Recovery in JOTM

Marek Procházka
INRIA Rhône-Alpes

Presentation Outline

- Recovery overview
- X/Open XA support for recovery
- OTS support for recovery
- Other points to solve
- JOTM: current state & what to do

Recovery Overview

- **Recovery = ability to process transaction in fault-tolerant manner**

- **How to do that? – By logging information and using it during the recovery procedure**

- **2 phases:**
  1. **Normal operation:** Storing required information to stable storage (*log file*)
  2. **Recovery:** After a failure, system is recovered by employing data stored in the log file

Logging

- **Selecting media**
  - Database-based log
  - File-based log

- **Log size**
  - Particular problem with removing log records that are not useful
  - Searching all transaction that were active at the time of system failure
  - Long operation of the system without any failure – potentially very large log file
    - Disk out-of-space
    - Uselessly long duration of the recovery phase

Storing Required Data: Transaction State

- **What to store:**
  - Transaction id, significant event + other important data

```
void begin(Xid trid)

void 2pcStart(Xid trid)
void voteEnd(Xid trid, int heuristicDecision)

void rollbackStart(Xid trid)
void rollbackEnd(Xid trid, int heuristicDecision)

void commitStart(Xid trid, boolean twoPhaseCommit)
void commitEnd(Xid trid, int heuristicDecision)
```

Storing Required Data: XA Resources

- Used for JDBC connections, JMS sessions
- What to store: dataSourceName, userName, userPassword

```
javax.sql.XADataSource ds =
    (DataSource) ctx.lookup(dataSourceName);

javax.sql.XAConnection con =
    ds.getXAConnection(userName, userPassword);

javax.transaction.xa.XAResource res =
    con.getXAResource();

javax.transaction.xa.Xid [] trid =
    xaResourceList[i].recover(
        javax.transaction.xa.XAResource.TMNOFLAGS);
```

Storing Required Data: OTS Resources

- **Used for maintaining tree of sub-coordinators**
- **Recovery Coordinator interface**

```
interface RecoveryCoordinator {
  Status replay_completion(in Resource r)
    raises (NotPrepared);
};
```

- **Recovery Coordinator retrieved during resource registration**

```
RecoveryCoordinator rc =
    Coordinator.register_resource(Resource)
```

Storing Required Data: OTS Resources

- **Resource object is itself responsible for recovery**
- **To be able to do so, what Resource object does:**
  - Before `VotePrepare`, it stores:
    - Resource object
    - Reference to `Resource` object
    - Reference to `RecoveryCoordinator`
  - During recovery
    - Determine transaction outcome by invoking `RecoveryCoordinator.replay_completion()`
    - Continue completion
- **`XAResource.recover()` has no equivalent in OTS**
- **`RecoveryCoordinator.replay_completion()` has no equivalent in X/Open X**

Container Recovery

- **Recovery of other resources related to containers?**
- **Examples:**
  - EJB: Session beans implementing `SessionSynchronization` (`afterCompletion()` method invocation during recovery)
  - EJB: Entity beans are not subject for recovery, since the `store()` method is called before two-phase commit
  - OTS: Synchronization objects “are not recoverable” – they do not take part in recovery
  - OTS: Subtransactions “are not durable” – they do not take part in recovery
- **Other middleware platforms – probably other container-specific data subject for recovery**

Recovery Algorithm

- **Transactions active at the time of JOTM failure are subjects for recovery**
  - All active transactions are rolled back
  - Transactions prepared to rollback rolled back
  - Transactions prepared to commit are attempted to commit

- **All resources (XA resources, OTS resources) are reconnected and take part in recovery**

Other Points To Solve

- **User intervention**
  - If recovery could not proceed
  - If heuristic exceptions used
- **Transaction timeouts**
  - Usually passed, but cannot be checked during recovery
  - Databases use timeouts, but there is no support in SQL (XA resource timeouts propagated?)
- **Independence on transaction model used**
  - Specific features cannot be employed (e.g., compensation)
  - API of an adaptable recovery service?
- **Independence on middleware platform**
  - Use X/Open XA, JTA, OTS

Recovery Requirements

- **JTA transaction demarcation (JOTM)**
- **Enlistment of XA resources for registering data connections**
  - Implementation of the `recover()` method and XA resource persistence required
- **Enlistment of OTS resources for registering sub-coordinators**
  - Implementation of `RecoveryCoordinator`, and OTS Resource persistence required

JOTM: What To Do

1. **Physical log: database- or file- based, log structure, efficient data retrieval**
2. **Persistent XA resources for storing XA resources:**
   - Implement the `recover()` method in the JDBC 1.0 wrapper (currently returns null)
   - Check how is it with JDBC 2.0 drivers
   - Check how is it with JMS XA sessions
3. **Persistent OTS resources for storing tree of transaction sub-coordinators**
4. **LogWriter and LogReader: storing and retrieving data required for recovery**
5. **RecoveryCoordinator: the recovery algorithm**
6. **Enable recovery in JOTM: add logging points to the current JOTM code**