CS380L

CS380L: Mike Dahlin

August 28, 2008

> We are concerned with great research here. Work that will get wide recognition, perhaps even wine Nobel Prize. As most people realize, the average published paper is read by the author, the referee, and perhaps one other person. Classic papers are read by thousands. We are concerned with research that will matter in the long run and become more than a footnote in history. *R. W. Hamming* You and your research

1 Preliminaries

1.1 Review

1.2 Outline

1.3 Preview

2 Class intro

2.1 Why?

Systems research For grad student: research is name of the game
OS is good area to learn systems research (even if you are not an OS or even a systems person)

- Exposure to experimental research
  Most of work we look at will be good.
  Most of work we look at could be improved.
- Gain breadth
  Learn the literature (more importantly: how to think about the literature)
  Cross-pollination of neighboring fields
  - Myth: Most great research is done by young researchers
  - There appears to be a grain of truth
  - Reality:[1] Most great research is done by researchers who entered the field about 7-10 years earlier
    - Usually a young researcher
    - Occasionally a senior researcher who had the guts to try something new
  - Unsubstantiated pet theory: This effect comes when you bring new ideas, new perspective from a different area into a new area

[1] I need the cite on this. I've heard roughly this penonemenom described and attributed to some good studies, but I have not read the studies myself. Still, it is at least a nice theory and it probaly doesn't hurt anything to operate on the assumption that it is true.

- Project opportunities
  Research is name of game for grad student and beyond; learn by doing.
  Rapidly evolving area: a great grad student project can go to top conference

Fundamental problems

- Mediate access to limited resources
- Guarantee security and fault tolerance
- Persistent storage, replication, naming
- Concurrency, sharing

These problems occur in many other systems: many big systems (e.g., database, emacs, JVM, browser), many parallel/distributed systems (e.g., internet services)

3 Admin

3.1 Workload

Goal: good discussions, good projects, actual goal: deep understanding/preparation for research
Non-goals: busy work, create grade distribution, cause stress

Readings Reading: will take a long time (especially at first)
Almost all are very influential papers
Not all are good papers
Most are good examples of experimental technique
Almost all could be improved
Were they successful?
Few ever really worked
All die at some point
Most you can hope for: some *ideas* live on
What to look for:
Primarily the ideas and approach
In most cases, not style
In many cases: no right or wrong answer
Homework/quiz: combination worked well in my last 395t

- Main goal: read and reflect on paper – written critique to focus thoughts. *Short and focused.*
- Critique: *not* paper summary
  What to write down:
  - What is good, what is bad.
  - Questions
  - Suggestions
  - Design of new experiments
  - Future directions
  - Impact today
  - How fits with related work

- Quiz: rather than grade each critique (busy work) we will have occasional quizzes
- Stress: reading and thinking about 1 paper per class is a lot; skip 5 critiques (and you are off the hook on a quiz if you don't turn in a critique

Scribe notes: cancel (for now)
Written homeworks: 4 homeworks focusing on experimental techniques

Exams open book (possibly), in class

Project

- 2 person group
- topic: I will provide list (or you choose)
- goal: USENIX/Middleware/USITS/SOSP/SIGMETRICS/etc paper
- deliverable: milestones + written report
- peer review and feedback: help others with their projects, don't just focus on your own
  - more work
  - Answer 1: Welcome to the community of scholars.
  - Answer 2: You may learn something yourself

3.2 Schedule

Topic schedule
Project schedule

3.3 Professional development

Extra reading and discussion: doing research, presenting research, managing career

4 Levin, Redell, and Lampson

Today two readings – how to build systems and how to write about systems research

As with many classes, also go off on some related discussions not from the readings (the reading is a starting point, not the Truth.)

