

Advanced Operating Systems (CS 523)

Introduction

Tianyin Xu

Metadata

- Instructor: **Tianyin Xu**
- Teaching assistant: **Tyler Gu**
- Course website (with course schedule)
 - <https://cs523-uiuc.github.io/spring23/>
- Piazza
 - Link is on the course website

\$ Whoami

- Ass prof @ CS
- Working on software and 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

What is this course about?

- It's all about ~~Operating~~ **Systems Research**
 - Develop a systematic understanding of *systems research*
 - Grasp the basic knowledge of *systems research*
 - Discuss the seminal *systems research* papers
 - Get feet wet in *systems research* (mini research project)
- This is a course about:
 - “**discussing systems research**”
 - + “**doing systems research**”

This course does **NOT** teach:

- Basic concepts of operating systems
- The skills of hacking an operating system kernel
 - Kernel hacking experience is not required for 523.
 - Systems research is much broader than OS kernel.
- **CS 423 / ECE 391** is the choice if you want to learn the above.
- There is a prerequisite quiz on the course website.

You are expected to:

- **Read research papers (before the class):**
 - 1—2 papers for each class
- **Discuss the reading (in class)**
 - Answer questions in the class
 - Again, it's hard to discuss if you don't read.
- **Present a paper in the class**
 - We will send out the signup sheet.
 - 30% of your grade.
- **Conduct a semester-long mini-research project**
 - The best way to learn is to do it
 - The main purpose of the course -- **55% of your grade**

No homework, MP, or final exam!

Who are the target students?

- Students who are actively doing systems research.
 - Review classic, seminal papers
 - Explore and discuss new ideas
 - Try out new, wild ideas
- Students who are interested in systems research.
 - Evaluate if systems research is something for you.
- If you are neither of the above, you may reconsider.

Paper Reading

- **Reading papers is one of the most important skill sets in grad school.**
 - You need to learn how to **efficiently** and **effectively** read research papers
 - You will be slow in the beginning and be faster and faster
 - If you don't practice reading, you never know how.
- **We will read a lot of classic papers.**
 - Those are the must-to-read papers for systems research.
 - Some of them will appear in the SysNet qual exam.
 - It's hard to innovate if you don't know the literature well.

Rule #1:

Do NOT worship any paper or author.

- **A paper is *not* a “truth” but an “opinion”**
 - You should have your own judgement
- **Critical thinking is a must in grad school**
 - 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.
- **Plenty of horrible papers published in top venues.**
 - But you need a legit reason to “attack.”

How to read a research paper? (Griswold's version)

1. What are the **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?

Topics we will be discussing

- Historical Perspectives
- Unix and Plan 9
(and MINIX and Linux)
- Microkernel
- Library OS
- Synchronization
- Scheduling
- Memory Management
- Virtualization
- Storage and File Systems
- Communication
- Distribution
- Protection
- Reliability

Class Presentation and Discussion

- **You will be presenting one paper in a class.**
 - Tyler will send out the signup sheet.
 - The presentation counts **20%** of your grade.
 - Make sure you really understand the paper you present.
- **You're expected to ask questions.**
 - That counts your class participation (**15%**).
 - If I don't know you in the end of the semester, then you won't get the points.
- **If you don't have questions, I'll ask you questions via cold calls 😊**

Open Discussion (right after class)



- Discuss and debate your research ideas
- See what other teams are working on
- Anything you need help from me or Tyler

Course Project

- A **research** project fitting in the broad definition of “computer systems.”
 - In a group of **1** or **2**.
 - If you have strong reasons to do a **large** project in a team of more than 2, talk to us first.
 - Exceptions are never problems, as long as they make sense.
- **Please form groups before the end of next week.**
 - Send us an email by the end of next week identifying who is in your group

We take a very **broad** and **inclusive** view of systems research.

- It is well connected to areas like architecture, PL, SE, HPC, networking, and embedded/mobile.
- Security and reliability are essential aspects of system design and implementation.
- Everyone is talking about Sys4ML and ML4Sys.
- It can be even broader, e.g.,
 - Visualizing large-scale system data (e.g., logs/traces)
 - Human factors in system operations
 - OS Education
 - Crypto for OS

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)**

Most projects fall into the following categories:

- **Study**: qualitatively or quantitatively analyze an important aspect of a type of systems.
- **Measurement**: measure/characterize an important aspect of a type of systems through experiments.
- **Tooling**: design and implement a new tool that can address an important problem in modern systems
- **System**: design and implement a novel system with new capabilities or properties

Examples

- **Study**

- Chou et al., An Empirical Study of Operating Systems Errors, SOSP 2001.

