

Distributed Architectures

Info 341

1

Objectives

- What are basic aspects of an operating system?
- What is a system architecture?
- What are the characteristics of distributed systems?
- How does RPC work?
- Why is XDR necessary?
- What is a processor pool model?
- What are the tiers in a 3 tier model?
- What is the resource discovery problem?
- Multiprocessing, multithreading, distributed computing
- Examples from the real world :-)

2

Slight Digression

- Before we can really talk about distributed architectures and distributed systems ...
- Operating systems manage machine/system resources and mediate between the hardware and the applications

3

[Basic OS Concepts]

- Jobs/Process/Application/Thread
- Monitoring
- IO Buffering
- IO Spooling
- Time Sharing
- Multiprogramming/Multiprocessing
- MultiCore/MultiCPU/Hypterthreading
- IO Protection
- Memory Protection
- CPU Protection/System vs. User Space

4

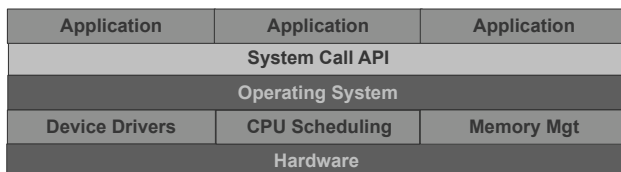
[OS Organization]

- OS Services
 - Program execution
 - Input/Output operation
 - File System Manipulation
 - Error Detection
- Services provided through OS system calls

5

[OS Design]

- Much like networking ... a layered model



6

[Some terms ...]

- procedure call
- parameter passing
- process
- thread
- daemon
- mutual exclusion
- semaphore

7

[Conceptual Background]

- What is a 'process'?

8

[Conceptual Background]

- What is a 'process'?
 - A process is a program that is running
 - What is necessary for a process?

9

Conceptual Background

- What is a 'process'?
 - A process is a program that is running
 - What is necessary for a process?
 - The state of execution of a program
 - Program Counter(s) (PC)
 - Register Values
 - Stack
 - Heap
 - Memory Page Set

10

Underlying Assumption

- The assumption underlying the entire discussion for today is the presence of a low latency, high bandwidth network.
- Originally, this could only be through a LAN, but now the concepts apply to the Internet in general.

11

What is an architecture?

- System Architecture
 - A description of the components and the relations among them
 - Hardware, Software, both
 - In a distributed system you have to have both

12

[The architectural views]

- Take three views of distributed systems
- Low level
 - Distributed function calls (RPC)
- Medium
 - Distributed system services
- High level
 - Applications as service
 - Multi-tier models
 - Peer-to-Peer

13

[Factors to consider]

- Three factors to consider (among others)
- Transparency
 - Does the system support seamless use of resources from the users perspective?
- Consistency
 - Does the system maintain consistent operation?
- Scalability
 - Does the system support small as well as large 'configurations' (system components, users)?

14

[Distributed Functions]

- Remote Procedure Calls (RPC)
 - Designed to facilitate writing client-server software
 - Generalization - many client-server applications have a similar structure
 - Abstraction - a procedure call to another machine
 - Quite low level, at the level of a function call
- Implementations
 - Sun - ONC (Open Network Computing)
 - OSF - DCE (Distributed Computing Environment)
 - Microsoft - Object RPC & COM/DCOM

15

General RPC Approach

- Programmer designs software as normal
- Considers the relation among the software components
- Determines which components are best organized remotely

16

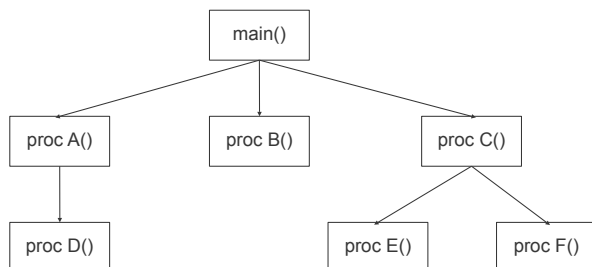
Software component relation

- Software has many possible relations
- Two critical relations
 - Procedural - the flow of control
 - Data - the flow of data
- Simple architectural descriptions are often called 'box and line' architectures

