Distributed Systems

Distributed Systems

- Better scalability and performance
    - Apps need more resources than one computer has
- Improved reliability and availability
- Easy to use with reduced operating expenses
    - Centralized management of all services and systems
    - Buy services instead of hardware
- Enabling new collaboration and business models
    - Collaborations that span system boundaries

Problems with DS

- Different machines don't share memory
    - Machines can't easily know the state of another
- Only way to interact remotely is to use a network
    - Usually async, slow, and error prone
    - Not controlled by any single machine
- Failures of one machine aren't visible to other machines

Transparency

- Ideally a distributed system would be like a single machine where each machine knows what other machines are doing
- TTransparent systems look as much like a single machine system as possible

Deustch's Seven Fallacies of Network Computing

1. Network is reliable
2. No latency
3. Available bandwidth is infinite
4. Network is secure
5. Topology of the network doesn't change
6. One administrator for the whole network
7. No cost of transporting additional data

- True transparency is impossible to achieve

Distributed System Paradigms

- Parallel processing
    - Relies on a few tightly coupled special hardware
- Single system images
    - Makes all node look like one large computer
- Loosely coupled systems
- Cloud computing
    - A highly specialized system for a specific purpose

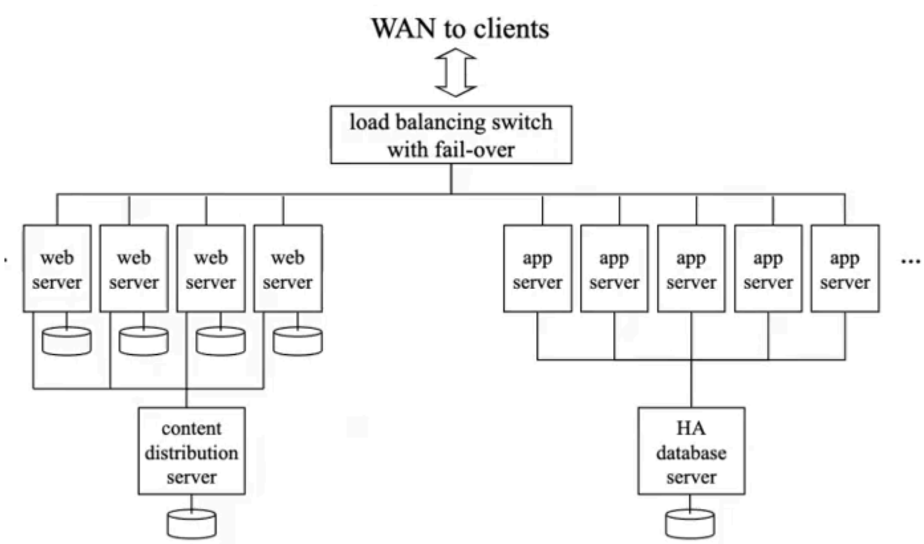
Loosely Coupled Systems

- Characteristics

- A parallel group of independent computers
- Connected by high speed LAN
- Serving similar but independent requests
- Minimal coordination and cooperation required

- Motivation
  - Scalability
  - Availability
  - Ease of management
  - Helps with web servers and app servers

- Horizontal Scalability
  - Each node is independent, and adding additional nodes is easy
  - Scalability is limited by the network instead of hardware or algorithms
  - High reliability, since one node can fail but the others continue running



- Load balancer can split the requests to different web servers

Elements of Loosely Coupled Architecture

- Group of independent servers
  - Servers run the same software and serve different requests
  - May share a common database
- Front end switch
  - Distributes incoming requests to different servers
  - Can do both load balancing and fail over
- Service protocol
  - Stateless servers and idempotent operations (applying an operation multiple times doesn't change the initial application)
  - Successive requests may be sent to different servers

Horizontally Scaled Performance

- Individual servers are inexpensive
- Good scalability
- Good service availability
- Challenge is managing thousands of servers
    - Automated installation and configuration
    - Self monitoring and self healing (can automatically detect and fix crashes)
    - Limited by the management of machines, not the hardware or the algorithm

Cloud Computing

- Tools that support particular kinds of parallel processing
- User does not need to be an expert at distributed systems

Map Reduce

- Single function that needs to be performed a lot
    - Such as searching for a particular string
- Divide the data into disjoint pieces
- Perform the function on each piece on a separate node (map)
- Combine the results to obtain output (reduce)

Reduce

- We might have 2 nodes assigned to doing the reduce operation
- They receive a share of data from the map node
- Reduce node performs a reduce operation to combine the shares, and outputs the final result

Synchronization in MapReduce

- Each map node produces an output file for each reduce node
- Produced atomically
- Reduce node only begins reducing when the entire output file is produced
- Forcing a synchronization point between map and reduce phases

Cloud Computing and Horizontal Scaling

- Rent some cloud nodes to the servers
- If the laid gets heavy, ask the cloud provider for another node
- If the load is reduced, release unneeded nodes

Remote Procedure Calls (RPC)

- One way of building a distributed program, procedure calls on a remote computer or server
- Procedure calls are used as an interface (function names)
- Natural boundary between the client and server

Limitations of RPC

- No implicit parameters or returns
- No call by reference parameters
- Slower than procedure calls

