Welcome! Course Staff

Instructors

Yuvraj Agarwal
Srini Seshan
Daniel Berger

10 TA’s

Lydia Mankins
Sandeep Agarwalla
Gaurangi Saxena
Zebing Lin
Leiyu Zhao
Junhong Li
Vamshi Reddy Konagari
Felipe Archondo
Devdeep Ray
Daehyeok Kim
Course Logistics

• Course Policies
  • Class web page: http://www.cs.cmu.edu/~15-440/
  • Piazza: https://piazza.com/cmu/fall2017/1544015640
  • Obligatory discussion of {late days, cheating, etc.}

• Waitlist!

• Optional Recitations: Primarily for project support

• Office hours / TAs are on class web page (check often)

• Go work through the Tour of Go!
  • https://tour.golang.org/welcome/1
Waitlist

- Waitlist of unprecedented size.
- Registered: 440 (94) + 640A (49) + 640B (34) => 178
- Waitlisted: 440 (20) + 640A (181) + 640B (45) => 245
- The bad news: Not everyone will get in. We are by law limited to physical class size, not subject to negotiation. Section A (144 people max), Section B (50)
- The plea: Not serious? PLEASE DROP SOON.
- The strategy:
  - 1st week, we will try and enroll to capacity. Attend class!
  - Buffer of 20 students (Section A), 8 students (Section B)
    - These get enrolled only if someone drops (we will email you)
Processing WL/Enrollment

• Unable to able to take the class if:
  • If not taken 213/513 at CMU *before*
  • If you are an UGRAD and lower than a “C” in 213
  • If you are a Grad and lower than a “B-” in 213/513

• Priority order
  • Required: CS UGrad, MS in SCS (MSCS, MSDC, MITS,..)
  • .. then WL rank + 213 Grade for ECE and INI students
  • Email from Faculty advisor in some cases

• Our apologies: Resource Limitations (Room, TAs, ..)
Recitations & TA hours

• Optional Recitations this year
  • Four 1hr sessions: 4:30pm – 8:30pm (most likely Tuesday/Wednesday)
  • Times based on survey, room availability, non overlap with classes
  • TAs will strictly enforce room size limit, if you find no seats please come to the next session (Overflow disallowed by fire code!).

• Recitations (6 or 7) primarily to support Programming Projects
  • Introduction to GO (9/5)
  • Introduction to P0, P1, P2, P3 + Discussion after projects due
  • Lead by TAs, are not meant to go over class lectures

• TA Office Hours (Mon – Friday, spread out during the day)
  • No office hours the day before projects or homeworks are due
Course Goals

• Systems requirement:
  • Learn something about distributed systems in particular;
  • Learn general systems principles (modularity, layering, naming, security, ...)
  • Practice implementing real, larger systems; in teams; must run in nasty environment;

• One consequence: Must pass homeworks, exams, and projects independently as well as in total.
  • Note, if you fail either you will not pass the class
Course Format

• ~24 lectures: Wed/Fr 4:30pm – 5:50pm in GHC4307

• Office hours see website
  • Practical issues for implementing projects; general questions and discussion

• 4 projects; 2 solo (p0, p2), 2 two-person team (p1,p3)
  • P1: Distributed (internet-wide) bitcoin miner
  • P2: Project with distributed systems concepts like replication or distributed commit/consensus (e.g. PAXOS)
  • P3: Building a distributed key value store (or similar)
Study Material

• Slides and notes on course website
  • Not identical to prior 15-440 instances

• Distributed Systems 3.0.1 (2017)
  • Free download
  • Link to purchase ($25) from syllabus page

• Several useful references on web page
About Projects

• Systems programming somewhat different from what you’ve done before
  • Low-level (C / GO)
  • Often designed to run indefinitely (error handling must be rock solid)
  • Must be secure - horrible environment
  • Concurrency
  • Interfaces specified by documented protocols

• Office Hours & “System Hacker’s View of Software Engineering”
  • Practical techniques designed to save you time & pain

• WARNING: Many students dropped during project 1
  => started too late!
Collaboration

• Working together important
  • Discuss course material
  • Work on problem debugging

• Parts must be your own work
  • Homeworks, midterm, final, solo projects

• Team projects: both students should understand entire project

• What we hate to say: we run cheat checkers...

