Unifying Remote Data, Remote Procedures, and Service Clients

William R. Cook
University of Texas at Austin
with
Eli Tilevich, Yang Jiao, Virginia Tech
Ali Ibrahim, Ben Wiedermann, UT Austin

Using a Mail Service

```
int limit = 500;
Mailer mailer = ...connect to mail server...;
for ( Message m : mailer.Messages )
    if ( m.Size > limit ) {
        print( m.Subject & m.Sender.Name);
        m.delete();
    }
```

Using a Mail Service

```
int limit = 500;
Mailer mailer = ...connect to mail server...;
for ( Message m : mailer.Messages )
    if ( m.Size > limit ) {
        print( m.Subject & m.Sender.Name);
        m.delete();
    }
```

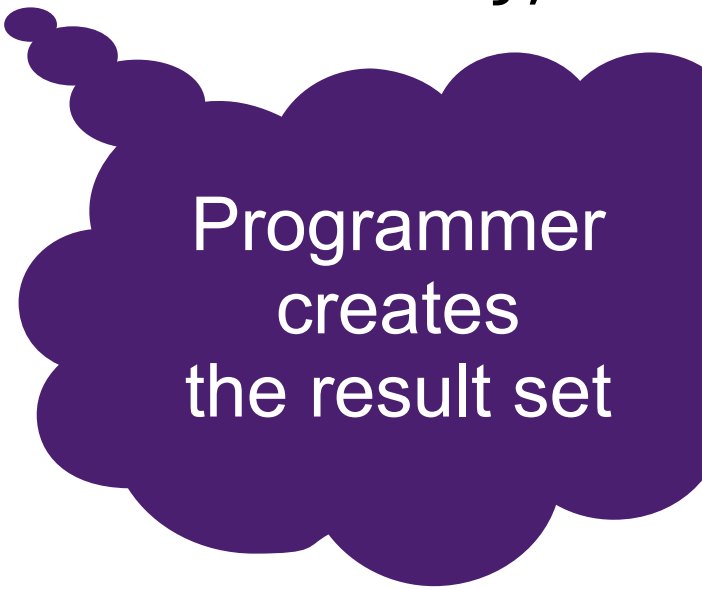
Works great if mailer is local object,
but is terrible if mail server is remote

Call Level Interface (e.g. JDBC)

```
// create a remote query/action
String q1 = "select Subject, Name
            from Messages join Person on ...
            where Size > " + limit;
String q2 = "delete from Messages
             where Size > " + limit;
// execute on server
ResultSet rs = conn.executeQuery(q1);
conn.executeCommand(q2);
// use the results
while ( rs.next() )
      print( rs.getString("Subject") & " Deleted" );
```

LINQ

```
// create the remote script/query
var results = from m in Messages
              where m.Size > limit
              select { m.Subject, m.Sender.Name };
// execute and use the results
for (var rs in results )
    print( rs.Subject & rs.Name );
// delete the items
var big     = from m in Messages
              where m.Size > limit
              select m;
db.Messages.Remove(big);
db.SubmitChanges();
// interesting issues about order of operations...
```



Why can't we just say this?

```
int limit = 500;
Mailer mailer = ...connect to mail server...;
for ( Message m : mailer.Messages )
    if ( m.Size > limit ) {
        print( m.Subject & m.Sender.Name);
        m.delete();
    }
```

Why can't we just say this?

```
int limit = 500;
Mailer mailer = ...connect to mail server...;
for ( Message m : mailer.Messages )
    if ( m.Size > limit ) {
        print( m.Subject & m.Sender.Name);
        m.delete();
    }
```

Original definition of "impedance mismatch":

"Whatever the database programming model, it must allow complex, data-intensive operations to be picked out of programs for execution by the storage manager..."

