Carnegie Mellon

isr institute for SOFTWARE RESEARCH

Deployment and Operations for Software Engineers 2nd Ed

Chapter 4 – The Cloud

Outline

- **Structure**
- Failure
- Scaling
- State management
- Sharing distributed data
- Security policies

Cloud characteristics

- A public cloud is available over the internet to anyone who can pay for what they use.
- A private cloud is owned by a single organization and usage is restricted to members of that organization.
- The public cloud has several characteristics
  - You pay only for what you use.
  - The storage and computation services are *elastic*, meaning that it can grow or shrink as your needs change.
  - Your use of the cloud is *self-provisioned*: You create an account and can immediately begin using it.

Data center

- Contains 50,000+ computers
- Independent power supply, security, air conditioning, etc.
- Computers are stored in racks.
- Computers are, mostly, commodity.

Hierarchical data center collections

- Multiple geographically related data centers comprise an availability zone.
- Multiple availability zones compromise a region.
- Cloud providers maintain multiple regions around the globe.
  - North America
  - South America
  - Europe
  - Asia-Pacific
  - South Africa

Allocating new VM in cloud

- You request a VM and specify
  - Region
  - Availability zone
  - Type of VM
  - VM image
  - Security settings
  - Other parameters.
- Cloud provider finds physical computer with spare capacity in the specified availability zone in the specified region.
- Hypervisor in that physical computer creates new VM, loads image, and returns IP address.

Service types

- Infrastructure as a Service (Iaas). Provides CPUs, storage, and networks.
- Platform as a Service (PaaS). Provides a collection of services such as storage services, hardware management services, compilers, language libraries.
- Software as a Service (SaaS). Total applications available from cloud providers such as email, customer management, invoicing, etc.

Discussion questions

1. What region are you in for AWS, Google, and Azure? How many availability zones for each provider in your region.
2. Why are commodity computers used in cloud providers data centers?
3. What are parameters in addition to the ones listed must be specified to allocate a VM?

Outline

- Structure
- **Failure**
- Scaling
- State management
- Sharing distributed data
- Security policies

Sometimes the whole cloud fails

Search for “cloud outages” to get enumerations of cloud failures

Outages affect all major cloud providers.

And sometimes just a part of it fails …

A year in the life of a Google datacenter

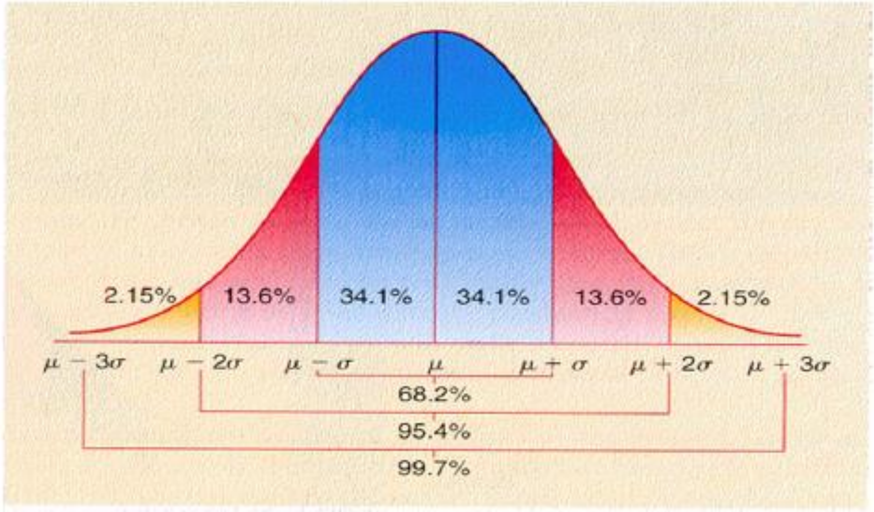
- ~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
- ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
- ~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
- ~5 racks go wonky (40-80 machines see 50% packetloss)
- ~8 network maintenances (4 might cause ~30-minute random connectivity losses)
- ~12 router reloads (takes out DNS and external vips for a couple minutes)
- ~3 router failures (have to immediately pull traffic for an hour)
- ~dozens of minor 30-second blips for dns
- ~1000 individual machine failures
- ~thousands of hard drive failures
- slow disks, bad memory, misconfigured machines, flaky machines, dead horses,

Amazon failure statistics

- In a data center with ~64,000 servers, each with two disks
- On a typical day
  - ~5 servers fail
  - ~17 disks fail

