CSE 224: OVERVIEW AND INTRODUCTION

George Porter
Jan 4, 2022
ATTRIBUTION

• These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
CSE 224: GRADUATE NETWORKED SYSTEMS

- Add networking support to software
  - Between two computers
  - Between computer and datacenter ("The Cloud")

- Develop software that is:
  - Scalable (handles 100s of M to 1+ billion users)
  - Fault-tolerant (survives failures)
  - Evolvable (how to support different versions?)
  - Secure
OUR LIVES ARE (LARGELY) ONLINE!
NETWORKED SERVICES DRIVEN BY DATA

Data + Amazon.com = Product Recommendations

Data + Spotify = Custom Stations

Data + Google = Personalized Search
DATA-DRIVEN, PER-USER CUSTOMIZATION + ML

Data + Amazon.com = Product Recommendations
HOW TO WRITE NETWORKED SOFTWARE?

- How to access data sets
- How to communicate between the user and the cloud?
- How to communicate between machines in the cloud to increase scale?
- How to handle failures and faults
DATACENTERS: THE HOME OF ALL THIS COMPUTING AND STORAGE

Microsoft

Google

Facebook
MASSIVE IMPACT

• To build:
  • Google spends about $3B per year
  • Microsoft spent $15B in total

• To operate:
  • 1-2% of global energy consumption\(^1\)
  • 140 billion kWh (50 power plants)
  • $13 billion in electricity bills
  • 100 metric tons of carbon pollution per year

1. LBNL, 2013
2. NRDC report
THE NETWORK HAS SEEN RAPID GROWTH
THE NETWORK HAS SEEN RAPID GROWTH
THE NETWORK HAS SEEN RAPID GROWTH


Web Created

Google’s 1st cluster
THE NETWORK HAS SEEN RAPID GROWTH

- 1989: Web Created
- 1993: Google’s 1st cluster
- 1997: Facebook
- 2001
- 2005
- 2009: 10 years
- 2013: 15 years
THE IMPORTANCE OF SCALE

• Network primitives are designed to scale

• Techniques we learn are directly applicable to global-scale services like Google, Facebook, ...

• Your projects will be tested in small scale
  • Yet could scale immensely with minimal to no modifications
HOW TO BUILD SUCH LARGE SYSTEMS?
HOW TO BUILD SUCH LARGE SYSTEMS?

- Systems...
- Built on top of abstractions...
- Built on software...
- Built on hardware...

*We will cover the software abstractions to enable you to write networked software*
IT’S NOT JUST WEBSITES AND SOCIAL MEDIA THOUGH!
SELF-DRIVING CARS
Smart, cleanly-powered grid

Interconnected grid with: 1. Distributed, regional, and central generation; 2. Hybrids (multiple means) of power generation at each scale; 3. Smart sensors in buildings for efficient use; 4. Smart technologies to designate critical areas during power losses; 5. New generation batteries and other storage technologies.

CSE 224 VS {221,222A,223B}

- 224: Graduate Networked Systems
  - How to program networked software
  - Socket programming, RPC, protocol design and implementation, consensus and consistency, security, TLS, ...
  - Designed as a *broad survey* of systems thinking
  - Learn through hands-on, programming-based projects
- 224 Target audience:
  - MS “comps” students and BS/MS students
  - Non-systems MS “thesis” and non-systems Ph.D. students
- Note:
  - Cannot receive credit for both 124 and 224

- Research-focused depth sequence
  - 221: Operating Systems
  - 222A: Networking
  - 223B: Distributed systems theory
  - Deep dives into peer-reviewed literature
  - Learn through close readings and in-class discussion of 4 research papers per week
- 221/222A/223B Target audience:
  - Systems MS “thesis” and Systems Ph.D students

Note:

- Cannot receive credit for both 124 and 224
THE CHALLENGE OF NETWORKING

• CS programs include:
  • Algorithms
  • Programming languages
  • Architecture
  • Data structures
  • Etc...

