3C03 Concurrency: Distributed Transactions

Wolfgang Emmerich

Outline

- *Roles in Distributed Transaction Processing*
- *Two Phase Commit Protocol (2PC)*
- *Impact of 2PC on Concurrency Control*
- *CORBA Object Transactions*
- *Summary*

Roles of Components

- ***Distributed system components involved in transactions can take role of:***
- ***Transactional Client***
- ***Transactional Server***
- ***Coordinator***

Coordinator

- ***Coordinator plays key role in managing transaction.***
- ***Coordinator is the component that handles begin / commit / abort transaction calls.***
- ***Coordinator allocates system-wide unique transaction identifier.***
- ***Different transactions may have different coordinators.***

Transactional Server

- *Every component with a resource accessed or modified under transaction control.*
- *Transactional server has to know coordinator.*
- *Transactional server registers its participation in a transaction with the coordinator.*
- *Transactional server has to implement a transaction protocol (two-phase commit).*

Transactional Client

- *Only sees transactions through the transaction coordinator.*
- *Invokes services from the coordinator to begin, commit and abort transactions.*
- *Implementation of transactions are transparent for the client.*
- *Cannot tell difference between server and transactional server.*

Two-Phase Commit

- *Multiple autonomous distributed servers:*
  - *For a commit, all transactional servers have to be able to commit.*
  - *If a single transactional server cannot commit its changes every server has to abort.*
- *Single phase protocol is insufficient.*
- *Two phases are needed:*
  - *Phase one: Voting*
  - *Phase two: Completion.*

Phase One

- *Called the voting phase.*
- *Coordinator asks all servers if they are able (and willing) to commit.*
- *Servers reply:*
  - *Yes: it will commit if asked, but does not yet know if it is actually going to commit.*
  - *No: it immediately aborts its operations.*
- *Hence, servers can unilaterally abort but not unilaterally commit a transaction.*

Phase Two

- Called the completion phase.
- Co-ordinator collates all votes, including its own, and decides to
  - commit if everyone voted ‘Yes’.
  - abort if anyone voted ‘No’.
- All voters that voted ‘Yes’ are sent
  - ‘DoCommit’ if transaction is to be committed.
  - Otherwise ‘Abort'.
- Servers acknowledge DoCommit once they have committed.

Server Uncertainty (1)

- Period when a server must be able to commit, but does not yet know if has to.
- This period is known as server uncertainty.
- Usually short (time needed for co-ordinator to receive and process votes).
- However, failures can lengthen this process, which may cause problems.

Recovery in Two-Phase Commit

- *Failures prior to start of 2PC results in abort.*
- *Coordinator failure prior to transmitting commit messages results in abort.*
- *After this point, co-ordinator will retransmit all Commit messages on restart.*
- *If server fails prior to voting, it aborts.*
- *If it fails after voting, it sends GetDecision.*
- *If it fails after committing it (re)sends HaveCommitted message.*

Complexity

- *Assuming N participating servers:*
- *(N-1) Voting requests from coordinator to servers.*
- *(N-1) Completion requests from coordinator to servers.*
- *Hence, complexity of requests is linear in the number of participating servers.*

Committing Nested Transactions

- Cannot use same mechanism to commit nested transactions as:
  - subtransactions can abort independently of parent.
  - subtransactions must have made decision to commit or abort before parent transaction.
- Top level transaction needs to be able to communicate its decision down to all subtransactions so they may react accordingly.

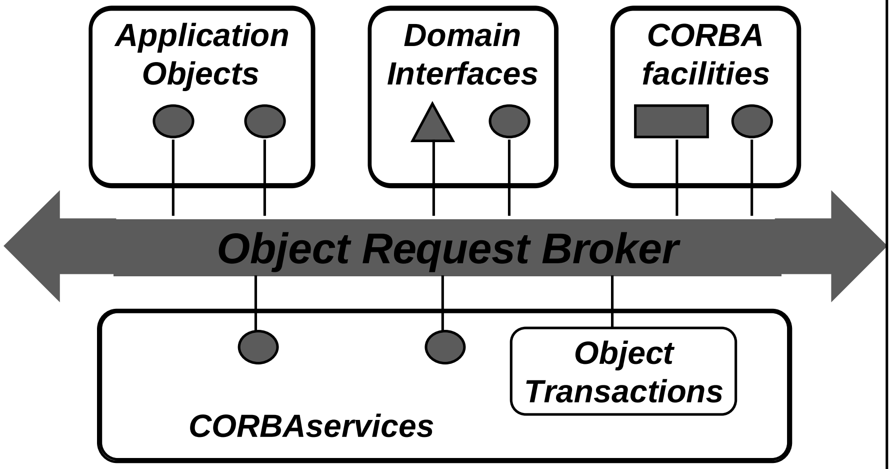
Provisional Commit

- Subtransactions vote either:
  - aborted or
  - provisionally committed.
- Abort is handled as normal.
- Provisional commit means that coordinator and transactional servers are willing to commit subtransaction but have not yet done so.

