COMMAND-QUERY RESPONSIBILITY INTEGRATION (CQRS)

Motivation for Distribution

- Organizational
  - Bounded context
- Separation of Concerns
- Scalability
- Fault Tolerance

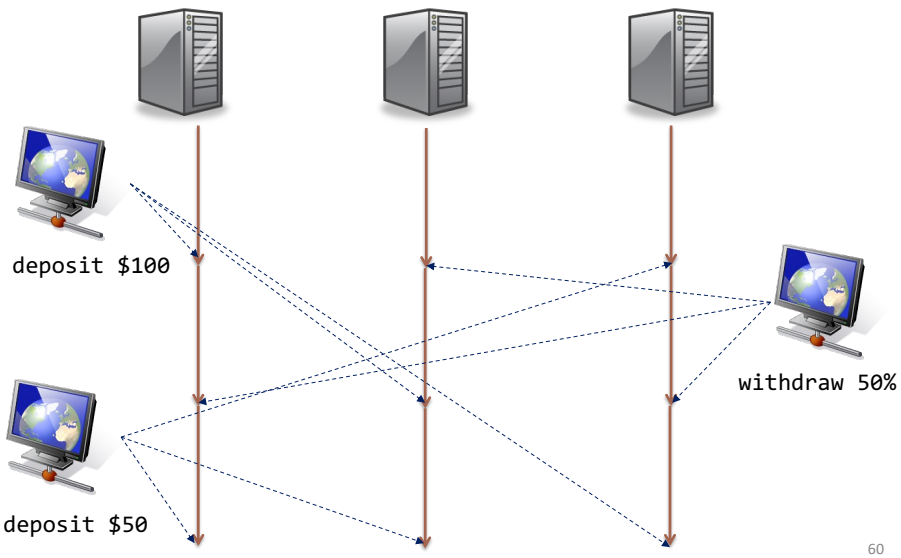
Motivation for Distribution

- Organizational
  - Bounded context
- Separation of Concerns
- Scalability
  - Asynchronous communication
  - Eventual consistency
- Fault Tolerance

