



Remote Batch Invocation--SOA with Distributed Objects



Yang Jiao, Marc Fisher II, and Eli Tilevich @ Virginia Tech

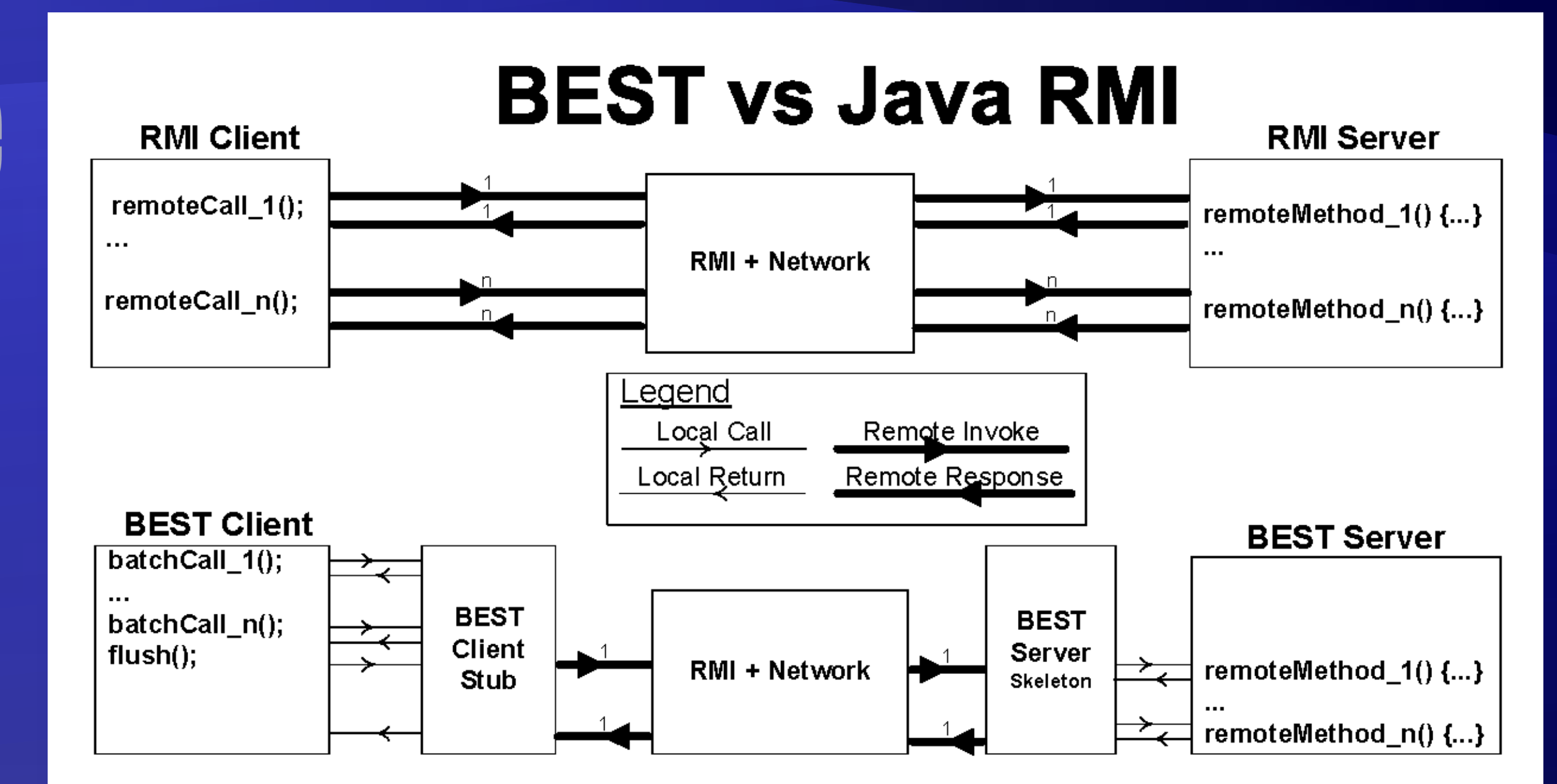
In collaboration with Ali Ibrahim, William R. Cook @ UT Austin

Introduction

- Most computing nowadays is distributed in nature
 - The Internet, scientific computation, mobile devices, etc.
- Programming distributed systems is difficult and labor-intensive
 - High latency, partial-failure, differences in memory access, etc.
 - Latency lags bandwidth
- **Goal:** create intuitive programming abstractions for programming efficient distributed systems using objects
 - Ensure good performance by effectively utilizing modern networks

BEST Runtime

- A library built on top of Java RMI [3]
- Records client calls, sending them to server in bulk.
- Returns all the results in bulk.

