Grades Statistics

- Midterm 1: average 9.94 (out of 15) with standard deviation 1.33
  - Many thanks to students who submitted question suggestions
  - Grading rules for multi-answer questions: (summation of partial point for each correct choice) - (summation of partial point for each incorrect choice)
  - Midterm1 scores will be final after Wednesday 10/21
  - Poll about whether we should have short answer questions in final exam
- Total points so far: average 33.68 (out of 40) with standard deviation 1.73
  - The grading so far: [35.41, 48]: A; [31.95, 35.41): B; [0, 31.95): C or worse
- Please talk to me if you feel difficulty in learning
- Try to get full points for your case-study, remaining homework/exercises

Case Study Notes

- The goal of case study
  - Learn the latest techniques in distributed systems
  - know and collaborate with other team members
  - Learn from other teams
- Process
  - Select topic: talk to your team members on which topic to work on after class
  - Inform your selection: post your topic on piazza (case-study folder) after class. New topics could be proposed.
  - Search and select paper/project you want to work on: https://scholar.google.com/, http://www.apache.org/, https://github.com, university library website, etc.
  - If one topic has more than one teams. Some coordination will be needed to make sure the team will work on different papers/projects.
  - Send me a piazza post (case-study folder) and get my approval. You could attach files in your post using menu item: Insert->Insert file
  - Work as a team
  - Present as a team on week 15 (12/03): every member needs to present
- Case Study Presentation Grading Rubric

Case Study Topics

	Team	Topic
1		NoSQL/NewSQL Database
2		Parallel Computing
3		Distributed File System
4		Peer-to-peer (P2P) computing
5		Micro Services
6		Cloud computing
7		Big Data
		…

Discussion #5

- A weather Web service maintains the current weather information which gives different results for the same place when you invoke it at different times. Whether it is a stateless or stateful service?
  - This Web service is a stateless service because the execution result doesn't rely on previous executions requested by the same client.
  - Stateless service is about execution independence.
  - The Web service could keep the weather state in database or external resources. So stateless doesn't mean no state at server side.

IS 651: Distributed Systems
Chapter 8: Distributed Systems Basics

Jianwu Wang

Fall 2020

Learning Outcomes

- After learning this chapter, you should be able to
  - Understand each technique (why we need it, how it works)
  - Understand differences between similar techniques

Distributed Systems Basics

- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Caching

- Caching is an optimization for distributed systems that reduces latency and decreases network traffic
- There are two categories of this kind of network-based cache:
  - Web caching
  - Application caching

Web Caching

- There are four locations for web cache:
  - Web browser cache (see figure)
  - Forward proxy cache
  - Open proxy cache
  - Reverse proxy cache
- A proxy (server) acts as an **intermediary** for requests from clients seeking resources from other servers

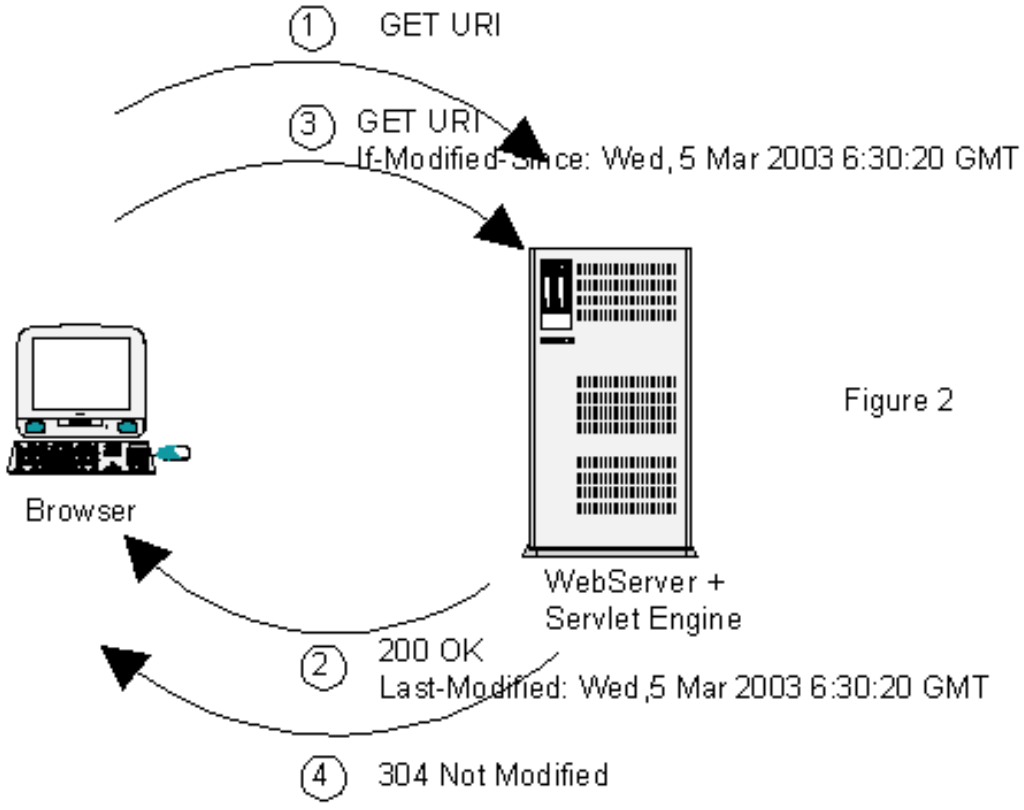
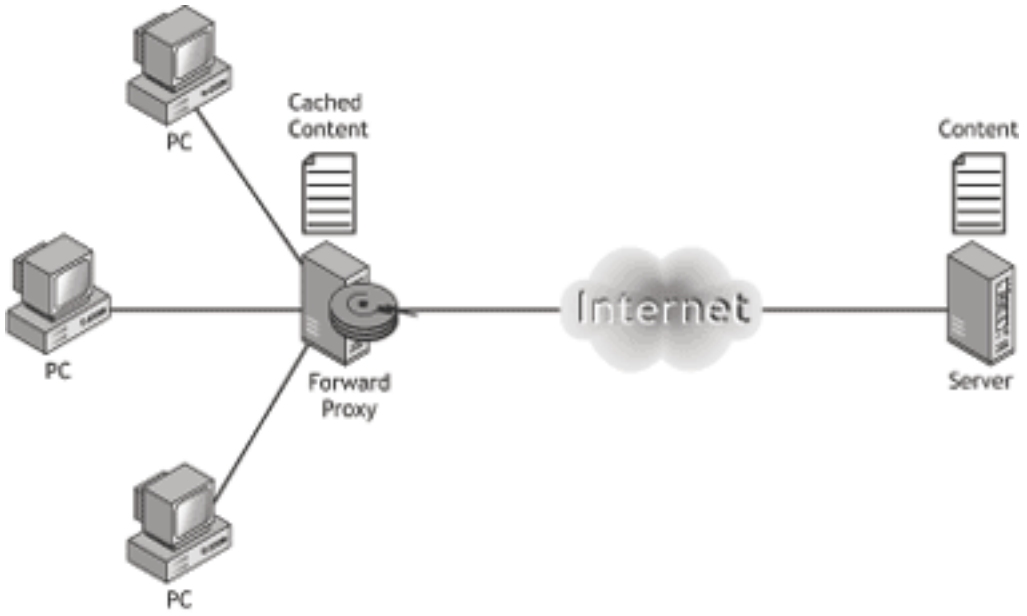