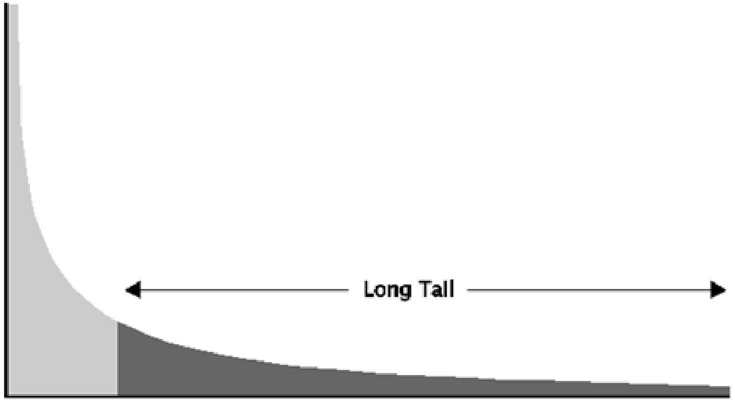
Normal distribution

- A distribution describes the probability than any given reading will have a particular value.
- Many phenomenon in nature are "normally distributed".
- Most values will cluster around the mean with progressively smaller numbers of values going toward the edges.
- In a normal distribution the mean is equal to the median



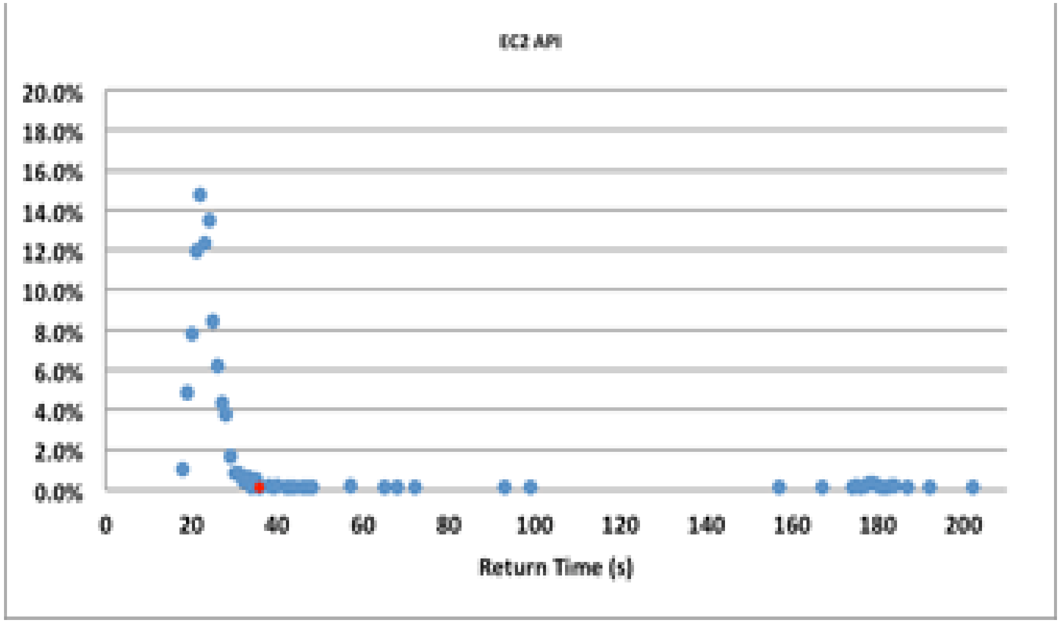
Long tail distribution

- In a long tail distribution, there are some values far from the median.
- These values are sufficient to influence the mean. The mean and the median are dramatically different in a long tail distribution.



What does this mean?

- If there is a partial failure of the cloud some activities will take a long time to complete and exhibit a long tail.
- The figure shows distribution of 1000 AWS "launch instance" calls.
- 4.5% of calls were "long tail"



You must be aware of the possibility of a long tail distribution

- Long tail latencies are a result of congestion or failure somewhere in the path of the service request.
- There are many possible contributors to congestion–server queues, hypervisor scheduling, or others–but the cause of the congestion is out of your control as a service developer.
- Your monitoring techniques and your programming techniques must reflect the possibility of a long tail distribution.

Discussion questions

1. Has there been a cloud outage in your region recently? What was the cause?
2. The Google failure statistics talk about a “rack failure”. What is a rack failure?

