Reliability of Cloud-Scale Systems (CS 598XU)

So, let the fun begin.

Tianyin Xu

Spring 2022

Metadata

- Instructor: **Tianyin Xu**
- TA: **Siyuan Chai**
- Website: https://cs598txu-uiuc.github.io/spring22/
- Piazza: Add yourself on Piazza if you are not there.
- Time: W/F 15:30—16:45pm
- Office hour: 16:45—18:00pm
- **Format:** Still subjective to changes ☹
  - Shall we make it online fully? Still debating

$ Whoami

- Ass prof @ CS
- Working on software/system reliability
- Played with Facebook data centers
  - happy to chat about industry versus academia
- Grad school at UC San Diego.
  - I thought nobody would come here from SD
- Applied twice for grad school.
  - I failed the first time.
  - persistence >> genius
  - understanding / experience >> wild ideas
  - doers >> talkers

Who is our TA?

- CS PhD student
- Working on OS/memory
- Undergrad: Northwestern
- Hobbies
  - Soccer
    - Weekly on campus soccer game for fun
    - Welcome to join!
  - Long distance running/cycling
    - Favorite trail: Route 66, Chicago to Santa Monica
  - Peking Opera
    - Try this video if you want to get a taste!

What is this course about?

- **It's all about System Reliability Research**
  - Develop a systematic understanding of *system reliability*
  - Grasp the basic knowledge of *system reliability techniques*
  - Discuss the important *research* papers
  - Get feet wet in *research* (by doing a mini research project)

- **This is a course about:**

  "discussing research" + "doing research"

  (You will be evaluated based on how well you discuss papers and how good your research is)

Goals and Non-goals?

- **Goals**
  - Explore range of problems in system reliability
  - Understand how to identify reliability issues in your own research and how to think about them
  - Figure out if system reliability is a topic of interest for you
  - If so, serve as a forcing function to read papers
  - Get feet wet in research (mini research project)
- **Non-goals**
  - Review of basic OS and distributed system concepts
    - Take CS 423 (with me!) or 425

What does the course look like?

- Watch videos/talks
- Chat about the talks
  - The problems
  - The solutions
  - The authors
- Do research projects
  - Could be something you plan to do anyway

What is the load of the course?

- **Reading**
  - 4 papers per week and small homework
  - 15% of your final score
- **Assignments**
  - Fun analysis to make sure you look into real stuff
  - 10% of your final score
- **Discussion in class**.
  - 10% of your final score is participation
  - It matters!
- **Project**
  - This is the main purpose of the class (70% of your final score)

No midterm or final exam.

It sounds like easy/free credits?

- Yes. It is a 598 course.
- The goal is not to make it hard.
  - I don't mind if you say it's an easy course.
- I hope we will have fun…
- And/or learn something useful.

It looks like a PhD-only course?

- If doing a research project is not something for you, the course is likely a bad fit.
  - 70% of your grade comes from a mini-research project
  - There won't be handheld instructions on how to find problems, how to read related work, and how to reason about novelty and contributions

Readings

- **There is no textbook for this class.**
  - Reference: The Google SRE Book (free!)
  - https://sre.google/sre-book/table-of-contents/
- **We will see two types of papers.**
  - one from industry about state-of-the-art practices
  - one from academia about novel ideas/proposals
- **You are required to read two papers before the class and submit a homework.**
  - The forms will be sent to you via Piazza right after the class.
    - **due 11:59pm Mon/Wed**
  - It is mostly in the form of asking you to write down your thoughts, questions, and discussion points.

How to read a research paper

1. What are **motivations** for this work?
2. What is the proposed **solution**?
3. What is the work's **evaluation** of the proposed solution?
4. What is your analysis of the identified problem, idea and evaluation?
5. What are the **contributions**?
6. What are **future directions** for this research?
7. What questions are you left with?
8. What is your take-away message from this paper?

W. Griswold, How to Read an Engineering Research Paper, https://cseweb.ucsd.edu/~wgg/CSE210/howtoread.html

Rule #1:

Do NOT worship.

