

DRb

Distributed Ruby

Johan Sørensen <johan@johansorensen.com>

DRb?

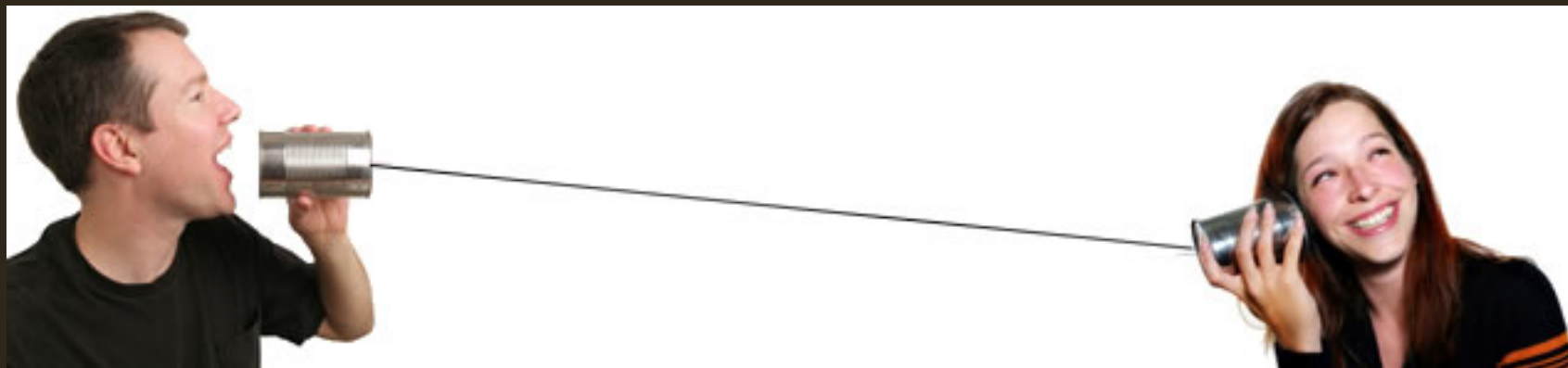
- Remote Method Invocation (“RMI”) for Ruby
- Af Masatoshi SEKI
- Peer to Peer
- Client, Server
- Crossplatform (pure ruby)
- Multiprotocol

DRb exempel

Object

??

Object



DRb exempel

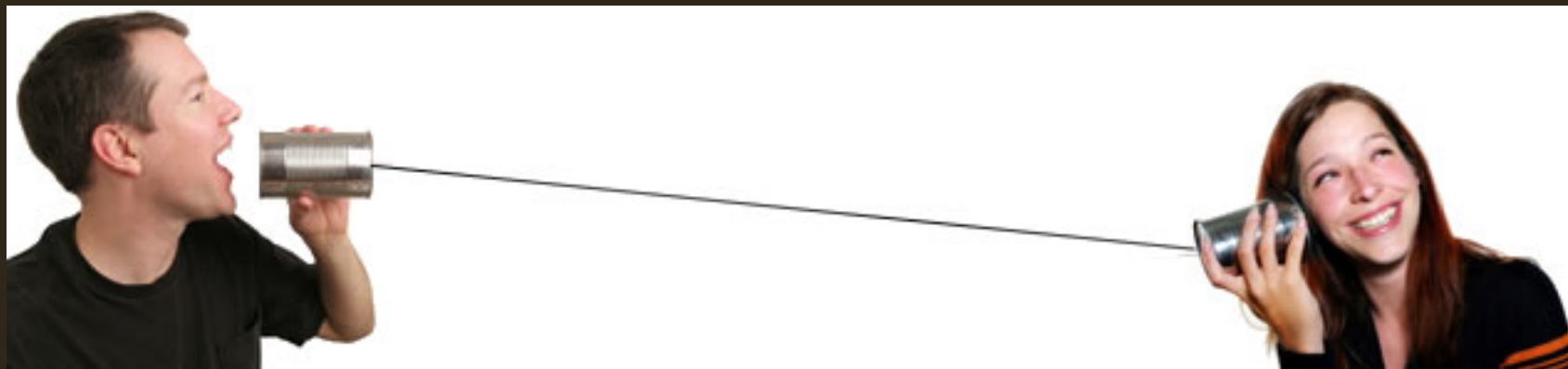
```
# Server
require "drb"
obj = [1, 2, 3, "four", "Five"]
uri = "druby://127.0.0.1:6666"
DRb.start_service(uri, obj)
DRb.thread.join
```

```
# Client
require "drb"
DRb.start_service
uri = "druby://127.0.0.1:6666"
obj = DRbObject.new_with_uri(uri)
obj.size #=> 5
obj.last #=> "Five"
obj.map{|v| v+v }
#=> [2, 4, 6, "fourfour", "FiveFive"]
```

Object

??

