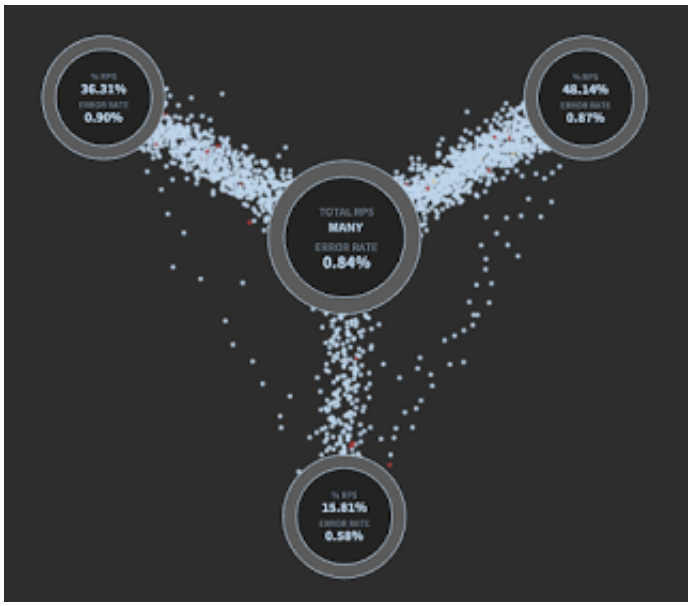
A New Approach to System Intuition

Flux: A New Approach to System Intuition



First level of Flux

On the Traffic and Chaos Teams at Netflix, our mission requires that we have a holistic understanding of our complex microservice architecture. At any given time, we may be called upon to move the request traffic of many millions of customers from one side of the planet to the other. More frequently, we want to understand in real time what effect a variable is having on a subset of request traffic during a Chaos Experiment. We require a tool that can give us this holistic understanding of traffic as it flows through our complex, distributed system.

The two use cases have some common requirements. We need:

- Realtime data.
- Data on the volume, latency, and health of requests.
- Insight into traffic at the network edge.
- The ability to drill into IPC traffic.
- Dependency information about the microservices as requests travel through the system.

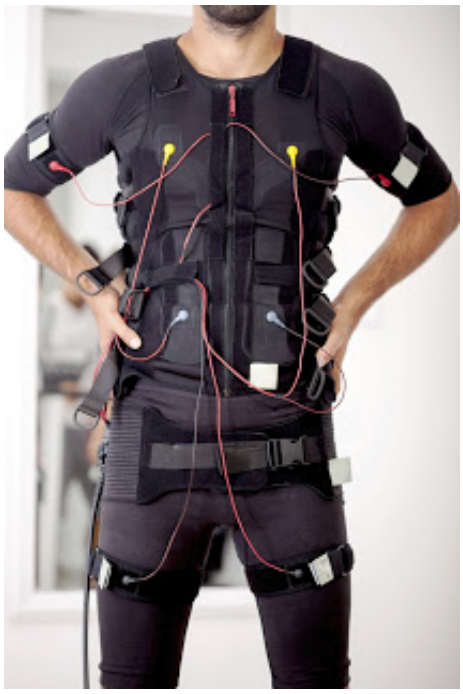
So far, these requirements are rather standard fare for a network monitoring dashboard. Aside from the actual amount of traffic that Netflix handles, you might find a tool at that accomplishes the above at any undifferentiated online service.
Here's where it gets interesting.
In general, we assume that if anything is best represented numerically, then we don't need to visualize it. If the best representation is a numerical one, then a visualization could only obscure a quantifiable piece of

information that can be measured, compared, and acted upon. Anything that we can wrap in alerts or some threshold boundary should kick off some automated process. No point in ruining a perfectly good system by introducing a human into the mix.

Instead of numerical information, we want a tool that surfaces relevant information to a human, for situations that would be too onerous to create a heuristic. These situations require an intuition that we can't codify.

If we want to be able to intuit decisions about the holistic state of the system, then we are going to need a tool that gives us an intuitive understanding of the system. The network monitoring dashboards that we are familiar with won't suffice. The current industry tools present data and charts, but we want a something that will let us feel the traffic and the state of the system.

