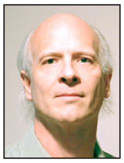
Capacity Engineering
An Interview with Rick Boone

RIK FARROW

Rick Boone is currently a Senior Engineer on the Capacity Engineering team at Uber, where he focuses on modeling and forecasting Uber's capacity needs. He has been at Uber for 3.5 years, where he primarily worked in SRE (Site Reliability Engineering). Previously, Rick worked at Facebook for three years as a Production Engineer and, before that, at a number of tech startups in Los Angeles. In his free time, he loves traveling, swimming, and gymnastics. boone@uber.com



Rik is the editor of *;login:*. rik@usenix.org

When I heard Rick Boone's talk at SREcon18 Americas, I was immediately struck by his approach. While capacity planning is really an art, relying partially on past behavior but just as much on intuition, Rick described uncovering the best metric for reliably predicting capacity as needed.

Uber's services run on their own hardware, and their goal is to always have sufficient capacity without ever having either too much or a shortage that will hurt business. Rick's approach [1] used machine learning to help pick out the appropriate metric and mathematically predict its impact on a service's capacity needs. You can watch the video of his talk to learn the approach used. In this interview, Rick discusses why Uber doesn't use capacity planning and instead relies on capacity engineering.

Rik Farrow: What's wrong with capacity planning?

Rick Boone: For those who are concerned with availability and reliability of services or platforms, service owners, Production Engineers, or SREs, capacity planning is typically one of the fuzziest and least understood parts of their job. When we speak of capacity planning, we're aiming for a "just right" amount of resources allocated for a service, which will allow that service to run both efficiently (i.e., "not using too many resources") and reliably (i.e., "not using too few resources"), even in the face of unexpected surges of traffic. This can be pretty difficult to achieve for a multitude of reasons, especially in a fast-moving, complex environment with lots of interactions between hundreds or thousands of services.

Typically, capacity planning involves a fair amount of back-of-the-napkin math and fuzzy methods that differ from service to service. Knowledge of what drives the service's needs and usage (i.e., "demand") is required, as is knowledge of how that demand will grow and change. Knowledge of the service's dependencies and operational particulars is also needed (e.g., "Does it speak to a database?"), along with an understanding of how those details affect the service's ability to serve its demand (and how it consumes resources). Once all of that is known and understood, the planner then needs to determine the best way to calculate expected demand in the future and then extrapolate expected needs from that.

All of these things tend to be very local and service-specific pieces of information, what I like to call "Jedi" knowledge—intuition which service owners tend to gain over time—which differ wildly across an engineering ecosystem. For instance, a search service will grow at a very different trajectory, and have very different resource needs, than a payment service.

You can start to see why traditional capacity planning can be so difficult across an engineering ecosystem. Its methods are typically not repeatable from one service to another, nor are they scalable beyond one or two teams. If one person or team does manage to use a method that is successful for their service, their skills and insight will not necessarily transfer. Methodology and process becomes wildly inconsistent, leading to confusion and, typically, overallocation and wastage of resources. As teams become less confident that they can capacity plan effectively, they begin to simply "throw hardware" at the problem and move on to more solvable things. Often, empirical data or mathematical reasoning is left out of the process, leading to further lack of repeatability and understanding. And even when data and analysis/mathematics are used, there is still a very worrying lack of confidence or certainty delivered with the results. If the "plan" is to go wrong, it is unknown by how much it will go wrong. Either it will work or it won't. This leaves stakeholders and dependent services with an inability to make informed decisions or tradeoffs concerning the service.

Beyond the issues of fuzzy methodology, there are also problems that arise from the nature of software and infrastructure. At Uber, like most large-scale engineering shops, we release a lot of code and changes to a lot of services on a lot of servers throughout the day. This all adds up to an ever-changing, complex, and highly coupled environment, the entirety of which is difficult for humans to consider when predicting future usage, especially months in the future.

RF: How is capacity prediction different?

RB: With capacity *prediction*, we aim to remove all of the fuzziness and *hand-waving* from our understanding of capacity usage and needs. We do this by applying statistical and mathematical methods to large amounts of past usage data via machine learning, allowing us to create mathematically sound models of every service, which we can then use to reliably predict each service's future capacity needs.

Each model takes in, as input, a value of Uber's primary business metrics, things like *Trips Currently Online*, and returns, as output, the amount of hardware resources needed to handle that particular volume of the metric. For example, for service "FooBar," the model might indicate that to handle 100K trips online, the service will need 1000 CPU cores.

By providing a method based on a *blackbox* model of any and every service, which takes in an input that is common across all of Uber, we now have a repeatable, scalable, interpretable, and simple way of both assessing capacity usage and predicting its future values. We don't need to know about a service's dependencies, its particulars, its architecture, how its software performs, etc.—all of that is represented mathematically by the model, and we can deal solely with representative numbers, instead of human/jedi knowledge.

As is typical with statistically derived models, we are also able to construct empirically derived measures of the model's accuracy, so that we can have a very precise idea of how much confidence we can place in the model's prediction. Whereas *plans* are often and easily broken, *predictions* are made with an expectation of success (along with an empirical measure of possible failure).

RF: How did you go about creating a method for capacity prediction at Uber?

RB: The primary things needed for us to bring capacity prediction to fruition were: (1) a consideration of the fundamental thing(s) that drive resource consumption and (2) a way to represent these things via data and mathematical models.

At Uber, the demand for most services is driven by a few key high-level metrics, such as the number of drivers online or the number of trips occurring. Because of this, the levels of these metrics typically have a close correlation with resource usage. We started by building multivariate data sets comprising these metrics and the CPU usage of a single service, at a granularity of one hour. We typically use about two weeks of historical data to ensure that we're only analyzing the most recent representation of the service, including its current software releases, dependencies, clients, payloads, etc. Because we have multiple high-level metrics that can drive a service's usage, we perform correlation analysis to mathematically determine which metric has the strongest correlation with the service's resource usage. Having determined the best metric, we then use machine learning methods to build a quantile regression model, which is a variant of a linear regression model, with the high-level metric as the feature/input and the resource usage as the outcome/output. With a quantile regression, we can retrieve 99% of all possible outputs, allowing us to greatly minimize the possibility of underpredicting.

We repeat this process for every service at Uber and store the resulting model in a database, with each model relying on one of a few high-level metrics as its input. Because our high-level metrics are key performance indicators for the entire company, we have accurate forecasting for them, extending months into the future. We simply pass these forecasts into our models and are able to get a prediction for resource usage for each service for the next few months.

Any service owner, SRE engineer, etc., can now query for their service's predicted resource needs for any week within the next 2–3 months and adjust their allocations accordingly.

RF: Is capacity prediction something unique for Uber or is it easy for others to also do this?

RB: This is very doable anywhere! The toughest thing that others might run into is acquiring both historical and forecasted high-level business metric data. Once that is acquired, along with service-level resource usage data (CPU, memory, etc.), you'll need machine-learning methods to apply to the data and train a model. ML libraries are readily available in a number of libraries, primarily in Python or R. Or you could also write your own model trainer. Finally, you'll need a place to store the models (we use Cassandra) and a way to retrieve and apply them. We built a light API in front of the Cassandra model store.

Reference

[1] Rick Boone, "'Capacity Prediction' instead of 'Capacity Planning': How Uber Uses ML to Accurately Forecast Resource Utilization": https://www.usenix.org/conference/srecon18americas/presentation/boone.