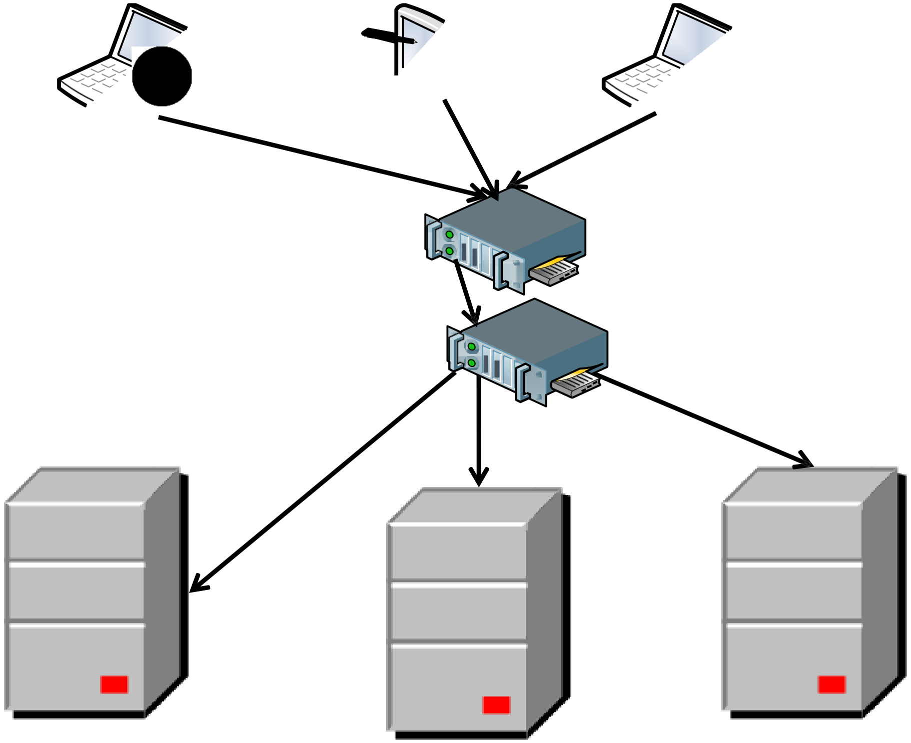
Outline

- Structure
- Failure
- **Scaling**
- State management
- Sharing distributed data
- Security policies

Load balancer

- One server may not suffice for all of the requests for a given service.
- Have multiple servers supplying the same service.
- Use “load balancer” to distribute requests.
- Server is registered with load balancer upon initialization
- Load balancer monitors health of servers and knows which ones are healthy.
- All servers managed by one load balancer are identical
- Load balancer IP is returned from DNS server when client requests URL of service.

