Topics covered in this lecture

- Course overview and expectations
- Broad brushstroke coverage of distributed systems
Course Overview and Such

August 27, 2019

Course webpage

- All course materials will be on the course webpage
  - http://www.cs.colostate.edu/~cs555
  - Schedule
  - Lectures
  - Assignments
  - Announcements

- Grades will be posted on Canvas
- Piazza: For discussions

August 27, 2019
Professor: SHRIDEEP PALICKARA
On the schedule page

- You will see the **topics** that will be covered and the **order** in which I will cover them
- The readings section will list the books (and the chapters therein) that form the basis for the materials
- You will also see the schedule for when the **assignments** will be posted and when they are due

Office Hours

- Shrideep Pallickara
  
  Computer Science (CSB 364)
  
  Office Hours: 2:00-3:00 pm Friday or by appointments

- GTA: Daniel Rammer
  
  Office Hours: Will be finalized soon.
Why Distributed Systems?

- Your hard-drive's primacy has been eroding
- Data sources and programs are delivered over the network
  - No single hard drive can hold all the data you need
- Services themselves are distributed
  - Google search is backed by a massive distributed cloud

Distributed systems builds on a diverse set of areas

- Algorithms and Graph Theory
- Cryptography
- Networking
- Failure recovery and consistency models
- Probability theory
- Machine learning
- Information Retrieval
- Transactional Systems
Distributed Systems: Challenges [1/2]

- Scale with increases in data and users
- Responsiveness
  - Regardless of data size, responses must be prompt
- Intelligent
  - Correlate all sorts of information
- Dealing with system conditions
  - Murphy’s Law
  - Malicious Users
  - Byzantine failures

Distributed Systems: Challenges [2/2]

- Security
  - Detection
  - Privacy and Accountability
  - Authorizations
About me

- I do research in this area
- Areas that I have worked or actively work in include
  - Cloud computing
  - Internet of Things, Edge/Fog Computing
  - Grid computing
  - Peer-to-Peer systems
  - Content dissemination systems
  - Object Request Brokers
  - Collaborative systems
Expectations

- You will attend all classes
- You will focus on the discussions, and not on ...
  - Other assignments
  - Social networking updates
- Programming Assignments have to be done **individually**
- You will be **challenged** in this course
  - Assignments are geared toward real systems

ASSIGNMENTS

August 27, 2019
Professor: SHRIDEEP PALLICKARA

CS555: Distributed Systems (Fall 2019)
Dept. Of Computer Science, Colorado State University
Programming Assignments: Logistics

- Programming assignments will be due **at 5:00 pm on a Wednesday**
- You are allowed to submit up to 2 days late
  - There is a **7.5%** deduction for each day that you are late
- All programming assignments are **individual** assignments

Assignments: What to expect

- There will be no busy work
  - No GUI
- Complexity will not be through obfuscation
- You will be able to look back and feel good about them
  - Delayed gratification
There will be 3 programming assignments and a Term Project

- Contrasting replication and erasure coding in a distributed file system
- Structured P2P system
- Cloud computing assignment
- Term Project & Report [35]
  - Team Effort: Groups of 2-3 (Must include at least one on-campus and distance student)

WHAT IT TAKES TO SUCCEED
What it takes to succeed (1/2)

- You are required to work at least **12 hours** per-week outside of class
  - Coding and reviewing material from class

- If you miss a lecture
  - Add about 3 hours per missed lecture

What it takes to succeed (2/2)

- **Work on the assignments** *every day*
  - There is no such thing as waiting for inspiration to strike

- **Reflect** about how you could have done things differently for better performance
  - Even after you have submitted an assignment
  - It will improve your choices in the next assignment
How to fail this course?

- Believing that you can learn via osmosis
- **Missing lectures**
  - If you don't have the discipline to show up, you will most likely not have the discipline to catch up
- **Procrastinating**
  - The assignments cannot be done in a week
  - **Organize your schedule** so that you can succeed

Use of laptops, cell phones, tablets, and other electronic devices

- **Is not allowed**
- If you must use a laptop you will have to:
  - Turn off wireless
  - And use it only for taking notes
- **Authorized laptop users**
  - Will sit in the back row starting at the corners
Why attend lectures if all the slides are posted?