- **A paper is *not* a "truth" but an "opinion"**
  - You should have your own judgement
- **Critical thinking is a must in grad study**
  - Papers are arguments based on the authors' work.
  - You are welcome to reject the arguments, criticize the approaches, and question the results.
  - You will need to back up your criticisms and rejections.
- **Professors do not know everything.**
  - Treat me as a senior graduate student.
  - I will learn the materials together with you.

Topics we will be covering

- Cloud systems and availability
- Understanding root causes
- Bug detection
- Software testing
- Chaos engineering
- Load and drain tests
- Monitoring
- Recovery
- Tracing
- Diagnosis
- **I'm open to more topics… got any?**

Video Watching and Discussion

- **We are going to watch videos together.**
  - Typically better than me blah-blah-ing
  - We will discuss the technical materials after the video.
  - I also want to learn.
- **You will be co-leading the discussion of one course with me together (yes, reliability needs redundancy)**
  - And, you won't be let know till 3 days before the class
  - Summarize the questions and lead the discussion

Projects

- **A research project related to the broad definition of software/system reliability.**
  - In a group of **2**
  - You will need to know each other to find partners.
    - Additional efforts are needed in this remote world.
  - If you have strong reasons to do a project in a team of more than 2 students, talk to me.
- **Please try to form groups this week.**
  - Send us an email by the end of this week identifying who is in your group

Most projects will fall into:

- Most projects will fall into the category of:
  - **Analysis**: evaluate the reliability of a system of interest by empirical characteristic studies (e.g., bug study, failure analysis) or measurement (e.g., fault injection)
  - **Tool**: develop a new tool that can improve the systems' reliability (e.g., CPMiner, Veriflow)
  - **System**: build a new system that are more reliable in certain aspects than the state of the art (microkernel, SeL4, new consensus protocols).

Examples

- **Study**
  - Yuan et al., Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems, OSDI 2014.
  - Ganesan et al., Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions, FAST 2017.
- **Tooling**
  - Li et al., CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code, OSDI 2004.
  - Sun et al., Testing Configuration Changes in Context to Prevent Production Failures, OSDI 2020.
- **System**
  - Hawblitzel et al., IronFleet: Proving Practical Distributed Systems Correct, SOSP 2015.
  - Ongaro and Outsterhout, In Search of an Understandable Consensus Algorithm, USENIX ATC 2014.

Project Timeline (12 Weeks in Total)

- **End of Week 3: Submit project proposal**
  - A well-defined research problem and feasible solutions.
  - Show the feasibility by concrete examples, datasets, and tools for system building.
  - **You can try me the idea before deciding on the project.**
- **End of Week 7: Submit Checkpoint 1 report**
  - Show your system/tool prototype and preliminary results.
  - Your prototypes should be able to work with your motivating examples.
- **End of Week 11: Submit Checkpoint 2 report**
  - (At this point, you are expected to build your system/tool and start evaluation)
  - Describe the detailed evaluation plan in your report.
- **Final project demo (15 min)**
- **Submit final project report (6 pages)**

Exploring your project NOW!

- **Initial project proposals due in 3 weeks (one page)**
  - What you plan to do
  - Why is it interesting
  - How you'll do it (feasibility)
  - What you're not sure about

} **Problem Statement**

Project Grading

- **A to A+:** significant results and publishable work;
- **A- to A:** strong results and a clear roadmap to publishable work;
- **B+ to A-:** interesting results but quite far from being significant;
- **B to B+:** a good exploration but leads to nothing;
- **B- to B**: some efforts of exploration but no conclusion.

(You should feel free to explore and fail)

- **Hope: sufficiently interesting to be real papers**
  - at least something you are proud of talking about

Where to find papers to read?

- Best practice: venues where the papers in the reading list are published.
  - OSDI/SOSP
  - ASPLOS (PL and arch)
  - NSDI (networked systems)
  - FAST (file and storage systems)
  - EuroSys (European)
  - SOCC (Cloud systems)
  - USENIX ATC (everything)

Tips

- **Pick a good problem**
  - Why is this problem interesting?
  - What is the impact of solving this problem?
  - Look at what others are doing:
    - Academic conferences: OSDI/SOSP, NSDI, EuroSys, ATC , etc.
    - Engineering blogs and postmortems
- **Pick a problem that is achievable.**
  - Start from small (you only have one semester)
  - What resources would you need to investigate the problem? (ask if you're serious)
- **Think about how to evaluate your work.**

FAQ

- **I am a MCS student. I have no interested in research.**
  - You are likely to suffer (sorry).
  - You can potentially do a project that implements an existing research idea, or build an infrastructure that measure the reliability of a set of systems.
- **I don't want to come to lectures. Can I skip?**
  - Yes, but you will lose many points.
- **Can I simply *reuse* my own research projects?**
  - Unfortunately, yes

What is “reliability”?

