CSE 403

Review

Announcements

- Projects due on 12/08
- Final on 12/07, in ARCH 160 at 10:30a-11:20a
- Presentations on 12/10, in ARCH 160 at 8:30a-10:20a

Announcements

- Additional TA office hours in Atrium
  - 2:00 PM Wednesday (Tom)
  - 1:00 PM Thursday (Megan)
  - 3:30 PM Thursday (Anton)
- Additional instructor office hour: 11:30 AM Friday (presentation, wrap up questions)

Final release

- Project deliverable
  - Quantity: How much did you deliver?
  - Quality: How good was it? We'll "deep dive" into at least 2 features (Design/UI/UX matters, bugs, usability)
  - Achievement of vision
- Software engineering principles
  - See wrap up and Joel on Software article
  - Answer all the questions in the wrap up (and more)

Presentation

- What you said you were going to do (1 minute)
- What you did (1 minute)
- What you would do next
  - Next release (1 minute)
  - Strategically (1 minute)
- Issues (1 minute)
- Demo (3 minutes)
- Important things you learned (2 minutes)
- Discussion of checklist (2 minutes)
- Questions and answers (2 minutes)

Wrap up write up

- Goal is to understand your product process and infrastructure in addition to what you built
- Answer all the questions
- Document contains the answers or you make submit a link to an online document
- Okay to hyperlink to the answers
- Anything additional you'd like to add

Presentation Logistics

- You are turning in:
  - Release notes (due 12/08, 11:59PM)
  - Project presentation (due 12/09, 11:59PM)
  - Wrap up write up (due 12/08, 11:59PM)
- Final presentations are on 12/10, 8:30-10:20, in ARCH 160
- You don't have to attend all off them but you can
- Show up at least 20 minutes before your presentation -- we'll let you in on the breaks ("on-deck" -- so you are ready to go in your time slot)

Schedule of presentations

- 8:30: Tile to the Top
- 8:45: Event Hub
- 9:00: Bulletin
- 9:30: Instafeed
- 9:45: Full House
- 10:00 Change My Mood

Final logistics

- December 7, 2012 at 10:30, ARCH 160
- 50 minutes
- Closed book, closed notes, no electronic devices
- Bring a pencil
- True/False but with an explanation
- Short answer

What we've done this quarter

Objectives of this class

- An understanding of the fundamentals of software engineering
- Experience building a software project using software engineering fundamentals
- Useful "stuff" to help you as a software professional

Important takeaways

- "Divide and conquer" is your friend
- Change, ambiguity, and uncertainty are inevitable
- Can't have it all: Cost-benefit analyses are key
- Modularity and abstraction are important tools
- You might be late: What are you going to do?
- You are in a great position for a great career in software engineering!

Tips of the day

- Eight percent of success is showing up
- There are only two hard problems in computer science: Naming, cache invalidation, and off-by-one errors
- Worse is better: Worse is better than better, sooner is better than later, something is better than nothing
- Talk about compensation with your peers
- The grinf*ck
- One-on-one's with your manager are important (look out for the sh*t sandwich)
- Send holiday cards

Software engineering?

More like this...

...or more like this?

Don’t under-estimate your potential for greatness

- You’re getting a great education in computer science and software engineering
- Software is eating the world
- You have knowledge, skills, and tools to make a large impact today

It's a great time to be a software engineer!

Good luck!

Last lecture

- Last minute questions about your project?
- Complete course evaluations
- Review for the exam