Figure 2

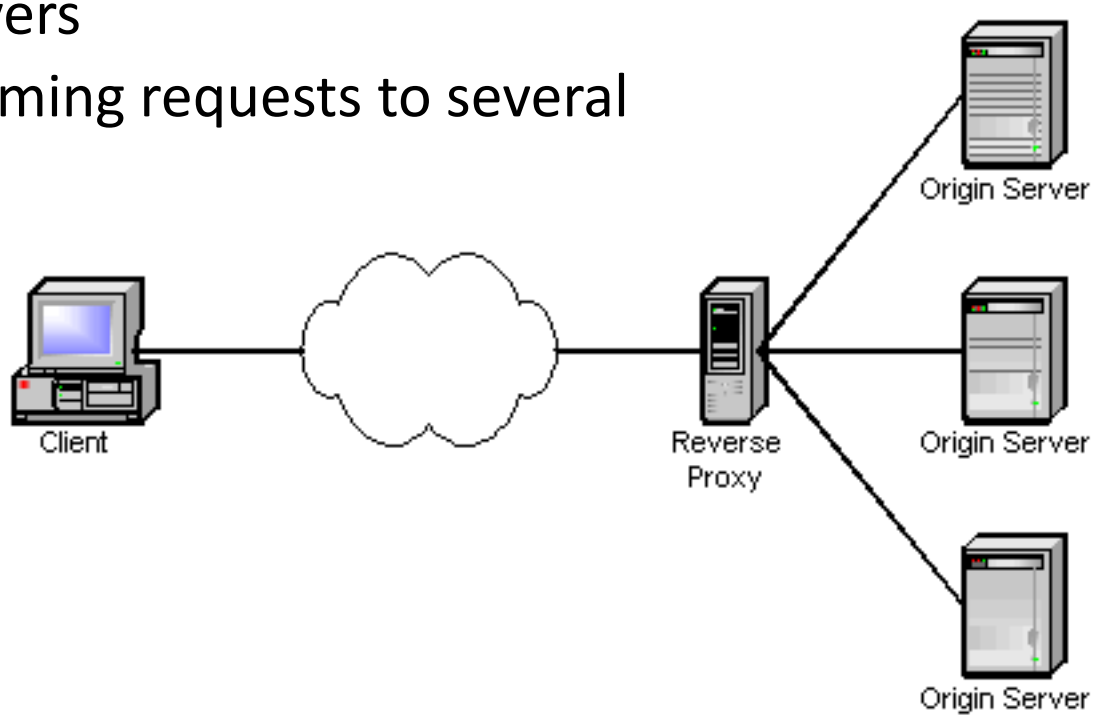
Forward Proxy Cache

- A forward proxy cache is located at the organization (as at UMBC) or at the internet service provider (ISP)
- Two approaches used by forward proxy cache
  - Configure the browser
  - Interception caching



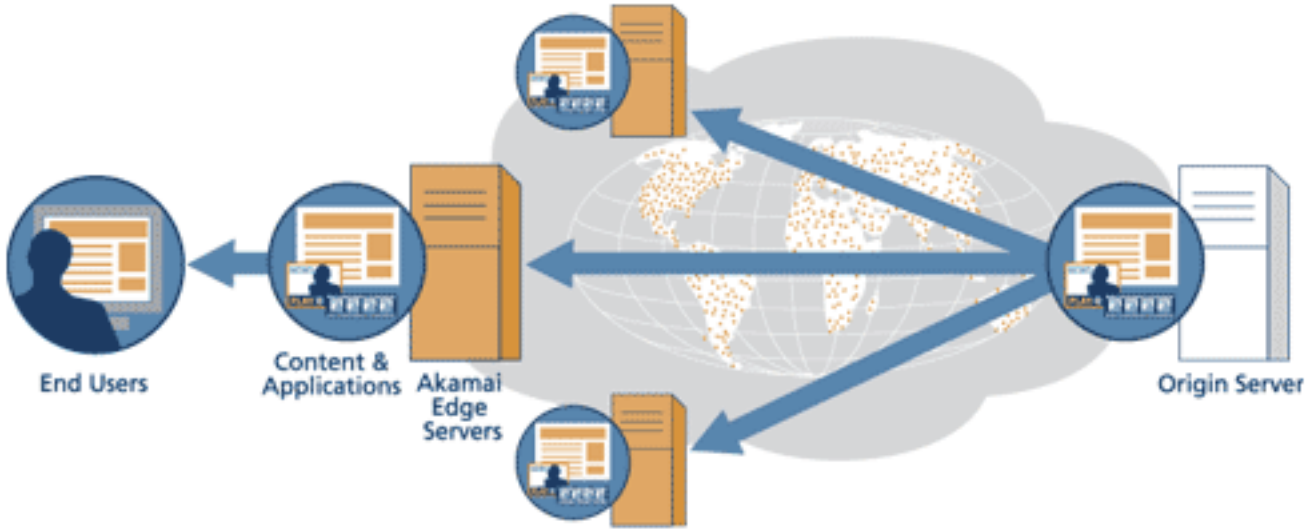
Reverse Proxy Cache

- A reverse proxy cache is on the internal network of the server
  - Can reduce load on its origin servers
  - Can distribute the load from incoming requests to several servers (see load-balancing)