In trying to explain this requirement for a visceral, gut-level understanding of the system, we came up with a metaphor that helps illustrate the point. It's absurd, but explanatory.



Let's call it the "Pain Suit."

Imagine a suit that is wired with tens of thousands of electrodes. Electrode bundles correspond to microservices within Netflix. When a Site Reliability Engineer is on call, they have to wear this suit. As a microservice experiences failures, the corresponding electrodes cause a painful sensation. We call this the "Pain Suit."

Now imagine that you are wearing the Pain Suit for a few short days. You wake up one morning and feel a pain in your shoulder. "Of course," you think. "Microservice X is misbehaving again." It would not take you long to get a visceral sense of the holistic state of the system. Very quickly, you would have an intuitive understanding of the entire service, without having any numerical facts about any events or explicit alerts.

It is our contention that this kind of understanding, this mechanical proprioception, is not only the most efficient way for us to instantly have a holistic understanding, it is also the best way to surface relevant information in a vast amount of data to a human decision maker. Furthermore, we contend that even brief exposure to this type of interaction with the system leads to insights that are not easily attained in any other

way.

Of course, we haven't built a pain suit. [Not yet. ;-)]

Instead, we decided to take advantage of the brain's ability to process massive amounts of visual information in multiple dimensions, in parallel, visually. We call this tool Flux.

In the home screen of Flux, we get a representation of all traffic coming into Netflix from the Internet, and being directed to one of our three AWS Regions. Below is a video capture of this first screen in Flux during a simulation of a Regional failover:

Flux Failover Simulation

The circle in the center represents the Internet. The moving dots represent requests coming in to our service from the Internet. The three Regions are represented by the three peripheral circles. Requests are normally represented in the bluish-white color, but errors and fallbacks are indicated by other colors such as red.

In this simulation, you can see request errors building up in the region in the upper left [victim region] for the first twenty seconds or so. The cause of the errors could be anything, but the relevant effect is that we can quickly see that bad things are happening in the victim region.

Around twenty seconds into the video, we decide to initiate a traffic failover. For the following 20 seconds, the requests going to the victim region are redirected to the upper right region [savior region] via an internal proxy layer. We take this step so that we can programmatically control how much traffic is redirected to the savior region while we scale it up. In this situation we don't have enough extra capacity running hot to instantly fail over, so scaling up takes some time.

The inter-region traffic from victim to savior increases while the savior region scales up. At that point, we switch DNS to point to the savior region. For about 10 seconds you see traffic to the victim region die down as DNS propagates. At this point, about 56 seconds in, nearly all of the victim region's traffic is now pointing to the savior region. We hold the traffic there for about 10 seconds while we 'fix' the victim region,

and then we revert the process.

The victim region has been fixed, and we end the demo with traffic more-or-less evenly distributed. You may have noticed that in this demonstration we only performed a 1:1 mapping of victim to savior region traffic. We will speak to more sophisticated failover strategies in future posts.

RESULTS

Even before Flux v1.0 was up and running, when it was still in Alpha on a laptop, it found an issue in our production system. As we were testing real data, Justin noticed a stream that was discolored in one region. “Hey, what’s that?” led to a short investigation which revealed that our proxy layer had not scaled to a proper size on the most recent push in that region and was rejecting SSO requests. Flux in action!

Even a split-second glance at the Flux interface is enough to show us the health of the system. Without reading any numbers or searching for any particular signal, we instantly know by the color and motion of the elements on the screen whether the service is running properly. Of course if something is really wrong with the service, it will be highly visible. More interesting to us, we start to get a feeling when things are right in the system even before the disturbance is quantifiable.

STAY TUNED

This blog post is part of a series. In the next post on Flux, we will look at two layers that are deeper than the regional view, and talk specifically about the implementation. If you have thoughts on experiential tools like this or how to advance the state of the art in this field, we’d love to hear your feedback. Feel free to reach out to traffic@netflix.com.

-Traffic Team at Netflix

Luke Kosewski, Jeremy Tatelman, Justin Reynolds, Casey Rosenthal