59

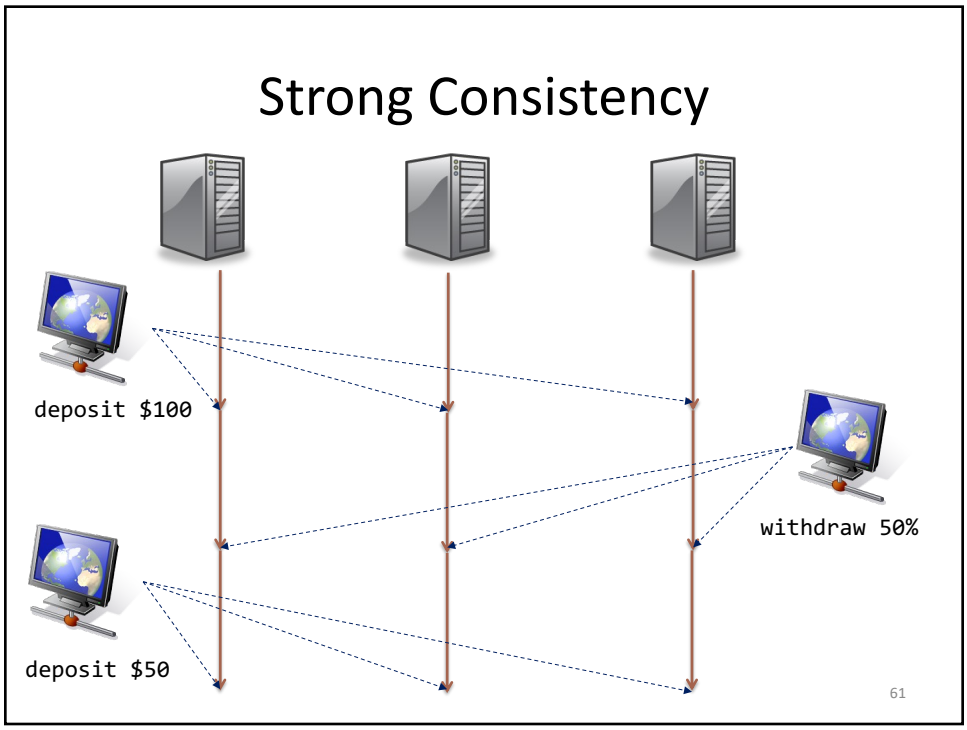
59

No Consistency

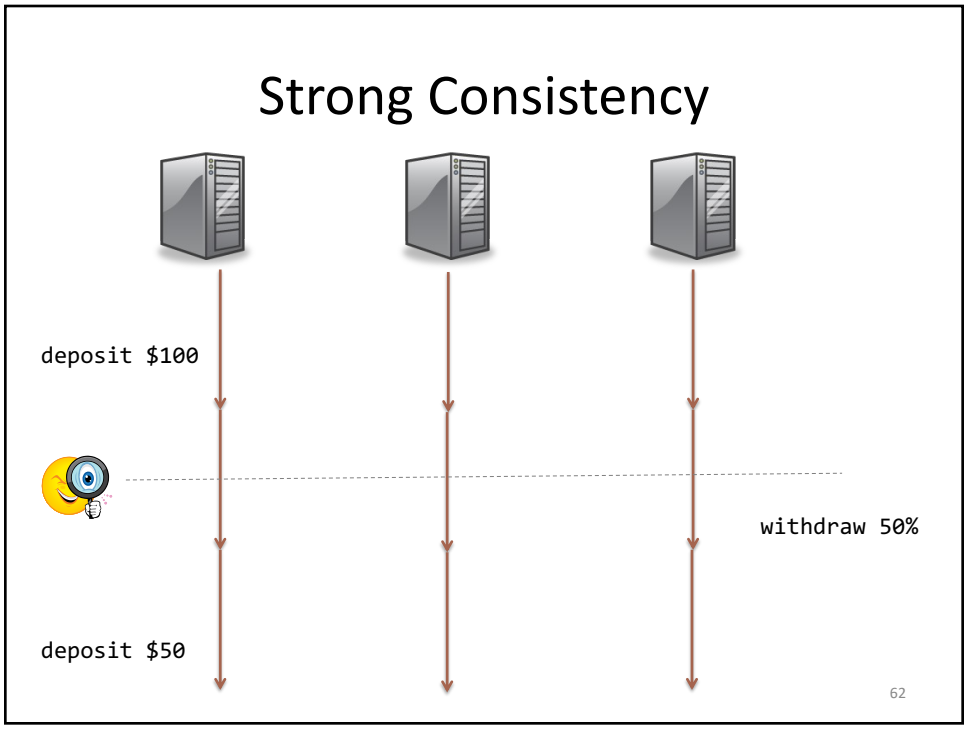


60

60


Strong Consistency
deposit $100
withdraw 50%
deposit $50
61

61


Strong Consistency
deposit $100
withdraw 50%
deposit $50
62

62

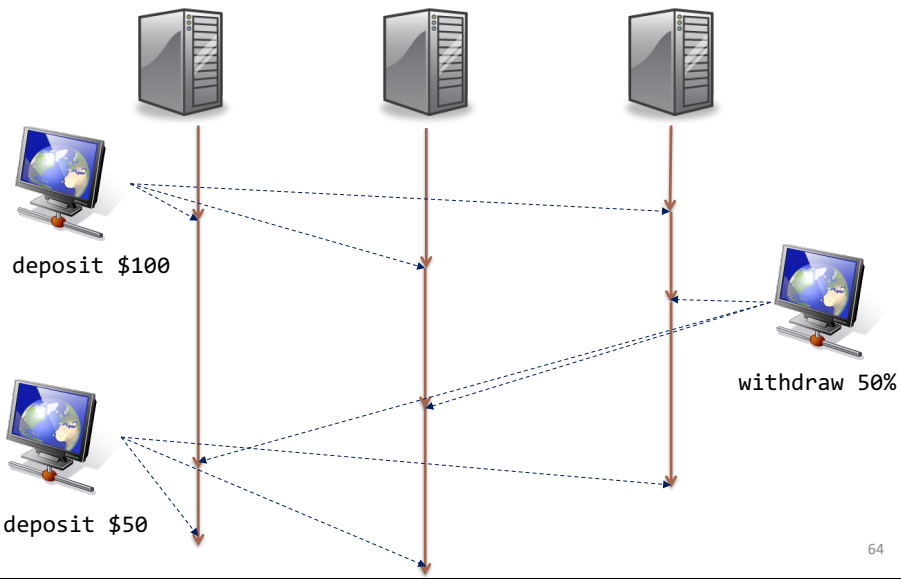
Strong and Eventual Consistency

- Strong Consistency: Effect of an update is visible by any operation that follows it
  - No stale reads
- Eventual Consistency
  - Consistent ordering: Updates are done in same order on all replicas
  - Total propagation: Updates are performed on all replicas *eventually*