4.1 Levin and Redell

Levin and Redell "AN evaluation of the 9th SOSP Submissions - or - How (and How Not) to write a Good Systems Paper"

Key quote: *These thirty-odd questions can help you write a better technical paper. Consult them often as you organize your presentation, write your first draft, and refine your manuscript into its final form*

- Summary: rules of thumb for writing good systems papers
  - Most papers bad (easy to reject, not worth reading, etc.)
  - A few researchers publish stuff at top conferences year after year
  - Learn from them or risk a second-rate career

4.1.1 Original ideas

- One key theme: "Doing things right is not as important as doing the right things"
  - *Are the ideas in the paper new?*
  - *How do you know?*
  - *Can you state the new idea concisely*
  - *What, exactly, is the problem being solved?*
- My take
  - Much of this class will focus on *execution.* I often say that execution is what separates SOSP/OSDI authors from the rest. We have lots of ideas good enough to generate an SOSP/OSDI paper, but not too many people are successful at executing the research that 20% better to get it over the top at these conferences.
  - But, if execution separates the top 10% from the remaining 90%, ideas separate the top 1% from the top 10%
    - Work on important problems
    - More likely to have impact
    - Easier to turn into a paper
    - More satisfying!
- Lots of other good advice and themes in this paper.

4.1.2 Types of papers

Levin and Redell: 3 categories

1. Real implemented system (survey of entire system of focus on one part)
2. Unimplented system
3. Theoretical topic

Continuum:

		Effort	Frequency	
			SOSP accept	SOSP reject
Real system	Public production use for extended period of time	huge	rare	occasional
	Production use by people other than author	large	occasional	occasional
Worked once	Benchmarks run	large	common	occasional
Simulated	Simulations run	med	occasional+	common
Paper design	Sounds good	med-	rare	common

- Update:– in above table since 1984, a lot more submissions centered on "worked once" prototypes; fewer paper designs and simulations; fewer real systems
- Very few groups in the world build and evaluate interesting research prototypes
- Majority of SOSP/OSDI papers are about prototypes
- Note: differs by field
  - Networking: more theoretical and simulation (why?)
  - Architecture: more simulation (why?)

- Prototype is not enough
  - "We spent a lot of effort building this, let me tell you about it..."

Why build?

- Patterson circle
  - Is it really a good idea?
    - *With sufficient thrust, pigs fly just fine*
    - Ousterhout: *By the end of page 1, I can tell whether they built it or not*
  - Generate new research ideas
    - What is the real problem?

4.1.3 Evaluation criteria

Original ideas

- Can you state the original idea crisply?
  - Focus the paper (v. "we spent a long time working hard")
- Is the idea significant?
- What lessons were learned?
  - If you didn't learn anything, your readers probably won't either
  - "Close the circle"
  - Papers that discuss limitations of the work are much more interesting than those that don't
- Do you *know* the related work?
  - Literature review is not a chore to avoid wasting time
  - You probably do have a good idea; make it better/crisper by standing on shoulders
    - Word to the wise: the authors of related work are likely to be your reviewers. Politically expedient to cite them, summarize their work accurately, and explain your advance crisply (in addition to being good science).
  - Techniques
    - Follow references back from papers you know
    - Read TOC of recent related conferences
    - Know the groups working in the area; check their pages (And talk to them at conferences)
    - Citation indecies: Citeseer, etc.

4.1.4 Another take on similar themes

- From Guy Kawasaki *the Art of the Start*
  - *Set a timer for one minute. Give your current pitch until the timer goes off. Ask the audience to write down one sentence that explains what your organization does. Collect the answers and compare them to what you think you said.*
  - 10/20/30 rule: a good pitch should be 10 slides, 20 minutes, 30 point font

    - *You're probably thinking,* Guy's referring to the hoi polloi, great unwashed masses, and bozos. They should use only ten slides and twenty minutes, but not us. We have curve-jumping, paradigm-shifting, first-moving, patent-pending technology. *I am, in fact, referring to* you.

