Distributed Systems

A distributed system is ...

"one on which I cannot get any work done because some machine I have never heard of has crashed".

- Loosely-coupled
- network connection
- could be different OSs, or different parts of the OS
- processes must communicate via messages

What advantages does a network of sites offer?

- More work can get done
- Ability to share devices, programs and data
- Greater reliability
- Easier to expand

Two Phase Commit Protocol

With distributed systems we want to ensure that if something goes wrong at one site we don't end up with inconsistent data.

A "transaction" is some event that has to be completely successful or not done at all (atomic).

We need stable storage – usually replicated on several devices – can be done with two copies.

- make the change to one (check for success)
- make the change to the other (check for success)
- if ever the copies disagree copy the original data from the second back to the first

2PC – transaction coordinator and all sites involved in a transaction have stable storage logs.

All transactions can be undone and redone safely.

Log entries and messages

Commit request phase

<prepare> - started the protocol, sent to all sites

<ready> - recorded and returned if ok, <abort> if not ok

Commit phase

<commit> - if all reply in time, sent to all sites

<abort> - sent by coordinator to all sites if something went wrong

Network & Distributed Operating Systems

Network OS

- Communications layer on top of a normal OS.
- Possibly different OS.
- User is aware of different machines.
- Some can copy files across the network but not share them. e.g. ftp

In this case the file location is explicitly known.

- Others can share files but the location is still part of the name.

Distributed OS

Aim to have the system look like one machine.

There is no difference (except speed) between accessing local and remote resources (location transparency).

The Distributed OS can move resources and processes (migration transparency).

Remote Procedure Calls (RPC)

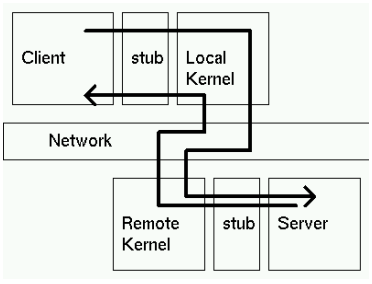
Birrell & Nelson 1984

Hide the message passing system so that it looks like a series of procedure calls.

Most requests for service wait until the request is fulfilled – semantically just like a procedure call.

Programmer doesn't have to package and unpackage data in the messages.

Also adds flexibility because sometimes the service might actually be local.



Client and Server stubs

- Client makes ordinary procedure call to the local stub.
- Stub marshals parameters (may need to locate server as well).
- Stub sends request via local kernel.
- Remote kernel passes request to Server stub.
- Server stub unpacks message request and parameters.
- Makes ordinary procedure call to Server.
- And then vice-versa.

The stubs at both ends need to be constructed from the same interface specification - to ensure consistency.

Care has to be taken about different versions of the service. A different version may take slightly different parameters or return different types etc.

RPC messages

Messages are highly structured

what procedure to execute
parameters
version number (service may survive a long time and code may be written to different versions)
timestamp (could be used for synchronization purposes)
source address
where to send the results
possibly the type of machine the request comes from

Finding the server

- include port numbers at compile time or
- have a binder or rendezvous/matchmaker service

Server usually registers with a binder or name server

client end sends the request to find the server

multiple servers – the binder can spread the load

binder can periodically check on servers - deregistering those that don't respond

binder could do the security work
otherwise servers have to check each call

Service Discovery

There are many different zero configuration services which don't require a binder.

Zeroconf (Apple's Bonjour)

Multicast DNS - is "service" here?
Flexible - protocols can be decided at access time

UPnP - Universal plug and play

A UPnP device can join a network and get an IP address
Announce its name to control points
Provide a list of its functions
Interrogate other devices
Protocols are strictly defined

Both are designed for small LANs.

Marshalling

Heterogeneous networks.
Data formats in the different machines may be different.

- ASCII, Unicode, EBCDIC for strings
- Big or little endian for integers
- Different floating point formats

Stubs need to know about this.

<u>Different solutions</u>

client stub converts to the server format

server stub converts from the client format

convert to and from a canonical format
no one needs to know other machines formats
(we don't want to have to convert from format A to canonical to A and then back again)
e.g. NFS's XDR - external data representation

Reference parameter problem

The references no longer make sense.

- either disallow reference parameters
- or copy the parameter to and fro
- either in one chunk or whenever the server makes a change to it

(this is just like distributed shared memory)

Copy/restore semantics aren't quite the same.

- e.g. pointer as a reference parameter
- the same parameter passed twice in the parameter list

we can check for this

Sometimes we don't need to copy the data both ways
e.g. input buffer (only needs to be copied from the server to the client)
Our interface specification should be able to express this.

RMI & CORBA & DRb

Remote Method Invocation

Java technology

RPC with objects

Solves object reference problem by sending serialized versions of the object. If the object passed as a parameter is itself actually on the remote site it is just sent by reference.

