Applying the Saga Pattern

Caitie McCaffrey
Caitie McCaffrey
Distributed Systems Engineer

@Caitie
CaitieM.com
• Why Sagas?
• Sagas Paper
• Distributed Sagas
• Sagas in Halo 4
Systems Used to Be Simple
Serializability & ACID
SOA/Microservices
Two Phase Commit
2PC: Prepare Phase

Coordinator

Propose→ Resource 1
Vote←

Propose→ Resource 2
Vote←

Propose→ Resource 3
Vote←

Car

Building

Airplane
2PC: Commit Phase
2PC Doesn’t Scale

• $O(n^2)$ Messages
• Coordinator: Single Point of Failure
• Reduced Throughput
Spanner
Google’s Globally-Distributed Database
Corbett et. al.
“Spanner is Google’s scalable, multi-version, globally distributed, and synchronously-replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions.”

–Corbett et al.
“The key enabler of these properties is a new TrueTime API and its implementation...using multiple modern clock references (GPS and atomic clocks).”

–Corbett et al.
Spanner is Expensive & Proprietary

• Custom Hardware & Data-Centers
• Synchronization Not Solved
Distributed Transactions are Hard & Expensive
Can We Do Better?
Sagas
Hector Garcia-Molina, Kenneth Salem
Princeton University 1987
Sagas are Long Lived Transactions
“A Saga is a Long Lived Transaction that can be written as a sequence of transactions that can be interleaved.

All transactions in the sequence complete successfully or compensating transactions are ran to amend a partial execution.”
A Saga is a Collection of Sub-Transactions

\[ T_1, T_2 \ldots T_n \]
Each Sub-Transaction has a Compensating Transaction

\[ C_1, C_2 \ldots C_n \]
$C_n$ Semantically Undoes $T_n$
Saga Guarantee

Either

• $T_1, T_2 \ldots T_n$ or

• $T_1, T_2 \ldots T_j, C_j, \ldots C_2, C_1$
Trade-Off: Atomicity for Availability
Sagas are a Failure Management Pattern
Large Single Transaction
Sagas

- Book Hotel ($T_1$)
- Book Car ($T_2$)
- Book Flight ($T_3$)
- Cancel Hotel ($C_1$)
- Cancel Car ($C_2$)
- Cancel Flight ($C_3$)
Saga Execution Coordinator (SEC)
Saga Log

- Begin Saga
- End Saga
- Abort Saga

- Begin $T_i$
- End $T_i$
- Begin $C_i$
- End $C_i$
Successful Saga

Begin Saga
Successful Saga

Begin Saga

Start Book Hotel ($T_1$)
Successful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)
Successful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)
Successful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

End Book Car Rental (T₂)
Successful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

End Book Car Rental (T₂)

Start Book Flight (T₃)
Successful Saga

Begin Saga

Start Book Hotel ($T_1$)

End Book Hotel ($T_1$)

Start Book Car Rental ($T_2$)

End Book Car Rental ($T_2$)

Start Book Flight ($T_3$)

End Book Flight ($T_3$)
Successful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

End Book Car Rental (T₂)

Start Book Flight (T₃)

End Book Flight (T₃)

End Saga
Unsuccessful Saga
Backwards Recovery
Unsuccessful Saga

Begin Saga
Unsuccessful Saga

Begin Saga

Start Book Hotel ($T_1$)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

Abort Saga
Unsuccessful Saga

Begin Saga

Start Book Hotel ($T_1$)

End Book Hotel ($T_1$)

Start Book Car Rental ($T_2$)

Abort Saga

Start Compensate Car Rental ($C_2$)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

Abort Saga

Start Compensate Car Rental (C₂)

End Compensate Car Rental (C₂)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

Abort Saga

Start Compensate Car Rental (C₂)

End Compensate Car Rental (C₂)

Start Compensate Book Hotel (C₁)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

Abort Saga

Start Compensate Car Rental (C₂)

End Compensate Car Rental (C₂)

Start Compensate Book Hotel (C₁)

End Compensate Book Hotel (C₁)
Unsuccessful Saga

Begin Saga

Start Book Hotel (T₁)

End Book Hotel (T₁)

Start Book Car Rental (T₂)

Abort Saga

Start Compensate Car Rental (C₂)

End Compensate Car Rental (C₂)

Start Compensate Book Hotel (C₁)

End Compensate Book Hotel (C₁)

End Saga 😞 🚫 🌴
Sagas in Distributed Systems

“Due to space limitations, we only discuss Sagas in a centralized System, although clearly they can be implemented in a distributed database system.”

–Molina et. al
SOA/Microservices

- Travel App
  - Book Trip
  - Travel Agent Svc
    - Book Hotel
      - Hotel Reservation Svc
      - Rent Car
        - Car Rental Svc
      - Book Flight
        - Airline Reservation Svc
Requests instead of Transactions

- Book Hotel (T_1)
- Book Car (T_2)
- Book Flight (T_3)
- Cancel Hotel (C_1)
- Cancel Car (C_2)
- Cancel Flight (C_3)
A Distributed Saga is a Collection of Sub-Requests

$$T_1, T_2 \ldots T_n$$

Each Sub-Request has a Compensating Request

$$C_1, C_2 \ldots C_n$$
Successful Distributed Saga

Begin Saga

Start Book Hotel Request (T₁)

End Book Hotel Request (T₁)

Start Book Car Rental Request (T₂)

End Book Car Rental Request (T₂)

Start Book Flight Request (T₃)

End Book Flight Request (T₃)

End Saga
Saga Log
Durable & Distributed
Saga Execution Coordinator (SEC)

- Interprets & Writes to Saga Log
- Applies Saga Sub-Requests
- Applies Saga Compensating Requests when Necessary
Apply Compensating Requests

- Aborted Saga Response
- Start Request Fails
- SEC Crashes (non-safe state)
What Happens when Compensating Requests Fail?
Compensating Requests Must Be Idempotent
What Happens when SEC Fails?
Safe States

• All Executed Sub-Requests are Complete
  (Start Ti & End Ti both logged)

• Saga has been Aborted, Proceed with Compensating Transactions
Un-Safe State

- Start Ti logged, no End Ti logged
  - Abort Saga
  - Start Compensating Requests
Request Messaging Semantics

- Sub-Requests ($T_i$): At Most Once
- Compensating Requests ($C_i$): At Least Once
Distributed Saga Guarantee

Either

- $T_1, T_2 \ldots T_n$ or

- $T_1, T_2 \ldots T_j, C_j, \ldots C_2, C_1$
Distributed Sagas

- Distributed/Durable Saga Log
- SEC Process
- Compensating Requests: Idempotent
Halo Statistics
Halo Statistics Service with Sagas
Forward-Recovery
Forward Recovery
Sub-Requests Must Also Be Idempotent
Sagas

• Long Lived / Distributed Transactions
• Trade Atomicity for Availability
• Failure Management Pattern
Thank You

- @pbailis
- @randommood
- @tsantero
- @aphyr
- @jmhodges
- @clemnsv
Questions?

Please remember to evaluate via the GOTO Guide App

@Caitie