Compensating Transactions: When ACID is Too Much

Dr Paul Robinson
paul.robinson@redhat.com
@pfrobinson
Agenda

• Background
• Non-transactional resources
• Cross-domain distributed transactions
• Long-lived transactions
• What we have today & planned
• Getting started
ACID Transactions

Positives
- Strong guarantees
- Tolerate (certain) failures
- Easy API to develop against

Negatives
- Locking resources
- Can reduce throughput
- Requires two-phase aware resources
- Tight-coupling
ACID transactions not suitable?
Don’t give up on transactions altogether!
Extended Transactions

- Alternative to ACID transactions
- Many different types
- Relaxes some ACID properties
Compensation-based Transactions

- Based on Sagas
  - Academic research from 1987
- ACID vs Compensation-based transactions
  - ACID
    - Phase <=1: acquire locks & log
    - Phase 2: commit/rollback changes
  - Compensation-based
    - Phase <=1: commit changes (TODO: how to explain?)
    - Phase 2: confirm? / compensate change
Hang on….
What effect does this have on Isolation and Consistency?
Looks a bit dubious to me!
Isolation: ACID
Isolation: Compensations
Existing Support in Narayana

- WS-BusinessActivity
  - Oasis Web Services standard
  - Standardizes
    - Message format
    - State machine
    - No API
  - Problems?
    - Web services only
    - Low-level API
Compensations API

- JTA-like API
  - Application API only
  - Maybe AS and RM in future
- Interop with JTA 1.2
- Transport Neutral
Onto the examples...
Non-Transactional Resources

- Transactional Resources
  - Participate in a two-phase protocol
    - Prepare and later commit/rollback

- Non-transactional resources
  - Can’t participate in a two-phase protocol

Compensation-based transaction is an option
Code Example

• Ecommerce site, selling books
• Coordinates:
  • Updating databases
  • Dispatching package
public class BookService {

    @Inject
    PackageDispatcher packageDispatcher;

    @Compensatable
    public void buyBook(String item, String address) {
        packageDispatcher.dispatch(item, address);
        //other activities, such as updating inventory and charging the customer
    }
}
Non-transactional Resource

```java
public class PackageDispatcher {

    @Inject
    OrderData orderData;

    @TxCompensate(RecallPackage.class)
    public void dispatch(String item, String address) {
        orderData.setAddress(address);
        orderData.setItem(item);
        //Dispatch the package
    }
}
```
@CompensationScoped
public class OrderData {

    private String item;
    private String address;
    ...
}

public class RecallPackage implements CompensationHandler {

    @Inject
    OrderData orderData;

    @Override
    public void compensate() {
        //Recall the package somehow
    }
}
Cross-Domain Distributed Transactions

- Distribution => Increased chance of failure
  - Use transactions!
- Distribution => Increased latency
  - Maybe not ACID transactions then
- Cross-domain => loose coupling
  - ACID transactions => tight-coupling

Compensation-based transaction is an option
Code Example

- Travel Agent
- Coordinates:
  - Booking of a Taxi
  - Booking of a Hotel
- Taxi and Hotel Service are remote Web Services
  - WS-BA
public class Client {

    @Compensatable
    public void makeBooking() {

        // Lookup Hotel and Taxi Web Service ports here...

        hotelService.makeBooking("Double", "paul.robinson@redhat.com");
        taxiService.makeBooking("Newcastle", "paul.robinson@redhat.com");
    }
}
@WebService
public class HotelService {

@Inject BookingData bookingData;

@Compensatable(MANDATORY)
@TxCompensate(CancelBooking.class)
@TxConfirm(ConfirmBooking.class)
@Transactional(REQUIRES_NEW)
@WebMethod
public void makeBooking(String item, String user) {

    //Update the database to mark the booking as pending...
    bookingData.setBookingID("the id of the booking goes here");
}
}
public class ConfirmBooking implements ConfirmationHandler {

    @Inject
    BookingData bookingData;

    @Transactional(REQUIRES_NEW)
    public void confirm() {
        // Confirm order for bookingData.getBookingID() in Database (in a new JTA transaction)
    }
}
public class CancelBooking implements CompensationHandler {

    @Inject
    BookingData bookingData;

    @Transactional(REQUIRES_NEW)
    public void compensate() {
        // Cancel order for bookingData.getBookingID() in Database (in a new JTA transaction)
    }
}
Long Lived Transactions (LLT)

- ACID Transactions => all or nothing
  - OK for short lived transactions (SLT)
  - Failure in LLT leads to significant loss of work

- Compensation-based Transactions
  - Split work into many SLT
  - Find alternative if one fails
  - Compensate all if no alternative
Code Example

• Travel Agent
• Coordinates:
  • Hotel service
  • 2 Taxi Services
• Aim to book 1xHotel & 1xTaxi
• Hotel booking is important
Compensation Handler

```java
public class Agent {

    @Inject ...

    @TxCompensate(cancelOn=BookingException.class)
    public void makeBooking(String email, String room, String dest) throws BookingException {

        hotelService.makeBooking(room, email);

        try {
            taxi1Service.makeBooking(dest, email);
        } catch (BookingException e) {
            taxi2Service.makeBooking(dest, email);
        }
    }
}
```
What we Have Today

- Initial version of the API
  - @Compensatable
  - With handlers
  - REQUIRED only
  - @CompensationScoped
  - LLT
- Distribution over WS-BA
- Blog post series
- Today: Available in Narayana 5.0.0.M3 and WildFly nightly build
- June 11th: Available in WildFly 8.0.0.Alpha2
What we Have Planned

- Quickstarts
  - All examples seen today
- Support more complex use-cases
  - Multiplicity
- Recovery
  - @CompensationScoped
  - JTA interop
- EJB support
- Specification
- Throughput comparisons
- 2PC participants
- NoSQL RM integration
Getting Started

- Explore the quickstarts (when they are ready)
  - http://github.com/jbosstm/quickstart/
- Give feedback
  - http://community.jboss.org/en/jbosstm
- Track issues
  - http://issues.jboss.org/browse/JBTM
  - TXFramework component
- Subscribe to Blog
  - http://jbossts.blogspot.co.uk/
- Contact me
  - paul.robinson@redhat.com, @pfrobinson