RPC in the modern world

CS 414: Advanced Systems
Oliver Kennedy
RPC Overview

- Remote procedures can be called as if local.
  - ... but they execute remotely
- The RPC system deals with the network so you don’t have to.
Can we do better?

- Object oriented languages exist.
- Can we abstract objects?
  - If we can abstract methods, why not?
  - Object mobility
- Can we abstract data?
  - Do we care where code runs?
Network Objects: RPC++

- RPC allows us to virtualize a method.
- Why not virtualize the whole object?
- Network objects implement this idea.
Deja Vu

Client

Bind Object
Lookup Object
Create Object
Register Object
List Object
Reply
Create Stub
Use Object

NetObjects
User Code

Server

Create Object
Register Object
List Object
Reply
Create Stub
Use Object

NetObjects
User Code
Why Objects?

- RPC can’t pass pointers.
  - Centralized data structures become complicated.
- RPC has problems with interactable data structures.
  - Streams, Sockets, …
Getting from here to there...
Share the love
The fine print

- To Send or to Reference?
  - Tag each class as sendable or not.
- How can unknown classes be encoded?
  - Use Introspection to break a class down.
- How do you encode special datatypes?
  - Add hooks to let a class to encode itself.
- How do target machines know what code to run?
Is RPC/NetObjects worth it?

- Can we really trick the programmer?
  - Programmers need to know about network issues.
  - Hello world can fail.

- Does it matter?
  - With minor changes, most error conditions can be detected.
    - This breaks the abstraction.
  - Reliability is a problem.
    - Programmers lose fine-grained control.
Applications

- *Network Objects has been used in several projects.*
- **Packagetoold**
  - A dpkg-like tool
- **Siphon**
  - A repository merge tool
Linda

- We’ve seen tools that are used for distributed computing...
- But what is distributed computing?
  - A task is broken down into smaller tasks.
  - Each processor takes on a subtask.
Approaches to Distribution

- Let a compiler figure out how to parallelize.
  - Can code be parallelized?

- RPC: Central system issues commands to processing servers.

- Producer/Consumer
  - Producers need to know who the consumers are or visa versa
A tuple is analogous to a C struct.

We can create a distributed database.

The database can store tuples.

We can create tasks that exist to create more tuples.
Tuple Space

(out, 2, $53.2)

(machine1: "pi", computePi())

"bar", ?s

(machine2: "pi", computePi())

(machine3: "foo", 2, $53.2)
Searching Tuple Space

- Tuple Space queries are done via pattern matching.
- ?, the formal operator.
- ("foo", ?x) would match ("foo", 3) and would set x to 3 (if x is an int).
  - Matching is done by type and length.
- A hash table can be used to speed up lookups.
Linda Primitives

- **Out:** Create a tuple in tuple space
- **Exec:** Create a live tuple for execution
- **Read:** Find a tuple that matches a specific pattern and return its contents
- **In:** Like read, but destroy the tuple
- **Implementations can have both blocking and nonblocking versions of read/in.**
Complex Operations

- Operations on Tuple Space are guaranteed to be atomic and fair.
- Tickets: $x$ tuples in tuple space
- Locks: put the locked data in a tuple
- Queues: drop requests into tuple space
Mindsets

- Linda forces programmers into a new mindset.
- Programs become nuggets of code.
- Programs become more task oriented.
- The idea of a program running on a single computer vanishes.
Problems

- Requires programmers to re-think their approach to programming.
  - Is this really different from other languages?
- Distributed databases? (Implementation isn’t as easy as it seems)
- This approach can have a lot of overhead.
  - Dumb scheduling/Protection.
  - Synchronizing tuple access.
- Crash recovery?
  - Tuples can vanish from the system.
  - A “lock” held by a process might never be released.
Implementation

- Implementations exist on many platforms.
- Linda has been used in several projects:
  - DNA Sequencing, Raytracing, etc...
- Performance measurements?
Conclusions

- The RPC Model can be vastly improved upon.
- By adding objects to RPC, we can create primitives that RPC can’t approach.
- By using a distributed database, we can let the runtime find concurrency in our programs.
- But is it good enough?
  - Networks are inherently unstable.
  - Abstracting away instability can have dire consequences for some applications.