Object



DRb Exempel

```
# Server
require "drb"
obj = [1, 2, 3, "four", "Five"]
uri = "druby://127.0.0.1:6666"
DRb.start_service(uri, obj)
DRb.thread.join

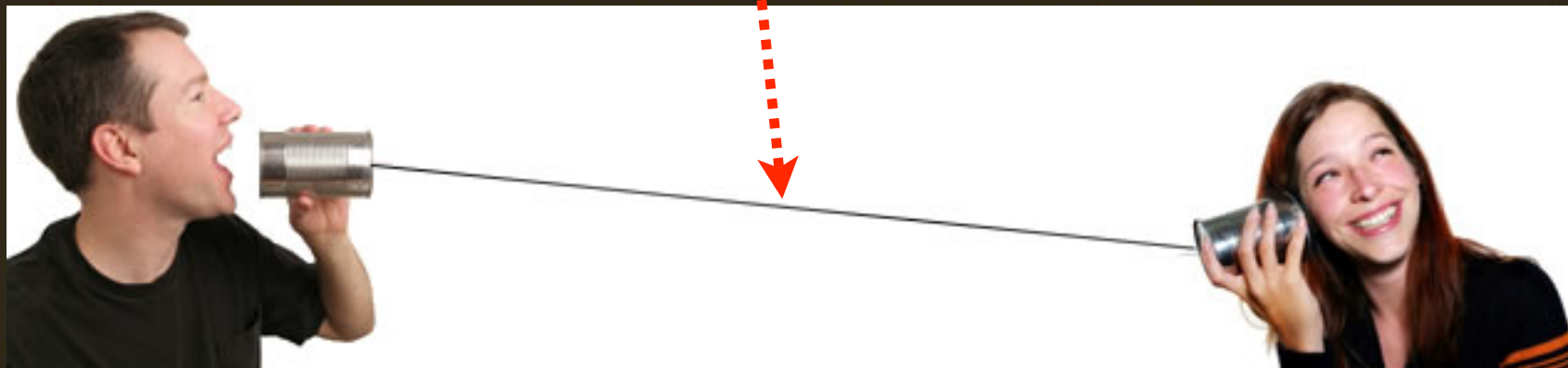
# DRb biblioteket
# Objektet vi kommer att dela över DRb
# URI'n servern ska lyssna på
# Start server som lyssnar på +uri+, med +obj+
# Vent tills DRb er klar
```

DRb Exempel

```
# Client
require "drb"
DRb.start_service
uri = "druby://127.0.0.1:6666"      # Vår server uri
obj = DRbObject.new_with_uri(uri)  # obj ska vara ett DRbObject ifrån uri
obj.size #=> 5                     # Vi kan kalla metoder på det andra objektet!
obj.last #=> "Five"
obj.map{|v| v+v }
#=> [2, 4, 6, "fourfour", "FiveFive"]
```

DRb?

magic??



DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1676)
[ call] with_friend DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1101)
[ call] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1755)
[ return] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1757)
[ call] open DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1152)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
[ call] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:110)
[ return] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:111)
[ return] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:133)
[ call] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1187)
[ call] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1683)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1687)
[ call] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:728)
[ call] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:822)
[ call] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:807)
[ return] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:817)
[ call] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:883)
[ call] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:549)
[ return] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:550)
[ call] set_sockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:961)
[ return] set_sockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:962)
[ return] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:884)
[ return] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:823)
[ return] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:729)
[ return] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1188)
[ call] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1193)
[ call] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:905)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
```


DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1676)
[ call] with_friend DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1101)
[ call] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1755)
[ return] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1757)
[ call] open DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1152)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
[ call] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:110)
[ return] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:111)
[ return] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:133)
[ call] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1187)
[ call] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1683)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1687)
[ call] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:728)
[ call] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:822)
[ call] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:807)
[ return] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:817)
[ call] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:883)
[ call] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:549)
[ return] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:550)
[ call] set_sockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:961)
[ return] set_sockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:962)
[ return] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:884)
[ return] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:823)
[ return] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:729)
[ return] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1188)
[ call] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1193)
[ call] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:905)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
```

DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1676)
[ call] with_friend DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1101)
[ call] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1755)
[ return] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1757)
[ call] open DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1152)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
[ call] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:110)
[ return] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:111)
[ return] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:133)
[ call] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1187)
[ call] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1683)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1687)
[ call] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:728)
[ call] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:822)
[ call] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:807)
[ return] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:817)
[ call] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:883)
[ call] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:549)
[ return] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:550)
[ call] setsockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:961)
[ return] setsockopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:962)
[ return] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:884)
[ return] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:823)
[ return] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:729)
[ return] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1188)
[ call] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1193)
[ call] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:905)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
```

DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1676)
[ call] with_friend DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1101)
[ call] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1755)
[ return] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1757)
[ call] open DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1152)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
[ call] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:110)
[ return] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:111)
[ return] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:133)
[ call] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1187)
[ call] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1683)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1687)
[ call] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:728)
[ call] open DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:822)
[ call] parse_uri DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:807)
[ return] parse_uri DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:817)
[ call] initialize DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:883)
[ call] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:549)
[ return] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:550)
[ call] set_sockopt DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:961)
[ return] set_sockopt DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:962)
[ return] initialize DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:884)
[ return] open DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:823)
[ return] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:729)
[ return] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1188)
[ call] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1193)
[ call] send_request DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:905)
[ call] stream DRb::DRbTCPsocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
```


DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1676)
[ call] with_friend DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1101)
[ call] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1755)
[ return] fetch_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1757)
[ call] open DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1152)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
[ call] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:110)
[ return] unlock Mutex (/opt/local/lib/ruby/1.8/thread.rb:111)
[ return] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:133)
[ call] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1187)
[ call] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1683)
[ call] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1647)
[ return] current_server DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1648)
[ return] config DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1687)
[ call] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:728)
[ call] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:822)
[ call] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:807)
[ return] parse_uri DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:817)
[ call] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:883)
[ call] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:549)
[ return] initialize DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:550)
[ call] set_socketopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:961)
[ return] set_socketopt DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:962)
[ return] initialize DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:884)
[ return] open DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:823)
[ return] open DRb::DRbProtocol (/opt/local/lib/ruby/1.8/drb/drb.rb:729)
[ return] initialize DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1188)
[ call] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1193)
[ call] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:905)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
```

DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRb|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
# ~~~~~ #
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ return] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ call] send_request DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:597)
[ call] __drbref DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1058)
[ return] __drbref DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1060)
[ call] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:554)
[ return] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:555)
[ call] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:554)
[ return] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:555)
[ call] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:554)
[ return] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:555)
[ call] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:554)
[ return] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:555)
[ return] send_request DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:609)
[ return] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:907)
[ call] recv_reply DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:920)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ return] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ call] recv_reply DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:631)
[ call] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:564)
[ call] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:29)
[ return] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:30)
[ return] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:565)
[ call] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:564)
[ call] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:29)
[ return] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:30)
[ return] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:565)
[ return] recv_reply DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:632)
[ return] recv_reply DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:922)
[ return] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1194)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
```


(Marshal crash course)

```
>> class Foo; def initialize(x); @x = x; end; end  
=> nil
```

```
>> f = Foo.new("a value")  
=> #<Foo:0x4bd20 @x="a value">
```

```
>> dump = Marshal::dump(f)  
=> "\004\bo:\bFoo\006:\a@x\"fa value"
```

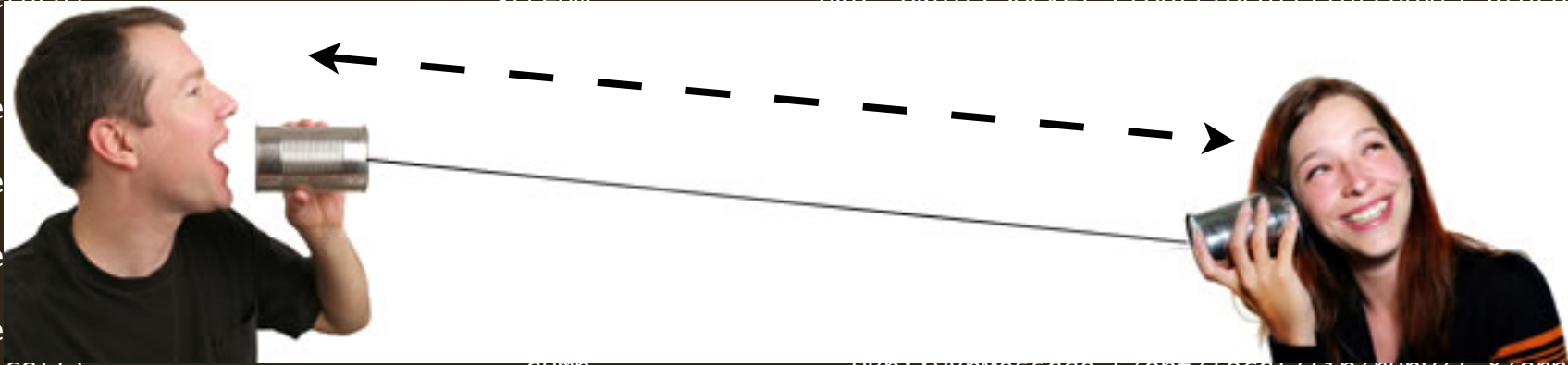
```
>> f = nil  
=> nil
```

```
>> g = Marshal::load(dump)  
=> #<Foo:0x19e88 @x="a value">
```

```
>> g  
=> #<Foo:0x19e88 @x="a value">
```

