

CS 518

Advanced Operating Systems

(formerly CS597d)

Randy Wang
Fall, 2000
Princeton University

Outline

- Introduction: what, why, and who
- Nuts and bolts/course work
- Course topics
- Historical perspective

Outline

- Introduction: what, why, and who
- Nuts and bolts/course work
- Course topics
- Historical perspective

What

- Literature survey
 - Classic papers since 1960s
 - A few contemporary papers
- Format
 - Mostly lecture
 - Some discussion
- A “substantial” class project
- Basic goal: learn to do a bit system research

Who

- Me: Randy Wang
- Guest lecturers:
 - Prof. Kai Li
 - Prof. J. P. Singh
 - Prof. Larry Peterson
 - Prof. Ed Felten
 - Prof. Andrew Appel
 - Prof. Vivek Pai
- You
 - Know undergrad OS (equivalent of cs318)
 - Advanced undergrads
 - Any junior grads (not limited to systems people)

CS518

4

Randy Wang

Systems Research and Religion

- A lot of scripture reading
- Re-incarnation happens a lot
- Solves a lot of hard problems
- Follow the leaders
- Supposed to cater to everyone

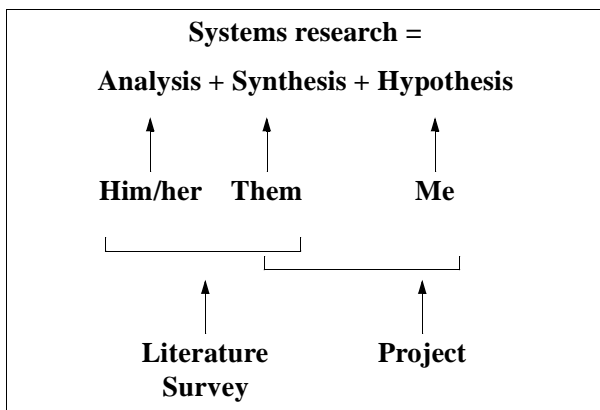
CS518

5

Randy Wang

Why (1)

(scripture-reading)



- For grad students, the name of the game is research

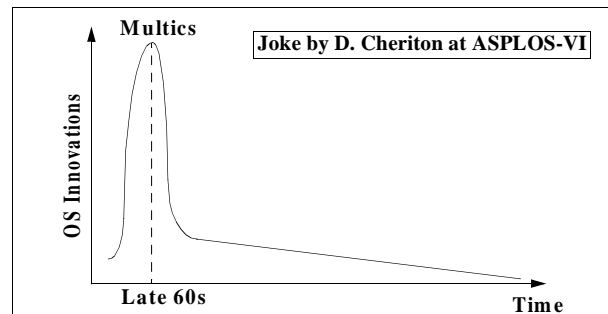
CS518

6

Randy Wang

Why (2)

(reincarnations)



- OS research keeps “going back to the future”
- Good
 - Synthesis
- Bad
 - Re-invent wheels
 - Repeat mistakes

CS518

7

Randy Wang

Why (3) **(solving hard problems)**

- Operating systems solve many hard problems
 - Mediate access to limited resources
 - Provide some guarantees of security and fault-tolerance
 - Provide persistent storage
 - Manage concurrency and sharing
- These problems occur in many other systems
 - Many big program smells like an OS: databases, emacs, Java, browsers, ...
 - Many distributed and parallel applications need to solve these hard issues: eg.: scalable internet servers

Why (4) **(following leaders)**

- Get to know the other systems faculty
 - Their area
 - Their interests
- Maybe do a project related to their interests
 - Undergrads: research, letters
 - Grads: research seminars, research advisors

Why (5) **(catering to the mass)**

But I'm not even a systems person!

- Exposed to how to do research
- Gain breadth
- A chance to find that "special" buddy to work on projects with
- Cross-pollination of neighboring (or not so neighboring) fields

Why Not

- If you don't want to read a lot of papers
- If you're not planning to work hard
- (This is not a seminar.)

Outline

- ~~Introduction: what, why, and who~~
- **Nuts and bolts/course work**
- Course topics
- Historical perspective

Course Web Page

- www/courses/cs518
- protection
 - username: xxxxxxxx
 - password: xxxxxxxx
- from me:
 - readings
 - announcements
- from you:
 - **fill out signup form**
 - **don't forget to include link to a picture of you! (this counts as homework 0)**
 - submit homeworks
 - read others' posts
 - send anonymous feedback to me

Grading

- 30% homework
- 20% two quizzes
- 50% project

Readings

- Almost all are very influential papers
 - **Not** all are good papers
- What constitutes success of a system?
 - Few ever really worked
 - All die at some point
 - Most you can hope for: some **ideas** live on
- What you're looking for
 - Primarily the ideas
 - In most cases, not style
 - In many cases, there are no right or wrong answers
- Read before class, write observations, read again later (before quizzes, for example)

Quizzes

- A few short questions
- Test whether you read the papers
- Test whether you understood the papers

Homework

- 1/4 to 1/2 page “observations” of papers
- Post on-line **before** class, and read others’
- Goals
 - Keep up with reading schedule
 - Encourage creative thinking
 - Think projects early
- Possible contents
 - Do **NOT** copy from paper summary
 - Weaknesses of paper
 - Questions
 - Suggestions of improvement
 - Designs of new experiments
 - Future directions (how to extend or apply ideas)
 - Comparison against related work
 - Whatever!