63

63

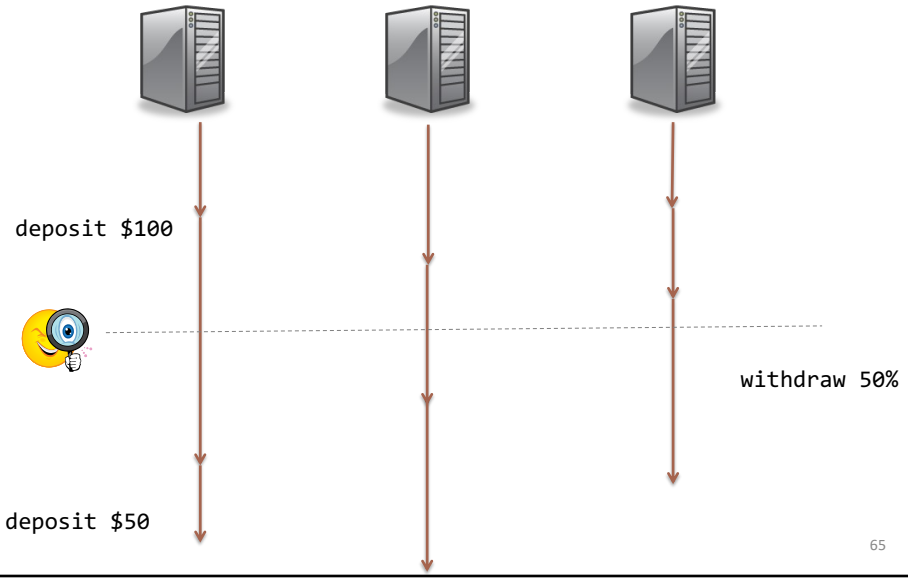
Eventual Consistency



64

64

Eventual Consistency



65

65

CRUD

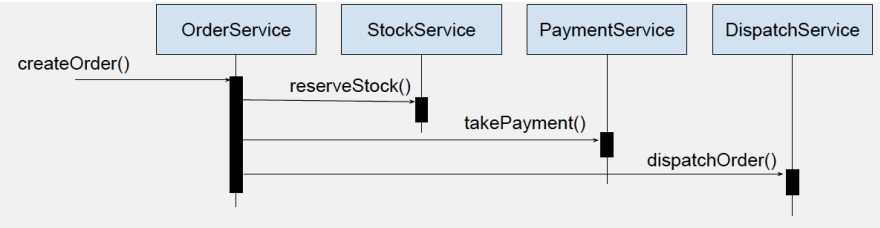
- Strong consistency via transactions
- Shortcomings
  - Scalability
  - Transaction coordination
  - **Reproducibility** (audit logs)