DRb?

```
>> obj = DRbObject.new_with_uri(uri)
=> #<DRb::DRbObject:0x53d66c @ref=nil, @uri="druby://127.0.0.1:6666">
>> enable_trace # set_trace_func{...}
Enabling method tracing with event regex /^(call|return)/ and class exclusion regex /IRB|Wirble|RubyLex|RubyToken/
[ return] enable_trace Object (/Users/johan/.irbrc:67)
=> nil
>> obj.size
[ call] method_missing DRb::DRbObject (/opt/local/lib/ruby/1.8/drb/drb.rb:1077)
[ call] here? DRb (/opt/local/lib/ruby/1.8/drb/drb.rb:1674)
# ~~~~~ #
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ return] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ call] /drb.rb:597)
[ return] /drb.rb:1058)
[ call] /drb.rb:1060)
[ return] /drb.rb:554)
[ call] /drb.rb:555)
[ return] /drb.rb:554)
[ call] /drb.rb:555)
[ return] /drb.rb:554)
[ call] /drb.rb:555)
[ return] /drb.rb:555)
[ call] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:554)
[ return] dump DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:555)
[ return] send_request DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:609)
[ return] send_request DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:907)
[ call] recv_reply DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:920)
[ call] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ return] stream DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:902)
[ call] recv_reply DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:631)
[ call] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:564)
[ call] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:29)
[ return] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:30)
[ return] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:565)
[ call] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:564)
[ call] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:29)
[ return] exclusive Thread (/opt/local/lib/ruby/1.8/thread.rb:30)
[ return] load DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:565)
[ return] recv_reply DRb::DRbMessage (/opt/local/lib/ruby/1.8/drb/drb.rb:632)
[ return] recv_reply DRb::DRbTCPSocket (/opt/local/lib/ruby/1.8/drb/drb.rb:922)
[ return] send_message DRb::DRbConn (/opt/local/lib/ruby/1.8/drb/drb.rb:1194)
[ call] synchronize Mutex (/opt/local/lib/ruby/1.8/thread.rb:132)
[ call] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:97)
[ return] lock Mutex (/opt/local/lib/ruby/1.8/thread.rb:98)
```



DRb Exempel Redux

```
# Client
require "drb"
DRb.start_service
uri = "druby://127.0.0.1:6666"
obj = DRbObject.new_with_uri(uri)
obj.size #=> 5
obj.last #=> "Five"
obj.map{|v| v+v }
#=> [2, 4, 6, "fourfour", "FiveFive"]

# ...
# Initiera DRb
# Server URI att connecta till
# Nytt DRbObject ifrån uri
# Vår Array finns tillgänglig över DRb!
```

Remote Method Invocation

- Client
 - Marshal method arguments
 - Send message to server
 - Reference, method name, arguments
- Server
 - Receive message
 - Unmarshal arguments
 - Find local object from reference
 - Invoke method with arguments

DRbUndumped

- Utan DRbUndumped
 - Pass by value
 - Måste dela kod
- Med DRbUndumped included
 - Pass by reference (ingen Marshalling)
 - Måste inte dela kod

```
require "drb"  
class Foo  
  include DRbUndumped  
  # ...  
end  
  
obj = [1,2,3]  
obj.extend DRb::DRbUndumped
```

Varför bruke DRbUndumped?

- Stora objekt
- Singleton objekt kan brukes
- Mindre/”tunna” klienter
- Inget behov av att hålla klassar i synk

Sockets

- TCP Sockets (“druby://127.0.0.1:6661”)
- Unix socket (“drbunix:///tmp/foo.sock”)
- SSL socket (“drbssl://127.0.0.1:6661” + .pem cert)

DRb sikkerhet

- Går ikke helt
- ACL
- Ruby's \$SAFE level
- default_load_limit
- SSL
- Firewall, user permissions

ACL Exempel

```
require "drb"  
require "drb/acl"  
acl = ACL.new %w[  
  deny all  
  allow 127.0.0.1  
]  
DRb.install_acl(acl)  
DRb.start_service # blabla..
```

DRb med Rails

- Enkelt!
- Bra att få “tunga ting” ut ifrån Rails
- Backgroundrb

Backgroundrb

- <http://backgroundrb.rubyforge.org/>
- “ BackgroundDRb is a ruby job server and scheduler. Its main intent is to be used with Ruby on Rails applications for offloading long running tasks. Since a rails application blocks while servicing a request it is best to move long running tasks off into a background process that is divorced from the http request/response cycle. ”
- Rails plugin, `queue+workers` based at
- `SomeWorkerClass#do_work`

Bingo!