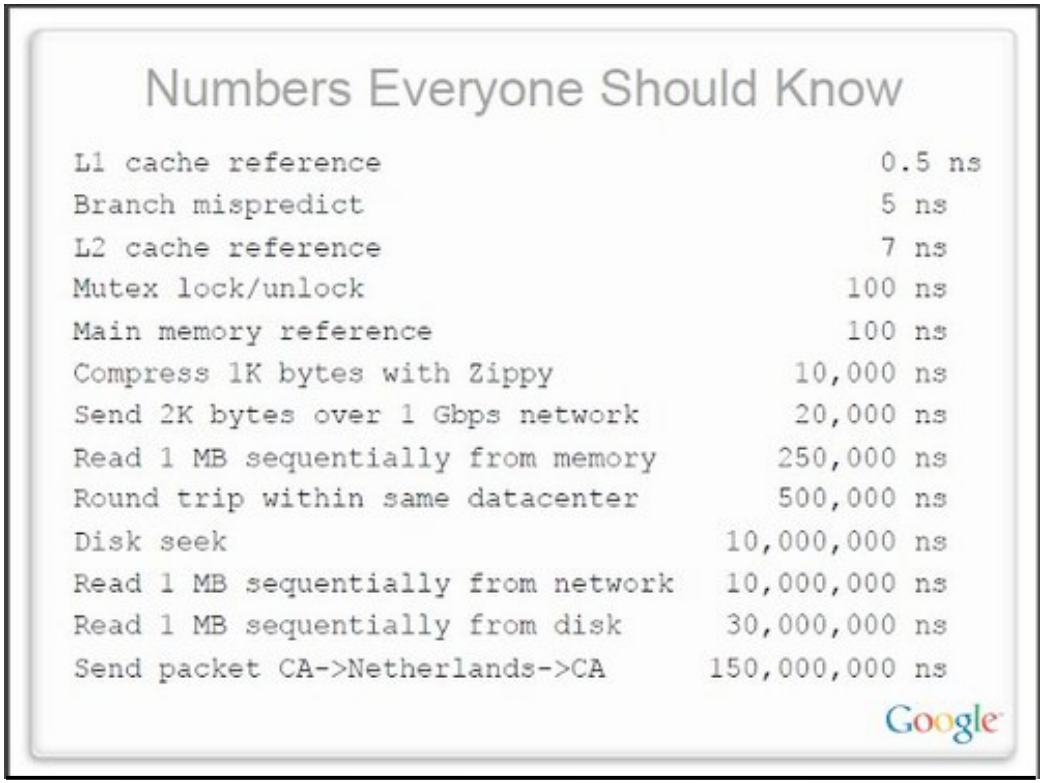
Course evaluations

Review for final

Scale up vs. scale out

- Terms not clear, sometimes interchangeable
- “Up”: beefing up a single system to make it perform better
- “Out”: added more systems (boxes/blades) to make it perform better
- Up == “up” and/or “out”
- Up == vertical, Out == horizontal

Aside: Rules of thumb



Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Google

But 2 years old!

Major topics

- Teams
- Lifecycle
- Requirements
- Architecture
- Design
- Testing
- Women in technology
- Version control
- Deployment
- Testing
- Scale/Megascale
- Careers
- Customers and Users (AARRR!: a pirate's approach)
- Feedback
- Enterprise software
- Lean startup

Key concepts

- Divide and conquer
- Abstraction
- Modules
- Coupling and cohesion
- Parallelism
- Cost-benefit analysis
- Feedback
- Scale
- Examples in software, your project, and in real life

Teams

- Why work in teams?
- More people->more productivity
- More people->less productivity
- Organizational structures
- Roles
- Individuals

Lifecycles

- What are the components of a software lifecycle?
- What are the different kinds of lifecycles?
- When does software get released?

Requirements

- How do we specify requirements?
- Why is specifying requirements hard?
- Who specifies requirements?
- Why do we specify requirements?

Design

- What is the design stack?
- How do we do design?
- How do we specify designs?
- How does UML fit in?
- Who does design?

Architecture

- Why architecture?
- What are the "-ilities" and why are they important?
- Choosing tools: How and why?
- Modularity -- why is this good?
- Web, MVC, pipelines, layer cakes

Testing

- Types of testing
- What is regression testing
- Why do we test?
- When should we test?
- Why do we do code reviews?
- Examples of unit test boundary conditions
- What are bugs?

Women in Technology

Version control

- Why use version control?
- Distributed vs. centralized version control
- Branch strategies and merging
- Development, testing, staging, build, live environments: what are these? how are they different
- Github: Why did we favor this in this class?

Deployment

- How do we deploy software?
- Provisioning vs. deployment
- Tools for deployment
- Operations/maintenance tasks
- Web application deployment vs. software deployments

Scale/Megascale

- How do we measure and profile?
- Scale out: how do we do it?
- How do we scale up a single database?
- Caching
- Rules of thumb: Numbers every software engineering should know
- Scaling through replicating databases: Read/Writing partitioning, horizontal scaling, vertical scaling
- Consistency, CAP theorem
- Megascale

Careers

- 8 tips
- Personal brand
- Interview structure, techniques, strategies

Feedback

- What is feedback?
- Types of feedback
- “Pre-real” feedback
- Live site feedback/testing
- Software engineering performance evaluation

Customers and users (AARRR!: a pirates approach

- Acquisition
- Activation
- Retention
- Referral
- Revenue
- (Internet) Advertising: CPC, CPM, SEO, SEM, viral equation
- Why should engineers care?

