Managing Data in Microservices

Randy Shoup
@randyshoup
linkedin.com/in/randysshoup
Background

• VP Engineering at Stitch Fix
  o Combining “Art and Science” to revolutionize apparel retail

• Consulting “CTO as a service”
  o Helping companies scale engineering organizations and technology

• Director of Engineering for Google App Engine
  o World’s largest Platform-as-a-Service

• Chief Engineer / Distinguished Architect at eBay
  o Multiple generations of eBay’s infrastructure

@randyshoup  
linkedin.com/in/randyshoup
Stitch Fix

Create Your Style Profile.

Get Five Hand-picked Items.

Keep What You Like. Send Back the Rest.

@randyshoup

linkedin.com/in/randyshoup
Stitch Fix

Create Your Style Profile.

Get Five Hand-picked Items.

Keep What You Like. Send Back the Rest.

How do you prefer clothes to fit the top half of your body?

- Mostly Tight / Form Fitting
- Prefer Fitted / Showing my Figure
- Straight
- Mostly Loose
- Oversized

How do you prefer clothes to fit the bottom half of your body?

- Pants (Choose your waist)
- Skirts

@randyshoup

linkedin.com/in/randysihoup
Stitch Fix

Create Your Style Profile.

Get Five Hand-picked Items.

Keep What You Like. Send Back the Rest.

@randyshoup

linkedin.com/in/randysouph
Stitch Fix

Create Your Style Profile.

Get Five Hand-picked Items.

Keep What You Like. Send Back the Rest.

@randyshoup

linkedin.com/in/randystoup
Combining Art and [Data] Science

• 1:1 Ratio of Data Science to Engineering
  o Almost 100 software engineers
  o Almost 100 data scientists and algorithm developers
  o Unique in our industry

• Apply intelligence to *every* part of the business
  o Buying
  o Inventory management
  o Logistics optimization
  o Styling recommendations
  o Demand prediction

• Humans and machines augmenting each other

@randyshoup
linkedin.com/in/randyshoup
Styling at Stitch Fix

Inventory

@randyshoup

Personal styling

linkedin.com/in/randyshoup
Personalized Recommendations

Inventory

Machine learning

Algorithmic recommendations

@randyshoup

linkedin.com/in/randysyouh
Expert Human Curation

Algorithmic recommendations

Human curation

@randyshoup
linkedin.com/in/randyshoup
How do we work, and why does it work?
Modern Software Development

Organization
Practices
Technology
Culture
Modern Software Development

- Small Teams
- TDD and Continuous Delivery
- Microservices
- DevOps
Small “Service” Teams

• Teams Aligned to Business Domains
  o Clear, well-defined area of responsibility
  o Single service or set of related services

• Cross-functional Teams
  o All skill sets needed to do the job

• Depend on other teams for supporting services, libraries, and tools

@randyshoup  
linkedin.com/in/randyshoup
Test-Driven Development

- Tests help you go faster
  - Tests “have your back”
  - Development velocity

- Tests make better code
  - Confidence to break things
  - Courage to refactor mercilessly

- Tests make better systems
  - Catch bugs earlier, fail faster
“We don’t have time to do it right!”

“Do you have time to do it twice?”
Test-Driven Development

• Do it right (enough) the first time
  o The more constrained you are on time and resources, the more important it is to build solid features
  o Build one great thing instead of two half-finished things

• Right ≠ Perfect (80 / 20 Rule)

• ➤ Basically no bug tracking system (!!)
  o Bugs are fixed as they come up
  o Backlog contains features we want to build
  o Backlog contains technical debt we want to repay
Continuous Delivery

• Most applications deployed multiple times per day

• More solid systems
  o Release smaller units of work
  o Smaller changes to roll back or roll forward
  o Faster to repair, easier to understand, simpler to diagnose

• Rapid experimentation
  o Small experiments and rapid iteration are cheap
Modern Software Development

- Small Teams
- TDD and Continuous Delivery
- Microservices
- DevOps
DevOps

• End-to-end Ownership
  o Team owns service from design to deployment to retirement

• Responsible for
  o Features
  o Quality
  o Performance
  o Operations
  o Maintenance
You Build It, You Run It.

-- Werner Vogels
Modern Software Development

- Small Teams
- TDD and Continuous Delivery
- Microservices
- DevOps
Evolution to Microservices

• eBay
  • 5th generation today
  • Monolithic Perl → Monolithic C++ → Java → microservices

• Twitter
  • 3rd generation today
  • Monolithic Rails → JS / Rails / Scala → microservices

• Amazon
  • Nth generation today
  • Monolithic Perl / C++ → Java / Scala → microservices

@randyshoup  linkedin.com/in/randyshoup
First Law of Distributed Object Design:

Don’t distribute your objects!

-- Martin Fowler
If you don’t end up regretting your early technology decisions, you probably over-engineered.

-- me
Microservices

- Single-purpose
- Simple, well-defined interface
- Modular and independent
Microservices are nothing more than SOA done properly.