David Maier, DBLP 1987

It's not impedance of **data formats**,
it's impedance in **processing models!!**

Object-Relational Impedance Mismatch


Object-
Relational
Impedance
Mismatch

PL/DB Impedance Mismatch

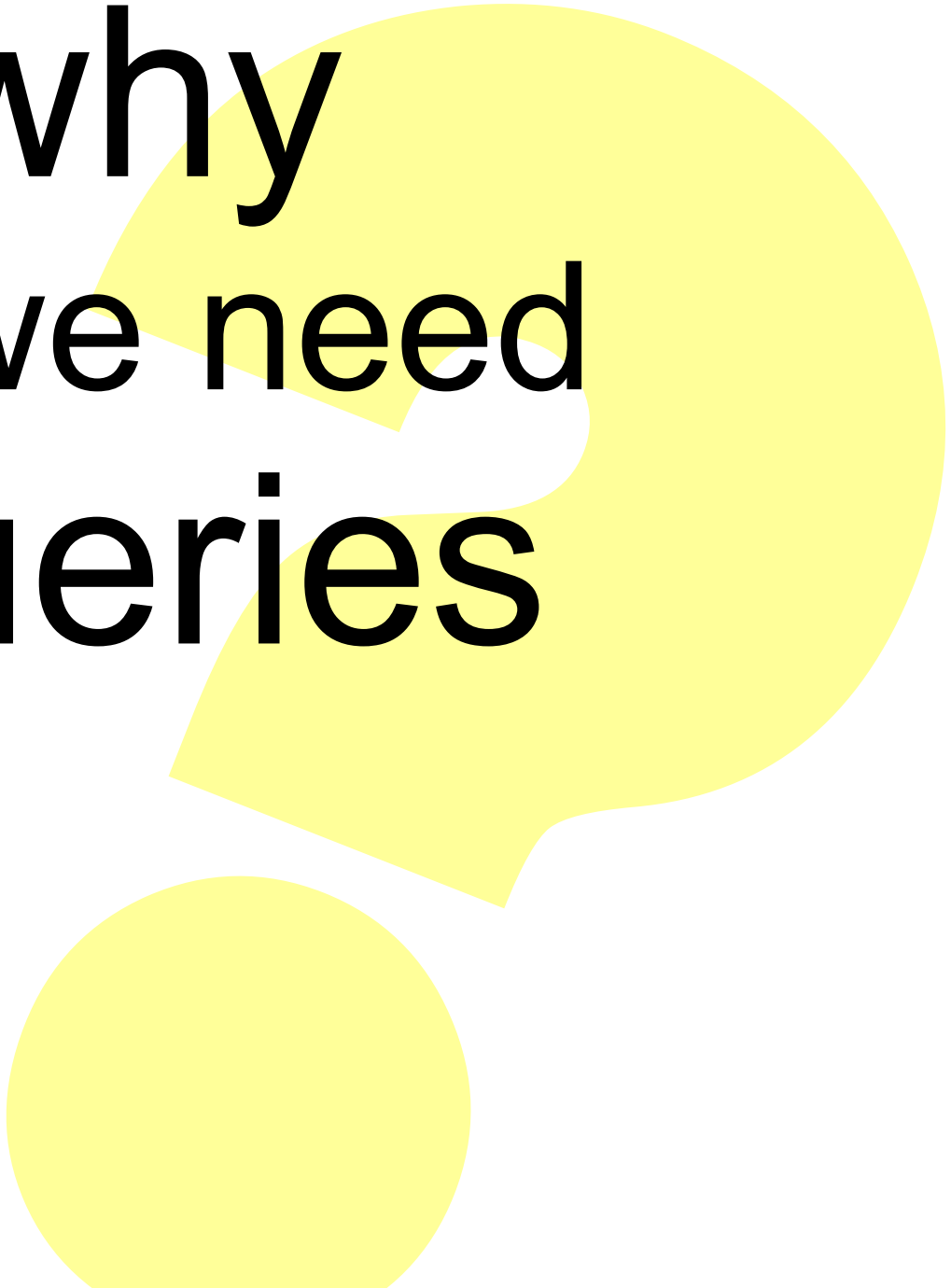
Programming Language to Database

record-at-a-time

versus

bulk

why do we need queries



Queries are ***efficient***

Orders of magnitude more efficient

(This applies to SQL as well as MapReduce)


ACID
Transactions
do **not** require
queries


"Queries"
can be
actions

Are queries just an optimization technique?

Do programmers

want

to write queries?

If you are a

database person,

the answer is usually

“Yes”

“They should write queries...
its good for them, like vegetables...”

But the **truth** is...

programmers
do not want to
write queries

Queries are a

means to an end

not

an end in themselves

let's try again...

