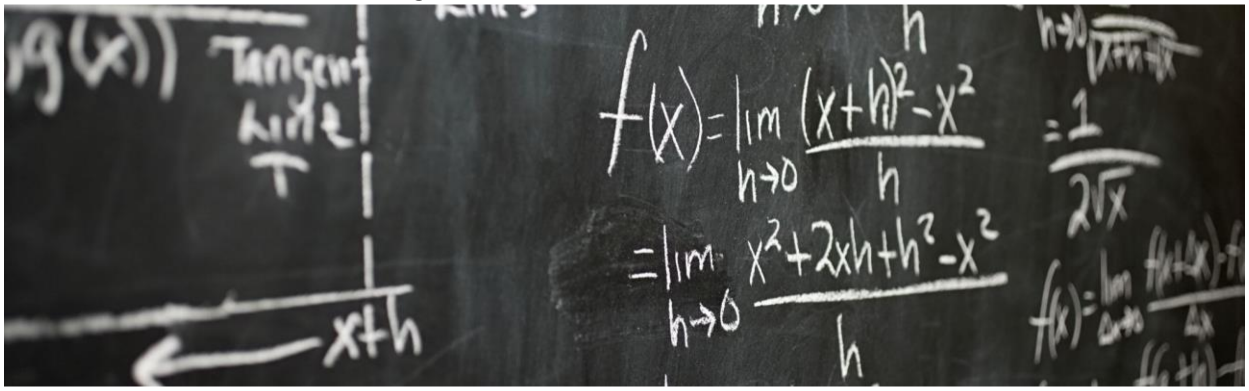
SRE: Service Availability Mathematics



There should be no such thing as boring mathematics. — Edsger W. Dijkstra

In complex systems it can be tricky to reason things so then we find ourselves guessing, not rooting ourselves in logic. Like in the movie Moneyball when they stopped going by the stats and started going by how they felt.

Peter Brand - “*It's about getting things down to one number. Using stats to reread them, we'll find the value of players that nobody else can see. People are over looked for a variety of biased reasons and perceived flaws. Age, appearance, personality. Bill James and mathematics cuts straight through that. Billy, of the twenty thousand knowable players for us to consider, I believe that there is a championship team of twenty five people that we can afford. Because everyone else in baseball under values them. Like an island of misfit toys.*”

Availability mathematics is about getting things down to one number and it is both an art and a science. It is science; the math is clear. The art is finding how to manipulate the systems in order to give you the desired availability and thus value.

If you have a service with an SLA of 99.9% availability (.999a and .001u), and have 2 independent identical service running in parallel the SLA becomes 99.9999% (take the error .001, square it, subtract it from 1, multiply by 100). A single instance of the service has an error budget of 8.76 hours but 2 has about 30-ish minutes . But what if now both servers depend on each other for the application to function (.999, squared, multiply by 100) it drops down to 99.8%, which results in 17.5 hours of downtime. A simple mistake in architecture could have drastic consequences on the overall system.

For example, you can run your application on one of the below platforms. A Kubernetes system that has 3 different tiers, a service mesh with 2 nodes, a control plane of 3 nodes and an application tier of 2 nodes (let's pretend you need all 3 for it to work). The SLA is 99.5% for each node. You are also offered to run your app on 3 vanilla nodes, with an SLA of 99% each. Which is better? It’s B (believe it or not). The composite nature brings down the availability of platform A. Now how do you fix it: run a parallel system or increase the individual component SLAs.

For more information : https://cloud.google.com/blog/products/devops-sre/composite-cloud-availability

Option A)

	Node Count (99.5)	Component SLA	Composite Platform SLA
Service Mesh	2	99.9975	99.9949875628125
Control Plane	3	99.9999875	
Application	2	99.9975	

Option B)

	Node Count	Composite Platform SLA
App Pair 1	3	99.9999