Notes

• Midterm reminder
  ▪ Second midterm next week (04/03), regular class time
  ▪ 20 points, more questions than midterm 1
  ▪ *non-comprehensive* exam: no need to study modules before midterm 1
  ▪ Online testing like Midterm 1. Same exam structure.
  ▪ Remember to bring your computer

• Send me up to 5 good questions in your opinion, I’ll use top ones
  ▪ Via direct message at slack. Can be a group effort. Try to add some explanation.

• If you want to learn more on Distributed Systems (including Cloud Computing) and/or Big Data, please consider enroll 'Independent Study' and/or 'Master Thesis' with me
  ▪ You could do in the summer/winter or regular semesters
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
  ▪ Team building: results have been sent out
  ▪ Select topic: talk to your team members on which topic to work on after class
  ▪ Inform your selection: post your topic on slack (case-study channel) after class. For a topic, the team who post earliest get it.
  ▪ Search and select paper/project you want to work on: https://scholar.google.com/, http://www.apache.org/
  ▪ Send me an email and get my approval
  ▪ Work as a team
  ▪ Present as a team on week 14 (05/01): every member presents
## Case Study Topics

<table>
<thead>
<tr>
<th>Team</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NoSQL/NewSQL Database</td>
</tr>
<tr>
<td>2</td>
<td>Parallel Computing</td>
</tr>
<tr>
<td>3</td>
<td>Distributed File System</td>
</tr>
<tr>
<td>4</td>
<td>Peer-to-peer (P2P) computing</td>
</tr>
<tr>
<td>5</td>
<td>Micro Services</td>
</tr>
<tr>
<td>6</td>
<td>Cloud computing</td>
</tr>
<tr>
<td></td>
<td>Big Data</td>
</tr>
</tbody>
</table>
Common Mistakes for HW4

• The Guardian html doesn't have required 2 additional variables from the guardian JSON

• The Flickr urls should be REST based, not static xml urls

• No actual content for the last Flickr url
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
  ▪ 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
Spring 2018
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
• Load-balancing
• Distributed naming
• Database replication
• 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.
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)
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
Content Delivery Network (CDN)

• This improves access to data by locating content in various places on the Internet in order to get it closer to clients
  ▪ Example: Netflix
Application Caching

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

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>NFS</td>
</tr>
<tr>
<td>presentation</td>
<td>XDR</td>
</tr>
<tr>
<td>session</td>
<td>RPC</td>
</tr>
<tr>
<td>transport</td>
<td>UDP/TCP</td>
</tr>
<tr>
<td>network</td>
<td>IP</td>
</tr>
<tr>
<td>data link</td>
<td></td>
</tr>
<tr>
<td>physical</td>
<td>network interface</td>
</tr>
</tbody>
</table>
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
Attribute-based Naming

• Attribute-based naming is known as directory services
• Allows searches by attributes
• Implementations/standards
  ▪ X.500
  ▪ Lightweight Directory Access Protocol (LDAP)
  ▪ Active Directory
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
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)

- One process, one thread
- Multiple processes, one thread per process
- Multiple processes, multiple threads per process
- One process, multiple threads
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
  - Many 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
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
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](https://martinfowler.com/articles/microservices.html)
Discussion #6

- What are the commonalities and differences between caching and replication?

<table>
<thead>
<tr>
<th>Difference 1</th>
<th>Caching</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commonality 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commonality 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>