Language Design: Remote Calls

Starting point

```
print( r.getName() );
print( r.getSize() );
```

Notes:
print is local
r is remote

Goals

Fast: one round trip

Stateless communication

do not *require* persistent connection

Platform independent

no serialization of complex user-defined objects

Clean programming model

A Novel Solution: Batches

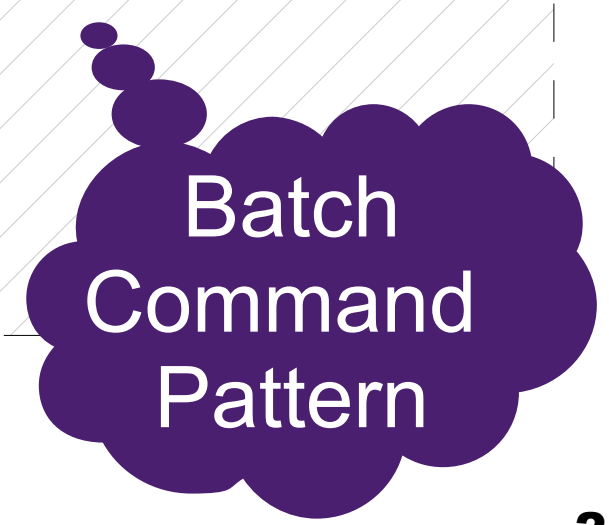
```
batch ( Item r : service ) {
    print( r.getName() );
    print( r.getSize() );
}
```

Execution model: Batch Command Pattern

1. Client **sends *script*** to the server
   (Creates Remote Façade on the fly)
2. Server **executes** two calls
3. Server **returns results in bulk** (name, size)
   (Creates Data Transfer Objects on the fly)
4. Client **runs the local code** (print statements)

Compiled Client Code (generated)

```
// create remote script
script = <[
   out_A( *.getName() )
   out_B( *.getSize() )
]>;
// execute on the server
Forest x = service.execute( script );

// Client uses the results
print( x.get("A") );
print( x.getInt("B") );
```



A larger example

```
int limit = 500;
Service<Mailer> serviceConnection =...;
batch ( Mailer mail : serviceConnection )
    for ( Message msg : mail.Messages )
        if ( msg.Size > limit ) {
            print( msg.Subject & " Deleted" );
            msg.delete();
        }
```

Remote part as Batch Script

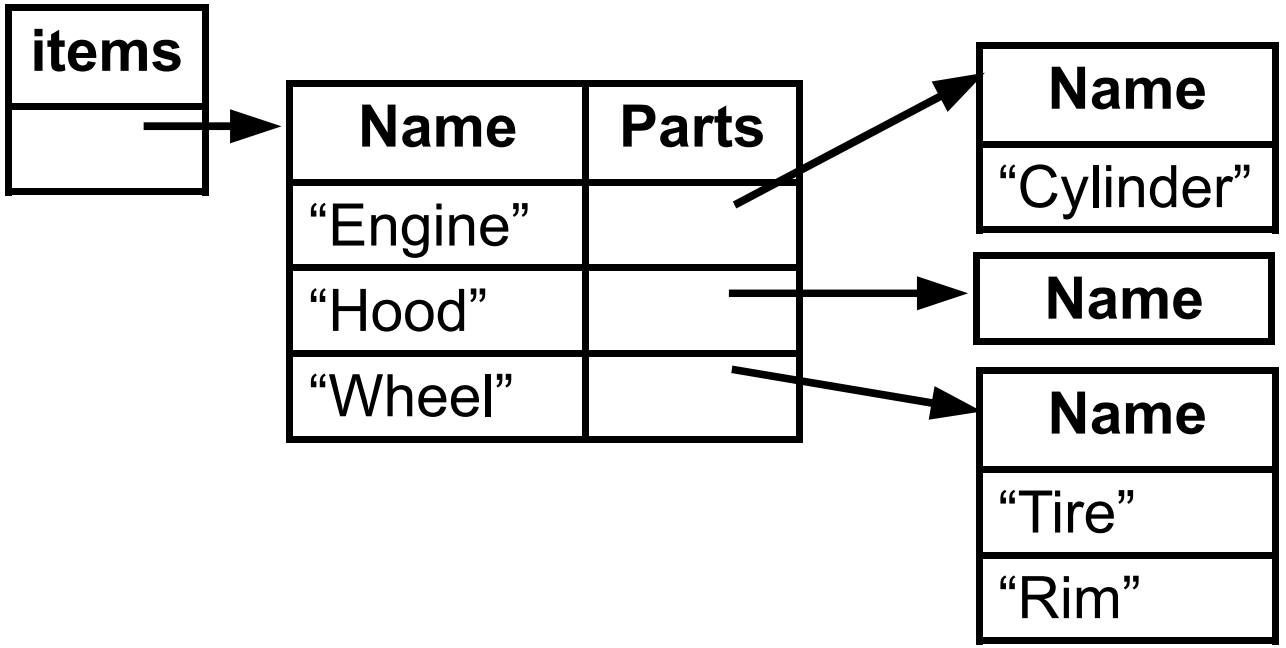
```
script = <[
   for ( msg : *.Messages ) {
      out_A( msg.Subject );
      if ( out_B( msg.Size > 500 ) ) {
         msg.delete();
      } else {
         out_C( msg.Date );
      }
   ]>
```

