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

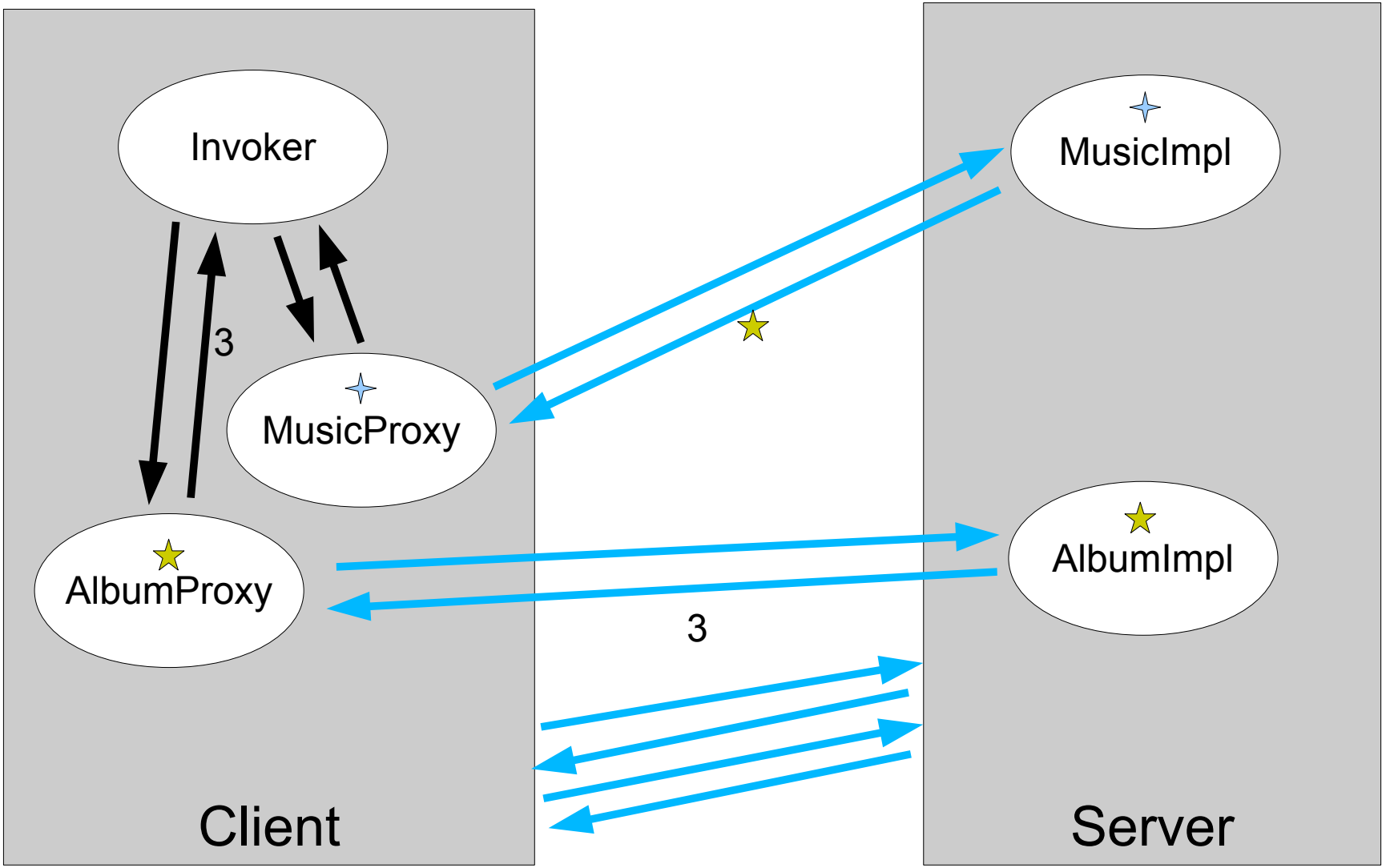
```
interface Music {
  Album[] getAlbums();
  ...
}

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

Remote Procedure Calls (RPC)

```
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



Too Many RPC calls

n: number of albums
worst case: 4n + 1 remote 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());
}}
```

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*

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 Facade and Data Transfer Objects

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

```
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());
}}
```

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())
    if (album.rating() > minimumRating) {
      System.out.println("Played: " +
                         album.rating() + " " +
                         album.getTitle());
      album.play();
    } else {
      System.out.println("Skipped: " + album.getTitle());
}}}
```

