# Web Services versus Distributed Objects

William R. Cook, Janel Barfield
University of Texas at Austin

## How many times have you heard…

## “Web Services suck…” ?

## “WS are a bad version of distributed objects”

(CORBA, DCOM, RMI, etc.)

## Criteria

Performance

Ease of Use

## Past Studies

Distributed Objects
10 to 100
times faster than
Web Services

## Test Case

Call a remote service that
returns an integer
(or an array, etc)

Demare 2005, Elfwing 2002, Juric 2004

## Biased

Typical of RPC…

typical of
Web Services?

Web services
support
“document-oriented
messaging”

## My Study

Also biased

…in the
other direction

HTTP designed
with distributed
objects

## RMI Document Server

```
interface Item {
  File   Get();
  File   Invoke(HashMap p);
  Item  Sub(String n)
    throws FileNotFound;
}
```

```
interface File {
  int       Length();
  String Text();
  String Type();
  String Encoding();
  long    Modified();
}
```

Abstraction
is great

Distance
doesn’t
matter

unless…

property being abstracted
=
essence of problem

-Steve Cook

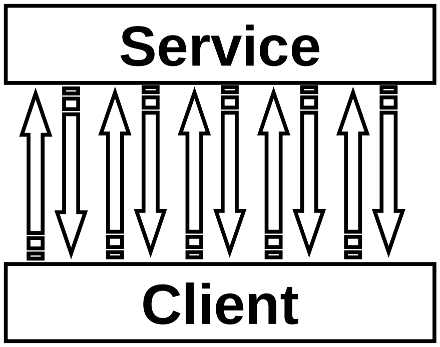
distance
=
essence of problem

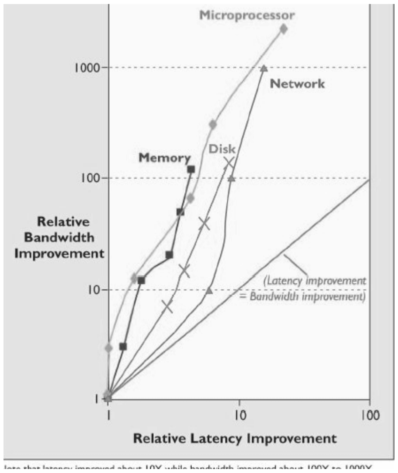
## Calling RMI

```
Container c;
c = root.Sub(“papers").Sub("index.htm");
String s = c.Get().Text();
```

“.” = Round-trip

Can’t take something that works locally and make it remote. But taking something that works remotely and using it locally is ok. – **Don Box**





Things are only going to get worse

Latency Lags Bandwidth

David Patterson
CACM
October 2004

The Web would have failed

“but nobody would do it that way…

…you need:
value objects,
custom marshalling / serialization
custom client stubs,
blah…blah…blah…
blah…blah…blah…
blah…blah…blah… blah…blah…blah…
blah…blah…blah… blah…blah…blah…
blah…blah…blah… blah…blah…blah…
blah…blah…blah… blah…blah…blah… blah…blah…blah…
blah…blah…blah… blah…blah…blah… blah…blah…blah… blah…blah…blah…

# Result…

- Defeat abstraction
- Tightly couple client and server
- Complex

# Web Services

# Can't easily convert RMI interface into a web service

```
interface Item {
  File   Get();
  File   Invoke(HashMap p);
  Item  Sub(String n)
    throws FileNotFound;
}
```

# Actions vs. Docs

Objects:

 methods = Actions

Web Service:

 in/out  = Document

Description of work

# Nominalization

verb → noun

traverse → traversal

collect → collection

implement → implementation

# WS Document Server

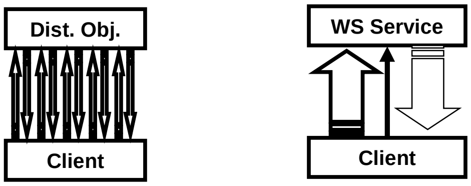
```
class Request {
  String      name;
  bool        doGet;
  Request[]  subItems;
  Param[]    params;
}
```

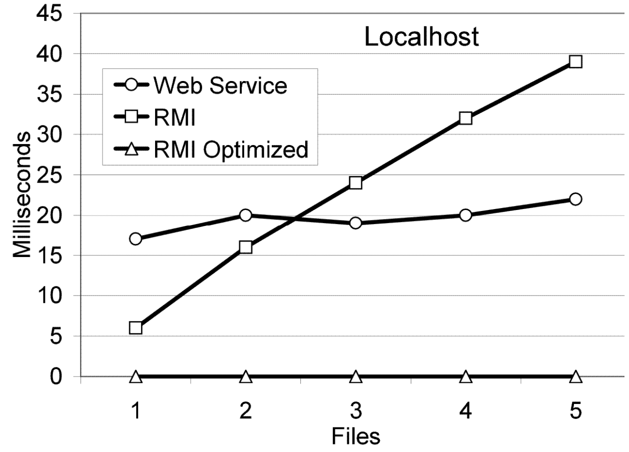
```
class ResourceServer
{
  File[] perform(Request doc);
}
```