Compiled code (auto generated)

```
Service<Mailer> serviceConnection =...;

Forest result =
    serviceConnection.execute(script);

for ( r : result.getIteration("msg") )
  if ( r.getBoolean("B") )
    print( r.get("A") & " Deleted" );
  else
    print( r.get("A") & ":" & r.get("C") );
```

Forest Structure == Control flow

```
for (x : r.Items) {
  print( x.Name );
  for (y : x.Parts)
    print( y.Name );
}
```



Batch = One Round Trip

Clean, simple performance model

Some batches would require more round trips

```
batch (..) {
    if (AskUser("Delete " + msg.Subject + "?")
        msg.delete();
}
```

Pattern of execution

OK: Local → Remote → Local

Error: Remote → Local → Remote

Can't just mark everything as a batch!

What about Object Serialization?

Batch only transfers primitive values, not objects

But they work with any object, not just *remotable* ones

Send a local set to the server?

```
java.util.Set<String> local = … ;
batch ( mail : server ) {
    service.Set recipients = local; // compiler error
    mail.sendMessage( recipients, subject, body);
}
```

Serialization by Public Interfaces

```
java.util.Set<String> local = … ;
batch ( mail : server ) {
    service.Set recipients = mail.makeSet();
    for (String addr : local )
        recipients.add( addr );
    mail.sendMessage( recipients, subject, body);
}
```

Sends list of addresses with the batch

Constructors set on server and populates it

Works between different languages

Serialization can be encapsulated in a procedure

Batch Summary

Client

Batch statement: compiles to Local/Remote/Local

Works in any language (e.g. Java, Python, JavaScript)

- Cross-language and cross-platform

Server

Small engine to execute scripts

- Call only public methods/fields (safe as RPC)
- Stateless, no remote pointers (aka proxies)

Communication

Forests (trees) of primitive values (no serialization)

- Efficient and portable

Batch Script Language

e ::= x	c	variables, constants
|	**if** e **then** e **else** e	conditionals
|	**for** x : e **do** e	loops
|	**let** x = e **in** e	binding
|	x = e | e.x = e	assignment
|	e.x	fields
|	e.m(e, ..., e)	method call
|	e … e	primitive operators
|	$\mathbf{out}_X$ e	parameters and results
|	**fun**(x) e	functions

= + – * / % ; < <= == => > & | not

Agree on *script format*, not on *object representation*

Batches ==> SQL

```
batch ( Service<FileSystem> directory : service ) {
    for ( File f : directory.Files )
        if ( f.Size > 90 ) {
            print( f.Name );
            print( f.Size );
        }}
```

Batch Script:

```
for (f : *.Files)
    if (f.Size > 90) { out_A(f.Name); out_B(f.Size) }
```

SQL:

```
SELECT f.Name, f.Size
FROM Files
WHERE f.Size > 90
```

Data Schema = Server Interface

```
public class Northwind {
  Set<Customer> Customers;
  Set<Order> Orders;
  void insertCustomer(Customer c);
  void insertOrder(Order o);
}
```

```
@Table(name="Orders")
public abstract class Order {
  @Id public int OrderID;
  public Date OrderDate;
  public Date RequiredDate;
  public Date ShippedDate;

  @Column(name="CustomerID")
  public Customer Customer;

  delete();
}
```