Message sequence client makes a request



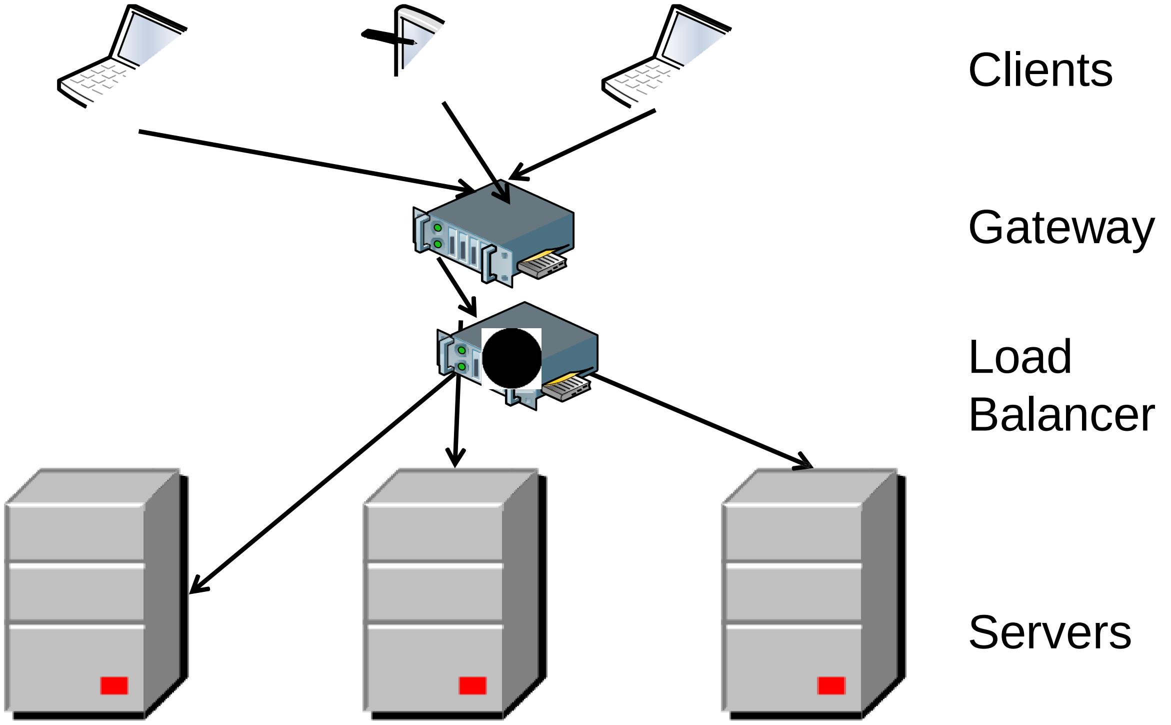
Clients

Gateway

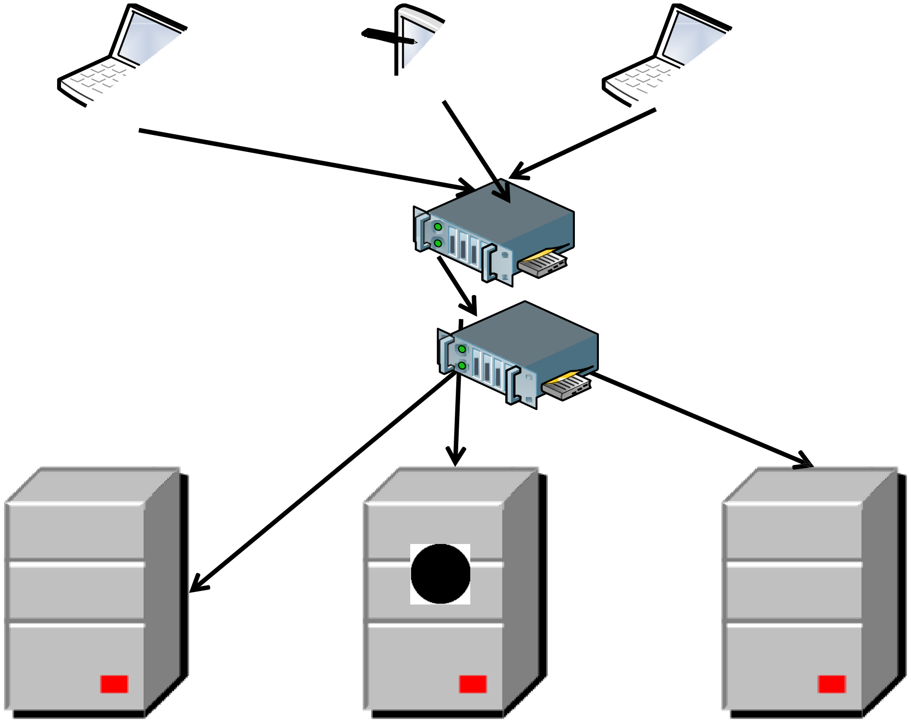
Load Balancer

Servers

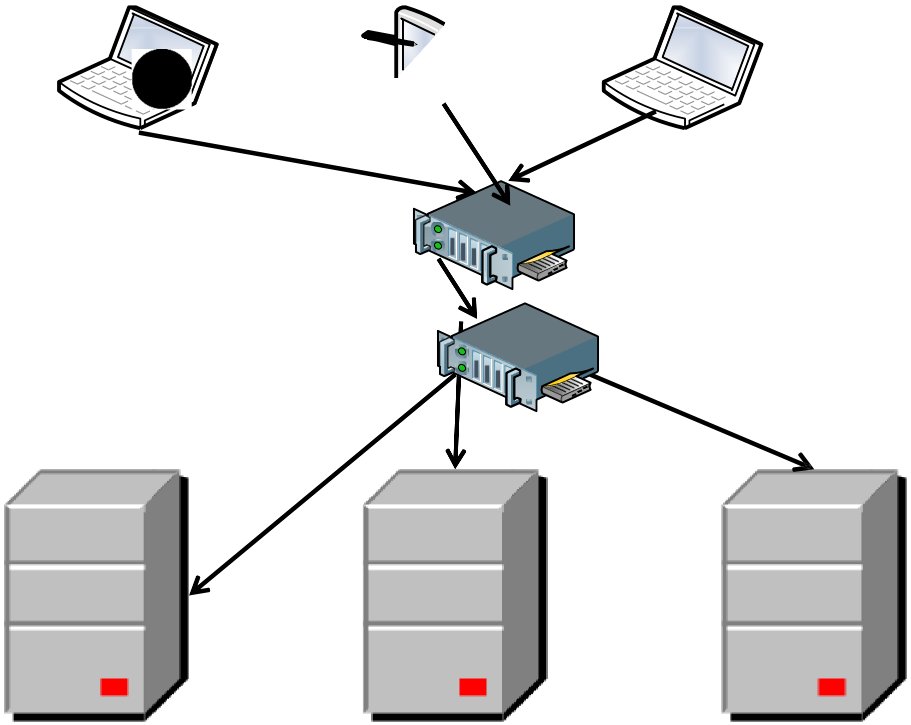
Message sequence message goes through gateway and then to load balancer