- Merriam-Webster online dictionary:
  - **Reliability:** The quality or state of being **reliable**
  - **Reliable:** suitable or fit to be **relied** on
  - **Rely:** to be **dependent**
  - **Dependent:** **relying** on another for support

- “The ability of a system or component to perform its required functions under stated conditions for a specified period of time”

What is reliability?

- Availability, reliability, safety, security

Availability	System is available for use at any time.
Reliability	The system operates correctly and is trustworthy.
Safety	The system does not injure people or damage the environment.
Security	The system prevents unauthorized intrusions.

What is reliability?

- **Most of CS is about providing functionality**
  - User interface
  - Software design
  - Algorithms
  - Operating systems/networking
  - Compilers/PL
  - Vision/graphics
  - Microarchitecture
  - VLSI/CAD

There are reliability problems in all of these domains

- **Reliability is not about functionality**
  - It is about how the embodiment of functionality behaves in the presence of errors and failures

Reliability is like security... somewhat

- Threat model
- Zero day
- Bugs and misconfigurations
- Fail static versus default deny
  - If you don’t know what to do, do the least harmful thing
- Need to mitigate before you know what’s going on
- Both conflict with velocity
- Defense in depth

* Mogul, Thinking about availability in large-scale infrastructures, HotOS, 2017.

Why reliability techniques matter?

- We are living in an unreliable world.
  - but we desire highly available services.

- Hardware breaks.
  - Assume you could start with super reliable servers (MTBF of 30 years)
  - Build a computing system with 10 thousand of those
  - Watch one fail per day

- -> Building fault-tolerance into the software.

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.

  - Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

J. Dean, Designs, Lessons and Advice from Building Large Distributed Systems, LADIS, 2009.

Why reliability techniques matter (cont.)?

- **Software has (many) bugs.**

- **Some bugs could be hard to find.**
  - concurrency bugs
    - not exposed in every simple run
    - manifested in a larger scale
  - latent bugs
    - only manifested under certain circumstances

- **-> Prioritize critical bugs and correctness properties**
- **-> Recover from bugs**
  - revert buggy code changes
  - remove the triggering conditions
- **-> Formal verification**

Consequences of system failures

- **Corrupted:** committed data that are impossible to regenerate, are lost, or corrupted
- **Unreachable:** service is down or otherwise unreachable by the users
- **Degraded:** service is available but in some degraded mode
- **Masked:** faults occur but are hidden from users by the fault-tolerant software/hardware mechanisms
  - Why is this still bad?