-- me
Microservices

- Single-purpose
- Simple, well-defined interface
- Modular and independent
- Isolated persistence (!)
Microservice Persistence

• Approach 1: Operate your own data store
  o Store to your own instance(s) of {Postgres, MySQL, etc.}, owned and operated by the service team

• Approach 2: Use a persistence service
  o Store to your own schema in {Dynamo, RDS, Spanner, etc.}, operated as a service by another team or by a third-party provider
  o Isolated from all other users of the service

• ➔ Only external access to data store is through published service interface

@randyshoup  
linkedin.com/in/randyslop
Events as First-Class Construct

• “A significant change in state”
  o Statement that some interesting thing occurred
  o 0 | 1 | N consumers subscribe to the event, typically asynchronously

• Traditional 3-tier system
  o Presentation ➔ interface / interaction
  o Application ➔ stateless business logic
  o Persistence ➔ database

• Fourth fundamental building block
  o State changes ➔ events
Events as First-Class Construct

- Events represent how the real world works
  - Finance
  - Software development process
  - Any “workflow”

@randyshoup  
linkedin.com/in/randyshoup
Microservices and Events

• Events are a first-class part of a service interface

• A service interface includes
  o Synchronous request-response (REST, gRPC, etc)
  o Events the service produces
  o Events the service consumes
  o Bulk reads and writes (ETL)

• The interface includes any mechanism for getting data in or out of the service (!)
Extracting Microservices

- Problem: Monolithic shared DB

- stitchfix.com
- Styling app
- Warehouse app
- Merch app
- CS app
- Logistics app
- Payments service
- Profile service

- Clients
- Shipments
- Items
- Styles, SKUs
- Warehouses
- etc.
Extracting Microservices

- Decouple applications / services from shared DB

- stitchfix.com
- Styling app
- Warehouse app
- Merch app
- CS app
- Logistics app
- Payments service
- Profile service

- Clients
- Shipments
- Items
- Styles, SKUs
- Warehouses
- etc.

@randyshoup

linkedin.com/in/randysouph
Extracting Microservices

- Decouple applications / services from shared DB
Extracting Microservices

• Step 1: Create a service
Extracting Microservices

• Step 2: Applications use the service
Extracting Microservices

- Step 3: Move data to private database
Extracting Microservices

• Step 4: Rinse and Repeat
Extracting Microservices

- Step 4: Rinse and Repeat

Diagram showing:
- Styling app
- Warehouse app
- client-service
- item-service
- style-service
- core_client
- core_item
- core_sku

@randyshoup

linkedin.com/in/randysouhp
Extracting Microservices

- Step 4: Rinse and Repeat
Microservice Techniques: Shared Data

• Problem
  - Monolithic database makes it easy to leverage shared data
  - Where does shared data go in a microservices world?
Microservice Techniques: Shared Data

- Principle: Single System of Record
  - Every piece of data is owned by a single service
  - That service is the **canonical system of record** for that data

- Every other copy is a **read-only, non-authoritative cache**

@randyshoup

linkedin.com/in/randyshoup
Microservice Techniques: Shared Data

• Approach 1: Synchronous Lookup
  o Customer service owns customer data
  o Fulfillment service calls customer service in real time
Microservice Techniques: Shared Data

- Approach 2: Async event + local cache
  - Customer service owns customer data
  - Customer service sends \textit{address-updated} event when customer address changes
  - Fulfillment service consumes event, caches current customer address
Microservice Techniques: Shared Data

• Approach 3: Shared metadata library
  o Read-only metadata, basically immutable
  o E.g., size schemas, colors, fabrics, US States, etc.
Microservice Techniques: Joins

• Problem
  o Monolithic database makes joins very easy
  o Splitting the data into separate services makes joins very hard
Microservice Techniques: Joins

• Approach 1: Join in Client Application
  o Get a single customer from customer-service
  o Query matching orders for that customer from order-service

• Best for single A, multiple Bs (1:N join)
Microservice Techniques: Joins

• Many common systems do this
  o Web application “mashup”
Microservice Techniques: Joins

• Approach 2: “Materialize the View”
  o Listen to events from item-service and order-feedback-service
  o Maintain denormalized join of items and order feedback in local storage

• Best for high cardinality A and B (M:N join)
Microservice Techniques: Joins

- Many common systems do this
  - Most NoSQL approaches
  - “Materialized view” in database systems
  - Search engines
  - Analytic systems
  - Log aggregators
Microservice Techniques: Workflows and Sagas

- Problem
  - Monolithic database makes transactions across multiple entities easy
  - Splitting data across services makes transactions very hard

@randyshoup  
linkedin.com/in/randyshoup
Microservice Techniques: Workflows and Sagas

- Transaction ➔ Saga
  - Model the transaction as a state machine of atomic events

- Reimplement as a workflow

- Roll back by applying compensating operations in reverse
Microservice Techniques: Workflows and Sagas

• Many common systems do this
  o Payment processing
  o Expense approval
  o Any multi-step workflow
Microservice Techniques: Workflows and Sagas

- Ideal use for Functions as a Service ("Serverless")
  - Very lightweight logic
  - Stateless
  - Triggered by an event

@randyshoup
linkedin.com/in/randylshoup
Thanks!

- Stitch Fix is hiring!
  - [www.stitchfix.com/careers](http://www.stitchfix.com/careers)
  - Based in San Francisco
  - **Hiring everywhere!**
  - More than half remote, all across US
  - Application development, Platform engineering, Data Science

- Please contact me
  - @randyshoup
  - linkedin.com/in/randyshoup