Strategies for Decentralized Data Management in Microservices Architectures

O’Reilly Software Architecture Conference
2/27/2018
@jpmorgenthal
Microservices is the new <put popular silver bullet solution here>
What I Like About Microservices

- Ease of scaling in and out
- Low likelihood of introducing entanglements as a technical debt
- Separation of concerns
- Isolation of change
- Simplified Testing
What is the goal?
Goals of Decentralized Data Management (DDM) in μServices Architectures

- Remove scaling limits due to data access
- Isolation of change
  - Shifting to microservices facilitates the possibility of staging data before commit
- Reduced fragility
  - Data model changes have less potential for application-wide impact
- Easier introduction for emerging data management technologies
- Easier modeling of multi-pathing for performance
- Simplified detection and correction of exceptions in production
- Removal of assumptions that lead to increased technical debt
DDM Addresses Both Messaging and Persistence

Systems of Record

XML

Transform

ODS

Business Logic

XML

JSON

Monolith
What Is A Monolith Anyway?
To create individual Web and Distributor μServices we must deal with the entanglement of data and process.
Quantum entanglement is a physical phenomenon which occurs when pairs or groups of particles (services) are generated or interact in ways such that the quantum state of each particle cannot be described independently of the state of the other(s), even when the particles are separated by a large distance—instead, a quantum state must be described for the system as a whole.
Phase 1 Decomposition

Web Orders ➔ Consumer Orders ➔ Warehouse

Distributor Orders ➔ Distributors ➔ Orders ➔ 3PL
Phase 1 Decomposition

PROS:
- Service failures don’t lead to indeterminable state
- Simplifies troubleshooting in production
- Allows other services to subscribe to same content reducing need for dependencies
- Additional scaling options

CONS:
- Increased complexity of overall system
DDM Patterns

- Read Only Replica
- Write-Thru Cache
- Streaming
- ACID
**DDM Patterns**

Generate Pick Tickets

Warehouse DB

Select * from inv
join Picks on inv.sku=picks.sku

Inventory Service

Oltp

Fulfillment Service

Oltp

POST /picks/generatePickTickets
GET /fulfillement/openOrders
GET /inventory/orderId/lines/sku

Join Pattern