highscalability.com

The QuickBooks Platform - High Scalability -

11-13 minutes

This is a guest post by Siddharth Ram – Chief Architect, Small Business. Siddharth_ram@intuit.com.

The QuickBooks ecosystem is the largest small business SaaS product. The QuickBooks Platform supports bookkeeping, payroll and payment solutions for small businesses, their customers and accountants worldwide. Since QuickBooks is also a compliance & tax filing platform, consistency in reporting is extremely important.. Financial reporting requires flexibility in queries – a given report may have dozens of different

dimensions that can be tweaked. Collaboration requires multiple edits by employees, Accountants and Business owners at the same time, leading to potential conflicts. All this leads to solving interesting scaling problems at Intuit.

Solving for scalability requires thinking on multiple time horizons and axes. Scaling is not just about scaling software – it is also about people scalability, process scalability and culture scalability. All these axes are actively worked on at Intuit. Our goal with employees is to create an atmosphere that allows them to do the best work of their lives.

Background

The QuickBooks Online product is a decade and a half old. It was built as a monolith, without clear separation of concerns. The monolith has served Intuit well – millions of customers use it with hundreds of millions of transactions in a given day. This was partly possible because swimlaning was built into the product. This allowed scaling by formulaic splits (customers are ‘homed’ in a particular shard). Today, we are well on our way to splitting the monolith into multiple services and moving to AWS as the primary hosting solution.

The Numbers

- Largest SaaS product for Small Businesses worldwide
- Over 5M customers across desktop and web
- Serves 250M + requests per day
- 40%+ annual growth
- Expanding globally with AWS
- ~2,000 engineers spread globally work on the products.
- 2000+ 3rd party applications built on the Intuit Platform

The Technology

- Java/JVM is the primary services/backend
- Polyglot persistence– different use cases require use of different storage technology,the QuickBooks Platform uses RDBMS (Oracle/MySQL), Neo4J and Cassandra for OLTP and Hadoop & Vertica for analytics
- The Frontend has been rebuilt using ReactJS,we have learnt that scaling applies not just to back end - front end scaling is a new skill we have developed.
- Monitoring is done using multiple in-house and off the shelf tools, the primary off the shelf tools are Splunk and New Relic

- Code Management via enterprise github
- ActiveMQ and Kafka process async messages
- Ansible and Chef used for configuration management
- Heavy use of edge caching and Akamai WAA

The Culture

- Every service has HA/DR with RTO's and RPO's defined. We execute HA/DR plans on a weekly basis to ensure that there is little to no customer impact in the event of a regular incident. This is a best practice that all SaaS products need to plan and execute.
- Heavy focus on resiliency. A service being unavailable typically does not mean that customers know about it.
- Services are published in an internal services portal. This allows engineers to reuse services that have been constructed by other team, and reduces service clones.
- Performance is measured in terms of TP99, TP90 and TP50 (and often higher). TP50 represents the experience that the 50th percentile of customers experience. Our goals is TP50 of 1 second and TP90 of 2 seconds. (i.e. less than 10% of customers experience page load times on any given page greater than 2 seconds).

- Failed Customer Interactions (FCI) is another key metric we track. Each failed interaction with a customer (HTTP 4xx/5xx) is considered a failed interaction. Our goal is to have FCI's for every service under 0.025%.
- Service teams own services end to end. They are responsible for maintenance and uptime of the service, consistent with the devops model. PagerDuty is used to alert on-call personnel of problems and participation in recovery.
- You cannot improve what you cannot measure. There is wide visibility into the company on our availability, performance, scalability and quality metrics via a near real-time dashboard. This is key in driving behavior in the organization.
- We are in the process of moving to a reactive, loosely coupled system that increases our ability to scale. This however requires careful thinking about how to deal with eventual consistency in the system.
- Engineers perform RCA's (Root Cause Analysis) of every failure in the system. RCA's are reviewed by Engineering leadership. We apply the 5 why's and fish-boning to every RCA. Failures are a wonderful teacher.
- Strong sense of code and domain ownership.