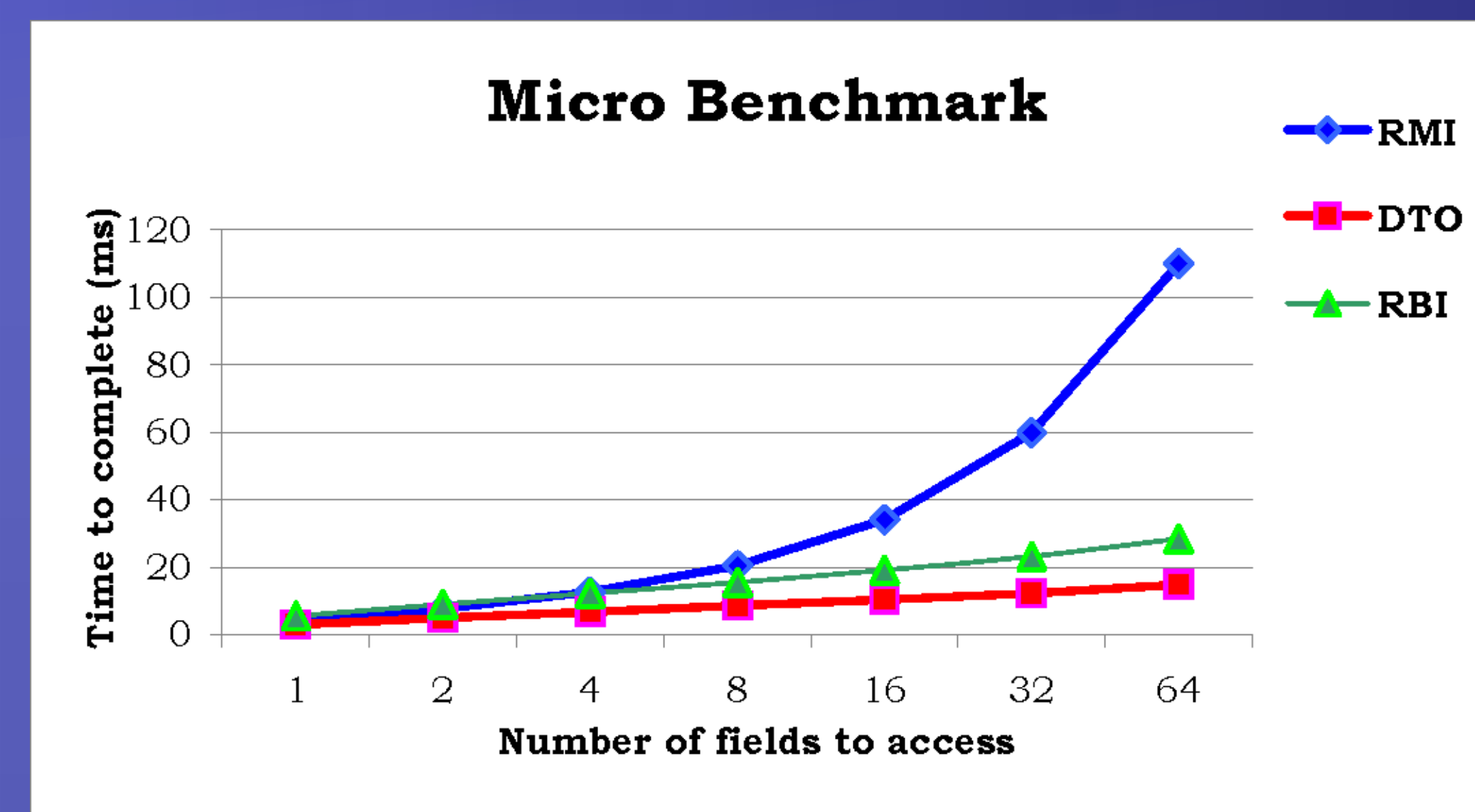


Challenges

- Latency lags bandwidth [1]
- Remote Procedure Calls do not compose efficiently,
 - Data Transfer and Remote Facade patterns hardcode large-granularity interfaces for particular client use cases.

Approach

- **Remote Batch Invocation** [2]
 - a language abstraction
 - optimize remote communication
 - leverage higher bandwidth, especially in high-latency environments.
- **Batch Execution Service and Translation (BEST)** [2, 3]
 - efficient middleware library for RBI