Content Delivery Network (CDN)

- As a type of reverse caching, this improves data access by locating content in various places on the Internet in order to get it closer to clients
  - Example:



Cache Info in HTTP Response Headers

- Date: response time
- Cache-Control: directives that MUST be obeyed
- Expires: how long the cache should be kept before the cache refreshes
- Last-Modified
- ETag (Entity Tags): a short unique identifier that the server generates for each object such as a web page

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

Application Caching

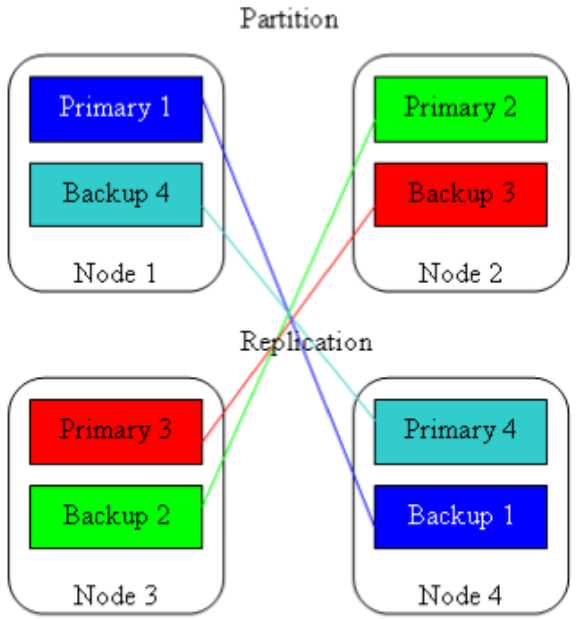
- Application caching is caching that is managed by the application itself to improve performance rather than the web and Internet infrastructure

Distributed Systems Basics

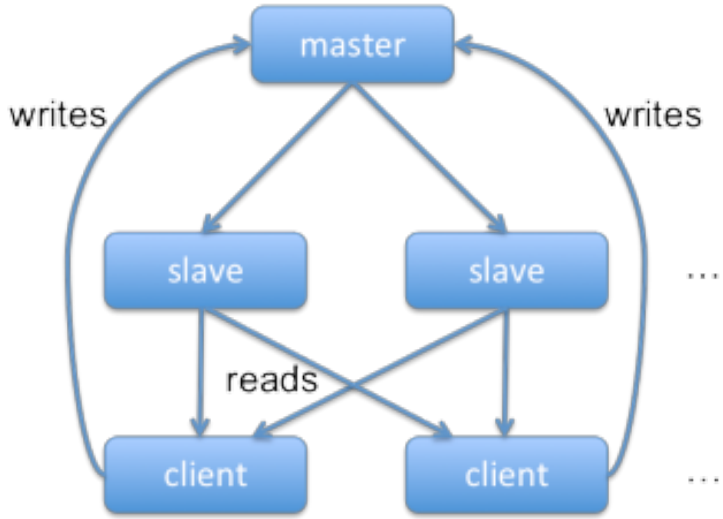
- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Replication

- General optimization on any network for scalability and fault-tolerance
  - making copies of information on different nodes on a network
  - having consistency mechanism between the replicas
- Eventual consistency



Partition and replication



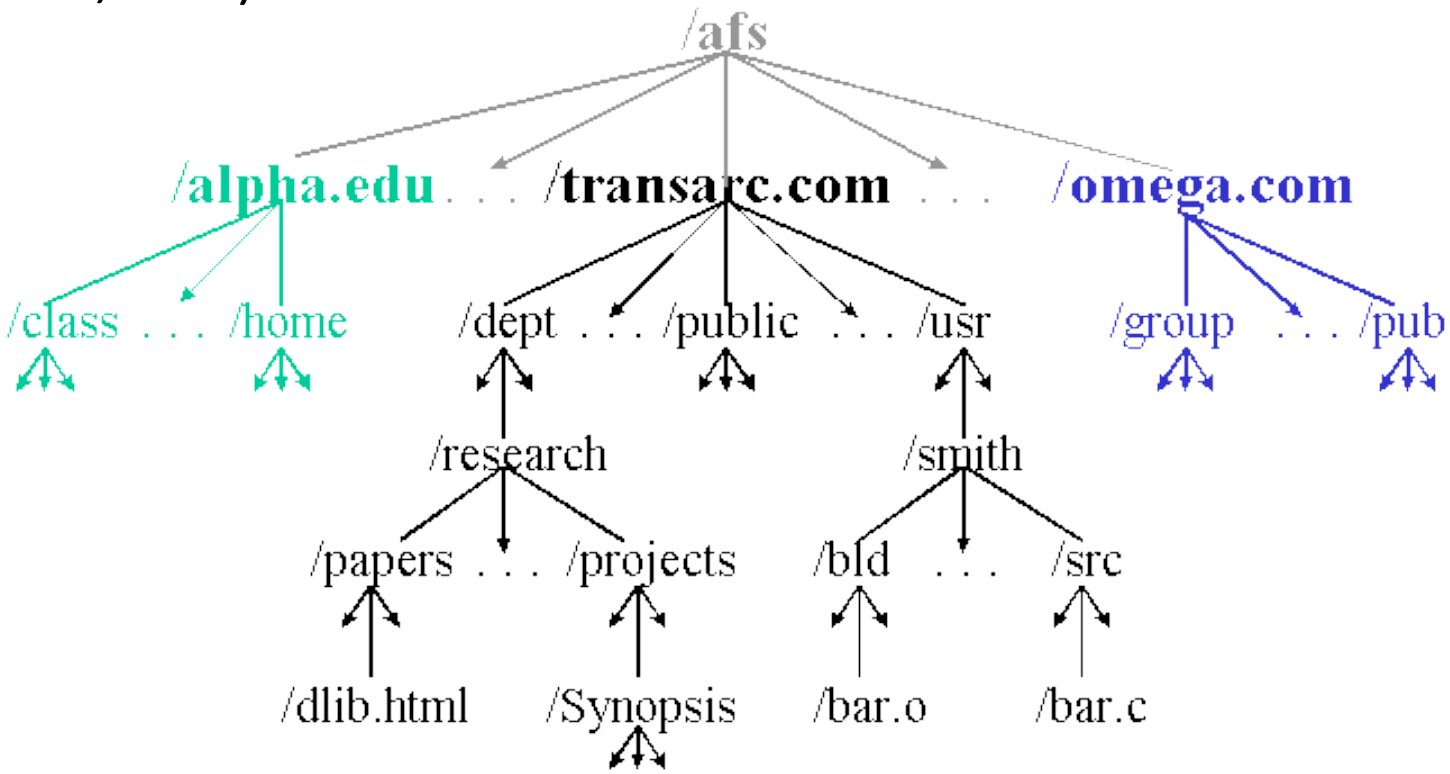
Read/write operation with master-slave replication