17

Procedure Call Graph

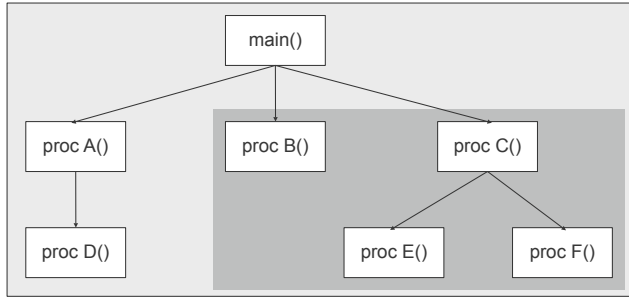
- Example box and line architecture
- Also known as a 'uses hierarchy' or 'uses graph'



18

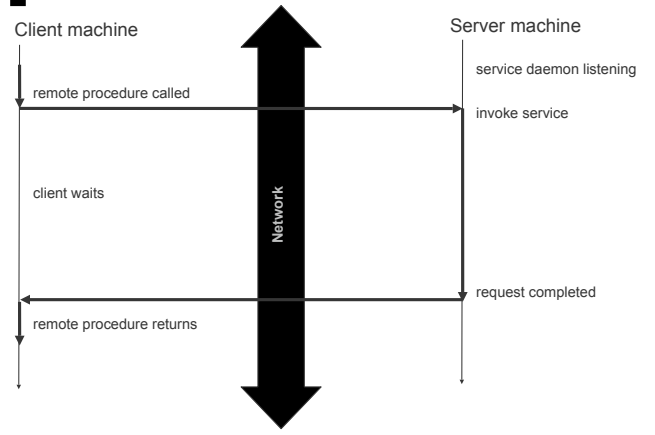
Partition the application

- Which parts of the application belong where
- Which parts are 'client' and which are 'server'?



19

How does this work?



20

Procedures still pass data...

- How is data passed between the client and server?
- Note there are architecture issues
 - Big Endian (PowerPC)
 - Little Endian (Intel)

21

[Procedures still pass data...]

- How is data passed between the client and server?
- XDR - eXternal Data Representation
 - htons(), htonl(), ntohs(), ntohl() is the concept, but way more complex
 - Imagine doing this for all possible data representations
 - Writing the data out on the network and reading it back in is called marshaling

22

[Considering RPC]

- Transparency
 - For the user quite transparent, you don't know where the server is
 - For the programmer, not transparent at all
- Consistency
 - High level of consistency
 - Semantics are of a procedure call, very clear
- Scalability
 - Not very scalable - mostly just a single client and single server, anything more gets unwieldy

23

[Distributed system services(*)]

- Level up from functional distribution
 - Service distribution
- The goal is to seamlessly share a large number of services
 - Many processors look like one
 - Share high performance print services
 - Share special peripherals

24

[Three models]

- Workstation/Server model
 - This should be familiar to you, much like our labs
- Processor pool model
 - Terminals connect to a collection of processors that look a single machine
- Integrated
 - Mixture of workstations and terminals where a processor pool handles larger tasks

25

[Workstation/Server Model]

- Each user provided with a workstation
- Processes run locally
- Generally for GUI tasks
- High-end peripherals are managed by remote dedicated servers
- Servers manage 'services' for the workstations

- Similar to the labs in the iSchool

26

[Consider the Workstation Model]

- Transparency
 - Nothing is transparent, you have to keep track of everything (e.g. are your files on this machine?)
- Consistency
 - Low level of consistency, software will not always run on each machine
- Scalability
 - Scalability is dependent on the servers

27

Processor Pool Model

- Users access the system through 'terminals'
- Processes are run remotely through a processor 'service'
- Heterogeneous equipment (processors) can be easily accommodated
- Execution can be load balanced
- High-end peripherals are still managed by a specific server

28

Consider Processor Pool Model

- Transparency
 - High process transparency
- Consistency
 - High level of consistency
- Scalability
 - Very scalable
- Why didn't this win?
 - Not at all good for GUI applications

29

Integrated model

- Machines are independent
- Global naming scheme
- Shared directory service (file service)
- Load/Login balancing
- Process migration
- High-end peripherals can be connected to a single machine and are seamlessly shared
- Similar to mead/vergil

30

Consider the Integrated Model

- Transparency
 - High transparency for processes
 - Low transparency for services
- Consistency
 - High level of consistency
- Scalability
 - Very scalable, but you need many workstations

31

Application Services

- As network technology improves
 - Transmission latency decreases
 - Reliability improves
- Things that could only be done in a LAN environment are now being done on the Internet at large
- This was the hope/promise of the Application Service Provider (ASP) dot com movement. An ASP was like an ISP but better/bigger.