• Please *do not* put code on *public* repositories

• Partner problems: Please address them early
Late Work

• 10% penalty per day

• Cannot be more than 2 days late
  • (no exceptions after 48 hours of due date/time)

• Usual exceptions:
  • documented medical, emergency, etc.

• Talk to us early if there’s a problem!

• Regrade requests in writing to course admin
Why take this course?

• Huge amounts of computing are now distributed...
  • A few years ago, Intel threw its hands up in the air: couldn’t increase GHz much more without CPU temperatures reaching solar levels
  • But we can still stuff more transistors (Moore’s Law)
  • Result: Rise of multi-core and GPUs.
  • Result 2: Your computer has become a parallel/distributed system

• Oh, yeah, and that whole Internet thing...
  • my phone syncs its calendar with google, which I can get on my desktop with a web browser, ...
    • (That phone has the computing power of a desktop from 10 years ago and communicates wirelessly at a rate 5x faster than the average american home could in 1999.)
  • Stunningly impressive capabilities now seem mundane. But lots of great stuff going on under the hood...
  • Most things are distributed, and more each day
If you find yourself ...

• In Hollywood....
  • ... rendering videos on clusters of 10s of 1000s of nodes?
  • Or getting terabytes of digital footage from on-location to post-processing?

• On Wall Street...
  • tanking our economy with powerful simulations running on large clusters of machines
  • For 11 years, the NYSE ran software from Cornell systems folks to update trade data

• In biochem...
  • using protein folding models that require supercomputers to run

• In gaming...
  • Writing really bad distributed systems to enable MMOs to crash on a regular basis

• Not to mention the obvious places (Internet-of-Things Anyone?)
What Is A Distributed System?

“A collection of independent computers that appears to its users as a single coherent system.”

• Features:
  • No shared memory – message-based communication
  • Each runs its own local OS
  • Heterogeneity

• Ideal: to present a single-system image:
  • The distributed system “looks like” a single computer rather than a collection of separate computers.
Characteristics of a DS

• Present a single-system image
  • Hide internal organization, communication details
  • Provide uniform interface

• Easily expandable
  • Adding new servers is hidden from users

• Continuous availability
  • Failures in one component can be covered by other components

• Supported by middleware
Distributed System Layer

Figure 1-1. A distributed system organized as middleware. The middleware layer runs on all machines, and offers a uniform interface to the system
Goal 1 – Resource Availability

• Support user access to remote resources (printers, data files, web pages, CPU cycles) and the fair sharing of the resources

• Economics of sharing expensive resources

• Performance enhancement – due to multiple processors; also due to ease of collaboration and info exchange – access to remote services

• Resource sharing introduces security problems.
Goal 2 – Transparency

• Software hides some of the details of the distribution of system resources.
  • Makes the system more user friendly.

• A distributed system that appears to its users & applications to be a single computer system is said to be transparent.
  • Users & apps should be able to access remote resources in the same way they access local resources.

• Transparency has several dimensions.
## 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 &amp; resource access (enables interoperability)</td>
</tr>
<tr>
<td>Location</td>
<td>Hide location of resource (can use resource without knowing its location)</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide possibility that a system may change location of resource (no effect on access)</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide the possibility that multiple copies of the resource exist (for reliability and/or availability)</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide the possibility that the resource may be shared concurrently</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide failure and recovery of the resource. How does one differentiate between slow and failed?</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that resource may be moved during use</td>
</tr>
</tbody>
</table>
The Joys of Real Hardware

Typical first year for a new cluster:

~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)
~1 **rack-move** (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 **network rewiring** (rolling ~5% of machines down over 2-day span)
~20 **rack failures** (40-80 machines instantly disappear, 1-6 hours to get back)
~5 **racks go wonky** (40-80 machines see 50% packet loss)
~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, etc.**

slide from Jeff Dean, Google
Goal 3 - Openness

- An **open distributed system** “…offers services according to standard rules that describe the syntax and semantics of those services.” In other words, the interfaces to the system are clearly specified and freely available.