Message sequence message is sent to one server



Message sequence reply goes to gateway and then back to client



Clients

Gateway

Load
Balancer

Servers

Note IP manipulation

- Client sends message to load balancer.
- Gateway intercepts message and changes source to be its IP
- Gateway sends message to load balancer
- Load balancer changes destination to be a server.
- Load balancer changes destination IP but not the source.
- Server always sends message back to what it thinks is the source. In this example, it is the gateway. The reply goes directly back to gateway.
- Gateway sends reply to client.

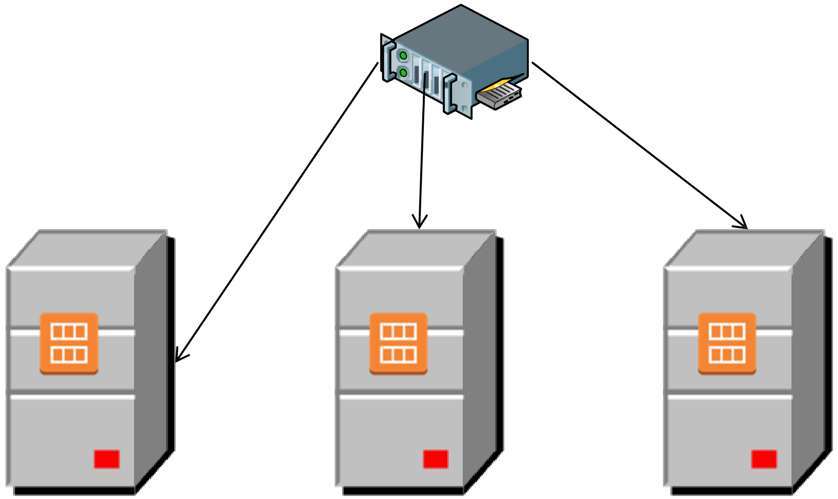
Routing algorithms

- Load balancers use variety of algorithms to choose instance for message
- Round robin. Rotate requests evenly
- Weighted round robin. Rotate requests according to some weighting.
- Hash IP address of source to determine instance. Means that a request from a particular client always sent to same instance. This is not recommended.
- Note that these algorithms do not require knowledge of an instance’s load.

Suppose servers are overloaded

- As load grows, existing resources may not be sufficient.
- Autoscaling is a mechanism for creating new instances of a server.
- Set up a collection of rules that determine
  - Under what conditions are new servers added
  - Under what conditions are servers deleted

