems extremely hard to estimate, but there is a nice trick we can deploy that we explain after this numbered list.
3. On receiving the request, the server typically sends the first chunk of HTML to the client, time T2.

   This first chunk is independent of the request (e.g., query or URL parameters), so can be served quickly. It typically contains the basic page elements, such as the header, navigation elements, and JavaScript functions. Providing the user with visible feedback that the request was received is beneficial: the page typically clears, and a header displays with some page decorations, sometimes called the chrome or frame. Since the server takes time (to time T4) to compute the URL-dependent part of the page (e.g., query, or URL parameters), the more "code" that can be shipped, the faster the page will be, as the client and network are typically idle.
4. At time T4, the server starts sending the rest of the page, which can involve additional round trips for other assets (e.g., images).

5. At time T6, the browser fires the Onload event, indicating the page is ready. At this point, it makes a log request, typically a simple 1×1 image request (beacon) or equivalent. That request reaches the server at time T7. There may be other activities that happen after the Onload event and additional logging (e.g., user actions like scrolls, hovers, and clicks).

The Page Load Time (PLT) the user experiences is T6−T0, which we approximate by measuring T7−T1. Because the time the initial request takes to reach the server is likely to be very similar to the time it takes the Onload event beacon to reach the server (both are small requests), these two deltas will probably be very similar and allow us to approximate the user experience time.

In newer browsers that support new W3C (World Wide Web Consortium) standards, Navigation Timing calls provide multiple PLT-related information (see www.w3.org/TR/navigation-timing/). The above measurements are more generic, and the numbers from the W3C Navigation Timings match well.

The Slowdown Experiment Design

What may seem like a trivial experiment turns out to be more complex. One question is where to insert the slowdown? Bing initially slowed down sending Chunk1 (see Figure 5.2), but had too big an impact and was deemed unreasonable because of the following:

1. Chunk1 is sent from the server quickly because there is nothing to compute. It is therefore unreasonable to say that we can improve the latency of Chunk1.
2. Chunk1 is what gives the user the feedback that the request was properly received by painting the page chrome or frame. Delaying that has multiple negative effects unrelated to overall site performance.

The right place to delay is when the server finishes computing Chunk2, which is the URL-dependent HTML. Instead of the server sending the HTML, we delay the response, *as if* the server took longer to generate the query-dependent part of the HTML.

How long should the delay be? There are several factors at play here:

- Every metric we compute has a confidence interval. We would like the Treatment effect to be large so that we can estimate the "slope" more accurately. Figure 5.3 shows two possible measurements at 100 msec and 250 msec the delay time axis. If both have a similar confidence interval size, then measuring at 250 msec provides us with much tighter bounds on the slope. This factor calls for making the delay larger.

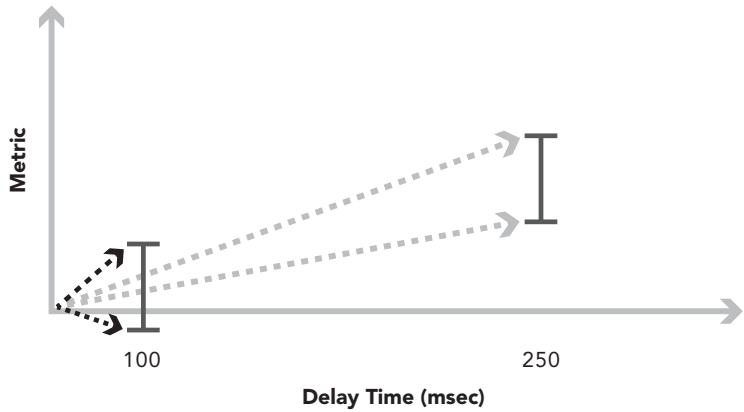

Figure 5.3 Confidence intervals around different delay times

- A longer delay implies that our first-order Taylor series approximation may be less accurate. This factor calls for a shorter delay.
- A long delay causes more harm to our users, as we strongly believe that faster is better and therefore slowing the experiences causes harm. This factor calls for a shorter delay.

Another question is whether the delay is constant or some percentage, to account for geographical network differences (e.g., Bing users in South Africa have very slow page load times, so 250 msec delay may not feel like much). Given that the experiment is modeling a back-end server-side delay, a constant delay was deemed a good choice. If we wanted to model what happens relative to network differences, then the experiment might be based on, for example, payload size instead of latency.