Projects

- Topics
 - A list of suggestions by middle of semester
 - Some suggestions by other systems faculty
 - You can suggest your own but need my approval
- Meta-content
 - Systems projects typically involve some hacking
 - Pure paper designs are ok, but are not encouraged and they must pass a higher bar
- Procedures
 - Proposal
 - Checkpoint meeting
 - (Selected?) final presentation (?)
 - Final report
- Typically, best 1 or 2 are conference quality publications

Outline

- ~~Introduction: what, why, and who~~
- ~~Nuts and bolts/course work~~
- **Course topics**
- Historical perspective

Course Topics

- Virtual memory
- Kernel structures
- Extensibility?
- Threads and synchronization
- Communication
- Storage
- File systems
- Fault tolerance
- Distributed systems
- Security
- Experiences

Virtual Memory, Kernel Structures, And Extensibility

- Multics -- extreme sharing
- VAX/VMS -- dealing with hardware advances
- Mach -- user level pagers
- Mach -- micro-kernel
- THE -- hierarchical kernel structure
- Hydra -- peer-to-peer subsystems
- Exokernel? -- exposing bare hardware securely
- Synthesis? -- hacks for extreme performance
- Disco? -- virtual machine
- ~3 weeks

Threads, Synchronization, And Communication

- Mesa -- threads and monitors
- SRC threads -- practical guide for programming threads
- TBD -- guest lectures by Prof. Larry Peterson
- RPC -- communication with an emphasis on semantics and ease of use
- Active Messages -- communication with an emphasis on minimalism and performance
- ~2 weeks

Persistence

- Storage: disks, RAIDs
- File systems
 - UFS -- where it all started
 - FFS -- what we all use today, hacks to make Unix work
 - LFS -- log structured, new organization
- Distributed file systems
 - NFS, AFS, Sprite -- client/server systems
 - Coda -- disconnected operations
 - LOCUS, xFS -- peer-to-peer systems
- Fault tolerance
 - Transactions?
 - TARGON -- recoverable Unix
- ~3 weeks

Distributed Systems

- Emerald -- distributed objects
- Ivy, Munin? -- shared virtual memory, guest lecture by Prof. Kai Li or Prof. J. P. Singh
- Grapevine -- an early internet mail system
- Porcupine? -- a state-of-art mail server
- Bayou? -- a peer-to-peer architecture
- LARD -- a state-of-art web server, guest lecture by Prof. Vivek Pai
- Themes: naming, scalability, load balance, availability
- ~2 weeks

Security

- Data security in a DBMS
- Kerberos?
- Java security, guest lecture by Prof. Ed Felten
- Proof carrying code, guest lecture by Prof. Andrew Appel
- ~1.5 weeks

Experiences

- Hoare -- simplicity
- Lampson -- folksy hints on functionality, speed, reliability
- How to give a talk
- ~1 week

Outline

- ~~Introduction: what, why, and who~~
- ~~Nuts and bolts/course work~~
- ~~Course topics~~
- Historical perspective

Historical Perspective of OS

- Phase 1: Expensive hardware, cheap humans
 - Batching processing
- Phase 2: Less expensive hardware, less cheap humans
 - Interactive time-sharing
- Phase 3: Cheap hardware, expensive humans
 - Personal computing
 - Distributed (client/server) computing
- Phase 4: Free hardware
 - ???
- Where are we heading?
 - Are we doomed to going-back-to-the-future?
 - Or are things fundamentally different this time?

Technology Advances Determine OS (the Good News)

	1981	1999	Factor
MIPS	1	1000	1,000
\$/MIPS	\$100K	\$5	20,000
DRAM Capacity	128KB	256MB	2,000
Disk Capacity	10MB	50GB	5,000
Network B/W	9600b/s	155Mb/s	15,000
Address Bits	16	64	4
Users/Machine	10s	<= 1	< 0.1

Hard Problems (the Bad News)

- Latency
- Sharing
- Reliability
- Security

Heavy Historic Baggage

- Game defined by Phase 1-3 of OS development
- Designs and voodoo constants frozen in the early days
- Capacity-centric (storage, computation)
- Generality-centric
- Client/server/hierarchy-centric
- Stationary-location-centric
- Protocol-centric

Flip-Sides

- ~~Designs and voodoo constants frozen in the early days~~
 - Question status-quo?
- ~~Capacity-centric (storage, computation)~~
 - Latency/throughput-centric?
 - Reliability and security?
- ~~Generality-centric~~
 - Special-purpose devices and OS?
- ~~Client/server/hierarchy-centric~~
 - Peer-to-peer?
- ~~Stationary location-centric~~
 - Mobility-aware?
- ~~Protocol-centric~~
 - Code-centric?

Going Back to the Future

- Avoid repeating mistakes
- Avoid re-inventing wheels
- Revive ideas whose time has come
- Ideas in one area of OS reincarnate in other areas
- Ideas in OS reincarnate in other systems