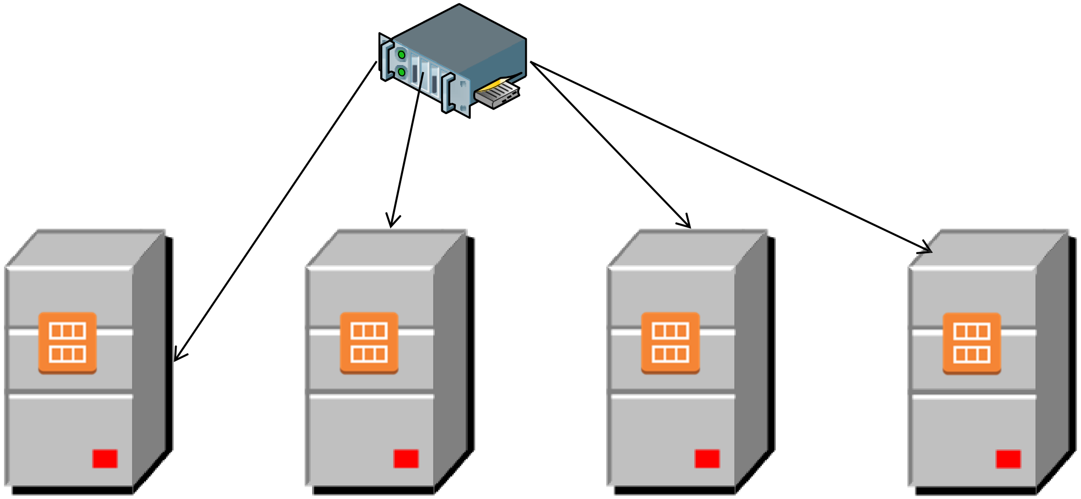
First there were three servers



Load Balancer

Servers

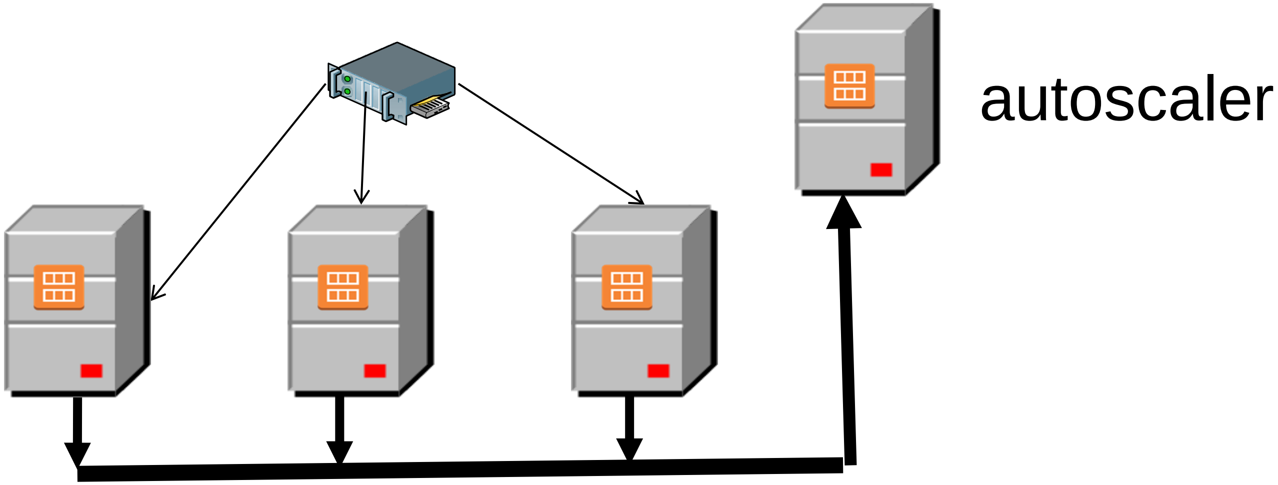
Now there are four



Issues:

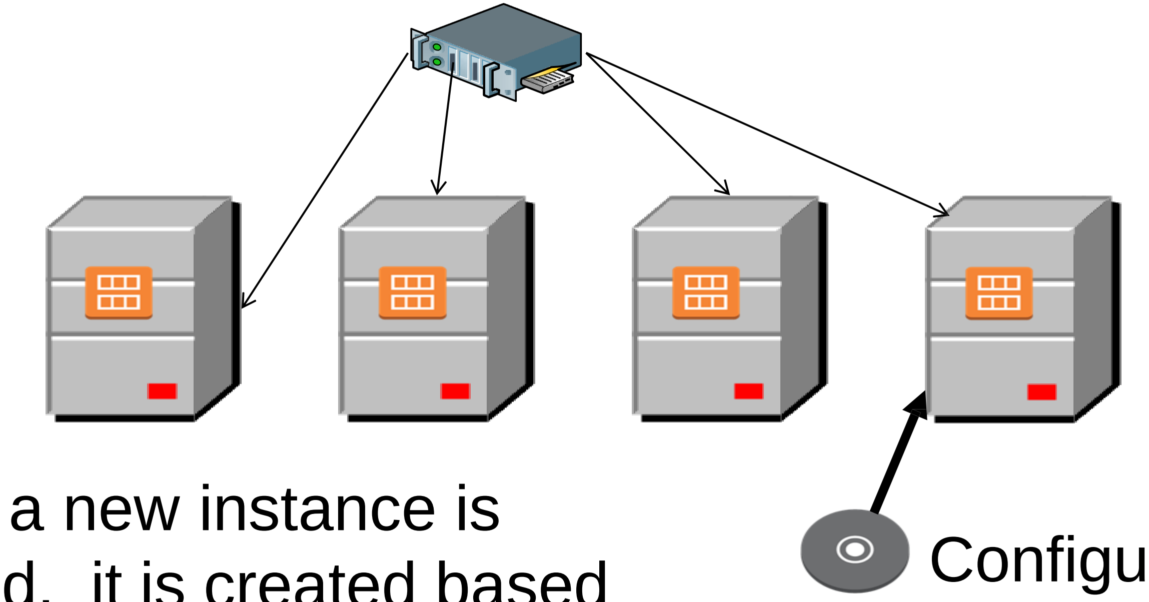
- What makes the decision to add a new server?
- How does the new server get loaded with software?
- How does the load balancer know about the new server?