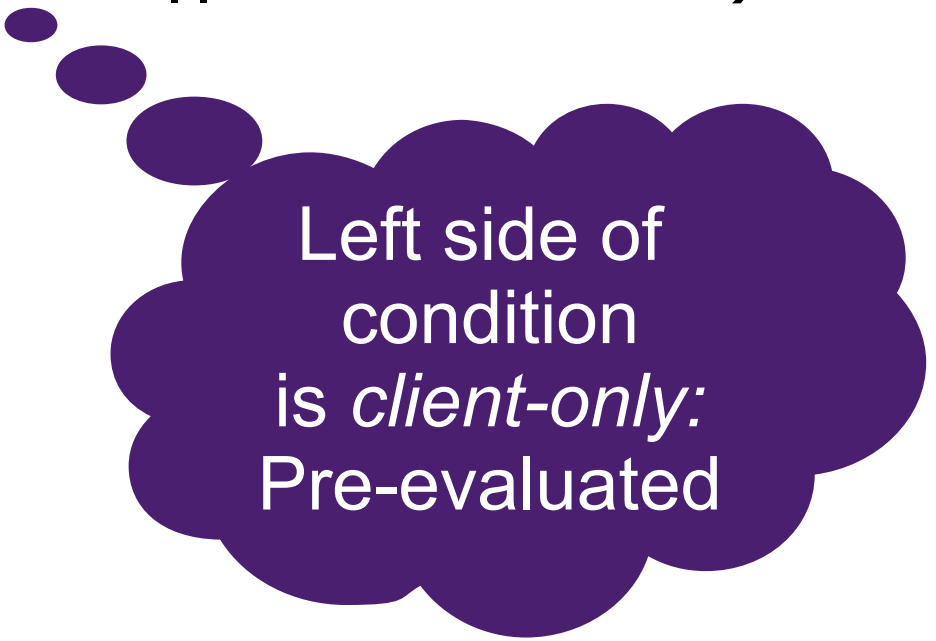
```
@Table(name="Customers")
public class Customer {
  @Id String CustomerID;
  String CompanyName;
  String ContactName;
  String Country;

  @Inverse("Customer")
  Set<Order> Orders;

  delete();
}
```

Dynamic Queries in LINQ

```
var matches = db.Course;
// add a test if the condition is given
if (Test.Length > 0)
      matches = matches.Where(
         c => c.Title == Test);
// select the desired values
matches = matches.Select(c => c.Title);
// iterate over the result set
for (String title : matches.ToList())
     print(title);
```

Dynamic Queries in Batches

```
batch (db : Database) {
  for (Ticket t : db.Course)
    if (Test.Length == 0 || c.Title == Test)
      print(c.Title);
}
```