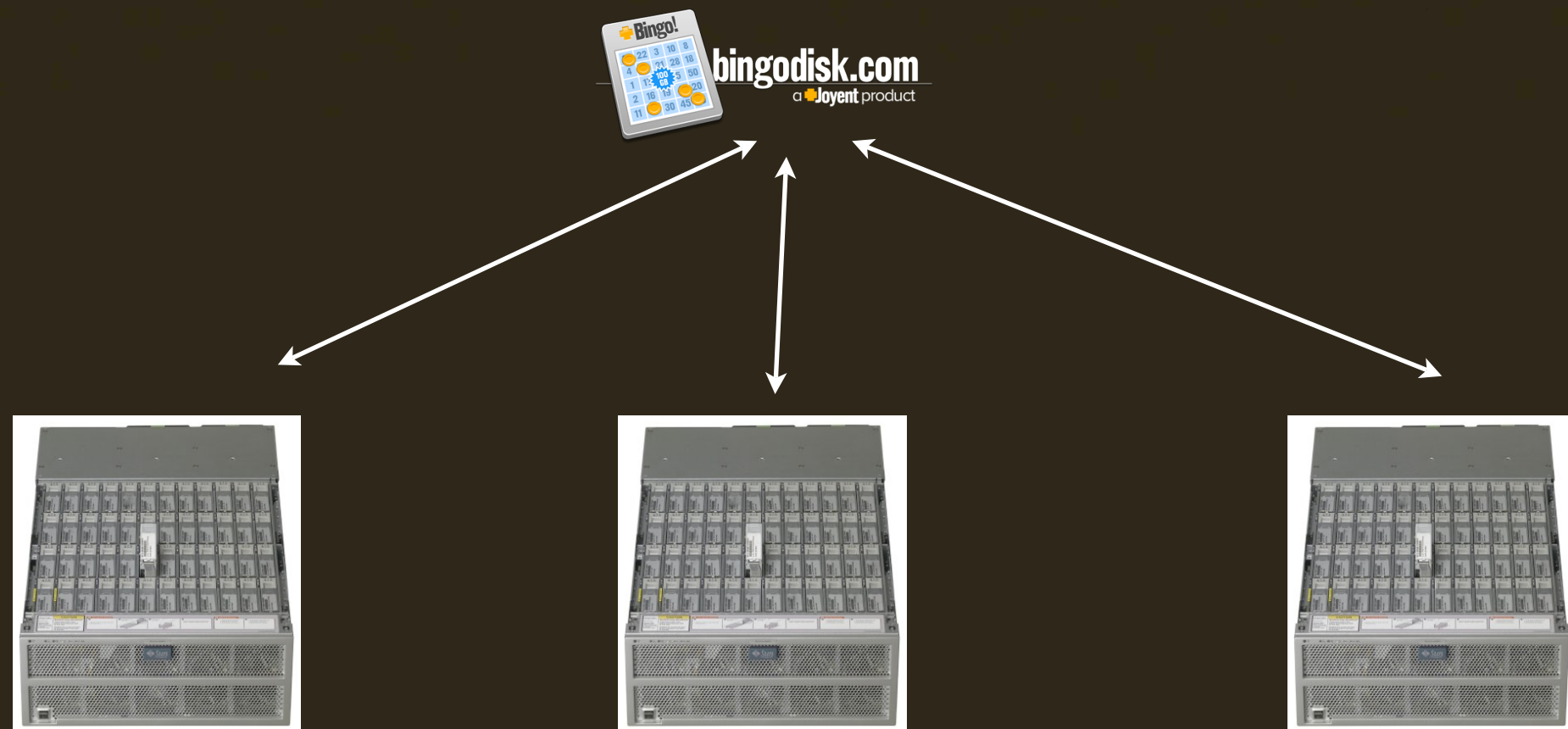


bingodisk.com
a + Joyent product

“Disk in the sky”

Bingo!

- En frontend app (www.bingodisk.com)
- X antal bingo nodes (diskserverar), tex johan.bingodisk.com, som kontaktas via DRb



Bingo!



Pseudo code*

The rails app

```
@customer.server = Server.find_least_utilitized
@customer.server.setup_customer!
```

```
class Server < ActiveRecord::Base
  def setup_customer!
    node_system = Bingo::Node.new(Config.node_config)
    node_system.setup_on_node(self, self.customer)
  end
end
```

node daemon (a DRb server)

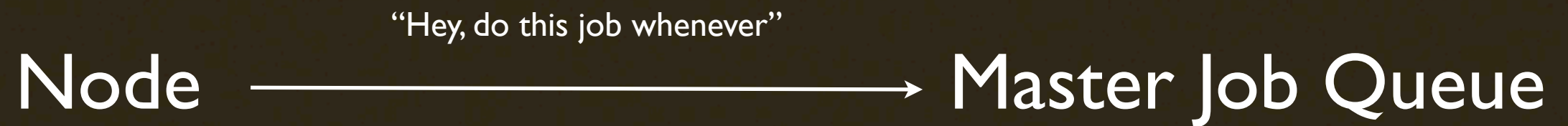
```
class Bingo::Node
  def initialize(config)
    # Hook up DRbconnection etc
    # from config values
  end

  def setup_on_node(server, customer)
    # create ZFS filesystem
    # config something
    # ...
  end
end
```

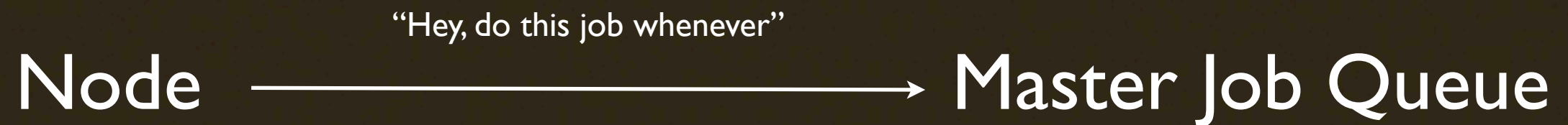
Bingo! Deux

- Varje node måste parsa en stor loggfil ca. 150 gånger om dagen
- Ruby fint*
- * Men treegt
- Green threads, inte “äkta” threads
- Distribuera workload med DRb!