RPC Components

- Interface Specification
    - Methods, parameter types, return types
- eXternal Data Representation (XDR)
    - Machine independent data type representations
    - May have optimizations for similar client/server
    - RPC always sends in external type, then client has to convert it
- Client stub
    - Client side proxy for a method in the API
    - Sends messages, converts to external data types
- Server stub
    - Server side recipient of the API invocation
    - Converts the client request to external data rep

Features of RPC

- Client app links against procedures
    - Calls local procedures
- All RPC implementation is inside those procedures

RPC process

1. Request sent by the client daemon
2. Server daemon calls the correct RPC
3. Server daemon returns the response

RPC Considerations

- Requires a client server binding model where there's a live connection
    - Request from client needs to be instantly received and processed
- Threading model
    - Multiple requests can be serviced by worker threads
- Limited failure handling
    - Client arranges for timeout and recovery
- Limited consistency support
    - Only between calling client and called server
- Limited consistency support
    - Only between calling client and called server
- Higher level abstractions improve RPC

Distributed Synchronization

- Spatial separation
  - Different processes run on different systems
  - No shared memory for atomic locks
  - Controlled by different operating systems

- Temporal separation
  - Can't totally order spatially separated events

Leases

- Robust locks
- Obtained from the resource manager
  - Gives the client exclusive right to update the file
  - Lease "cookie" must be passed to server on update
  - Lease can be released at end of critical section
- Only valid for a limited period of time
  - Lease expires after a fixed time frame
- Handles wide range of failures
  - Process, client node, server node, network

Lock Breaking and Recovery

- Revoking an expired lease is easy, based on server clock
- Makes it safe to issue new leases
- Object must be restored to last good state if there is a crash
  - Roll back to state prior to aborted lease
  - Implement all or none transaction

Distributed Consensus

- Achieving simultaneous, unanimous agreement
  - Even in the presence of node and network failures
  - Required: agreement, termination, validity, integrity
  - Desired: bounded time
  - Provably impossible in fully general case
- Reduce the amount consensus algorithms, which tend to be complex and takes time to converge
- Tend to be used more sparingly
  - Make a node a leader, and only the leader runs the consensus

Consensus Algorithm

1. Interested member broadcasts his nomination
2. All parties evaluate the received proposals according to a <u>fixed and well known</u> rule
3. After allowing a reasonable time for proposals, each voter acknowledges the bes proposal

4. If a proposal has majority vote, the proposing member broadcasts a claim that the question is resolved
5. Each party agrees with the winner claim

Distributed Security

- OS cannot guarantee privacy and integrity
  - Network activities happen outside the OS
- Authentication is harder
  - All possible agents may not be in the local password file
- Wire connecting the user to the system is insecure
- Internet is vulnerable

Goals of Network Security

- Secure conversations
  - Privacy: only you and receiver know the message
  - Integrity: nobody can tamper with the message
- Positive identification of both sides
  - Authentication of the identity of message sender
  - Assurance the message isn't a replay or forage
  - No non repudiation where the other side says they didn't send it
- Availability
  - Network and other nodes must be reachable when it's needed

Elements of Network Security

- Cryptography
  - Symmetric cryptography for protecting bulk transport of data
  - Public key cryptography primarily for authentication
  - Hashes used to detect message alterations
- Digital signatures and public key certificates
- Filtering technologies such as firewalls

Tamper Detection: Cryptographic Hashes

- Check sums used to detect data corruption
  - Add up all bytes in a block, send sum along with the data
  - Recipient adds up the bytes and checks the sums
- Crypto hashes are stronger
  - Unique: two messages unlikely to produce the same hash

Cryptographic Hashes

- Encrypt the hash (much less expensive than encrypting the data)

1. Compute a cryptographic hash for that message
2. Securely transmit the hash
3. Receiver does the same computation on received text
   a. If both hash results agree, message is intact

b. If not, message has been compromised

Secret Socket Layer (SSL)

- General solution for securing network communication
- Built on top of existing socket IPC
- Establishes secure link between two parties
  - Privacy: nobody can snoop on conversation
  - Integrity: nobody can generate fake messages
- Public key used to distribute a symmetric session key
  - Ex. Amazon has a known PK, which can be verified
- Certificate based authentication of server
- Optional certificate based authentication of client
  - Server requires authentication and non repudiation
- Data transport switches to symmetric crypto
  - Giving safety of public key and efficiency of symmetric

Digital Signatures

- Encrypting a message with private key signs it
  - Only you could have encrypted it, therefore it must be from you
- Cannot be tampered with after you wrote it
- Encrypting everything with private key is not a good idea
  - Asymmetric encryption is slow
- Only care about integrity
  - Compute a cryptographic hash of the message
  - Encrypt the hash with the private key
  - This is much faster than encrypting the entire message

Digital Signature Process

1. Message needs to be sent
2. Cryptographic hash and asymmetric encryption (own private key) used to generate digital signature
3. Receiver decrypts the digital signature using the sender's public key
4. Same cryptographic algorithm run on the message to get the hash, which is compared

- Web browser contains certificates and public keys

Signed Load Modules

- Load modules send with encrypted hash
- Designates a certification authority