- Slides are only part of the story
  - They anchor the discussion
- Any field has a language associated with it
- People who have worked in an area for a long time speak the language (and do so subconsciously)
  - Sitting in classes helps you learn how to frame questions and responses
- Often there are surprising questions
  - Some of these may be asked by interviewers

Help me help you

- We will have surveys at the end of every class
- You will provide a list of
  - 3 concepts you followed clearly
  - 3 concepts you had problems keeping up with
- Problem areas for the majority of the class will be addressed in the next class
Interactions

- You can have discussions with me and your peers

- There are **two constraints** to these discussions
  - No code can be exchanged under any circumstances
  - No one takes over someone else's keyboard

- Bumps are to be expected along the way
  - But you should get over this yourself
  - It will help you with the next problem you encounter

---

**Grading**

August 27, 2019
Grading Breakdown

- Programming Assignments: 55%
- Term Project & Report: 35%
- Quizzes: 10%

Grading: Letter Grades

- Letter grades will be based on the following standard breakpoints:
  - $\geq 90$ is an A, $\geq 88$ is an A-,
  - $\geq 86$ is a B+, $\geq 80$ is a B, $\geq 78$ is a B-,
  - $\geq 76$ is a C+, $\geq 70$ is a C,
  - $\geq 60$ is a D, and $<60$ is an F.

- I will not cut higher than this, but I may cut lower.
Grading and Late Submissions

- There is **no extra-credit**
  - Any credit you earn must be on a level-playing field with your peers
- There will be **no make-up exams**
- Assignments will have an interview portion

For the Quizzes

- I will only ask questions about what I teach
  - If I didn’t teach it, I won’t ask from that portion
- If the concepts were covered in my slides
  - You should be able to answer the questions
When I make slides

- I usually refer to several texts
  - Also technical papers, articles (with URLs)

- I always list my references at the end of every slide set

There is no textbook for this course, but we will rely on several texts/papers


There is no textbook for this course, but we will rely on several texts/papers

Topics covered in CS555

- Time
- Threads and Processes
- Role of Order
- Content Dissemination Systems
- Peer to Peer Systems
- Data intensive computing
  - MapReduce
- Stream Processing Systems
  - Spark

August 27, 2019
Professor: SHRIDEEP PALICKARA
Dept. Of Computer Science, Colorado State University

Topics covered in CS555

- Replication and Consistency Models
  - Brewers CAP
- Cloud Storage Systems
  - Google File System, Amazon Dynamo
- Distributed Objects and RPCs
- Virtualization
  - Server Consolidation

August 27, 2019
Professor: SHRIDEEP PALICKARA
Dept. Of Computer Science, Colorado State University
Distributed systems definition

- A *distributed system is a collection of independent computers that appears to its users as a single coherent system*  
  
  - Andrew Tanenbaum

- A *distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable*

  - Leslie Lamport
Distributed Systems:
GOALS

- Making resources accessible
- Distribution transparency
- Openness
- Scalability

GOAL 1: Making resources accessible

- **Share resources** in a controlled and efficient fashion
  - Printers, computers, storage facilities, etc.
- Reasons for sharing?
  - Economics: Often *cheaper* to share than to have copies
  - Easy to *collaborate* and exchange information
As connectivity and sharing increase security issues become important

- Protection against intrusion and eavesdropping of communications
- Entities can track communications to build profiles of users
  - Violates privacy
  - Unwanted communications
    - Spam?

GOAL 2: Distribution transparency

- **Hide** the fact that the constituent processes and resources are physically distributed
  - Across multiple computers

- The system should present itself to the application & users as if it were a single computer system
  - Transparency
Types of transparency

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide the fact that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource is replicated</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failures and recovery of a resource</td>
</tr>
</tbody>
</table>

Degree of transparency: Sometimes we cannot hide all distribution aspects from users

- Time zone variations
- Communications are not instantaneous
  - Signal transmission is limited by the
    - Speed of light
    - Limited processing capabilities of intermediate switches
Trade-off between transparency and performance of system

- EXAMPLE: Mask transient server failure by *retrying* connections periodically
  - Maybe better to give up earlier or try another server

- EXAMPLE: Several replicas need to be kept *consistent* at all times
  - If change is made to one copy, it must be propagated to all copies *before* a new change can be made

**GOAL 3: Openness**

- Offer services according to standard rules that describe *syntax* and *semantics* of those services
  - E.g., in networks, rules govern: format, content, and meaning of messages
  - Formalized in *protocols*

- Services are generally specified through interfaces
What are these interfaces?

- Specify names of functions
  - Along with types of parameters, return values, possible exceptions, etc.
  - IDL (Interface Definition Language), WSDL (Web Services Description Language)
- Hardest part is specifying the semantics of these interfaces
  - What exactly does the service do?