Distributed Systems Basics

- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Distributed Naming

- There are two major types of distributed naming systems:
  - Structured naming (NFS, AFS, DNS)
  - Attribute-based naming



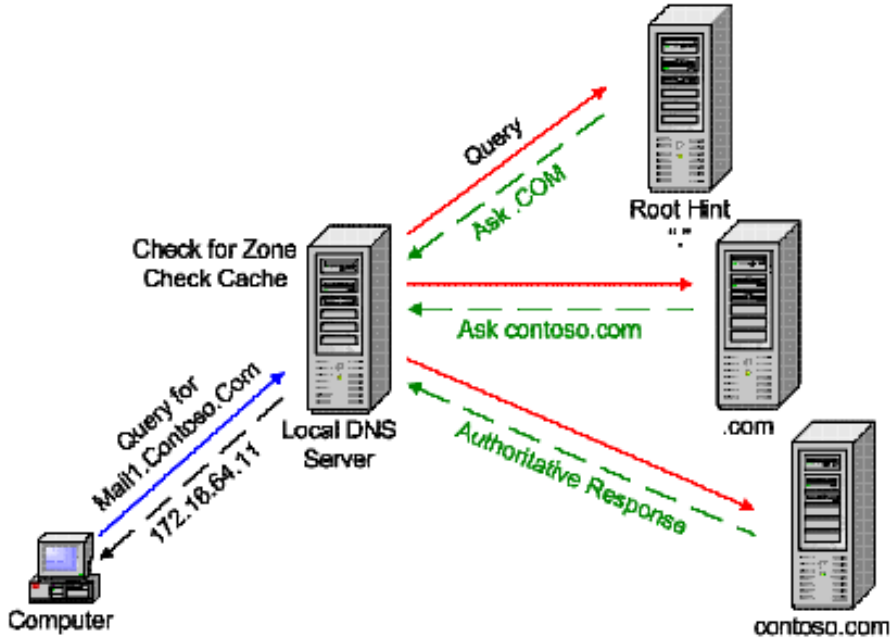
Network File System (NFS)

- A network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed
- AFS is a more modern version
  - We use it at UMBC

application	NFS
presentation	XDR
session	RPC
transport	UDP/TCP
network	IP
data link	network interface
physical	

Domain Name System (DNS)

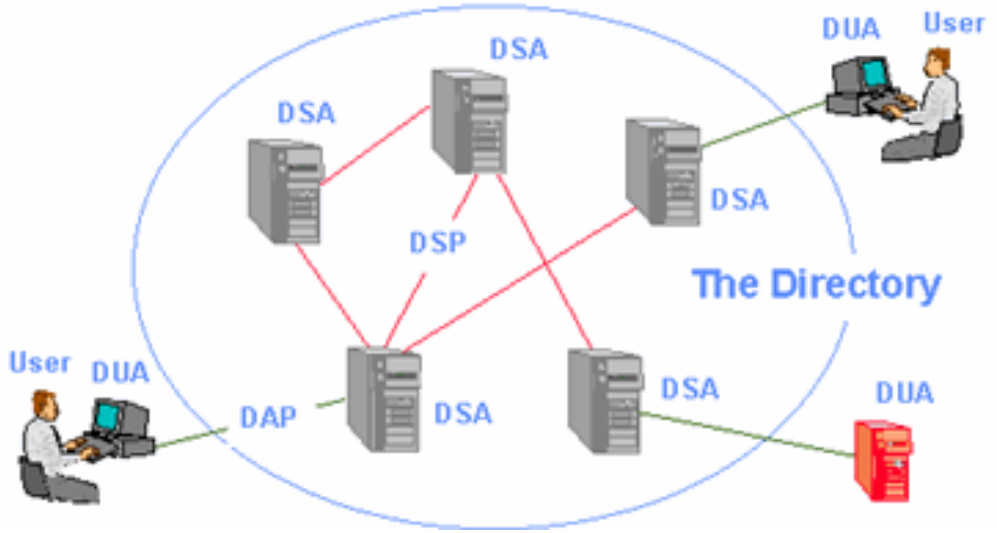
- It is the most widely used distributed naming system since it is used for looking up the addresses of hosts on the Internet
- Two roles
  - Server: respond to requests to convert names to IP addresses or the reverse
  - Client (resolver): ask other name servers to convert names to IP addresses
- Two optimizations: caching and replication
- Two resolver modes: recursive and iterative