• How does the network change each of these areas?
RESOURCES

- Canvas ([https://canvas.ucsd.edu](https://canvas.ucsd.edu))
  - Gradebook, links to assignments + deadlines, PDFs of lecture slides, in-class demos and exercises
- Piazza ([https://piazza.com/ucsd/winter2022/cse224](https://piazza.com/ucsd/winter2022/cse224))
- Github (for submitting your projects)
- Two books
- TA discussion section (1x week)
CLASS MEETINGS

• Mostly putting the material that you read into context
  • Examples, live coding demos, activities, some “mini lectures” on algorithms, protocols, etc.

• Designed to be active (not asynchronous)
  • Mixture of slides, notes, worksheets, demos, etc.

• You are responsible for everything that happens during class
  • Will record to Zoom

• Will be asking for feedback on what works and what doesn’t work a lot during the class
The Go Programming Language
Alan A. A. Donovan
Brian W. Kernighan

Free if accessed through the UCSD library

NETWORK PROGRAMMING WITH GO
Code secure and reliable network services from scratch
Adam Woodbeck

Free if accessed through the library
$29 e-copy ($32 printed)
We’ll be using the “Go” language

- golang.org
- Designed at Google in 2007
- Goals: improve programming productivity in an era of multicore, networked machines, and large codebases
- Kernighan (of ‘C’ fame) co-created

Why?

- Simple, readable, no mem allocation (similar to Python)
- High-performance networking
- Concurrency/parallelism
- Static typing and efficient runtime
- Industry-quality and deployed at massive scale
1. [5%] Single-node sort (Jan 11)
2. [10%] Distributed sort w/ sockets (Jan 20)
3. [25%] Build your own web server (Feb 1)
4. [15%] GRPC-based SurfStore client with single metadata store (Feb 15)
5. [30%] Fault-tolerant SurfStore server (Mar 3)
6. [10%] A web-based interface to your surfstore server supporting TLS supporting upload/download (Mar 11)
   - [5%] Research paper reflections (throughout the quarter)
GRADING SCHEME

<table>
<thead>
<tr>
<th>Grade</th>
<th>Percentage Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>100% to 97%</td>
</tr>
<tr>
<td>A</td>
<td>&lt; 97% to 94%</td>
</tr>
<tr>
<td>A-</td>
<td>&lt; 94% to 90%</td>
</tr>
<tr>
<td>B+</td>
<td>&lt; 90% to 87%</td>
</tr>
<tr>
<td>B</td>
<td>&lt; 87% to 84%</td>
</tr>
<tr>
<td>B-</td>
<td>&lt; 84% to 80%</td>
</tr>
<tr>
<td>C+</td>
<td>&lt; 80% to 77%</td>
</tr>
<tr>
<td>C</td>
<td>&lt; 77% to 74%</td>
</tr>
<tr>
<td>C-</td>
<td>&lt; 74% to 70%</td>
</tr>
<tr>
<td>D</td>
<td>&lt; 70% to 60%</td>
</tr>
<tr>
<td>F</td>
<td>&lt; 60% to 0%</td>
</tr>
</tbody>
</table>

I reserve the right to adjust these cut-offs in a way that benefits you (e.g. I might lower the A- range to 89, but I would never raise the B+ range to 91).
DEPLOYMENT PLATFORM: AMAZON CLOUD SERVICES

- Deploy your code on Amazon AWS to datacenters on five continents
  - Mumbai, India; Dublin Ireland; Sao Paulo Brazil; Seoul, Korea, California
YOUR SERVER IN THE CLOUD

- Every student gets about $50 in free credit for the Amazon cloud
- Can develop/run your code there
- Can develop on your own computer if you prefer to do that, but make sure it runs correctly on your cloud machine
- (The autograder runs on x86_64 Linux)
CLOUD NATIVE LANDSCAPE IN A 10-WEEK QUARTER
BACKEND DEVELOPMENT ROADMAP