Making the decision



- Each server reports its CPU and I/O usage to an autoscaler
- Autoscaler has a collection of rules to determine whether to add new server. E.g. one server is over 80% utilization for 15 minutes

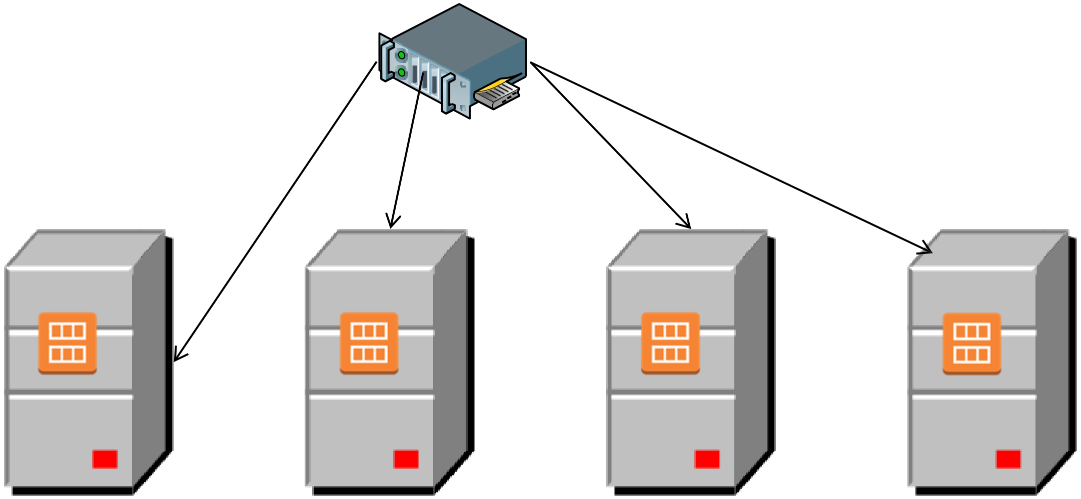
Loading the new server with software



When a new instance is created, it is created based on a launch configuration, security groups, etc

Configuration specification

Making the load balancer new server aware



The new server is registered with a load balancer by the autoscaler

Managing instance failure

- Replies from server may not go through load balancer.
- It must have another mechanism to determine health of servers.
- Each server sends a “I am alive” message periodically.
- This tells load balancer the server is healthy.
- If the server is unhealthy, the load balancer will not send it any more messages.
- If the server fails, the autoscaler will detect failure and create new instance, depending on its rules.

Discussion questions

1. Why is not recommended to use hashing as a load balancer distribution algorithm.
2. How does autoscaler know the launch configuration to load new server with software?
3. Suppose load balancer fails. How is that handled?
4. Suppose autoscaler fails. How is that handled?

Outline

- Structure
- Failure
- Scaling
- **State management**
- Sharing distributed data
- Security policies

How state is managed is important

- For both performance and logical reasons.

Performance numbers

- Transferring data times:

Read 1,000,000 bytes from memory	47 µsec
Round trip in same data center	~500 µsec
Read 1,000,000 bytes from SSD	~784 µsec
Read 1,000,000 bytes from disk	~10msec
Packet round trip Calif <-> Netherlands	150 msec

- Storing state in memory is faster than storing it on disk
- Sending state to same data center is faster than disk
- Sending state across data centers is relatively slow.

Logical consequences of state management

- Assume two clients and two instances of a server.
- Goal is to count number of requests from clients to servers.
- Three options:
  1. Store count in clients
  2. Store count in servers
  3. Store count on database
- Option 1 counts how many times each client calls any server
- Option 2 counts how many times each server is called
- Option 3 counts how many calls were made by both clients together.

Discussion questions

1. Code the three options in counting example and verify results are correct.
2. Time message round trip in same data center. Is the number given correct?