Resolver demo

Attribute-based Naming

- Attribute-based naming is known as directory services
- Allows searches by attributes
  - ldap[s]://*<hostname>*:*<port>*/*<base_dn>*?*<attributes>*?*<scope>*?*<filter>*
- Implementations/standards
  - X.500
  - Lightweight Directory Access Protocol (LDAP)
  - Active Directory



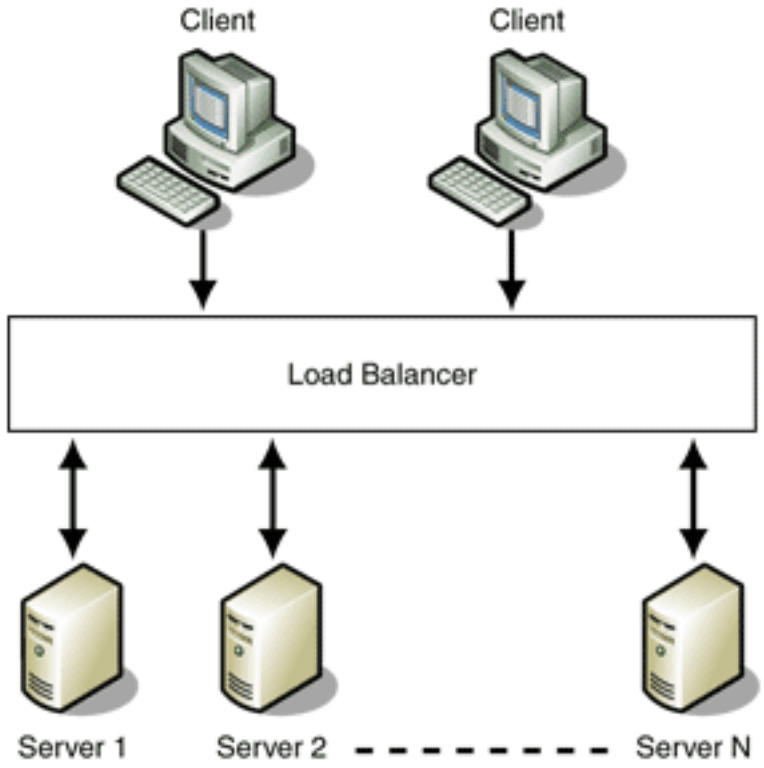
X.500 demo

Distributed Systems Basics

- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Load-balancing

- Load-balancing is a technique to make many servers appear as one server to clients and thereby getting performance increase
- Three major methods of load balancing
  - *Round-robin Domain Name System (RRDNS)*
  - *Load-balancing switches*
  - *Application servers*



Round-Robin Domain Name System (RRDNS)

- RRDNS is a low cost (in fact, free!), low performance way to load-balance
- Multiple IP addresses and servers for the same domain name
  - A new request uses the next IP address, until it wraps around to the first IP address again

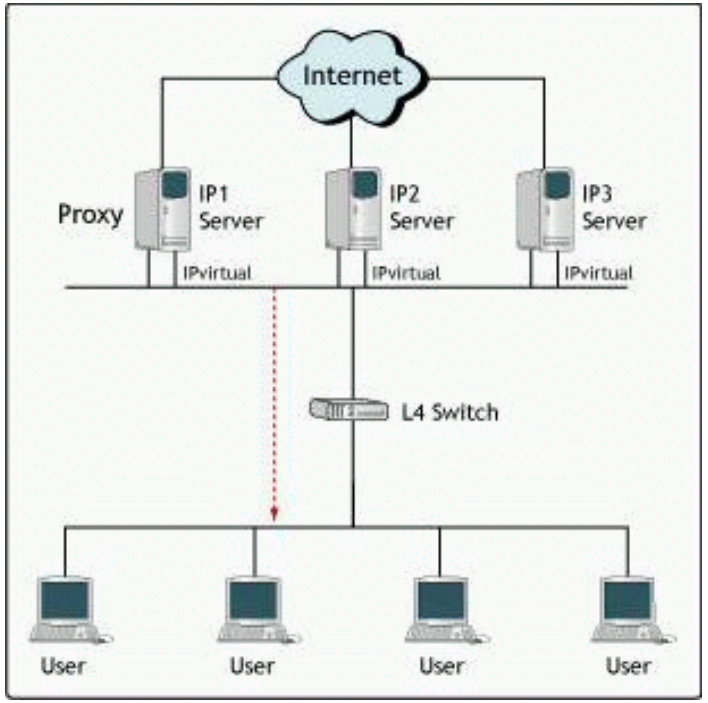
tsunami.abc.net.	A	IN	86400	66.98.218.34
tsunami.abc.net.	A	IN	86400	67.15.128.71

Limitations of the Round-Robin Domain Name System (RRDNS)

- Caching: DNS often caches the IP address of a domain. If the clients that have a lot of requests happen to cache the same IP address, the server will be overloaded.
- Lack of intelligence: RRDNS has no method of determining when a node in a cluster is overloaded or even available, it will just result in a bad request.