- **Measurement**

- Pillai et al., All File Systems Are Not Created Equal: On the Complexity of Crafting Crash-Consistent Applications, OSDI 2014.

- **Tooling**

- Li et al., CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code, OSDI 2004.

- **System**

- Rosenblum et al., The Design and Implementation of a Log-Structured File System, SOSP 1991.

Evaluation of Research Projects

- It will be evaluated using the same criteria as SOSP/OSDI submissions.
 - **Overall merit**
 - Importance of the topic
 - Originality and insightfulness
 - Validation and thoroughness
 - Presentation and clarify
- **Dream bar**: CP-miner, Veriflow
- **High bar**: Sufficiently interesting to be a real paper
- **Low bar**: Something you can brag about

Project Grading

- **A to A+:** significant results and publishable work;
- **A- to A:** strong results and a clear roadmap towards 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; no conclusion.

(You should have the courage to explore and fail)

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.**

This is an **easy course.**

(if you know how to do research)

- **No homework, no MPs, no exams!**
 - Research is not about following instructions
- **You're supposed to do research as a grad student**
 - You earn credits by doing what you're supposed to do.
- **You're supposed to read these papers.**
 - Qual exams for PhD students
 - Those are the papers every systems student reads.

This is a **hard course.**

(if you are not interested in research)

- **There is no lecture, but discussion.**
 - You are expected to understand the papers.
 - Forming study groups is always very helpful!
- **You are expected to come up with your own ideas.**
 - It's better to have some concrete directions
- **You're expected to independently design, implement, and evaluate your ideas.**
 - All what we will do is to push you.
 - It will be hard for us to help you debug your code.
- **It does not fit in the agenda of MCS program.**
 - You are welcome to try.

Systems Research Conferences

- SOSP/OSDI (one conference with two names)
- ASPLOS (arch + PL + OS)
- NSDI (networked systems)
- FAST (file and storage systems)
- EuroSys (European)
- MobiSys (mobile systems)
- USENIX ATC (everything)

Finding Teammates **NOW!**

- **Piazza (the “Search for Teammates!” section)**
 - I’m Tianyin Xu, a 9th year grad student
 - I’m interested in reliability – I enjoy watching failures.
 - I have an idea on configuration management [2000 words]
- **We will give you 5-minute to pitch ideas at the beginning of each class.**
 - Send me or TA an email if you want to do it.

Exploring your project **NOW!**

- **Project proposals due in 3 weeks (one page)**

- What do you plan to do?
- Why is it interesting?
- How you'll do it (feasibility)?
 - What is the basic idea?
- What's your plan and schedule?
- What you're not sure about?
- What resources you need?
 - We can provide VMs



Problem Statement

How to find good (research) ideas?



Marinov 4-way method:

- Talking
- Reading
- Hacking
- Dreaming

Marinov method, **the bad way**

- **Talking**
 - "Hey, you are the advisor – tell me a damn OSDI idea."
- **Reading**
 - Find ideas by reading the Limitation section
- **Hacking**
 - Let me read Linux kernel before doing research.
 - Let me take CS 423 before doing research.
- **Dreaming**
 - Sleep → Apple → Gravity

Marinov method, **the good way**

- **Talking**

- Articulate the problem you want to solve
 - Why it's important and why it hasn't been addressed?
- Pitch your idea with concrete examples or data points
 - Why your idea can solve the problem in a better way?

- **Reading**

- Seek for inspiration, but I don't find it too useful.

- **Hacking**

- Look at the design and implementation of existing systems
- Hack those systems to see if only a hack is needed

- **Dreaming**

- What's your wish list?

Tips that may be useful (I)

- **Understanding the problem first!**
 - Innovation without understanding leads to BS.
 - **Understanding itself could be a huge contribution.**
 - Empirical studies and measurements are great ways to develop understanding.
 - E.g., Goto statement considered harmful
 - If you have a topic/direction/problem but don't have a crisp idea, work on a study or a measurement.
 - Ask yourself questions – let your curiosity guide you.