- Mission driven. People do their best lives when they can see the impact they have on our customers.
- No compromises on quality, performance, availability and security.
- Continuous deployments and feature toggles enable us to deliver rapidly

Interesting challenges in scalability

Scaling characteristics

Products like QuickBooks Online have scaling characteristics that are different from other SaaS products. A query for a report may have a dozen of more filters applied on it. These queries are generally not the same across companies. Reporting has tax consequences and hence must always be consistent with the books. Collaboration may take place with multiple edits being made by an Accountant and a Small Business owner at the same time. The software need to determine how to handle conflicts. All this leads to interesting ways of scaling on the QuickBooks Platform at Intuit.

Variability

The QuickBooks platform serves millions of small businesses, their customers and Accountants globally. Customers are served for a diverse set of use cases: The variability the product needs to address include:

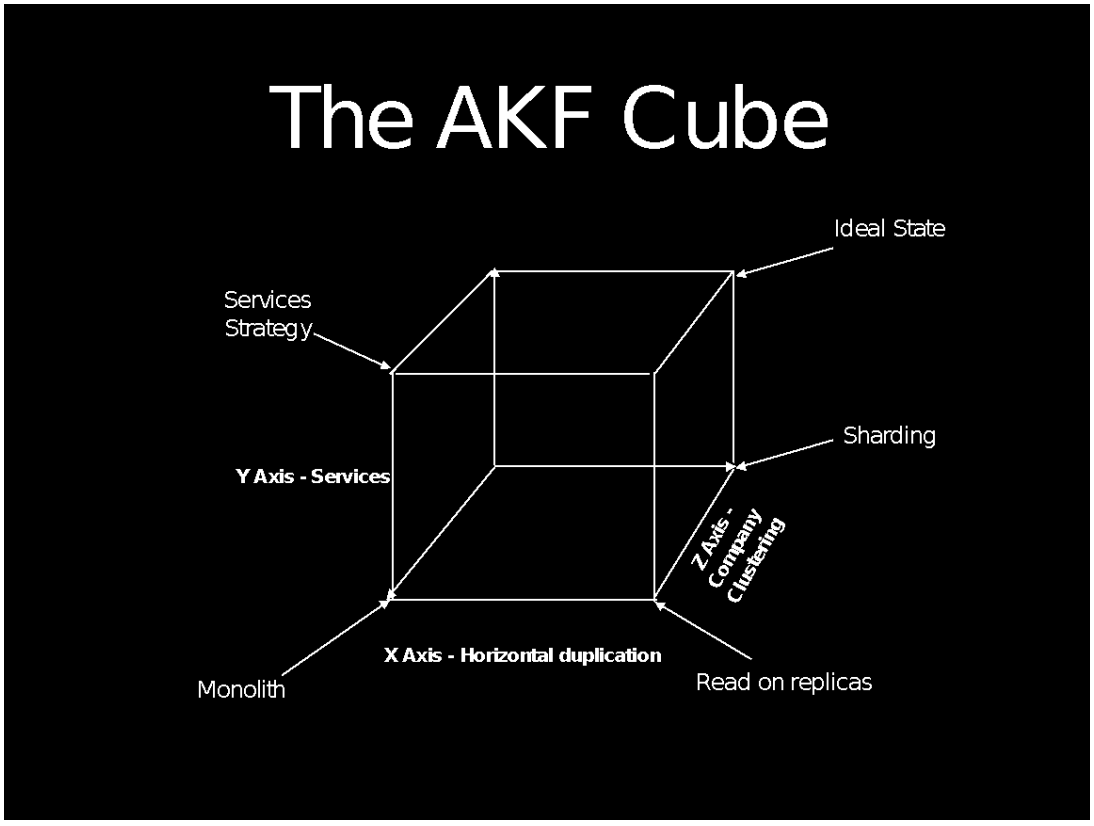
- Device Diversity – We service customers at their point of need - desktop, mobile devices, online
- Geographic diversity – used across the globe with the attendant complexity of regulations, taxation to be addressed that tend to be very localized and specialized
- Compliance diversity – Different government agencies (for example, payroll taxes) have different regulations for compliance
- Product Diversity – Different customer segments have different needs, based on the number of employees, the nature of the market they are addressing (e.g. Product based vs Services based businesses)
- Workflow diversity – companies may have workers who execute (and are given access to) a subset of the product. For example, an A/R clerk will see only accounts receivable flows. Employees generally do not execute business reports.