Batches for SQL

Batch compiler creates SQL automatically

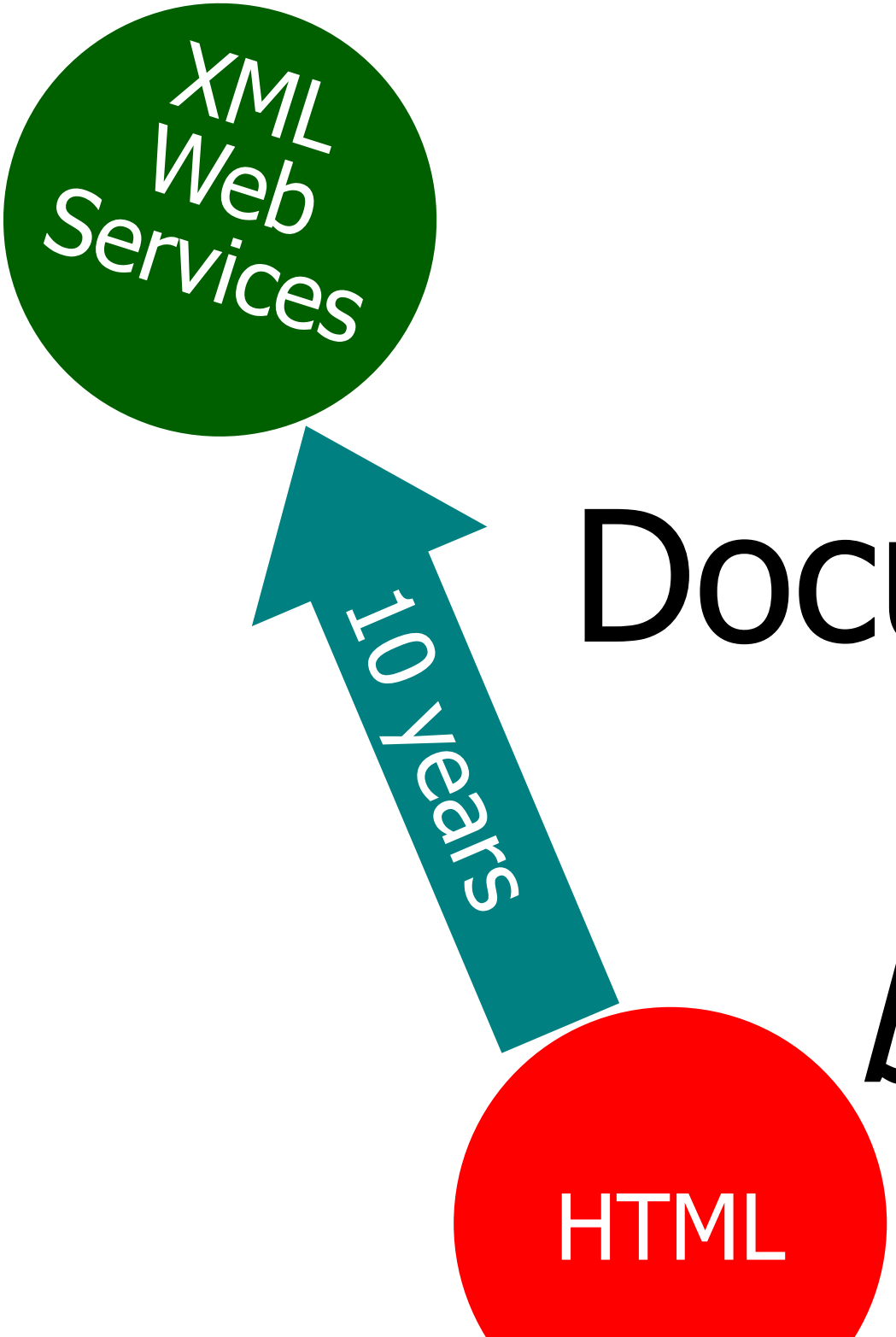
- Efficient handling of nested of loops
- Always a *constant* number of queries for a batch
  - No matter how many (nested) loops are used

Supports all aspects of SQL

- Queries, updates, sorting, grouping, aggregations

Summary

- Clean fine-grained object-oriented programming model
- Efficient SQL batch execution