Common Object Request Broker Architecture

Similar to RMI

Works with a wide variety of languages – not just Java

Needs a common description language to specify interfaces – IDL Interface Definition Language

Distributed Ruby

Very similar but simpler than Java RMI. Normally objects are passed by copying. If you

```
include DRbUndumped
```

in a class then all references to such objects are sent back to the original object.

Linda tuplespace

Another technique to share services over a network. Rather than explicitly sending messages we can share data in a tuplespace.

The tuplespace is a logically shared (sometimes distributed) memory consisting of tuples. A tuple is a list of parameters, some of which are empty, e.g, <"hi", 15, 12.5>

Tuples are put into the tuplespace by the "out" or "write" primitive and retrieved from the tuplespace by matching contents and types with the "in" or "take" primitive. The example above could be matched with <"hi", _, _>

JavaSpaces are an implementation of Linda tuplespaces. Rinda is the Ruby version of Linda. Linda originally worked with C and Fortran. lindypy is a Python 2.6 implementation.

Rinda example

```
# Provides a TupleSpace and waits for broadcast
# messages to find it.

require 'rinda/ring'
require 'rinda/tuplespace'

DRb.start_service
Rinda::RingServer.new(Rinda::TupleSpace.new)
DRb.thread.join
```

```
# Finds the TupleSpace by broadcasting,
# then acts as a simple service.

require 'rinda/ring'

DRb.start_service
tuple_space = Rinda::RingFinger.primary
# take the question
question = tuple_space.take([Numeric, Numeric])
# put the answer
tuple_space.write([question[0]+question[1]])
```

```
# Finds the TupleSpace by broadcasting,
# then acts as a client.

require 'rinda/ring'

DRb.start_service
tuple_space = Rinda::RingFinger.primary
# put something in the tuplespace
tuple_space.write([1, 2])
# read the answer
answer = tuple_space.take([Numeric])[0]
puts answer
```

Tuplespace advantages

Processes don't communicate directly with each other but instead access the tuplespace. It doesn't matter if one process dies, as long as another that deals with the same tuple is available (and the tuple hasn't been removed yet).

Tuplespace problems scale naturally. Adding more processes which handle particular requests is trivial.

(The tuplespace itself doesn't scale well. It is usually stored on one server. It can become a bottleneck. There are techniques to replicate tuplespaces.)

The space itself deals with synchronization. Each tuple operation is atomic.

Process Migration

Moving a process from one site to another while it is running.

Why would we like to do process migration?

- To enable us to do proper load balancing.
- If the process can be subdivided we can increase performance by having different parts running simultaneously on different machines.
- To move the process closer to the resources it is currently accessing.
- To move the process closer to the user.
- To enable us to keep a process going when the site it is executing on has to be taken down.

What do we need?

We need location and migration transparency of

processes

resources used by the process

What defines a process?

- PCB
- Resources
  - files
  - communication channels
  - memory
  - devices - including windows, keyboard, mouse
- Threads

Need some compatible machine (or virtual machine) architecture.

Internal and external reference problems

- References to resources within the program.
- References to the process from outside e.g. other processes communicating with it.

How can that be done?

Need a way of referring to all resources indirectly via global tables (like we did with our distributed file systems).

We can extend the ideas of a distributed file system to refer to other objects, including processes.

All process identifiers have no host information in the identifier.

e.g.

- A process table keeps track of which site each process is running on.
- When the process is moved the table is updated.
- Caching of information can be used for efficiency but we need ways to recover when the cache data is out of date.
- Not all processes need to be stored in this table.
  - Processes specific to a site which are not visible away from the site.

Doing the migration

Minimise the amount of down time

Process must be stopped at some stage

Stop, copy, notify

How much do we copy?

Only the working set
get the remaining pages by demand paging
Can't be used if the host is going down.

Everything, but don't stop the process
then copy pages which were dirtied during the copy

Both approaches only stop the process while the working set is moving.

Current uses

In reality process migration is not used for load balancing.

It is too expensive.

- Most processes only run for a few seconds.
- Transferring a process can easily take a few seconds.

It is still useful when a machine needs to be closed down for maintenance and it has running processes which we don't want to kill.
Common use - idle workstations
Move processes when no longer idle.

- Generally load balancing is only done when a process starts
- or when it has to move.
- Where is the best place to run this process?

The textbook also talks about Computation Migration – this means sending messages (or RPCs) to get work done on another site.

Before next time

Read from the textbook
8.1 – Background
8.3 – Contiguous Memory Allocation
8.5 – Paging
8.4 – Segmentation

Interesting read on Windows 8 memory management
http://arstechnica.com/information-technology/2011/10/how-windows-8s-memory-management-modifications-make-for-a-better-user-experience/