This is a subset of the diversity we handle. While there

is a lot of diversity that is addressed, there are some fundamental concerns that the platform always solves for. Security, Privacy, Scalability and Quality are addressed as part of the fundamental building blocks of the software.

Scaling Philosophy

The QuickBooks platform generally follows the 'scalability cube' model, by scaling on three dimensions:



X Axis – Read Replicas

The X axis is used primarily for read replicas. As is the

case for many SaaS products, there are a large number of reads compared to the number of writes. A common – and relatively expensive –read operation is reporting. Small Business and accountants need to understand how their business is doing. We deliver insights via a QuickBooks Money Bar, and more detailed insights through a large number of reports (e.g. Balance Sheet, Profit and Loss Reports). Read replicas allow us to both reduce the hotspots in access and deliver precomputed reports

Y Axis – Services

A second dimension to scaling is to break products in distinct services. The QuickBooks platform is modeled as 14 distinct domains, which are composed into products. The services API, is on its fourth revision now – the V3 QuickBooks API can be found at http://developer.intuit.com.

Z Axis – formulaic splits

The Z Axis refers to ‘sharding’ - de-normalized data that allows high scale. QuickBooks splits customers based on the attributes of the company. Each shard is then shared with other companies with similar

attributes.

The C axis

In addition to these aspects, we often talking about a ‘C’ axis – the culture axis, which is a key way for us to scale. Key reinforcements for us on the C axis are (a) a culture of metrics – you cannot fix what you cannot observe (b) a culture of ownership via a devops model and (c) customer backed thinking in whatever we do.

Scaling the front end

At over 200KLOC, QuickBooks onlike has a very large Javascript footprint. Part of scaling the front end has meant understanding how to ensure that changes are isolated and components are reusable – to the point that 3rd parties can plug in directly to the front end, and anyone following rules can contribute components. Scaling also means having uniformity in styles that are easy to adhere to.

Journey to Public Cloud

A key strategy at Intuit is to move all software assets to AWS. In the public cloud, given shared infrastructure, additional considerations need to be made for security

and privacy. We use the following general principles in moving software to AWS:

- Every endpoint must be secured. In the Public cloud we cannot assume that links between services are secure endpoints.
- Every Intuit and AWS platform service must be observable. This observability is key to being able to analyze access for fraud and security.
- Lift and shift + decomposition. Where appropriate, we decompose into services and move to public cloud. For larger systems, we move into the AWS infrastructure and decompose in place.
- Multi Region DR. Region outages happen. We want to be able to provide service even when a single region is impacted.
- Serve customers locally. Both Safe harbor and performance dictates storage of data close to customers. This requires customers to be ‘homed’ in a local region (e.g. EU customers served out of AWS data centers in Europe).
- Blast radius containment. A ‘shared nothing’ approach ensures that we restrict the blast radius. Outage of a system should cause a local event, not a global one.

Key Lessons

- Scaling on 3 axes is key. Knowing which axes are applicable to scale properly helps the scaling.
- Ensure that there is containment of impact. The blast radius of a system must be well understood and tested out.
- RCA – failures are a wonderful teacher. It is a shame to waste lessons from failures. Architects are responsible for analyzing failures and getting to root cause.
- Don’t underestimate the amount of rethinking for securing software in the public cloud.
- Dogfood. The service that engineers in the company use should be the same as those offered to 3rd party developers.
- Know your KPI’s and be able to predict what the performance curve should look like – this helps quickly identify that something is wrong.
- Scale the front end also, not just the back end.