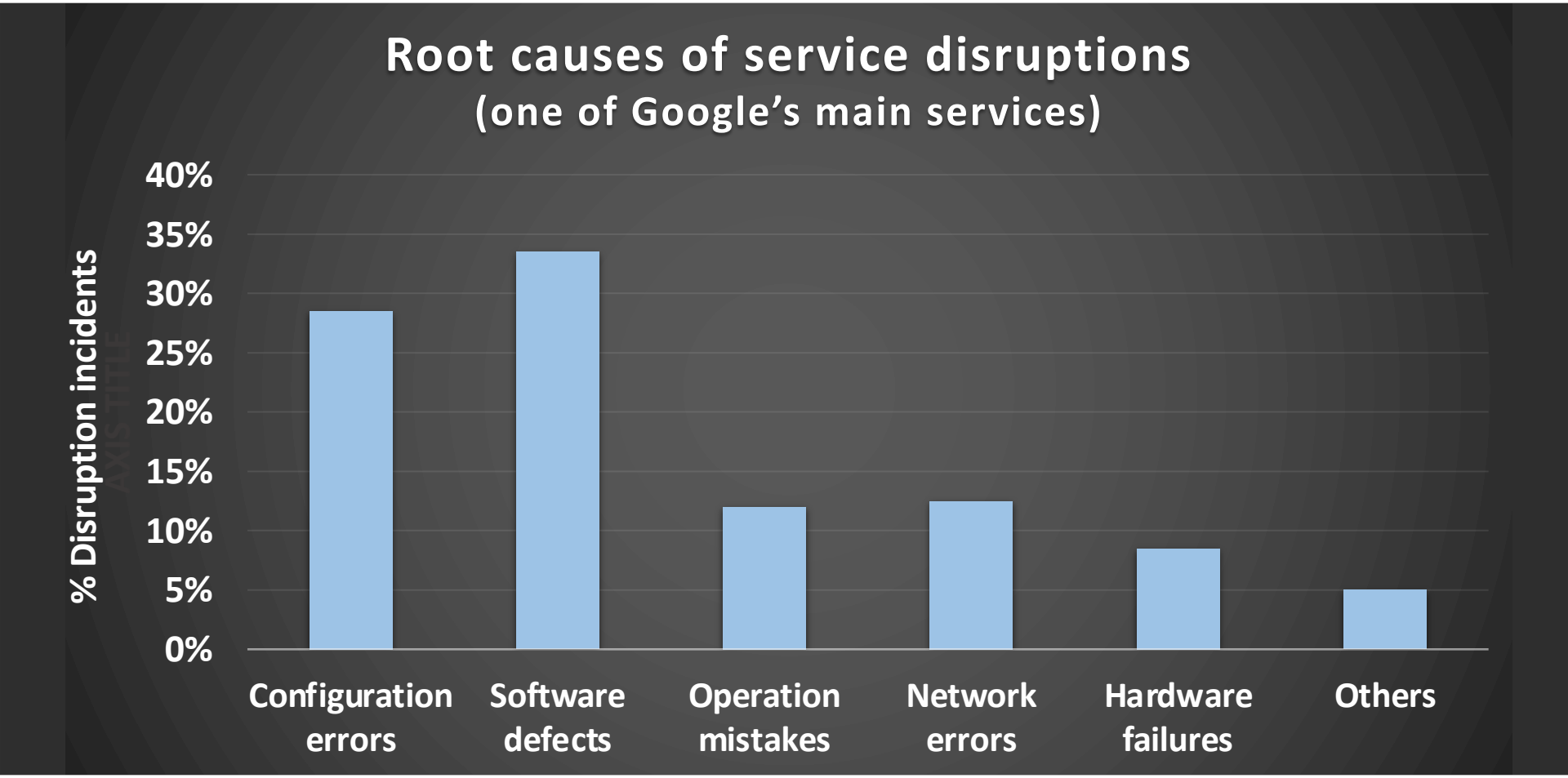
Redundancy for fault tolerance

- **Fault tolerance "is just" redundancy.**
  - Replication for component failures
    - space
  - Timeout and retry for message loss
    - time
  - Write ahead log and data pages
    - representation
  - Error correcting code
    - mathematical
  - K-version programming

Why do systems still fail? (despite all the redundancies)

- Not enough redundancies.
  - failure of entire datacenter
- Redundancies are also affected.
  - corrupted
- Redundancies are faulty (but unnoticed).
  - error handling code is often buggy.
- (Often Hidden) SPOF
  - e.g., failure recovery chain

How do systems fail in the field?



L. A. Barroso, J. Clidaras and U. Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse Scale Machines (2nd Edition), Morgan & Claypool Publishers, 2013.

How to study reliability?

- **Understand how systems break in the past**
  - postmortem and SEV reviews
  - bug study and analysis
  - field failure data
- **Prevent similar problems in the future**
  - understand failure modes
  - examine deficiency of existing mechanisms
  - learn from your own or others' mistakes

How to study reliability?

- **Examine how systems behave in the presence of realistic faults and errors**
  - thought experiments
    - reliability review
  - fault injection
    - fuzzing
  - production experiments
    - DiRT and Game Day

How to evaluate new reliability methods/tools?

- Mathematical proofs
  - Formal verification
- Error injection
  - Inject errors and see how systems behave
- Evaluation with historical data
  - Misconfiguration detection
- Finding defects in existing systems/software
  - Bug detection