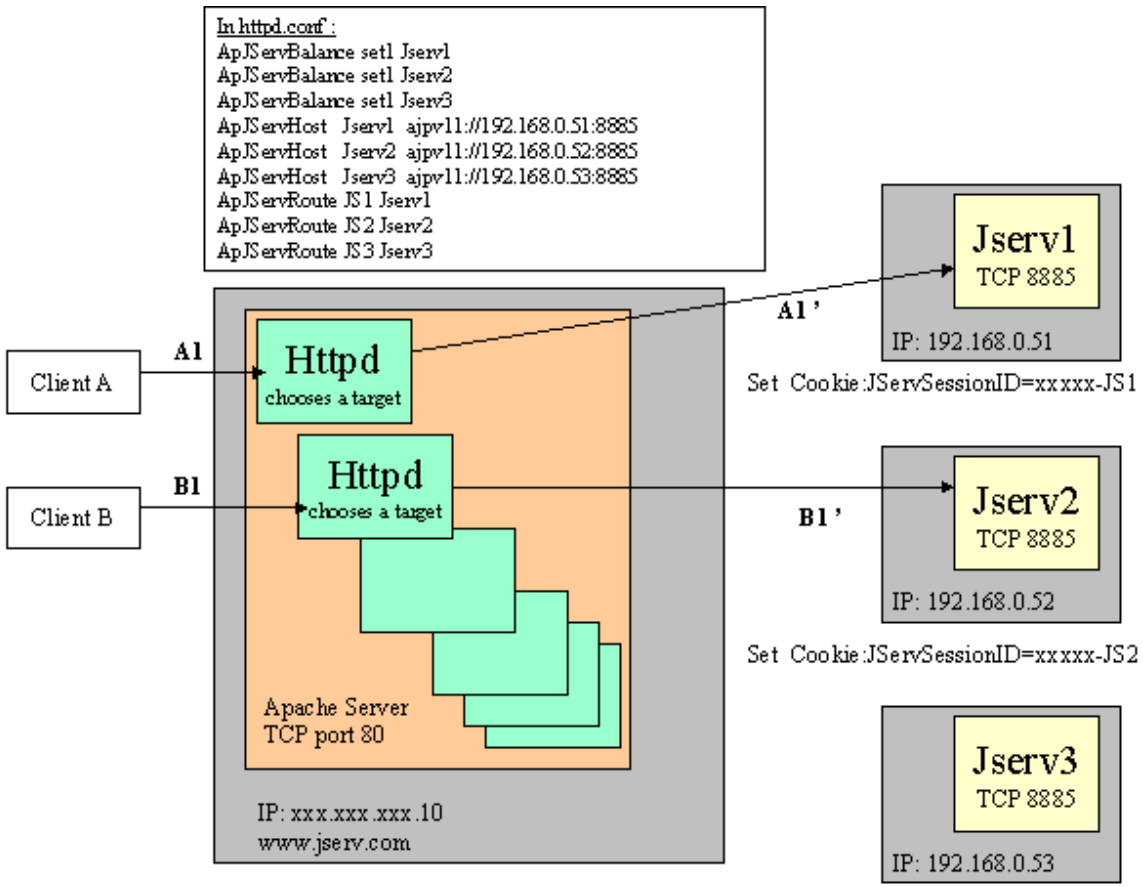
Load-Balancing Switches

- Load-balancing switches are the highest performance and most common method



Application Servers

- Application server load balancing approaches use the server to control the load-balancing.

