Subject: RMI
Posted by mirek on Sun, 22 Apr 2007 07:21:16 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 02:47
Quote:
and maybe another good point:
RMI for C++
http://rmi.sourceforge.net/pmwiki/pmwiki.php
The pages are password protected.
BTW I was thinking about something similar in UPP, because current callback system could be easily extended to work transparently on network.

So do I But my thinking is rather based on Serialize...

The real question(s) is:

- can we do RMI without IDL?
- is it worth the trouble?
- does it need to look like method invokation or we can can sustain a little bit less transparent solution (like passing structures with Serialize member)?

Mirek

P.S.: As this is an interesting topic, I am moving a copy to technology lab, if you have any ideas, let us continue there...

Subject: Re: RMI
Posted by zsolt on Sun, 22 Apr 2007 08:17:02 GMT
View Forum Message <> Reply to Message

I was thinking about creating some callback system for network communication, because in an event based system (GUI app) blocking calls are not the best.

BTW, RMI is used in Java internally only, because SOAP and XMLRPC is more portable and can be integrated easily in an enterprise environment.

So an ideal solution would be able to use a native C++ serialization or SOAP/XMLRPC as a transport layer as well.

Subject: Re: RMI
Posted by mirek on Sun, 22 Apr 2007 09:01:10 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 04:17I was thinking about creating some callback system for network communication, because in an event based system (GUI app) blocking calls are not the best.

BTW, RMI is used in Java internally only, because SOAP and XMLRPC is more portable and can be integrated easily in an enterprise environment.

So an ideal solution would be able to use a native C++ serialization or SOAP/XMLRPC as a transport layer as well.

Yes, but even for internal use, it is nice to have.

One thing that it could nicely address is clustering applications.

BTW, what you would relatively easy to achieve w.r.t. calls alone without any form of IDL:

```
struct MethodA {
    int a;
    Vector<String> b;

    void Serialize(Stream& s) { s % a % b; }
};

struct MethodB {
   ....
}

struct Server {
    void Do(MethodA& a);
    void Do(MethodB& b);
};

client:

Connection<Server> x;
....
MethodA a;
a.x = 123;
a.b.Add(123);
x.Do(a);
```

(Just to demostrate where I have ended the last time investigating this issue).

Mirek

Subject: Re: RMI
Posted by zsolt on Sun, 22 Apr 2007 10:25:10 GMT
View Forum Message <> Reply to Message

```
class RemotelyCallableClass
{
public:
 RCallback1<int> doSomething;
};

class ClientSide()
{
 RClient<RemotelyCallableClass> client; //some setup needed
 void doSomethingRemotely()
 {
  client.doSomething(33);
 }
};

class ServerSide()
{
public:
 ServerSide()
 {

  server.doSomething <<= THISBACK(OnDoSomething);
  //some setup (IP, port)
 }
 RServer<RemotelyCallableClass> server;
 void OnDoSomething(int v)
 {
  //some processing on server
 }
};
```

Of course this is an idea only
RCallback arguments should implement a serializable interface if they are not basic types.
The return value could be handled as a callback on the client side with something like RGate<int, int>.
I don't prefer blocking calls.

Edit: I changed RClient to RServer on server side.

Subject: Re: RMI
Posted by mirek on Sun, 22 Apr 2007 10:49:49 GMT
View Forum Message <> Reply to Message

Well, this is not a bad idea! If nothing else, it avoids the need to create those structs and to implement Serialize....

Mirek

Subject: Re: RMI
Posted by Ulti on Sun, 22 Apr 2007 11:07:56 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 04:17I was thinking about creating some callback system for network communication, because in an event based system (GUI app) blocking calls are not the best.

BTW, RMI is used in Java internally only, because SOAP and XMLRPC is more portable and can be integrated easily in an enterprise environment.

So an ideal solution would be able to use a native C++ serialization or SOAP/XMLRPC as a transport layer as well.
SOAP/XMLRPC is low efficiency.
for enterprise,it need much work to do:
such as corba(http://www.mico.org/) and ICE(www.zeroc.com)

better for internal first
thanks!