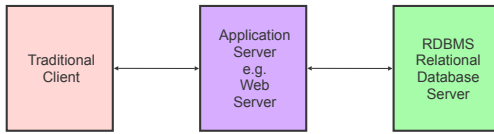
32

Multi-Tier Models

- Commonly we speak of client-server models
 - Client consumes some service
 - Server provides a service
- What happens when a server consumes a service?

33

Three-tier Model



- The logic of the application is spread across multiple servers
- Client handles user interaction (GUI)
- Server between client and back-end is known as the application server, it handles the 'business logic'
- The back-end server is often a relational database such as Oracle, MySQL, PostgreSQL et. al.

34

Considering Multi-Tier Models

- Transparency
 - Low transparency for users (need to know where)
 - Low transparency for application
- Consistency
 - High level of consistency
- Scalability
 - Very scalable, as one tier becomes loaded, add another server at that tier

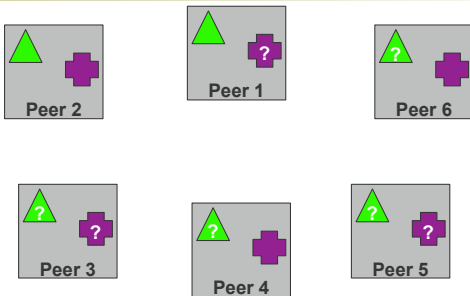
35

Peer-to-Peer Model

- In the basic peer-to-peer model every machine is both a client and a server
- Each peer can both produce and consume a service
 - Gnutella, Freenet, BitTorrent, Skype, Mixmaster remailers
- But this is not without problems ...

36

Peer-to-Peer Services



- How do peers find the services? Superpeers?
- How do we make sure that one peer is not overwhelmed?

37

Peer-to-Peer problems

- Resource Discovery
 - Services, data
- Load Balancing
 - Make sure a peer node is not overloaded
- Architecture (configuration of peers)
 - How to organize peers efficiently, ad-hoc
- Security, anonymity

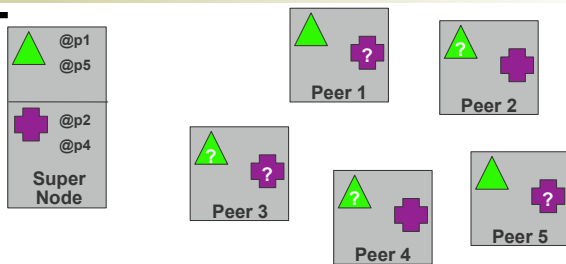
38

Hybrid Peer-to-Peer Model

- One attempt at solving the resource discovery problem...
- In a hybrid peer-to-peer model
 - Well known servers (super nodes) resolve the resource discovery problem for peers
 - Peers connect to well known servers then to each other

39

Peer-to-Peer Super Nodes



- The super nodes are well know
- Super nodes provide 'directory' service
 - Allow peer nodes to register available services and look up needed services

40

Consider Peer-to-Peer Models

- Transparency
 - Low transparency for services (resource discovery a problem) in pure model
 - High transparency in the hybrid model
- Consistency
 - Low consistency
- Scalability
 - Exceedingly scalable

41

Multiprocessing

- Two or more CPUs on the motherboard
- Same RAM, backplane, RAM, etc
- Symmetric multiprocessing OS
- Dual cores
- Hyperthreading
- Pipelining and parallelism
- FPU, AltiVec, MMX, SSE et al

42

[Multithreading]

- Process with multiple threads of execution
- Threads share process data structures
- Thread-safe vs non-thread-safe APIs
- Achieve concurrency on any computer
- Allows some threads to block as others continue processing in the background
- Java, Posix Threads (pthreads) API
- Ex Photoshop filters, QuickTime, Email clients
- Produces apps that are always responsive to the user even when doing processing

43

[Distributed Computing]

- More than one computer
- Networked
- Gigabit Ethernet and InfiniBand
- Cluster head allocates nodes to jobs
- OpenMosix, Beowulf, PVM, Grid
- ~ eCommerce sites (3-4 tiers)
- Load balancing (DNS, Cisco Localdirector)

44

[Distributed Computing Examples]

- [Folding@Home](#) from Stanford
- SETI@Home from Berkeley
- Virginia Tech G5 (Big Mac) Cluster
- Xcode from Apple Computer
- Rendering farms (ex Dreamworks)
- PVM (used with POV-Ray)
- P1 ("The Adolescence of P1" novel)
- Linux clusters and the top500.org
- Distributed Denial of Service attacks

45
