Partition and Reforest for Distributed Objects: Services and Data Access

William Cook
University of Texas at Austin
with
Eli Tilevich, Yang Jiao, Virginia Tech
Ali Ibrahim, Ben Wiedermann, UT Austin
Sun Microsystems, September 2009
Standard Approach to Distribution

- **Step 1: 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 … basically useless

C
C++
ML
Java
etc...
CORBA
DCOM
RMI
Example
Music Jukebox in the Cloud

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

interface Music {
    Album[] getAlbums();
    ... 
}

interface Album {
    String getTitle();
    void play();
    int rating();
    ... 
}
Remote Procedure Calls (RPC)

```java
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

Client

Invoker

AlbumProxy

MusicProxy

Server

MusicImpl

AlbumImpl

The University of Texas at Austin
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());
  }
```

\[ n: \text{number of albums} \]
\[ \text{worst case: } 4n + 1 \text{ remote calls} \]
What do people really do? (many many many experiments)

- Data Transfer Objects and Server Facade
  - 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
class TitleAndRatingAndCond implements Serializable {

    public String getTitle() { ... }  
    public int getRating() { ... }    
    public boolean getCond() { ... }  

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

Remote Facade 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());
    }
}
```

“In many ways, a Data Transfer Object is one of those objects our mothers told us never to write.” - Martin Fowler
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)

```java
int minimumRating = 4;
Service service = ... ;

batch (Music musicService : service) {
    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());
        }
}
```
```java
int minimumRating = 4;
Service service = ...; Music musicService = ...;
for (Album a : musicService.getAlbums())
    if (a.rating() > minimumRating) {
        // GET rating, title
        album.play();
    } else {
    }
for (?????) {
    if (???) {
        System.out.println("Played: " +
            rating + " " +
            title);
    } else {
        System.out.println("Skipped: " + title);
    }
} 
```
int minimumRating = 4;
Service service = ...; Music musicService = ...;

List<TitleAndRatingAndCond> results = new...;
for (Album a : musicService.getAlbums())
    if (a.rating() > minimumRating) {
        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() + " " + 
            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 proxies!*
- One round-trip per lexical batch block
- Two kinds of exceptions:
  - Remote exceptions (see paper)
  - Network exceptions (reduced!)
What can be executed remotely?

• Sequences and Composition
  – \textit{batch} \((r)\) \{ \texttt{r.foo(); r.foo().bar().getName(); }\}

• Loops and Conditions
  – \textit{batch} \((\texttt{music})\) \{ \texttt{Asynchrony does not help!} \}
    \begin{verbatim}
    for (Album a : music.getAlbums())
      if (a.rating() > 5)
        print( a.getName() + ":: " + a.rating() )
    \end{verbatim}

• Functional glue language, not specific to Java
  – truly cross-platform (like web services)
  – no assumption of internal serialization format
Reforestation

Introduce intermediate data structures
Deforestation [Wadler 89]

- Remove intermediate data structures (trees)
  \[
  \text{sum} \ (\text{square} \ (1 \ \text{to} \ 5))
  \]

- Deforested version
  \[
  \text{sum-square-interval}(1, 5)
  \]

The University of Texas at Austin
Identifying Remote Expressions

- Expressions: static local, non-static local, or remote.
- Remote expressions operate on data reachable from batch root.
- Intra-procedural flow sensitive data flow analysis.
Grouping Remote Operations

• Restrictions
  – Remote expressions should not depend on non-static local expressions.

• Local Compiler analysis
  – We want the programmer to be able to understand and predict the results.
    – Simple
    – Not sound! But okay...
Examples

Compiler Produces Error

```java
batch (Music musicService : service) {
    Album a = musicService.getAlbum(1);
    final int rating = a.rating();
    // non-static local
    boolean goodEnough = System.prompt(
            "Is this rating good enough? " + rating);
    if (goodEnough) {
        a.play(); // Can't batch with call to rating
    }
```
Examples

*Compiler does not catch possible error*