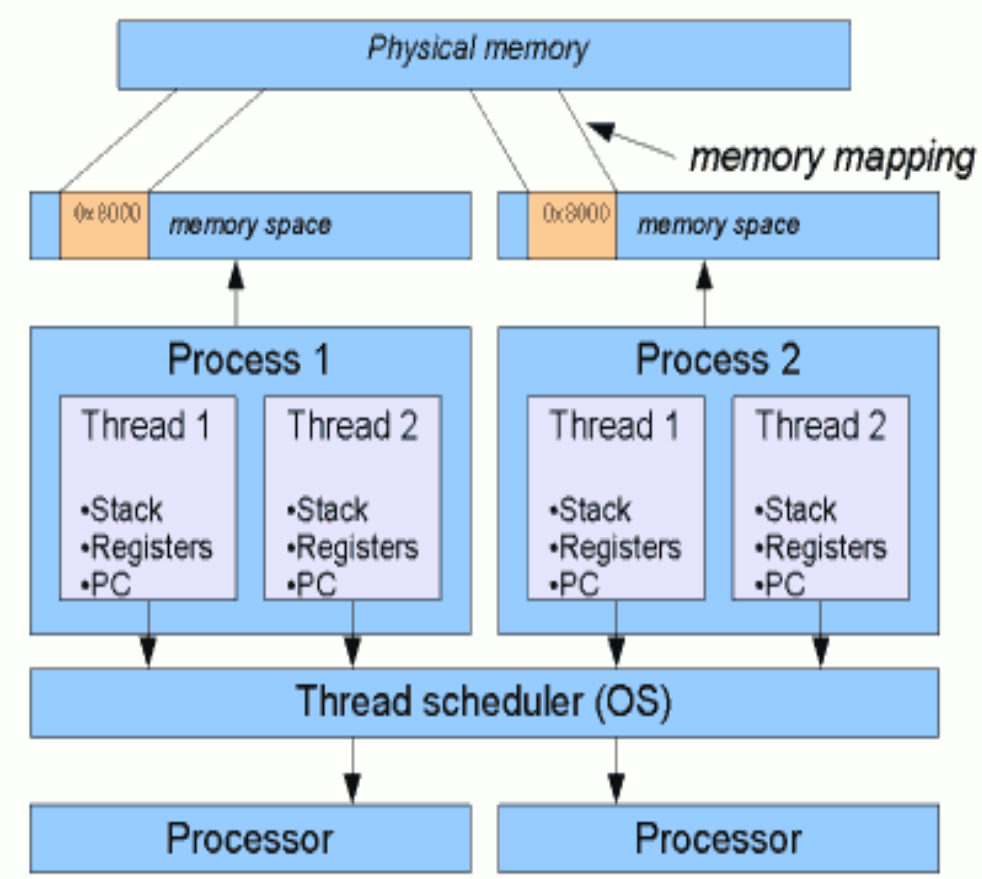


Distributed Systems Basics

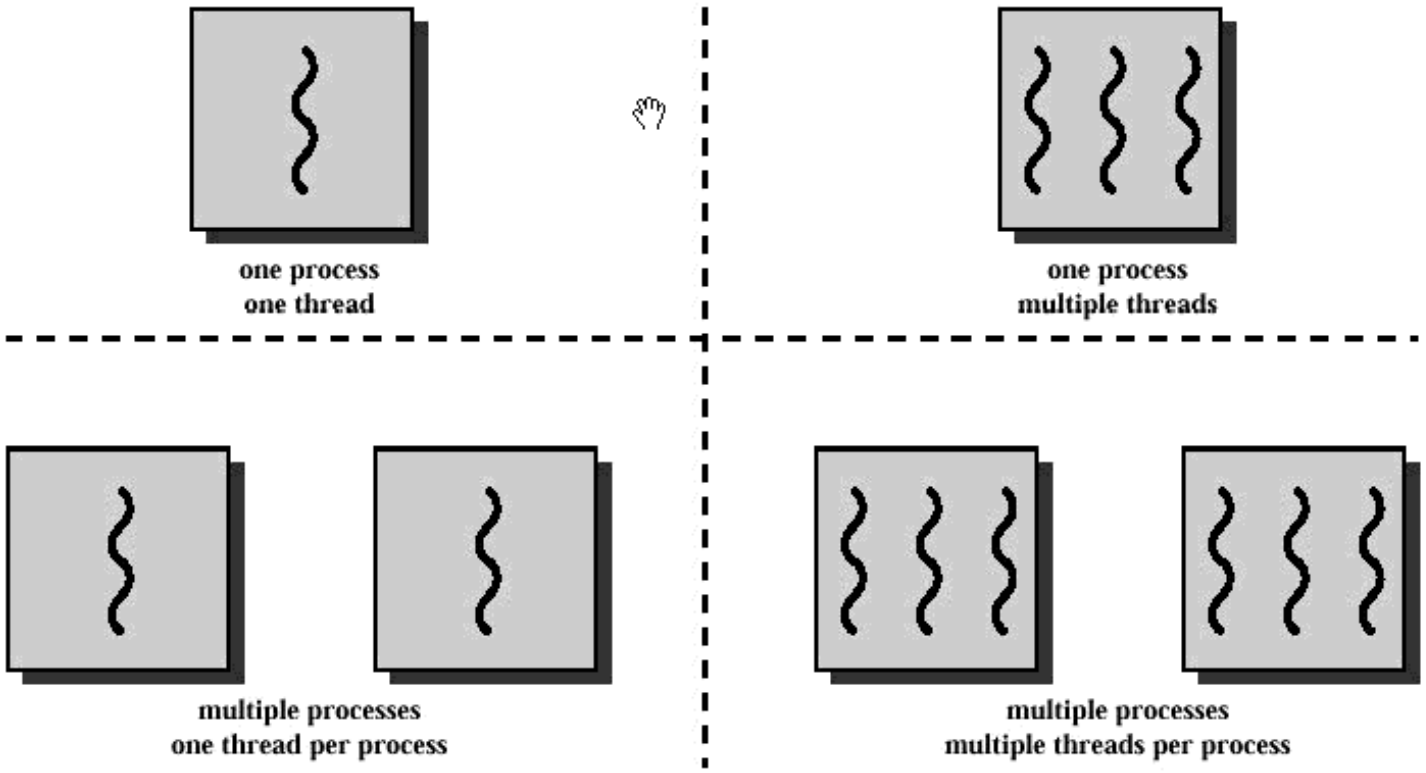
- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Processes and Threads

- Thread is often a component within a process
- Multiple threads can exist within one process, and share resources such as memory
- Benefits of Threads:
  - *Less time to create a new thread*
  - *Less time to terminate a thread*
  - *Less time to switch between two threads within the same process*
  - *Less communication overheads*



Processes and Threads (2)



Benefits of Multithreading

- Improve application responsiveness
  - The user of a multithreaded GUI does not have to wait for one activity to complete before starting another.
- Use multiprocessors more efficiently
  - Numerical algorithms and applications with a high degree of parallelism, such as matrix multiplications, can run much faster when implemented with threads on a multiprocessor.
- Improve program structure
  - Some programs are more efficiently structured as multiple independent or semi-independent units of execution instead of as a single, monolithic thread.
- Use fewer system resources
  - Each process has a full address space and operating systems state.
  - The inherent separation between processes can require a major effort by the programmer to communicate between the threads in different processes, or to synchronize their actions.