Bingo! Deux



Bingo! Deux



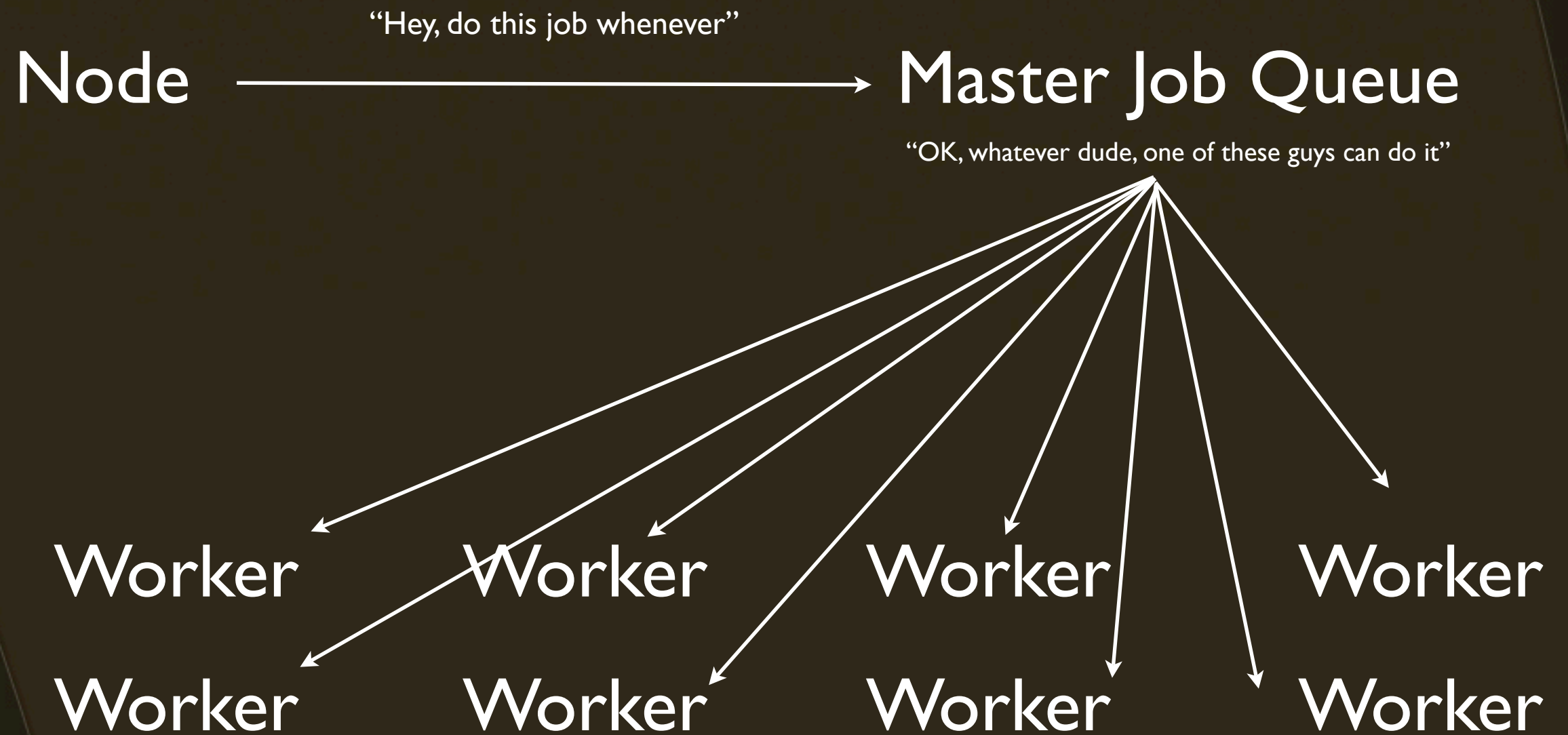
Worker
Worker

Worker
Worker

Worker
Worker

Worker
Worker

Bingo! Deux



Bingo! Deux

Med Ruby's Threads



ZZZZ

CPU2

ZZZZ

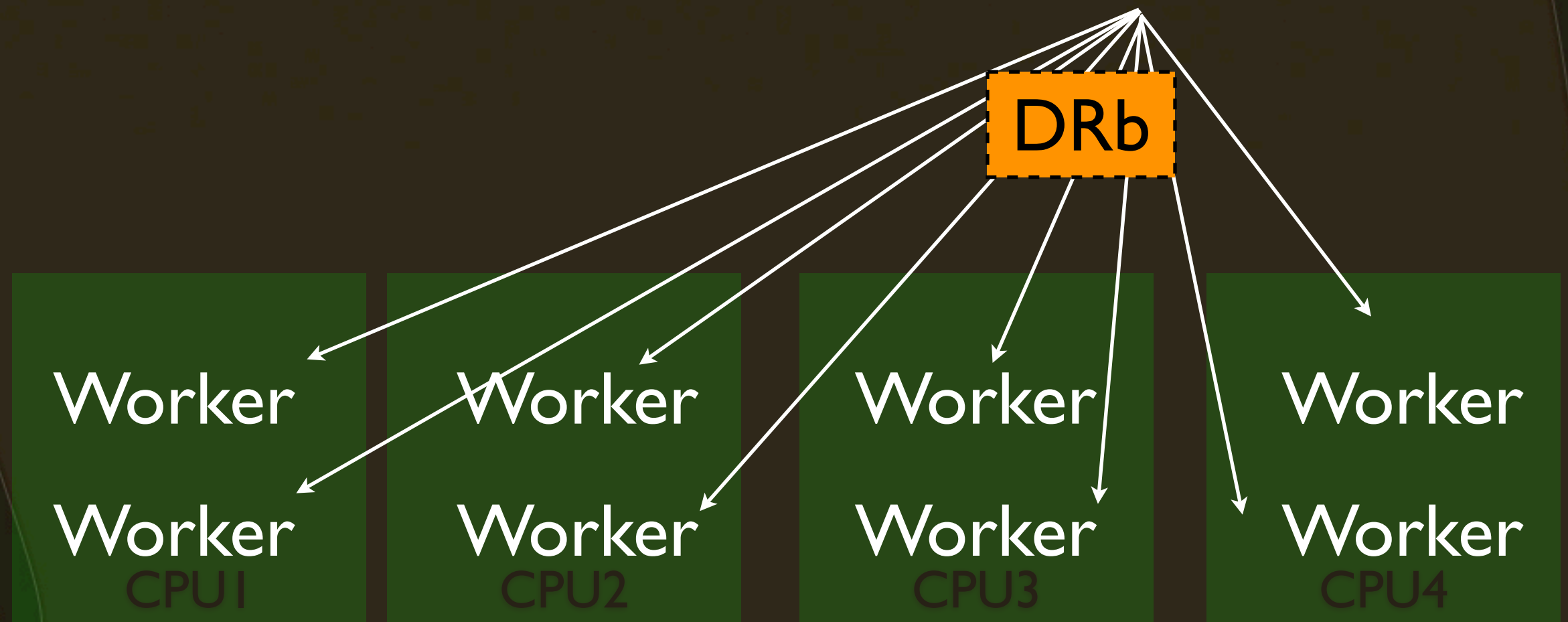
CPU3

ZZZZ

CPU4

Bingo! Deux

Master Job Queue



Bingo! Deux

Worker

Worker

SERVER1

Worker

Worker

SERVER2

Worker

Worker

SERVER3

Worker

Worker

SERVER4

Bingo! Deux

from `__future__` **import** yagni

DATACENTER 1

Worker

Worker

SERVER1

Worker

Worker

SERVER2

DATACENTER 2

Worker

Worker

SERVER3

Worker

Worker

SERVER4

... Heeeey!

Distributed programming

(med ruby!)

Rinda

- Ruby's Linda
- TupleSpace (delat object space)
- RingServer (zeroconf/bonjour for ruby)

Rinda Tuplespaces & RingServer

- Distribuerat whiteboard
- En tjänst lägger sig till på whiteboardet, säger vad den har, och whiteboardet vet om dens IP etc
- En klient frågar efter en tjänst på whiteboardet och får tillbaka dens IP+port
- klienten blir “automagiskt” kopplat till tjänsten

Rinda::TupleSpace

- En klient lägger till sina objekt på ett whiteboard och läsa/skriva/ta dom
- Tuple: [:name, class, (DRb)Object, “note”]
- Brukar ===
 - Matchar tuples med Regexp, Class===Class, #value === “value”
- ts#read(tmpl), ts#read_all(tmpl), ts#write(), ts#take(tmpl)

Rinda::TupleSpace

```
# server.rb
require "drb"
require "rinda/tuplespace"

ts = Rinda::TupleSpace.new
# share the ts over DRb
DRb.start_service("druby://127.0.0.1:6661", ts)
DRb.thread.join

# client.rb
# adds to the tuplespace
require "drb"
require "rinda/tuplespace"
require "pp"

# connect to the tuplespace
DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)

tuplespace.write([0, "Note", "first ts entry"])
tuplespace.write([1, "Note", "second ts entry"])
tuplespace.write([2, "Note", "third ts entry"])

#client_reader.rb
# reads from the tuplespace
require "drb"
require "rinda/tuplespace"

DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)

# grab all the notes, matching them against the Range
while note = tuplespace.take([nil, nil, /^([^\s]*)/])
  puts note.inspect
  # write it back into the tuplespace
  tuplespace.write([note.first, note[1], "READ -- #{note.last}"])
end
```