Locking and Provisional Commits

- ***Locks cannot be released after provisional commit.***
- ***Data items remain 'protected' until top-level transaction commits.***
- ***This may reduce concurrency.***
- ***Interactions between sibling subtransactions:***
  - ***should they be prevented (different)?***
  - ***allowed (part of the same transaction)?***
- ***Generally they are prevented.***

CORBA Transaction Service



IDL Interfaces

- ***Object Transaction Service defined through three IDL interfaces:***
- ***Current***
- ***Coordinator***
- ***Resource***

Current

```
interface Current {
  void begin() raises (...);
  void commit (in boolean report_heuristics)
       raises (NoTransaction, HeuristicMixed,
               HeuristicHazard);
  void rollback() raises(NoTransaction);
  Status get_status();
  string get_transaction_name();
  Coordinator get_control();
  Coordinator suspend();
  void resume(in Coordinator which)
       raises(InvalidControl);
};
```

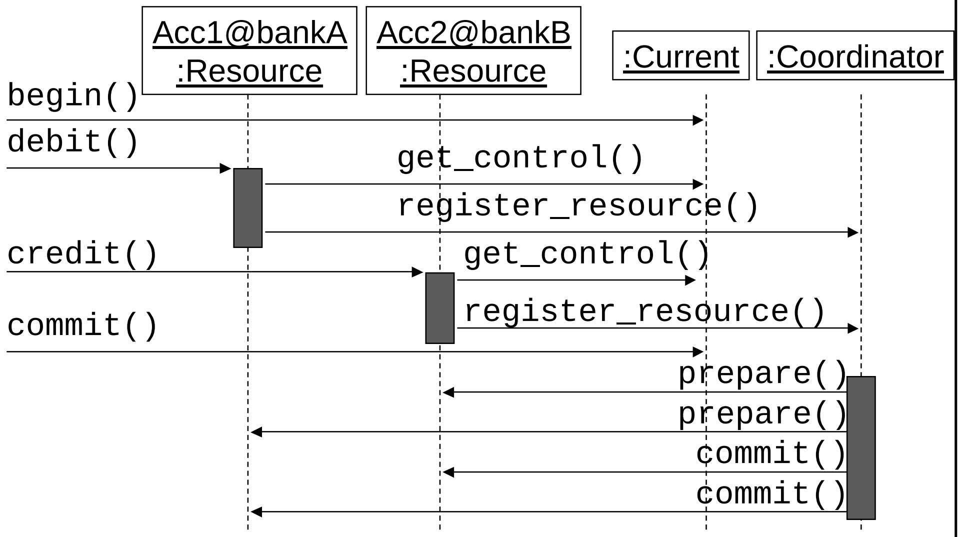
Coordinator

```
interface Coordinator {
 Status get_status();
 Status get_parent_status();
 Status get_top_level_status();
 boolean is_same_transaction(in Coordinator tr);
 boolean is_related_transaction(in Coordinator tr)
 RecoveryCoordinator register_resource(
         in Resource r) raises(Inactive);
 void register_subtran_aware(
         in SubtransactionAwareResource r)
         raises(Inactive, NotSubtransaction);
  …
};
```

Resource

```
interface Resource {
  Vote prepare();
  void rollback() raises(...);
  void commit() raises(...);
  void commit_one_phase raises(...);
  void forget();
};
interface SubtransactionAwareResource:Resource
  {
  void commit_subtransaction(in Coordinator p);
  void rollback_subtransaction();
};
```

Example: Funds Transfer



Summary

- ***Roles in Distributed Transaction Processing***
- ***Two Phase Commit Protocol (2PC)***
- ***Impact of 2PC on Concurrency Control***
- ***CORBA Object Transactions***
- ***Next Session: Questions & Answers***