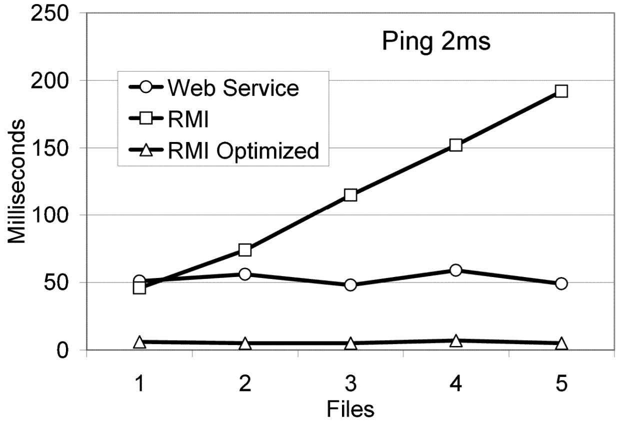
# Calling WS

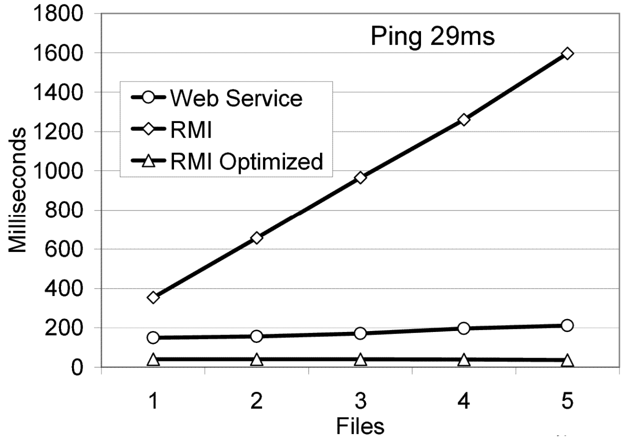
```
Request base = new Request("base");
Request index = new Request("index.htm");
index.doGet = true;
base.subItems.Add(index);

File[] docs = server.perform(base);

String s = docs[0].getText();
```

# Performance









# Ease of Use

# Actions

Open the “talk” folder and get “index.htm”

# Nominalization

Execute a retrieval of the document whose location is “talk” and name is “index.htm”

# Web Service

Execute a <u>retrieval</u> of the document whose <u>location</u> is “talk” and name is “index.htm”

# Better Wrappers

Object-oriented wrappers to document-oriented service

# Background

Explicit variant of Liskov's "batched futures"

# Calling Wrapper

Future

```
Container c;
c = root.Sub("papers").Sub("index.htm");
File doc = c.Get();
```

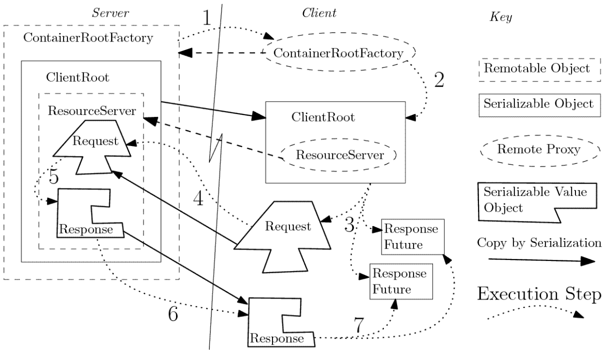
traverse

use results

```
root.perform();
String s = doc.Text();
```

Can't take something that works locally and make it remote. But taking something that works remotely and using it locally is ok. – **Don Box**

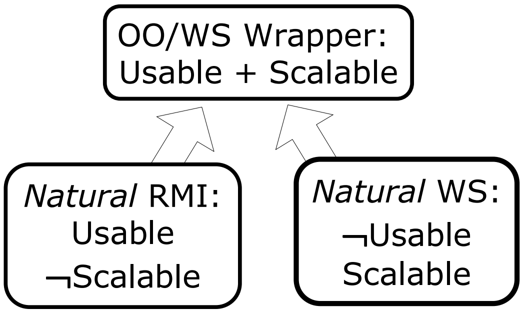
# Wrapper



# Benefits

Looks like RMI

Works like Web Service



OO/WS Wrapper: Usable + Scalable

*Natural* RMI: Usable ¬Scalable

*Natural* WS: ¬Usable Scalable

# Observation

WS Document can be

"thing": purchase order

"actions": insert, update, delete, search, traverse…

# Conclusion

- Web Services are <u>not</u> distributed objects
- Can simulate each other
- With *natural* design, WS are faster than RMI

Questions?