Examples of Studies

- Chou et al., An Empirical Study of Operating Systems Errors, SOSP 2001.
 - Study bugs in the OS kernels
- Lu et al., Learning from Mistakes — A Comprehensive Study on Real World Concurrency Bug Characteristics, ASPLOS 2008.
 - Study concurrency bugs
- Lu et al., A Study of Linux File System Evolution, FAST 2013.
 - Study code changes in file systems
- Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-intensive Systems, OSDI 2014.
 - Study catastrophic failures in data systems

Examples of Measurement

- Ren et al., An Analysis of Performance Evolution of Linux's Core Operations, SOSP 2019.
 - Measuring system call performance
- Ganesan et al., Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions, FAST 2017.
 - Measuring fault tolerance by fault injection testing
- Li et al., Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency, SOCC 2014.
 - Measuring and analyzing tail latency

Tips that may be useful (II)

- **Build on top of your experience**
 - Systems work is mostly about experiences.
 - You need to know the systems well, before doing research (or development).
 - Don't attack a type of systems that you have no experience in your project.
 - If you never hacked the Linux kernel, don't do a kernel project (taking 423 first).
 - Scratching your own itch
 - The dreaming idea – what do you wish to have?

Tips that may be useful (III)

- What system **properties** you want to improve?
 - **Performance** – making systems run fast
 - **Reliability** – dealing with failures
 - **Security** – dealing with attacks
 - **Usability** – easy to use / less error-prone
 - **Manageability** – systems need to be managed
 - **Compatibility / Portability**
 - **Scalability** – making systems run at large scale
 - **Energy-efficient / environment-friendly**
- *Pick one; don't try to do everything*

Tips that may be useful (IV)

- What are **driving forces** for new system research?
 - **Hardware** (multiprocessor, big memory)
 - **Network** (distributed systems, P2P)
 - **Scale** (datacenters, edge)
 - **Application** (GFS, MapReduce, Haystack)
 - **Computing model** (cloud, serverless)
 - **Operation** (DevOps, microservices)

Tips that may be useful (V)

- **Ideas are pipedreams without **execution**.**
 - **No system can be done in one day.**
 - **Manage your time well.**
 - Surprisingly, this is often what makes the difference.
 - **Insights often come from doing things.**
 - Get your hands dirty
 - **Think about evaluation at the design phase**
 - How will you evaluate?

“Can you post the projects that have been done in the past?” (I)

- Understanding Configuration Dependencies in Cloud Systems
 - A comprehensive study of configuration dependencies of cloud systems (including Hadoop, HBase, Spark, etc)
 - Published at FSE’20 (a top Software Engineering conference)
- Cozart: Automatic disaggregation for off-the-shelf OS kernels
 - A kconfig-based OS kernel debloating tool
 - Published at Sigmetrics’20
 - Selected as a research highlight by CACM
- Characterizing Reliability Issues in a Large Deployment of Lustre Distributed File System
 - A comprehensive analysis of storage failures of a distributed file system deployed in the Blue Water supercomputer
 - Published at SC’20 (a top Supercomputing conference)
 - Nominated for both Best Paper and Best Student Paper

“Can you post the projects that have been done in the past?” (II)

- Forensic Analysis of Configuration-oriented Cyber Attacks
 - Built a forensic analysis infrastructure for configuration-change tracking and integrate it with provenance-based forensic analysis
 - Will appear at NDSS'21 (a top Security conference)
- Leyenda: An Adaptive, Hybrid, Radix Sorting Algorithm for Large-scale Data
 - Sorting 60GB data in 290 seconds
 - Runner-up at SIGMOD'19 Sorting Contest.
- Toller: Improving Android UI testing infrastructure
 - Optimize the perf. of 3 key operations in Android UI testing framework
 - Published at ISSTA'21 (a top Software Engineering conference)

What's the secret of publishing my 523 project at top conferences?

- I have no magic – good problem + good execution
- It takes longer than a semester.
 - Unless you are working on some ML/DM stuff.
- The successful ones are typically connected to the student's own interests and/or research
 - Leo Chen (SRE and configuration management)
 - Austin Kuo (OS kernel specialization and debloating)
 - Wajih Ul Hassan (provenance and forensic analysis)
- Time management and hard work