TLA+ for Nike Stores

Brick & Mortar, the Cloud, Formal Methods, and Getting Things Done

Nathan Fairhurst, Principal Engineer

Please note that this presentation is contingent on approval of the final content by Nike Communications and Legal. All text here is solely the opinion of its author.

Abstract

Nike is well known for innovating in everything it does, including its store which operate at a global scale. Brick & Mortar retail of the future will be more connected than has ever been. However, this connectedness doesn't come for free, and means Nike must solve numerous challenges when building these larger and more distributed systems. At the same time, regulations and compliance around the world often require strong guarantees that other traditionally internet-only applications do not. Formal and agile methods can not only co-exist, they can serve one another. Tools like TLA+ help teams go faster, as they provide a strong foundations to build from, and prevent engineers from getting bogged in supporting systems wrought with design flaws.

TLA+ Use Cases

Please note that the following list is intentionally vague, pending approval of more detailed sharing.

1. Cloud to in-store device communication is idempotent
2. Actions promised to consumers are guaranteed to be live
3. Actions required by regulators are guaranteed to be live
4. Distributed systems synchronize at key moments
5. Applications can failover across multiple cloud regions without "lost writes"
6. Optimistic locking for (typically single-user) multi-user experiences
7. Monotonic never-skipping counters
8. Interval Tree implementation on top of DynamoDB

Engineering Process

Typically TLA+ specs are a pairing exercise. Only a couple engineers are well versed in TLA+, so they work as a team to translate prose and diagrams into a formal spec, and translate that spec back into prose. Design errors encountered are posed to everyone to solve as a team. Often a simple sequence diagram is enough to communicate the design or to communicate a design error. Once TLA+ can demonstrate an issue, we see a heightened interest in solving it.

High Performance Computing (HPC) is occasionally leveraged for very large models. However, we now try to avoid writing models large enough to need it!

Learnings

Be as abstract as possible, but no more so!

It is often a challenge to strike the balance between abstract (and checkable) vs concrete (and relatable). Under-specifying can also lead to bugs in implementation, as engineers are forced to tackle hard problems without being given the mental allowance to do so.

Safety is More Important, but Liveness is still Critical

For a DevOps person, liveness translates to "self-healing." We often cannot afford to build systems with livelock, even if they don't technically do anything wrong. We can rarely depend on a client to be live. These properties are much more effort, but they almost always pay off.

A bug is a bug, be it safety or liveness–product and engineering teams don't usually see a difference. TLA+ is one of the few ways to spot liveness issues.

Personally, my best contributions are all formally verified

Our most powerful and effective systems were formally verified. That strong foundation created emergent and unpredicted properties that enabled more features than initially expected.

TLA+ Gives you Super Powers

Not only can TLA+ let you see the outcome of billions of possible state transitions for your current application, but it trains your brain to see common errors. For myself, this is especially true of liveness. We saw all of the "marketing" quotes for TLA+ manifest.