66

66

Synchronous Microservice

- `orderService` orchestrates microservices

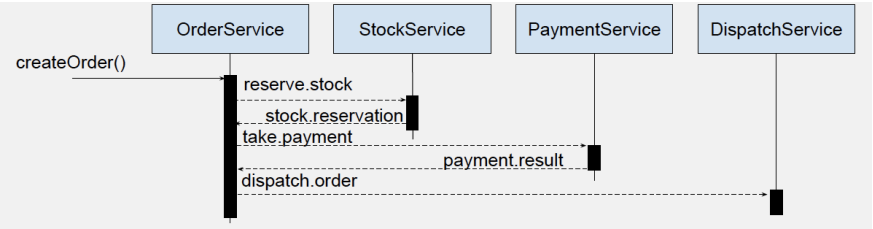


67

67

Asynchronous Microservice

- `orderService` **still** orchestrates microservices

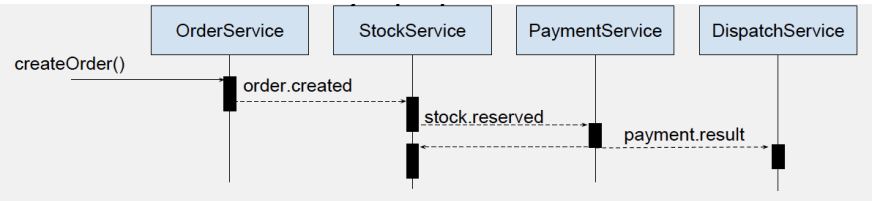


68

68

Choreography of Microservices

- Microservices react to events

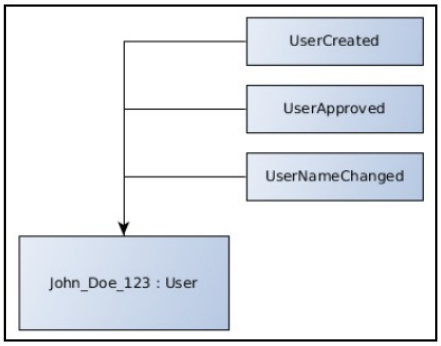


69

69

Event Sourcing

- Calculate current state from event log
- Events are immutable

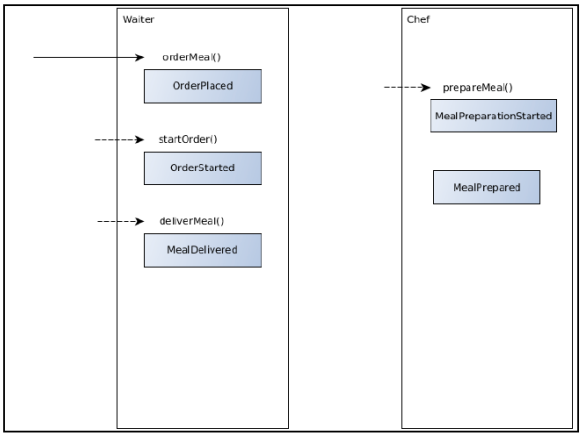


70

70

Eventual Consistency

- Example: Restaurant Order



71

71

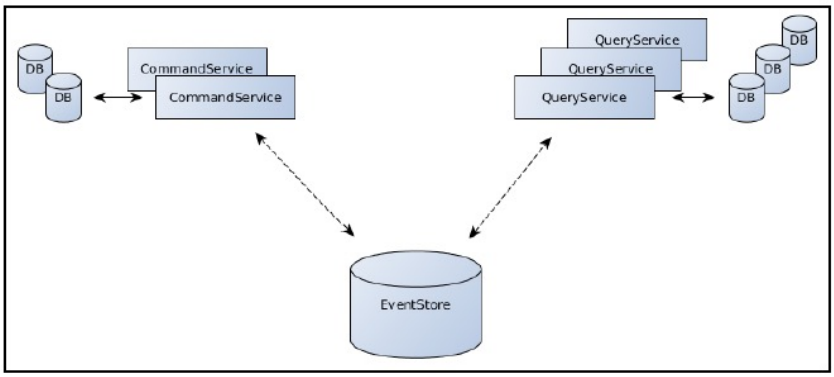
CQRS: Command Query Responsibility Segregation

- Command:
  - Succeed or fail (no return value)
  - Produces events
- Query
  - Return data
  - Cannot modify state
- Applications update state by consuming events

72

72

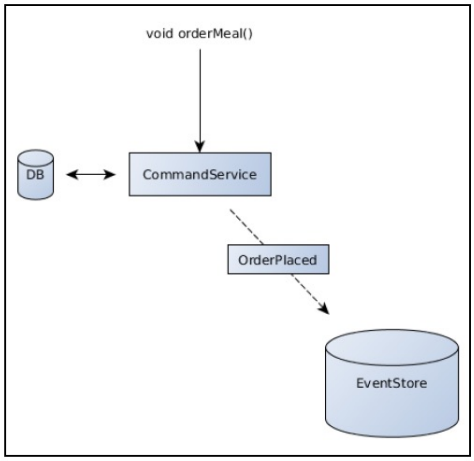
CQRS: Command Query Responsibility Segregation



73

73

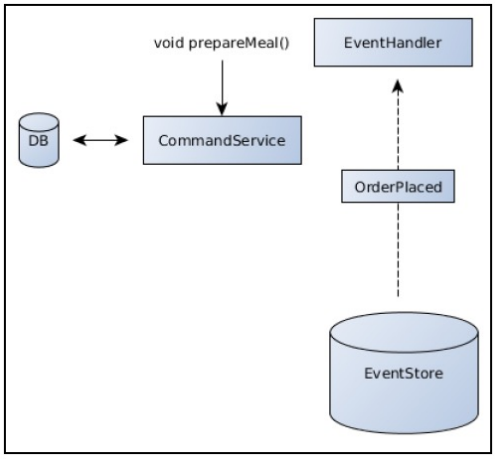
Example: Order a Meal



74

74

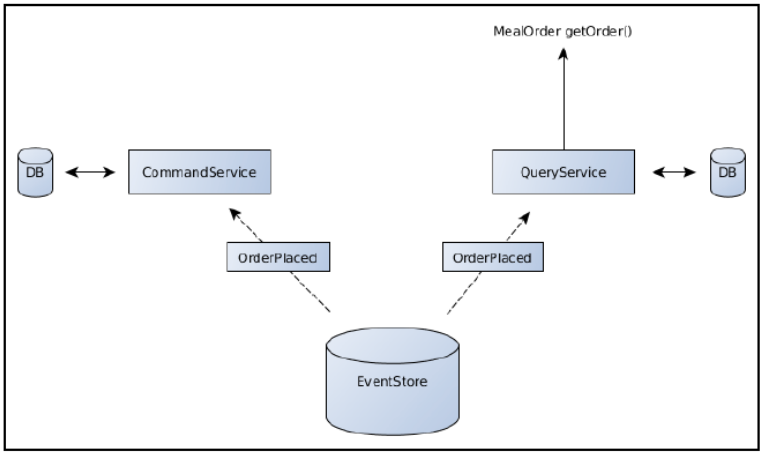
Example: Order a Meal



75

75

Example: Order a Meal



76

76

CQRS Benefits

- Horizontal scaling
  - Different for command and query services
- Optimize state representations
  - Appropriate to the service
- Read-side failover availability
  - Handle queries with event log down
- Scalability of event-sourced systems

77