Asynchronous Event Loop

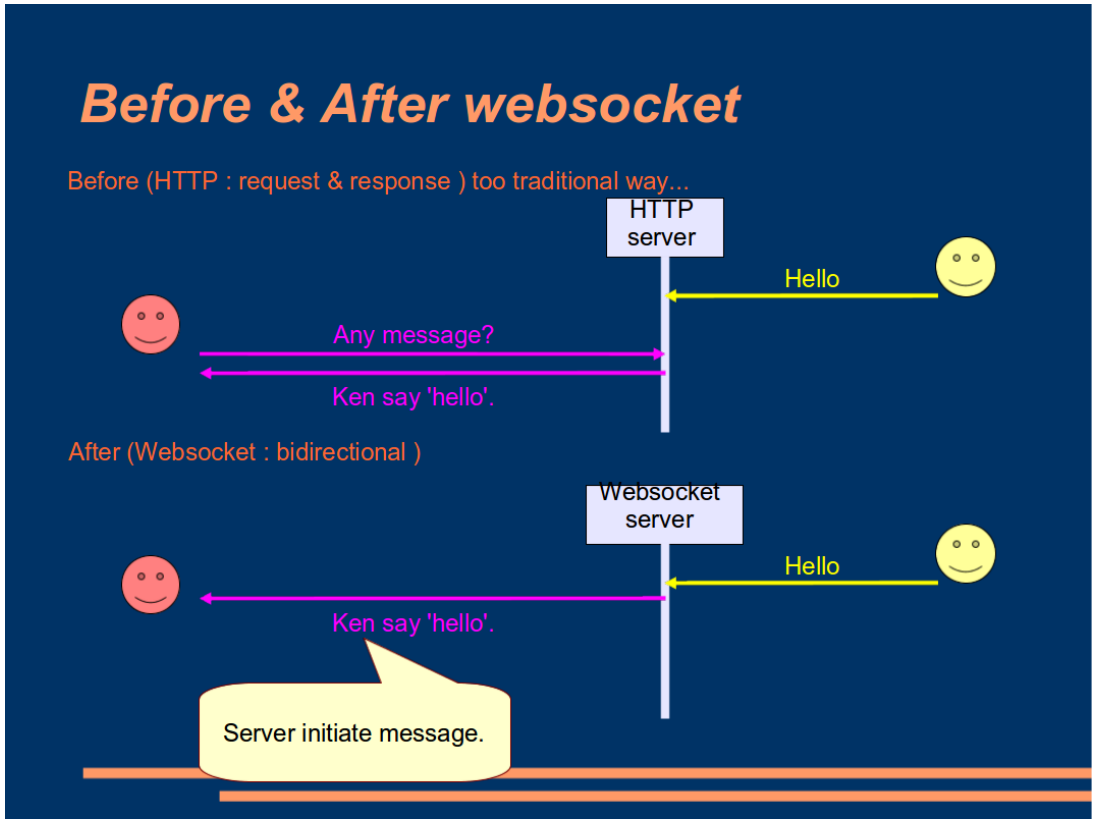
- Web server normally uses multi-threading for different requests, which has one problem
  - It creates a new thread for each synchronous request is that it is very memory intensive and Disk I/O (input/output) bound.
  - Each thread cannot respond to client until the data is read from disk.
- Asynchronous event loop is a new model to deal with the problem
  - These servers run as a single threaded process asynchronously.
  - The server just runs an event loop that gets requests and passes them on to other processes.
  - A **callback** mechanism informs the server process when data is ready.
- Node.js is an open-source runtime environment based on this model

Distributed Systems Basics

- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Push Technology

- Push technology, or server push: a style of Internet-based communication where the request for a given transaction is initiated by the publisher or server
- Pull technology: the request for information transmission is initiated by the receiver/client
- (short) polling: the client periodically (every few seconds) makes a request to check for new data
- WebSockets provides for a bi-directional, full-duplex communications channel over a TCP socket

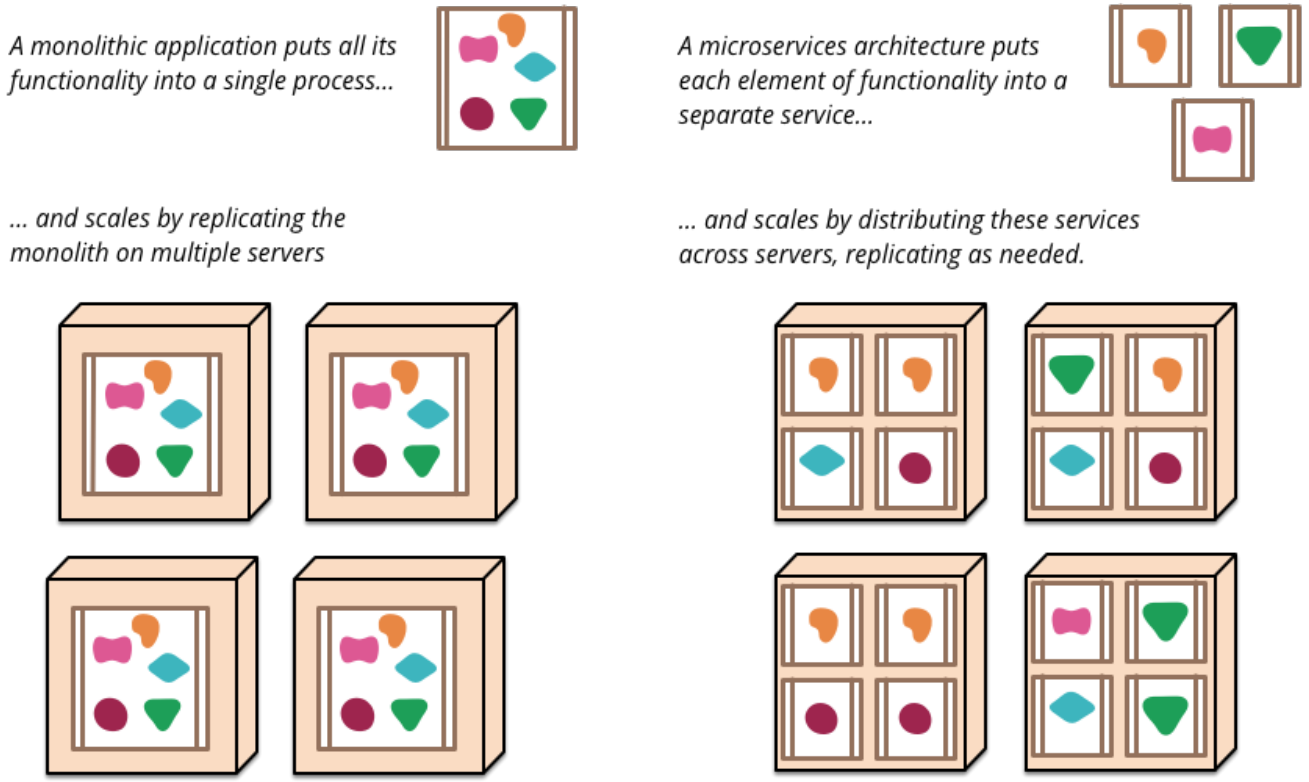


Distributed Systems Basics

- Caching
- Replication
- Distributed naming
- Load-balancing
- Processes and threads
- Push technology
- Microservice
- Server virtualization (see Chapter 13)

Microservice

- A system design pattern that follows Service-Oriented Architecture.
- Compared to monolithic applications where different functionalities are combined into a single program, microservice applications are easier to design, implement, deploy and maintain.



From https://martinfowler.com/articles/microservices.html