    1. title
    2. Problem: Describe the pain you alleviate
    3. Solution: explain how you alleviate this pain
    4. Business model
    5. Underlying magic: Describe the technology, secret sauce, or magic behind your product or service
    6. Marketing and sales
    7. Competition: Provide a complete view of the competitive landscape
    8. Management team
    9. Financial prospects and key metrics
    10. Current status, accomplishments to date, timeline, use of funds

  - Not all appropriate for research project, but I put the list here to point out
    - State the *pain* before the technology magic – make sure you are solving a real problem
    - Research the competition – surprisingly (?) start-ups spend more time reviewing the "literature"/competition than most researchers
    - another idea from start-ups that applies here – "ideas are a dime a dozen. What matters and what is rare is execution"

4.1.5 Presentation

- Lots to say about this...
- State the idea crisply, focus the paper
- Technique: top-down design, design reviews
  - *Write from an outline*
  - Step 1: Paragraph level outline of intro, subsection level outline of paper, graph-level outline (sketch graph, state hypothesis/lesson of each graph) of results section, paragraph level outline of related work
    - Well before paper deadline
      - Avoid useless work
      - Avoid missing important issue in work
    - Complete sentences *forbidden*
    - Talk through outline with colleague (will tighten argument, spot missed steps, etc.)
  - **Write from an outline**
  - Step 2: Point-level outline of intro
    - Complete sentences *forbidden*
    - Talk through with colleague
  - Step 3: paragraph-level outline of each section
    - Complete sentences *forbidden*
    - Talk through with colleague

- Write it
  - Trick: Read first sentence of each paragraph in section – do you have strong topic sentences
  - Trick: Most reviewers have a pretty good idea whether a paper will be accepted or rejected by the end of the intro
  - Expect to re-write from scratch (though design reviews drastically reduce the frequency)
  - Lots more to say...

4.2 Lampson: Hints for Computer System Design

A bunch of hints organized into 3 x 3 grid of categories

WHY →	*Functionality* Does it work?	*Speed* Is it fast enough?	*Fault Tolerance* Does it keep working?
WHERE ↓ Completeness Interface Implementation			

Lots of good advice...read it...

Key quote: *The designer usually finds himself floundering in a sea of possibilities, unclear about how one choice will limit his freedom to make other choices, or affect the size and performance of the entire system.*

Some points that stand out as particularly insightful (or that I disagree with...)

- A key theme is simplicity
  - *Do one thing at a time and do it well.*
    - On p 35 column 2, he gives a bunch of examples
    - My example: allowing read/write of arbitrary byte ranges in PRACTI
  - *Make it fast, rather than general or powerful*
    - I would say "Make it simple"
    - Beware premature optimization
    - Ousterhout: *The biggest speedup is when a system goes from working to not working. That's infinite speedup.*
    - Lampson touches on this *To find the places where time is being spent in a large system, it is necessary to have measurement tools ...*
    - → Design a system with a structure that facilitates incremental tuning
  - *The normal case must be fast, the worst case must make some progress*
    - Some countervaling opinion here – Amazon's Dynamo empasizes good performance for 99.99%-tile of requests; our Aardvark project emphasizes not letting failures knock you off of a cliff
    - Reconcile with lampson – predictable cost
- Another theme is that for performance, simple approaches work, complex ones dont
  - *When in doubt, use brute force, safety first*
    - Moores law v. queuing theory. GRAPH. Clever scheduling only matters when system is overloaded. Given moore's law, answer is often to buy more capacity (or wait for more capacity)

    - Lampson examples: virtual memory, clever job scheduling, ...
    - Other examples: clever networks scheduling for real time (video streaming gets deployed on internet once there is enough capacity to do it)
    - My example: big buffers for disk encryption (1GB sounds like a lot, but in 2008 it is $20 to make a $1K-$10K RAID run twice as fast...)
  - *Compute in background* – asynchrony and concurrency vital in emerging highly parallel world...

- Another theme is fault tolerance
  - *end-to-end error recovery*
    - Modern example – disk errors