Subject: Re: RMI
Posted by zsolt on Sun, 22 Apr 2007 11:10:18 GMT
View Forum Message <> Reply to Message

Yes
But my main purpose was non blocking behaviour.

Subject: Re: RMI
Posted by zsolt on Sun, 22 Apr 2007 16:14:02 GMT
View Forum Message <> Reply to Message

I was thinking a bit about it.
I think, that in some situations it could be useful, to have the same API for distributed and multi processed (multi thread/core) tasks.

Subject: Re: RMI

Posted by fallingdutch on Mon, 23 Apr 2007 06:04:08 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 10:17I was thinking about creating some callback system for network communication, because in an event based system (GUI app) blocking calls are not the best.

I am working on the same, but wanted to extend it to all Files/Devices.

On Linux: poll with read/write
on Windows ReadFile, WriteFile with OVERLAPPED and MsgWaitOnMultipleObjects

the Problem i have to solve at the moment is how to get the numbers of bytes in the buffer at a communication device on Windows. On Linux I use a nonblocking read, it returns the number of bytes read or gives the error wouldblock if nothing is in the buffer.

Do you have any Ideas?
Bas

Subject: Re: RMI
Posted by mirek on Mon, 23 Apr 2007 07:25:41 GMT
View Forum Message <> Reply to Message

fallingdutch wrote on Mon, 23 April 2007 02:04zsolt wrote on Sun, 22 April 2007 10:17I was thinking about creating some callback system for network communication, because in an event based system (GUI app) blocking calls are not the best.

I am working on the same, but wanted to extend it to all Files/Devices.

Are you sure this is the same problem? In fact, I think callback system for network communication would be a great implementation tool for RMI or for something built on RMI, but it is not RMI itself

Mirek

Subject: Re: RMI
Posted by Ulti on Mon, 23 Apr 2007 08:36:19 GMT
View Forum Message <> Reply to Message

personally like ICE very much(though ICE is not RMI,it's middleware),but it's GPLed.if U++ support RMI,then php will be easily supported(via writing PHP extension)

Subject: Re: RMI
Posted by Shire on Fri, 18 Jan 2008 08:38:41 GMT
View Forum Message <> Reply to Message

Quote:
personally like ICE very much(though ICE is not RMI,it's middleware),but it's GPLed.

I prefer ICE too, but it is based on STL. UPP can have its own implementation of network communication protocol.
IMHO, communication will be better, if it is based on queries, rather than RPC. Query is simple structure, consist of base types. Queries is better for asynchronous work, multi parameter passing and grouping multiple in one network packet. Handling queries can be easily chained. Query can build by accessors and query builders. Binding to other languages (USC, for example) also can be simple.
To do this, these components needed:

* serialization for marshalling queries
* platform-independent query description language (can be done by macros)
* platform-independent binary protocol description (IMHO, the most difficult part)
* local (per application) or/and global components registry (for dynamic creation)

Subject: Re: RMI
Posted by mirek on Fri, 18 Jan 2008 09:13:06 GMT
View Forum Message <> Reply to Message

Shire wrote on Fri, 18 January 2008 03:38Quote:
personally like ICE very much(though ICE is not RMI,it's middleware),but it's GPLed.

I prefer ICE too, but it is based on STL. UPP can have its own implementation of network communication protocol.
IMHO, communication will be better, if it is based on queries, rather than RPC. Query is simple structure, consist of base types. Queries is better for asynchronous work, multi parameter passing and grouping multiple in one network packet. Handling queries can be easily chained. Query can build by accessors and query builders. Binding to other languages (USC, for example) also can be simple.
To do this, these components needed:

* serialization for marshalling queries
* platform-independent query description language (can be done by macros)
* platform-independent binary protocol description (IMHO, the most difficult part)
* local (per application) or/and global components registry (for dynamic creation)

Interestingly, your new response to this old thread came at the moment I am about to start implementing some SOAP for/in U++

Mirek

Subject: Re: RMI
Posted by Shire on Fri, 18 Jan 2008 11:58:15 GMT
View Forum Message <> Reply to Message

Well, funny coincidence
SOAP support will be good.