Interoperability

- The extent to which two implementations of components from different manufacturers
  - Coexist and work together
  - Merely by relying on each other's interfaces
Portability

- Extent to which application developed for distributed system A
  - Can be executed, without modification, on a different distributed system B that implements the same interfaces as A

Extensibility

- Configure system out of different components
  - Possibly from different developers

- Add new components, replace existing ones
  - Without affecting components that stay in place

- Examples:
  - Add parts that run on a different OS
  - Replace an entire file system
Achieving flexibility

- System must be organized as a collection of small, replaceable or adaptable components.
- Definitions for not only highest-level interfaces but also for internal parts of the system.
  - Describe how these interact.

Flexibility example:
Caching within a browser

- Browsers allow you to adapt caching policy.
  - Specify size of the cache.
  - Consistency check of cached document.
- However you cannot influence other parameters such as:
  - How long can a document remain in the cache?
  - Which document should be removed when the cache fills up?
  - Cannot make caching decisions based on content.
    - For e.g., metro train timetables rarely change.
What we need is a separation of policy and mechanism

- Browser should ideally provide facilities only for storing documents
  - Users decide which documents are stored and for how long
  - Perhaps user can implement policy as a pluggable browser component

GOAL 4: Scalability Dimensions

- **Size**
  - Should be possible to add more users and resources to the system

- **Geographically scalable**
  - Users and resources may be spatially far apart

- **Administrative scalability**
  - Should be easy to manage even if it spans many independent administrative domains
Examples of scalability limitations

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single on-line telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Doing routing based on complete information</td>
</tr>
</tbody>
</table>

Characteristics of decentralized algorithms

- No machine has complete information about system
- Machines make decisions based only on local information
- Failure of one machine does not ruin algorithm
- No implicit assumption that a global clock exists
Scaling techniques: Asynchronous communications

- **Avoid waiting** for responses to requests that were issued
  - Works well in WAN and geographically dispersed settings
- Alternatively, use a **separate thread** of control for requests
- Does not work well in interactive applications

Scaling techniques: Distribution

- Take a component and **split** them into smaller parts
  - **Disperse** these parts across the system
- **Examples:**
  - Domain Name System (DNS)
  - World Wide Web
Scaling techniques:
Replication

- Scaling problems usually manifest themselves as performance problems
  - Slow responses, and such

- Replication involves having dispersed copies
  - Increases availability
    - Better fault tolerance
  - Balances load between components
    - Leads to better performance

Pitfalls: False assumptions made during development of a distributed application

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport costs is zero
- There is one administrator

List by:
Peter Deutch
Sun Microsystems
Trade-off space in distributed systems

1. CPU and memory utilization
2. Synchronization overhead
   - Number, type, and size of messages exchanged
3. Consistency
   - Brewers CAP theorem
4. Time
   - Completion time, response times, etc
5. Costs
After Charles Darwin’s book *The Origin of Species* was published

- German zoologist Ernst Haeckl stated
  - Ontogeny recapitulates Phylogeny

  Development of an embryo repeats the evolution of the species

  - i.e. human egg goes through stages of being a fish, … , before becoming human baby
  - Modern biologists think this is a gross simplification!

Something vaguely similar has happened in the computer industry

- Each new species (type of computer) goes through the development that its ancestors did
  - Both in hardware and software
  - Mainframe, mini computers, PC, handheld, etc
Much of what happens in computing and other fields is technology driven

- Ancient Romans lacked cars not because they liked walking
  - It is because they didn’t know to build cars

- PCs exist not because people have a centuries-old pent-up desire to own one
  - It is now possible to manufacture them cheaply

Technology affects our view of systems

- A change in technology renders some idea obsolete
  - Another change could revive it

- Especially true when change has to do with relative performance
  - Of different parts of the system
Let’s look at this relative performance

- When CPUs become faster than memories?
  - Caches become important to speed-up slow memory
- If new memory technology makes memories much faster than CPUs?
  - Caches will vanish!
- In biology extinction is forever
  - In computer science it is sometimes only for a few years

Systems stand on the shoulders of those that have come before it

- Client-Server
- 3-tier (or N tiers)
- Distributed Objects (DCOM, CORBA, RMI)
- Message Passing based
  - Queuing, Publish/Subscribe, Peer-to-Peer
- Grid Computing
- Service Oriented Architectures (XML)
- Cloud Computing
The contents of this slide-set are based on the following references