Enterprise software

- How is enterprise software different than web applications?
- What was/is the promise of enterprise software
- Why has been the problems of enterprise software in the past?
- How are enterprise software releases different than web application software releases?

Lean startup

- Team
- Hackathons
- Product-Market fit
- Minimum viable product
- Financing

Definitions

divide and conquer, waterfall model, agile, pipelining, product manager, project manager, user interface, information architecture, loose coupling, cohesion, agile, scrum, UML, use cases, modules, interface, implementation, metrics, abstraction, information architecture, runtimes, use cases, UI, UX, IA, regression testing, monkey testing, deployment, provision, scale, horizontal vs. vertical scale, r/w partitioning, important numbers, caching, CAP theorem, consistency, CPC, CPM, CTR, viral growth, ads, ad inventory, AARRR, enterprise software 80-20 promise, lean startup, minimum viable product, hackathon,

True/False Questions

Section 1. True/False. Answer each of the following questions either True or False by circling your answer. You may optionally give an explanation for the answer. Incorrect answers with reasonable explanations may receive partial credit. Correct answers with substantially incorrect explanations may not receive full credit.

True/False questions

1. **TRUE or FALSE.** Scaling through Read-Write partitioning works because there can be multiple "READ" replicants.

2. **TRUE or FALSE.** Scaling through Read-Write partitioning works because there can be multiple "WRITE" replicants.

3. **TRUE or FALSE.** You've made a bunch of changes in a branch that you need to commit. A teammate is also holding a bunch of changes in the same branch that need to be committed. Because there's likely to be a lot of conflicts that need to be resolved. *If you commit your changes second, you'll have fewer conflicts to merge than if you commit first.*

True/False questions

5. **TRUE or FALSE.** In the "Lean Startup" model, the goal of building a prototype is to achieve product market fit once the prototype is done.

6. **TRUE or FALSE.** In software engineering, we use feedback to change and improve a system.

7. **TRUE or FALSE.** Only when you have good feedback should you deploy a software feature.

8. **TRUE or FALSE.** Software deployment is what makes the software available to the customer.

True/False questions

9. **TRUE or FALSE.** It's better to look for bugs early in design and development rather than later.

10. **TRUE or FALSE.** Peer to peer code reviews propagate coding standards.

11. **TRUE or FALSE.** Regression testing uses bugs as a source for test cases.

12. **TRUE or FALSE.** If "strong consistency" is a requirement on your data, Read-Write partitioning is a unlikely to be viable solution for scaling out.

Short answer questions

Section 2. Short Answer. Each question is worth 6 points. Answer the following questions in 1 to 3 sentences. For questions requiring examples (or reasons), answer in (approximately) 1 sentence for each example.

Short answer questions

1. You've created a 30 minute television infomercial for your CSE 403 project that shows at 3:00AM. 2,500 people see it. It costs you $100. What's the CPM?

2. With regards to the previous question, explain an analysis whether or not this is a good deal.

3. Suppose you ascertain that the read-write ratio to your data store is 10:1. What scale up/out strategy might you consider?

4. Suppose further, that your write accesses are 90% creates/ appends as opposed to updates. How does this knowledge help you scale up and what optimization might you be able to make given this knowledge? (Think about how creates are different than updates.)

Short answer questions

5. When things are going really well, your software engineering project might be behind schedule. Why?

6. Your work is really demanding. You want more personal time, outside of work. What are your choices?

7. You've received project requirements that are too large, too complex, ambiguous, and incomplete. You need to do the design and implementation. What choices do you have to successfully execute?

8. You have a fix to show stopper that you want to deploy immediately to the live site. Your fix is "out of band" with your regular release process. How might you go about deploying it?

Short answer questions

9. Roughly speaking, how much faster is it to reference L1 cache vs. main memory? (Order of magnitude is fine)

10. What is viral customer acquisition?

Good luck!

Questions? Stop by for office hours or make an appointment.