DP Evaluation — Overview
6.033 Spring 2020

This document presents guidelines for how to evaluate your system even though you didn't build it. Each member of the team should read it before meeting about their evaluation. After that, we recommend that your team meet and work through the accompanying process document. As you do that, if there are places where you feel stuck or have a question, you should reach out to your TA.

Evaluation Overview

Even though you haven't built your system, you can still evaluate it for correctness, performance, scale, flexibility, etc. Many of these aspects can be calculated directly; others can be estimated or evaluated in other ways. As you begin to evaluate, keep in mind the following:

- **Different designs require different evaluations.** A metric or other element of an evaluation that is important for one design may not matter for another. For instance, if your design had no communication between components, it would not make sense to evaluate network overhead.

- **Evaluations require context.** Your evaluation should explain how the values that you calculated affect users or other entities in the system. If your system takes X seconds to respond to a failure, is that good or bad? If it takes Y seconds to perform a particular operation, is that acceptable? Under what conditions? Did you make any trade-offs that involve these metrics?

- **Your evaluation should justify your design choices** (although the evaluation section of the final report is not the only place where such justification should occur). For example, "Our method for performing operation X takes thirty seconds, compared to a design without this method, which would take five minutes."

Doing the evaluation

Direct calculations

Many evaluation metrics can be calculated directly, because you control the system. Given the limitations imposed by the DP specification — for instance, limitations on network speeds, storage, etc. — as well as the elements of your design — how often you send data over the network, what things you choose to store and for how long — you can often directly calculate elements such as network overhead, storage, etc.

Sometimes those values change depending on the use case. You can be explicit about that (e.g., “Under condition A, which happens 90% of the time, network overhead is X Kb/sec; under condition B, network overhead is Y Kb/sec”).

Estimations and Unpredictability

Sometimes there are quantities that you can’t calculate directly, but that you *can* estimate based on a reasonable model. You might do this by looking up reasonable values for a particular quantity (e.g., if the spec doesn’t provide exact numbers for CPU speeds, you could look up the speeds for similar devices) or by assuming some reasonable probability of a particular scenario (e.g., assuming some reasonable failure rate for part of your system, or giving a range of values). In the absence of any sort of reasonable model, you can often describe how your system would perform in a worst-case scenario.

For systems in which there is some degree of unpredictability, we often estimate the probability of meeting a target (for instance, the probability that a message arrives successfully at some component). As part of your evaluation, you can provide context/justification for that probability.

Evaluating Scale

Evaluating scale is something of a special case. Scale isn’t a binary quantity; it’s not the case that a system scales or it doesn’t. Typically, a system scales up to a point, and you need to figure out what that point is. We can also talk about scale in multiple facets: how many machines can a system support, how many users, how much data, etc.

Figuring out how far your system can scale requires you to recognize the bottlenecks in your system and figure out which of those bottlenecks will be reached first.

Examples of things that may affect scale include:

- Network overhead (does congestion become an issue?)
- Storage limits (at what point does storage fill up? At what point does disk access become a bottleneck?)
- Speed of processing large amounts of data (how fast can your system process data as the amount of data grows?)

No Calculable Metrics

Some design decisions may not correspond to calculable metrics; the most common case is preferring a simple, modular design over a complex one. Your design report should note when you made choices in the name of simplicity or other design principles. Depending on your report organization, it may be more appropriate to include this information in the design section than in the evaluation.

Other common cases are describing the user experience when appropriate, and how this system might evolve in the future.