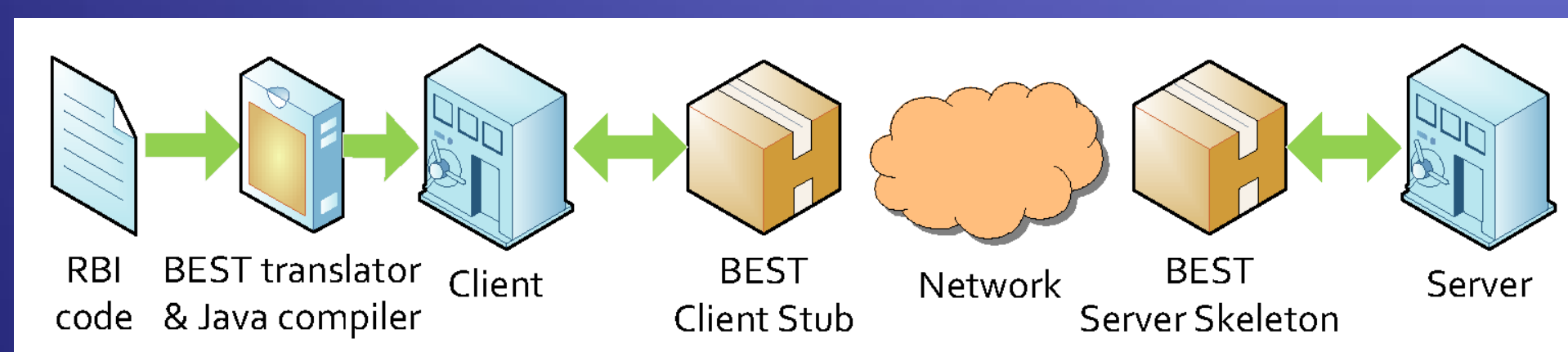


Benchmark Result

- Emulates a common usage scenario: the client retrieves the object from the server and updates its fields.
 - The RMI version is the slowest
 - The DTO and RBI versions exhibit comparable performance

Example

How RBI works



Branching

```

1 batch(Music music : musicService) {
2   final Album album = registry.getAlbum("Best ever");
3   if (album.isPlatinum()) {
4     album.play();
5     System.out.println("Album is platinum: " + album.getTitle());
6   } else {
7     System.out.println("Album is not platinum: " + album.getArtist().getName());
8   }
9 }

```

Loops

```

1 batch (Music music : musicService) {
2   final Album album = music.getAlbum("1");
3   System.out.println("Tracks: ");
4   for (Track t : album.getTracks()) {
5     ...
6     for (Artist a : t.getPerformers()) {
7       System.out.println(a.getName());
8     }
9   }
10 }

```

Exceptions

```

1 try {
2   batch (Music favoriteMusic : musicService) {
3     ...
4     try {
5       album.delete();
6     } catch (PermissionError pe) {
7       System.out.println("No permission to play album" + album.getTitle());
8     }
9   }
10 } catch (RemoteException re) {
11   System.out.println("Error communicating batch.");
12 }

```

RBI Code Snippet

```

1 Service musicService =
2   new Service("music-service", Music.class);
3 batch(Music music : musicService) {
4   final Album album = music.getAlbum("1");
5   if (album.getTitle().startsWith("A")) {
6     System.out.println("Tracks: ");
7     for (Track t : album.getTracks()) {
8       System.out.print(' ');
9       System.out.print(t.getTitle());
10    }
11  } else {
12    System.out.println("Album is not platinum: "
13      + album.getArtist().getName());
14  }
15 }

```

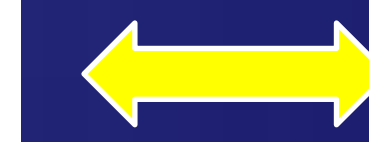
Future Work

- Employ static and dynamic program analyses to
 - Extract possible concurrency from recorded batch
 - Identify remote objects' fields needed for local use
- Consider collections of operations as 'documents'
 - Understand 'documents' as batches of operations

```

1 <ItemLookup>
2   <AWSAccessKeyId>XYZ</AWSAccessKeyId>
3   <Request>
4     <ItemIds>
5       <ItemId>1</ItemId>
6       <ItemId>2</ItemId>
7     </ItemIds>
8     <IdType>ASIN</IdType>
9     <ResponseGroup>SalesRank</ResponseGroup>
10    <ResponseGroup>Images</ResponseGroup>
11  </Request>
12 </ItemLookup>

```



```

1 interface Amazon {
2   void login(String awsAccessKey);
3   Item getItem(String ASIN);
4   ...
5 }
6 interface Item {
7   int getSalesRank();
8   Image getSmallImage();
9   ...
10 }
11 // calls specified in document
12 ...
13
14 aws.login("XYZ");
15 Item a = aws.getItem("1");
16 Item b = aws.getItem("2");
17 return new Object[] {
18   a.getSalesRank(), a.getSmallImage(),
19   b.getSalesRank(), b.getSmallImage()
20 }
21 ...

```

[1] D. A. Patterson. Latency lags bandwidth. *Commun. ACM*, 47(10):71–75, 2004

[2] Ali Ibrahim, Yang Jiao, Eli Tilevich, and William R. Cook. Remote Batch Invocation for Compositional Object Services. *The 23rd European Conference on Object-Oriented Programming (ECOOP 2009)*.

[3] Eli Tilevich, William R. Cook, Yang Jiao. Explicit Batching for Distributed Objects. *The 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009)*.

References