```java
private String newTitle = "foo"; // static local
void changeTitle() { newTitle = "bar"; }

batch (Music musicService : service) {
    Album a = musicService.getAlbum(1);
    changeTitle();
    a.setTitle(@newTitle);
}
```
Examples

*Compiler does not catch possible error*

private String newTitle = “foo”; // static local

void changeTitle() { newTitle = “bar”; }

batch (Music musicService : service) {
    Album a = musicService.getAlbum(1);
    a.setTitle(@newTitle);
    changeTitle(); // motion changed semantics
}

The University of Texas at Austin
Memory Model

- Only transfer primitive values
- No proxies (remote pointers)
  - Server is stateless, "service oriented"
  - No distributed garbage collection
- Serialization through public interfaces
  ```java
  batch (remote) {
      RemoteSet r = remote.makeSet();
      for (int elem : localSet().items() )
          r.add( elem );
  }
  ```

- Illegal: RemoteSet r = localSet;
- Need reusable helper functions/coercions
Execution Model

- **Client**
  - Language support for batches
  - Also library

- **Server**
  - Small engine to execute scripts
    - Can only call public methods
    - No constructors, static methods
    - Just as safe as current approach
    - Similar to existing server engines

- **Not completely transparent**
  - Programmer controls batching
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>RMI CORBA</th>
<th>Web Services</th>
<th>Remote Batch Invocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Interfaces</td>
<td>Good</td>
<td>Bad</td>
<td>Good</td>
</tr>
<tr>
<td>Latency</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Memory model</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Stateless</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Partial Failure</td>
<td>Bad</td>
<td>Better</td>
<td>Better</td>
</tr>
<tr>
<td>Programming Model</td>
<td>Good</td>
<td>Bad</td>
<td>Good... but...</td>
</tr>
</tbody>
</table>
Generalized Batches

- Parameterize by batch handler
  - $\sqrt{\text{batch}}$ RMI (remoteObject) { … }
  - $\sqrt{\text{batch}}$ WebService (service) { … }
  - $\sqrt{\text{batch}}$ SQL (db) { … }
  - $\text{batch}$ GPU (gpu) { …. }
  - $\text{batch}$ PartialEval (s) { … }
  - $\text{batch}$ $H(r)B = B_2(H(B_1, in))$

- Batch provides generalized program partitioning and reforestation capability
Web Services: Document = Batch Amazon Web Service

```xml
<ItemLookup>
  <AWSAccessKeyId>XYZ</AWSAccessKeyId>
  <Request>
    <ItemIds>
      <ItemId>1</ItemId>
      <ItemId>2</ItemId>
    </ItemIds>
    <IdType>ASIN</IdType>
    <ResponseGroup>SalesRank</ResponseGroup>
    <ResponseGroup>Images</ResponseGroup>
  </Request>
</ItemLookup>
```

```java
interface Amazon {
  void login(String awsKey);
  Item getItem(String ASIN);
  ...}

interface Item {
  int getSalesRank();
  Image getSmallImage();
  ...}
```

// calls specified in document
aws.login("XYZ");
Item a = aws.getItem("1");
Item b = aws.getItem("2");
return new Object[] {
  a.getSalesRank(), a.getSmallImage(),
  b.getSalesRank(), b.getSmallImage()   }
```

The University of Texas at Austin
Batching Database Access

```java
batch SQL (Database db : dbService) {
    for (Album album : db.getAlbums())
        if (album.rating() > 50)
            System.out.println("Played: " + album.getTitle());
}

DbResults data = dbService.executeQuery(
    "select title from albums where rating > 4");
for (item : data.items())
    System.out.println("Played: " + item.getTitle());

• Also updates, aggregation and grouping
```
Open Issues

• How hard is to add “batch statement” to your favorite programming language?
  – try it!

• What about multiple servers in batch?
  – Client → Server*
  – Client → Server → Server
  – Client ↔ Server

• Monadic interpretation??

• MPI
Related work

- Microsoft LINQ
  - Batches are more general than LINQ

- Mobile code / Remote evaluation
  - Does not manage returning values to client

- Implicit batching
  - Performance model is not transparent

- Asynchronous remote invocations
  - Asynchrony is orthogonal to batching

- Automatic program partitioning
  - binding time analysis, program slicing

- Transactions (batch/atomic)

- Cloud database system that sends javascript
“Whatever the database programming model, it must allow complex, data-intensive operations to be picked out of programs for execution by the storage manager, rather than forcing a record-at-a-time interface.”
Contributions

- New statement form:  
  \textit{batch} H (r) \{ body \}

- Interesting semantics, general applications
  - Partition
  - Reforest

- Unifies distribution and data access
  - Can be asynchronous too