Finally, there is a question of whether speedup is more important on the first page or later pages in the session. Some speedup techniques (e.g., caching of JavaScript) can improve the performance of later pages in a session.

Given the above factors, slowdowns of 100 msec and 250 msec were determined to be reasonable choices by Bing.

Impact of Different Page Elements Differs

Performance of different areas of the page differs. The speed of showing the algorithmic search results of Bing were critical and slowdowns had material impact on key metrics, such as revenue and key user metrics.

What about other areas of the page? It turns out they are much less critical. At Bing, some elements on the right pane are loaded late (technically, after the

`window.onload` event). A slowdown-controlled experiment was run, like that described above, delaying when the right-pane elements display by 250 msec. No statistically significant impact was detected for key metrics, despite having almost 20 million users in the experiment.

PLT is often measured using the `window.onload` to mark the end of the useful browser activity rather than the sequence described above. However, this measurement has severe deficiencies with modern web pages. As Steve Souders showed (2013), an Amazon page can render in 2.0 seconds above the fold (Wikipedia contributors, Above the Fold 2014), that is, on the visible part of the page, but the `window.onload` event fires at 5.2 seconds. Schurman and Brutlag (2009) reported that being able to progressively render a page so that the header displays early helps. The opposite is also true with Gmail, as a good example: the `window.onload` fires at 3.3 seconds, at which point only the progress bar is visible and the above-the-fold content displays at 4.8 seconds.

The term "perceived performance" often denotes the intuitive idea that users start to interpret the page once enough of it is showing. The concept of perceived performance is easier to state abstractly than it is to measure in practice, and `perception.ready()` isn't on any browser's roadmap (Souders 2013). Multiple proposals have been developed to estimate perceived performance, including:

- **Time to first result**. When a list displays, such as on Twitter, the time to the first tweet is a possible metric.
- **Above the Fold Time (AFT)**. You can measure the time until pixels above the fold are painted (Brutlag, Abrams and Meenan 2011). Implementations must use heuristics to handle videos, animated GIFs, rotating galleries, and other dynamic content that changes the page above the fold. Thresholds can be set for "percent of pixels painted" to avoid trivial elements of little consequence from prolonging the measured time.
- **Speed Index**. This is a generalization of AFT (Meenan 2012) that averages the time when visible elements on the page display. This does not suffer from trivial elements showing late, but still suffers from dynamic content changing above the fold.
- **Page Phase Time and User Ready Time**. Page Phase Time requires identifying which rendering phase satisfies perceived performance, and phases are determined by pixel-changing velocity. User Ready Time measures the time until the essential elements of the page (defined for each context) are ready to use (Meenan, Feng and Petrovich 2013).

One way to avoid coming up with definitions for perceived performance is to measure the time-to-user action, such as a click. This technique works well

when there's an expected action by the user. A more sophisticated variant of time-to-click is time-to-successful-click, where success can be defined as a click that does not result in the user coming back for 30 seconds, thus avoiding "bait" clicks. Such metrics do not suffer from the heuristics needed for many performance metrics and are robust to many changes. The main problem with such metrics is that they work only when an action is expected. If a user issues a query "time in Paris" and gets a good instant answer, there is nothing to click.

Extreme Results

While speed matters a lot, we have also seen some results we believe are overstated. In a Web 2.0 talk by Marissa Mayer, then at Google, she described an experiment where Google increased the number of search results on the Search Engine Result Page (SERP) from ten to thirty (Linden 2006). She claimed that traffic and revenue from Google searchers in the experimental group dropped by 20%. Her explanation? The page took half a second more to generate. Performance is a critical factor, but multiple factors were changed, and we suspect that the performance only accounts for a small percentage of the loss. See Kohavi et al. (2014) for details.

Conversely, Dan McKinley (2012), then at Etsy, claimed that a 200 msec delay did not matter at all. It is possible that for Etsy users, performance is not critical, but we believe a more likely hypothesis is that the experiment did not have sufficient statistical power to detect the differences. Telling an organization that performance doesn't matter will make the site slower very quickly, to the point where users abandon it in droves.

Finally, in rare scenarios, too fast may reduce user trust that some activity was done, so some products add fake progress bars (Bodlewski 2017).

When reviewing results of experiments, ask yourself what trust level to apply, and remember that even if the idea worked for a specific site, it may not work as well for another. One thing you can do is report replications of prior experiments (successful or not). This is how science works best.