Generated Facade by Partitioning

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

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

Remote

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

Local

Generated Code

```
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

Local

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
  - ***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()); }

Asynchrony does not help!

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

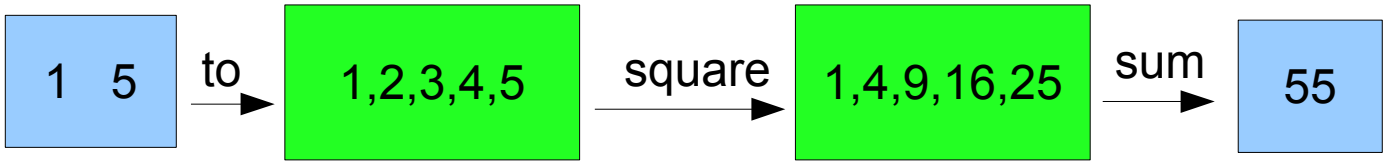
Reforestation

Introduce intermediate data structures

Deforestation [Wadler 89]

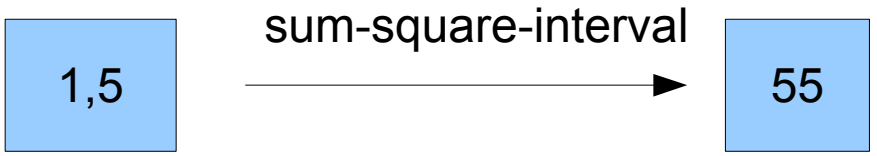
- Remove intermediate data structures (trees)

  sum (square (1 `to` 5))



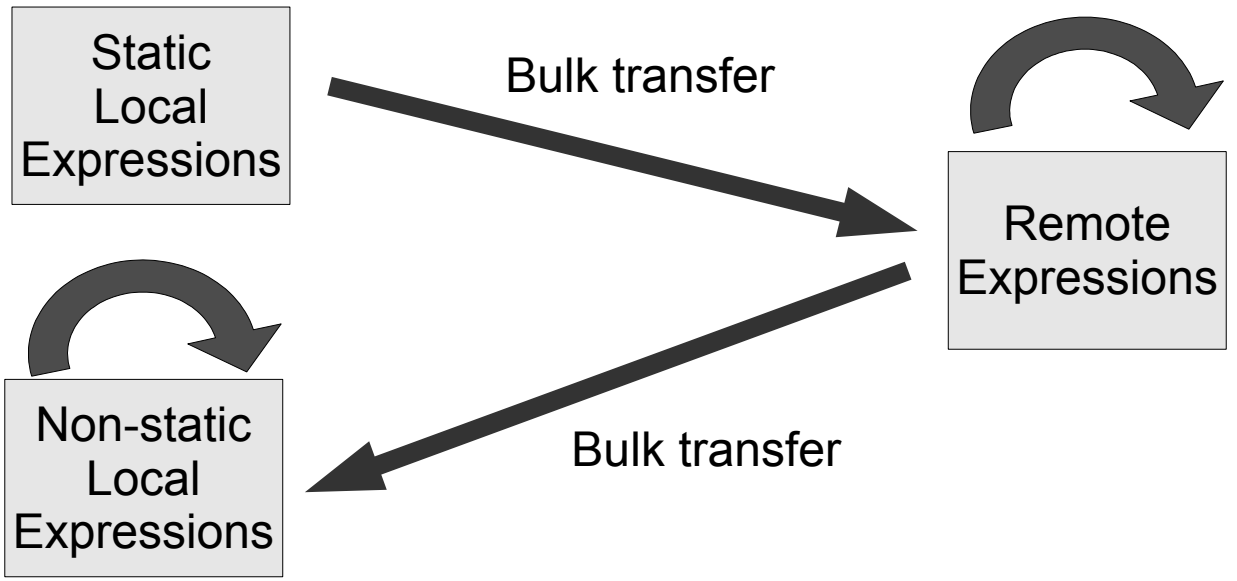
- Deforested version

  sum-square-interval(1, 5)



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

```
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

```
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
}}
```

Memory Model

- Only transfer primitive values
- No proxies (remote pointers)
  - Server is stateless, “service oriented”
  - No distributed garbage collection
- Serialization through public interfaces

```
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

	RMI CORBA	Web Services	Remote Batch Invocation
Clean Interfaces	Good	Bad	Good
Latency	Bad	Good	Good
Memory model	Bad	Good	Good
Stateless	No	Yes	Yes
Partial Failure	Bad	Better	Better
Programming Model	Good	Bad	Good... but...

Generalized Batches

- Parameterize by batch handler
  - √ ***batch*** RMI (remoteObject) { … }
  - √ ***batch*** WebService (service) { … }
  - √ ***batch*** SQL (db) { … }
  - ***batch*** GPU (gpu) { .... }
  - ***batch*** PartialEval (s) { … }
  - ***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

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

```
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()    }
```

Batching Database Access

```
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 favorate 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

David Maier 1987

"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:
  batch H (r) { body }
- Interesting semantics, general applications
  - Partition
  - Reforest
- Unifies distribution and data access
  - Can be asynchronous too