  - Compare to network protocols, Not proprietary

- **Interface Definition/Description Languages (IDL):** used to describe the interfaces between software components, usually in a distributed system

  - Definitions are language & machine independent
  - Support communication between systems using different OS/programming languages; e.g. a C++ program running on Windows communicates with a Java program running on UNIX
  - Communication is usually RPC-based.
Open Systems Support ...

- **Interoperability**: the ability of two different systems or applications to work together
  - A process that needs a service should be able to talk to any process that provides the service.
  - Multiple implementations of the same service may be provided, as long as the interface is maintained.
- **Portability**: an application designed to run on one distributed system can run on another system which implements the same interface.
- **Extensibility**: Easy to add new components, features
Goal 4 - Scalability

• Dimensions that may scale:
  • With respect to size
  • With respect to geographical distribution
  • With respect to the number of administrative organizations spanned

• A scalable system still performs well as it scales up along any of the three dimensions.
Summary: Goals of DS

- Resource accessibility
  - For sharing and enhanced performance
- Distribution transparency
  - For easier use
- Openness
  - To support interoperability, portability, extensibility
- Scalability
  - With respect to size (number of users), geographic distribution, administrative domains
Enough advertising

• Let’s look at one real distributed system
• That’s drastically more complex than it might seem from the web browser...
Lets say you were wondering what President Trump is upto today ... ?!? 

... wonder what the secret is of his amazing hairdo! ..
Domain Name System

- Naming! DNS translates names to IP addresses

- **Decentralized** - admins update own domains without coordinating with other domains

- **Scalable** - used for hundreds of millions of domains

- **Robust** - handles load and failures well
A Google Datacenter
How big? Perhaps one million+ machines

but it’s not that bad...

usually don’t use more than 20,000 machines to accomplish a single task.

[2009, probably out of date]
Search for “Trump hairdo”
2007: Universal Search

Ad System → Frontend Web Server

query → Super root

Super root → Cache servers

Images → Local → News → Video → Blogs → Books

Web

Indexing Service

slide from Jeff Dean, Google
Split into chunks: make single queries faster

Front-end

Replicate: Handle load & failures

GFS distributed filesystem Replicated + Consistent + Fast
How do you index the web?

• Get a copy of the web.
• Build an index.
• Profit.

There are over 1 trillion unique URLs
Billions of unique web pages
Hundreds of millions of websites
30?? terabytes of text
• Crawling -- download those web pages
• Indexing -- harness 10s of thousands of machines to do it
• Profiting -- we leave that to you.

• “Data-Intensive Computing”
Why? Hiding details of programming 10,000 machines!

Programmer writes two simple functions:

map (data item) -> list(tmp values)
reduce (list(tmp values)) -> list(out values)

MapReduce system balances load, handles failures, starts job, collects results, etc.
All that...

• Hundreds of DNS servers

• Protocols on protocols on protocols

• Distributed network of Internet routers to get packets around the globe

• Hundreds of thousands of servers

• ... to find out what’s the deal with Trump’s hair!
A glimpse on 15-640

• Frid 9/1: communication recap
• Wed 9/6: consistency and synchronization
• Frid 9/8: no lecture

• Sept: RPCs, concurrency, crash recovery
• Oct: consensus, midterm, real-world examples
• Nov: fault models & security

• CMU-Alert test this Friday (9/1)

• Optional communication review
  • this Saturday noon – 2pm, GHC 4307
Welcome! Course Staff

Instructors

Yuvraj Agarwal  Srini Seshan  Daniel Berger

10 TA’s

Lydia Mankins  Sandeep Agarwalla  Gaurangi Saxena  Zebing Lin  Leiyu Zhao

Junhong Li  Vamshi Reddy Konagari  Felipe Archondo  Devdeep Ray  Daehyeok Kim
WHEN I STARTED PROGRAMMING, WE DIDN'T HAVE ANY OF THESE SISsy "ICONS" AND "WINDOWS."

ALL WE HAD WERE ZEROS AND ONES -- AND SOMETIMES WE DIDN'T EVEN HAVE ONES.

I WROTE AN ENTIRE DATABASE PROGRAM USING ONLY ZEROS.

YOU HAD ZEROS? WE HAD TO USE THE LETTER "O."