Remote Batch Invocation for Compositional Object Services

CPS 296.1
9th Feb 2010

Vamsidhar Thummala
Content borrowed from William R Cook

Standard Approach to Distribution

- Step I: Design a language
  - Clean interfaces, modules
- Step 2: Add library for distribution
  - Remote procedure calls
    - Stub that send calls remotely
  - Distributed objects
    - Proxies: pointer to remote object
    - Create proxies on demand
- End Result
  - Clean, elegant, orthogonal
- Is it the case for Persistence too?
  - Example in later slides

- C, C++, ML
- RMI
- CORBA
- DCOM
- Cons?
  - Performance

Example: Music Jukebox in the Cloud

- Remote service which can play music on your home speakers
- Fine-grained interface
- OO design

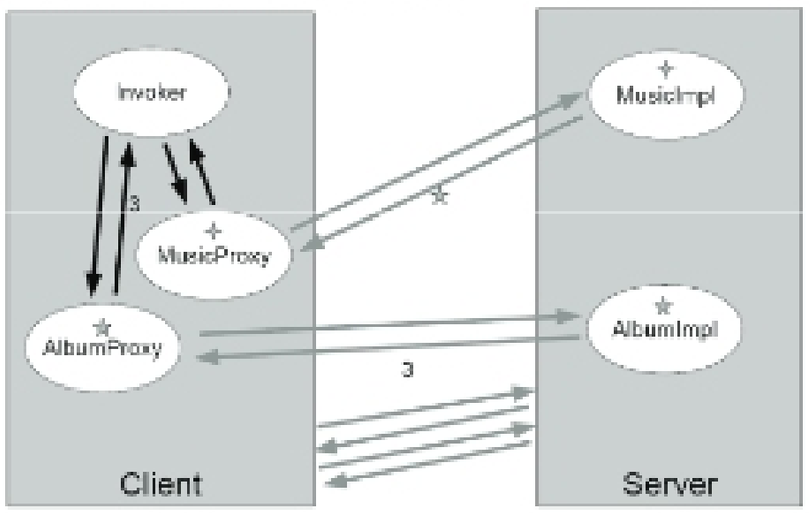
```
interface Music {
   Album[] getAlblums();
   .....
}

interface Alblum {
   String getTitle();
    void play();
    int rating();
    ....
}
```

Remote Procedure Calls (RPC)

```
int minimumRating = 4;
Music musicService = ... ;
for (Album album : musicService.getAlbums()) {
  if (album.rating() > minimumRating) {
        System.out.println("Played: " +
                           album.rating() + " " +
                           album.getTitle());
                           album.play();
  } else {
        System.out.println("Skipped: " +
                           album.getTitle());
  }
}
```

RPC behind the scenes



Result: Too many RPC calls

```
int minimumRating = 4;
Music musicService = ... ;
for (Album album : musicService.getAlbums()) {
  if (album.rating() > minimumRating) {
        System.out.println("Played: " +
                           album.rating() + " " +
                           album.getTitle());
                           album.play();
  } else {
        System.out.println("Skipped: " +
                           album.getTitle());
  }
}
```

n: number of albums

worst case: 4n + 1 remote calls

Current approaches

- Data Transfer Objects and Server Façade
  - Move data in bulk
  - Specialize to particular sequence of client calls
- Document-oriented Web Services
  - Stateless servers
- TCP-based command line interfaces
  - POP, IMAP, FTP, HTTP, etc...
- End result
  - Messy, non-compositional, rigid ... *fast*

Data Transfer Object

```
class TitleAndRatingAndCond implements
  Serializable {
  public String getTitle() { ... }
  public int getRating() { ... }
  public boolean getCond() { ... }
}
```

Remote Facade

```
interface MusicFacade
{
  TitleAndRatingAndCond[]
            playHighRatedAlbums(int minRating);
  ...
}
```

Remote Façade and Data Transfer Objects

```
int minimumRating = 4;
MusicFacade musicService = ... ;
TitleAndRatingAndCond[] results =
  musicService.playHighRatedAlbums(minimumRating);
for (TitleAndRating result : results) {
if (result.getCond()) {
        System.out.println("Played: " +
                result.getRating() + " " +
                result.getTitle());
} else {
        System.out.println("Skipped: " + album.getTitle());
}}
```

What are the problem with DTO and Remote Façade?

- Tight coupling with objects/code
  - Not service-oriented
- Can result in
  - under-approximation
    - Multiple calls to server
      - Client needs to print title of two different albums
  - over approximation
    - Single call to server

Remote Batch Invocation (RBI) - Insight

- We have an *incorrect assumption:*
  - Distribution can be solved in existing languages without any changes
- Goals
  - Fine-grained interfaces
  - Execute many remote operations in bulk
  - Create Facades and Transfer objects automatically

Remote Batch Invocation (RBI)

```
int minimumRating = 4;
Service service = ... ;
batch (Music musicService : service) {
   for (Album album : musicService.getAlbums())
        final Artist = musicService.addArtist("John");
        if (album.rating() > minimumRating) {
                System.out.println("Played: " +
                                album.rating() + " " +
                                album.getTitle());
                                album.play();
        } else {
                System.out.println("Skipped: " + album.getTitle());
        }
}}
```

Generated Façade by Partitioning

```
int minimumRating = 4;
Service service = ... ; Music musicService = ...;

for (Album a : musicService.getAlbums())
   if (a.rating() > minimumRating) {                    Remote
        // Get rating, title
        album.play();
   } else {
   }

for (????) {
   if (???) {
        System.out.println("Played: " +
                rating + " " +                          Local
                title);
   } else {
        System.out.println("Skipped: " + title);
   }
}
```

Generated Code

```
int minimumRating = 4;
Service service = ... ; Music musicService = ...;
List<TitleAndRatingAndCond> results = new...;
for (Album a : musicService.getAlbums())
   if (a.rating() > minimumRating) {                    Remote
        results.add(new TitleAndRatingAndCond(
        a.rating(), album.getTitle(), true));
        album.play();
   } else {
        results.add(new TitleAndRatingAndCond(0, null, false));
   }

for (TitleAndRatingAndCond result : results) {
   if (result.getCond()) {
        System.out.println("Played: " +
                result.getRating() + " " +              Local
                result.getTitle());
   } else {
        System.out.println("Skipped: " +
                result.getTitle());
   }
}
```

Remote Batch Invocation

- Clean server interface, decoupled clients
  - Fine-grained interfaces
  - Automatic bulk data transfer and facades
- Only primitive values can be transferred between clients and server
  - ***No stubs, No proxies!***
- One round-trip per lexical batch block
- Two kinds of exceptions:
  - Remote exceptions
    - Does the paper handle them?
      - Not gracefully
    - Proposed solution: transactional memory on server
      - Does this violate service-oriented (stateless) principle?
  - Network exceptions (reduced!)

What can be executed remotely?

- Sequences and Composition
  - *batch (r) { r.foo(); r.foo().bar().getName(); }*
- Loops and Conditions
  - *batch (music) {*

```
    for (Album a : music.getAlbums() )
    if (a.rating() > 5)
            print( a.getName() + ": " + a.rating() ); }
```

What cannot be executed remotely?

- Constructor calls
- Casts
- While loops
- Assignments
- Exceptions
- Are these Java compiler specific?

- What is the main drawbacks?
  - Think of SQL
  - Aggregate over collections cannot be done remotely