Rinda::TupleSpace

```
# server.rb
require "drb"
require "rinda/tuplespace"

ts = Rinda::TupleSpace.new
# share the ts over DRb
DRb.start_service("druby://127.0.0.1:6661", ts)
DRb.thread.join

# client.rb
# adds to the tuplespace
require "drb"
require "rinda/tuplespace"
require "pp"

# connect to the tuplespace
DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)

tuplespace.write([0, "Note", "first ts entry"])
tuplespace.write([1, "Note", "second ts entry"])
tuplespace.write([2, "Note", "third ts entry"])

#client_reader.rb
# reads from the tuplespace
require "drb"
require "rinda/tuplespace"

DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)

# grab all the notes, matching them against the Range
while note = tuplespace.take([nil, nil, /^[^READ]/])
  puts note.inspect
  # write it back into the tuplespace
  tuplespace.write([note.first, note[1], "READ -- #{note.last}"])
end
```


Rinda::TupleSpace

```
# server.rb
require "drb"
require "rinda/tuplespace"

ts = Rinda::TupleSpace.new
# share the ts over DRb
DRb.start_service("druby://127.0.0.1:6661", ts)
DRb.thread.join

# client.rb
# adds to the tuplespace
require "drb"
require "rinda/tuplespace"
require "pp"

# connect to the tuplespace
DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)
```

```
tuplespace.write([0, "Note", "first ts entry"])
tuplespace.write([1, "Note", "second ts entry"])
tuplespace.write([2, "Note", "third ts entry"])
```

```
#client_reader.rb
# reads from the tuplespace
require "drb"
require "rinda/tuplespace"
```

```
DRb.start_service
tuplespace = Rinda::TupleSpaceProxy.new(
  DRbObject.new_with_uri("druby://127.0.0.1:6661")
)
```

```
# grab all the notes, matching them against the Range
while note = tuplespace.take([nil, nil, /^[^READ]/])
  puts note.inspect
  # write it back into the tuplespace
  tuplespace.write([note.first, note[1], "READ -- #{note.last}"])
end
```

```
$ ruby client_reader.rb
[0, "Note", "first ts entry"]
[1, "Note", "second ts entry"]
[2, "Note", "third ts entry"]
```

Rinda::RingServer

```
# rinda_server.rb
require "rinda/ring"
require "rinda/tuplespace"

DRb.start_service

# Ett TupleSpace som innehåller namngivna tjänster
Rinda::RingServer.new(Rinda::TupleSpace.new)
DRb.thread.join

# sharing_a_tuplespace.rb
require "rinda/ring"
require "rinda/tuplespace"

DRb.start_service
ring_server = Rinda::RingFinger.primary # finds our "primary" RingServer

# Dela ut vårt tuplespace
# SimpleRenewer är en enkel Renewer som anger hur ofta den ska checka om tjänsten ever
ring_server.write([:name, :TupleSpace, Rinda::TupleSpace.new], 'Tuple Space',
                  Rinda::SimpleRenewer.new)

# using_the_tuplespace.rb
require "rinda/ring"
require "rinda/tuplespace"

DRb.start_service
ring_server = Rinda::RingFinger.primary
# Ask the RingServer for our TupleSpace
ts_service = ring_server.read[:name, :TupleSpace, nil, nil][2] # [2] är vårt DRbObject
ts = Rinda::TupleSpaceProxy.new(ts_service)

ts.write([:data, rand(100)])
puts "data is #{ts.read[:data, nil].last}"
# => data is 62
```

Starfish

`gem install starfish`

MapReduce for Ruby folk

dvs. nok till att få jobben gjort

Starfish

```
User.find(:all, :conditions => "opt_out=0").each{|user| user.calculate_solution_to_life }
```

Starfish

```
User.find(:all, :conditions => "opt_out=0").each{|user| user.calculate_solution_to_life }
```

```
# user_solution.rb
require 'config/environment'
require 'user'

server do |map_reduce|
  map_reduce.type = User
  map_reduce.conditions = "opt_out = 0"
end

client do |user|
  user.calculate_solution_to_life
end
```


Starfish

```
User.find(:all, :conditions => "opt_out=0").each{|user| user.calculate_solution_to_life }
```

```
# user_solution.rb
require 'config/environment'
require 'user'

server do |map_reduce|
  map_reduce.type = User
  map_reduce.conditions = "opt_out = 0"
end

client do |user|
  user.calculate_solution_to_life
end
```

```
my_computer$ starfish user_solution.rb
my_computer$ starfish user_solution.rb
my_computer$ starfish user_solution.rb
my_computer$ starfish user_solution.rb
bobs_computer$ starfish user_solution.rb
moms_old_scrapyard$ starfish user_solution.rb
the_mainframe_at_work$ starfish user_solution.rb
```

The End

- <http://segment7.net/projects/ruby/drb/rinda/ringserver.html>
- <http://blog.segment7.net/articles/2006/04/22/drb-an-introduction-and-overview>
- <http://www.ruby-doc.org/stdlib/libdoc/drb/rdoc/index.html>
- <http://www2a.biglobe.ne.jp/~seki/ruby/druby.html>
- Ruby tarball: drb/samples/
- <http://tech.ruby.com/2006/08/mapreduce-for-ruby-ridiculously-easy.html>
- <http://www.chadfowler.com/ruby/drb.html>