- **Basic Usage of Git**
- **Version Control Systems**
  What are they and why you should use one
- **Repo hosting services**
  Create account and Learn to use GitHub
- **Relational Databases**
- **NoSQL Databases**
- **More about Databases**
  - ORMs
  - ACID
  - Transactions
  - N+1 Problem
- **Database Normalization**
  Indexes and how they work
- **HATEOAS**
  Open API Spec and Swagger
- **Learn about APIs**
  - REST
    - Read Roy Fielding’s Paper
  - JSON APIs
  - SOAP
- **Caching**
- **Web Security Knowledge**
  - Hashing Algorithms
    - MD5 and why not to use it
    - SHA Family
    - scrypt
    - bcrypt
- **مهندカテゴリ**
  - PostgreSQL
  - MySQL
  - MariaDB
  - MS SQL
  - Oracle
- **PostgreSQL**
- **MySQL**
- **MariaDB**
- **MS SQL**
- **Oracle**
- **MongoDB**
- **RethinkDB**
- **CouchDB**
- **DynamoDB**
- **Data Replication**
- **Sharding Strategies**
- **CAP Theorem**
- **Cookie Based**
- **OAuth**
- **Basic Authentication**
- **Token Authentication**
- **JWT**
- **OpenID**
- **SAML**
- **Redis**
- **Memcached**
- **CDN**
- **Server Side**
- **Client Side**
BACKEND DEVELOPMENT ROADMAP

- **Testing**
  - Integration Testing
  - Unit Testing
  - Functional Testing

- **CI / CD**

- **Design and Development Principles**
  - GOF Design Patterns
  - Domain Driven Design
  - Test Driven Development
  - SOLID
  - KISS
  - YAGNI
  - DRY

- **Search Engines**
  - Elasticsearch
  - Solr

- **Message Brokers**
  - RabbitMQ
  - Kafka

- **Graph Databases**
  - Neo4j

- **GraphQL**

- **Containerization vs Virtualization**
  - Docker
  - rkt
  - LXC

- **WebSockets**

- **Web Servers**
  - NGINX
  - Apache

- **Architectural Patterns**
  - Monolithic Apps
  - Microservices
  - SOA
  - CQRS and Event Sourcing
  - Serverless

- **Content Security Policy**
  - HTTPS
  - CORS
  - SSL/TLS
  - OWASP Security Risks
BACKEND DEVELOPMENT ROADMAP

**Graph Databases**
- Neo4j

**GraphQL**
- Relay Modern

**Web Sockets**
- Mitigation Strategies
- Understand the Diff.

**Web Servers**
- Nginx
- Apache
- Caddy
- MS IIS

**Building for Scale**
- General topics that you should learn and care about for the sustainability of the product.

**Migration Strategies**
- Horizontal vs Vertical Scaling

**Keep Learning**
- Monitoring
- Telemetry

**Graceful Degradation**
- Throttling
- Backpressure
- Loadshifting
- Circuit Breaker

**Instrumentation**
TOGETHER, WE CAN HELP FIGHT COVID-19.

CA COVID Notify uses the Exposure Notifications System from Google and Apple to alert you when you may have been exposed to COVID-19.

ADD CA COVID NOTIFY

CAMPUS STATUS

UC San Diego continues to monitor the spread of COVID-19, working closely with local, state and national officials. For the latest updates to the campus community, visit the Current Campus Status page »
THE COVID-19 PANDEMIC AND 224

• Please be kind...
  • To me and the teaching staff
    • We are changing the course from in-person to online (temporarily?) then back to in-person (maybe??). As a result we can’t necessarily have every day of the whole term mapped out perfectly
  • To your fellow students
    • They’re under a ton of stress
  • To campus staff
    • They’re also under a ton of stress
  • Most importantly: to yourself