Outline

- Structure
- Failure
- Scaling
- State management
- **Sharing distributed data**
- Security policies

Sharing data

- Many services in a distributed system want to share data with each other.
- Use cases:
  - Locks
  - Configuration information
  - Discovery
  - Sequencing tasks
  - …

Options for sharing data

- Persistent database.
  - Takes time. See Performance numbers on State Management slide.
  - Suitable for large data sets.
- Distributed caching system such as Memcached.
  - In memory and fast.
  - Suitable for small data sets
  - Failure will cause data loss.
- Distributed coordination systems
  - In memory and fast
  - Suitable for small data sets
  - High availability
    - Complicated.

Memcached

- Key value store
- Limited to 1Mbyte
- Multiple servers that do not communicate with each other.
- Each client stores data in a server
  - Data is not shared across servers.
  - Clients with which data is shared must access same server
- Use cases
  - Session data
  - Web page caching
  - API caching
  - Caching of objects such as images, files, and metadata

Distributed coordination system

- Fault tolerant with high availability
- Manages coordination among clients.
- Operations appear atomic
- Multiple open source implementations
  - Zookeeper
  - etcd
  - Consul
- Use cases
  - Distributed locks
  - Service discovery
  - Health checks
  - Schedulers

Locks

- We will use locks to understand more about distributed coordination systems.
- Locks give the appearance of atomicity to clients.
- Critical resources are locked to prevent race conditions.
- A lock is a software construct.
- Associated with each resource a lock bit.
  - If bit is 0, no lock on resource
  - If bit is 1, resource is locked
- In practice, locks have other information in addition to lock bit.

Problems with locks

- Can lead to deadlock – two processes waiting for each other to release critical resources
  - Process one gets a lock on row 1 of a data base
  - Process two gets a lock on row 2.
  - Process one waits for process 2 to release its lock on row 2
  - Process two waits for process 1 to release its lock on row 1
  - No progress.

More problems with locks

- Getting a lock across distributed systems is not an atomic operation.
- It is possible that while requesting a lock another process can acquire the lock. This can go on for a long time (it is called livelock if there is no possibility of ever acquiring a lock)
- Suppose the virtual machine holding the lock fails. Then the owner of the lock can never release it.

Locks in a distributed system

- Utilizing locks in a distributed system has a more fundamental problem.
- It takes time to send a message (create a lock, for example) from one computer to another.

Synchronizing data

- The general problem is that you want to manage synchronization of data across a distributed set of servers where up to half of the servers can fail.
- Paxos is a family of algorithms that use consensus to manage concurrency. Complicated and difficult to implement.
- An example of the implementation difficulty
  - Choose one server as the leader which keeps the “authoritative” state.
  - Now that server fails. Need to
    - Find a new leader
    - Make sure it is up to date with the authoritative state.

Zookeeper as an example

- Zookeeper provides a guaranteed consistent (mostly) data structure for every instance of a distributed system.
- Definition of “mostly” is within eventual consistency lag (but this is small).
- Zookeeper keeps data in memory. This is why it is fast.
- Limited to 1 Mbyte
- Zookeeper deals with managing failure as well as consistency.
- Done using consensus algorithm.

Zookeeper API

- All calls return atomic views of state – either succeed or fail. No partial state returned. Writes also are atomic. Either succeed or fail. If they fail, no side effects.

Function	Type
Create	Write
Delete	Write
Exists	Write
Get children	Read
Get data	Read
Set data	Write

Lock creation

- Locks have names.
  - Client N
    - Create Lock 1
    - If success then client owns lock.
    - If failure, then it is placed on waitlist for lock 1 and control is returned to Client N
  - When owner of Lock 1 finishes, it deletes Lock1
  - If there is a waitlist, then those clients are informed about Lock 1 deletion and they try again to create Lock 1
  - If Client N fails, Zookeeper will delete Lock 1 and waitlist clients are informed of deletion

Discussion questions

1. How does Zookeeper check the health of its clients?
2. What happens if one of the servers (not the leader) in Zookeeper fails?
3. Compare Memcached and Redis.

Outline

- Structure
- Failure
- Scaling
- State management
- Sharing distributed data
- **Security policies**

Security policies

- A security policy is a set of rules that govern which accounts or services have access to which resources.
- The cloud service that provides access control is called Identity and Access Management (IAM).
- IAM rules are checked by invoking the IAM service when an API call is made to a cloud service.
- Amazon reports over 400 million IAM calls per second.

Discussion questions

1. Many coud users do not use IAM services. Why not?
2. How does a client identify itself to an IAM service?