Web Service Documents *are batches*

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>
```

Available Now...

Jaba: Batch Java

100% compatible with Java 1.5

Transport: XML, JSON, easy to add more

Batch statement as "for"

for (RootInterface r : serviceConenction) { ... }

Full SQL generation and ORM

Select/Insert/Delete/Update, aggregate, group, sorting

Future work

Security models, JavaScript/Python clients

Edit and debug in Eclipse or other tools

Available now!

Opportunities

Add batch statement to your favorite language

Easy with reusable partitioning library

Scala, C#, Python, JavaScript, COBOL, Ruby, etc...
Monads?

Optimization by partial evaluation

What about multiple servers in batch?

Client → Server*

Client ↔ Server

Client → Server → Server

Generalize "remoteness": MPI, GPU, ...

Concurrency, Asynchrony and Streaming

Related work

Microsoft LINQ

Batches are different and 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

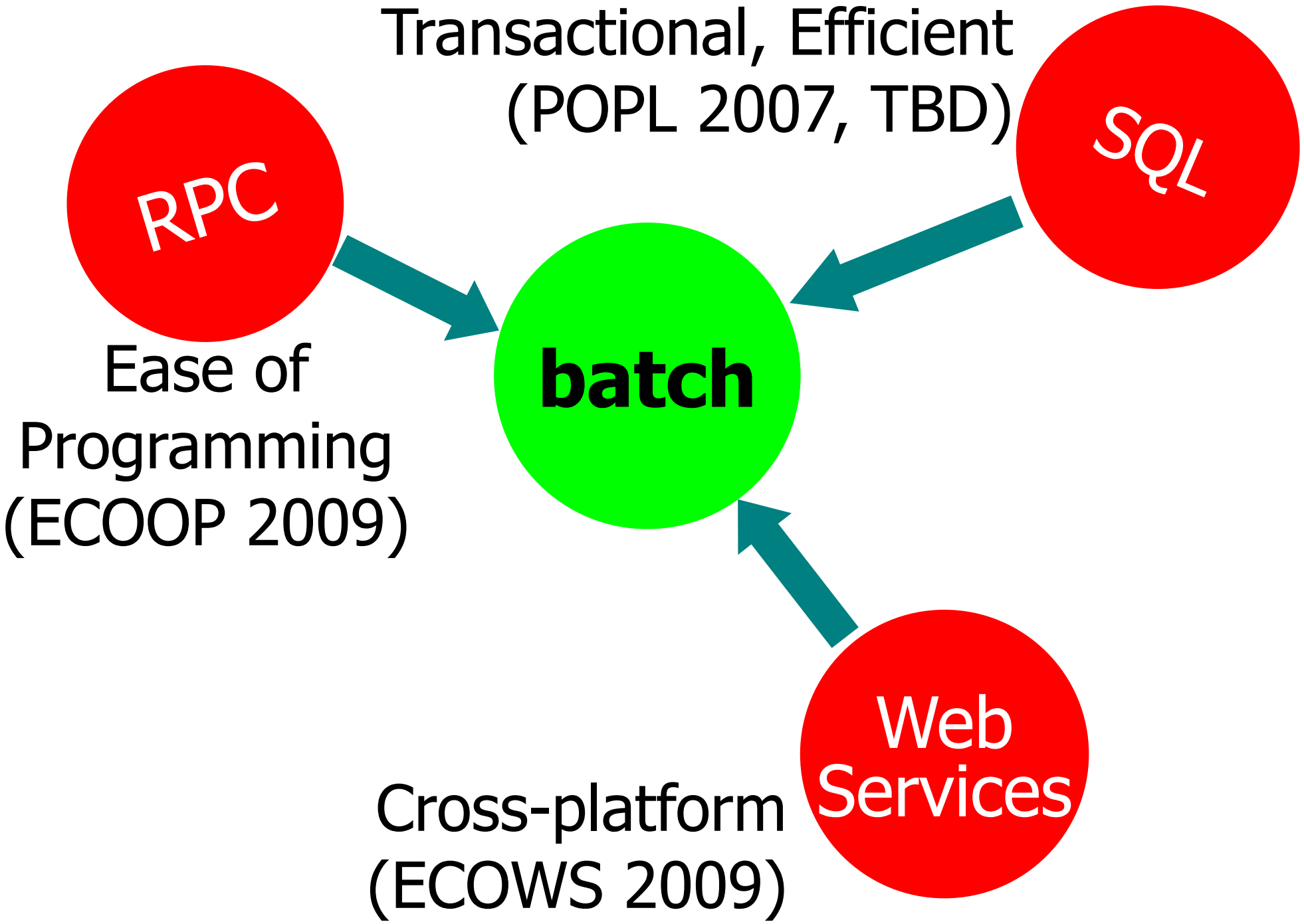
Asynchrony is orthogonal to batching

Automatic program partitioning

binding time analysis, program slicing

Deforestation

Introduce inverse: ***reforestation*** for bulk data transfer


Transactional, Efficient
(POPL 2007, TBD)
RPC
SQL
batch
Ease of
Programming
(ECOOP 2009)
Web
Services
Cross-platform
(ECOWS 2009)

Conclusion

Batch Statement

General mechanism for partitioned computation

Unifies

Distributed objects (RPC)

Relational (SQL) database access

Service invocation (Web services)

Benefits:

Efficient distributed execution

Clean programming model

No explicit queries, stateless, no proxies

Language/transport neutral

Requires adding batch statement to language!