Teaching a Mandatory Performance Evaluation Course to Software Engineering Undergrads

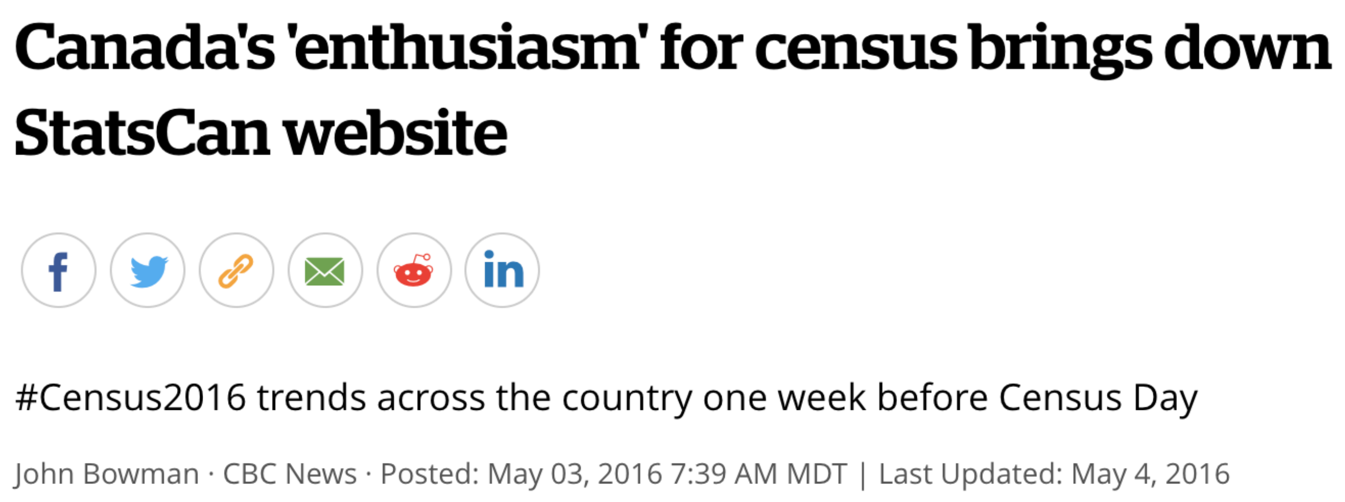
Diwakar Krishnamurthy
Department of Electrical & Software Engineering
(dkrishna@ucalgary.ca)

UNIVERSITY OF CALGARY

Overview

- Importance of performance evaluation (PE) for undergrad software engineers
- Challenges in teaching PE at undergrad level
- Reflections from teaching a mandatory undergrad PE course
    - Strategies to improve student buy-in
    - Open challenges

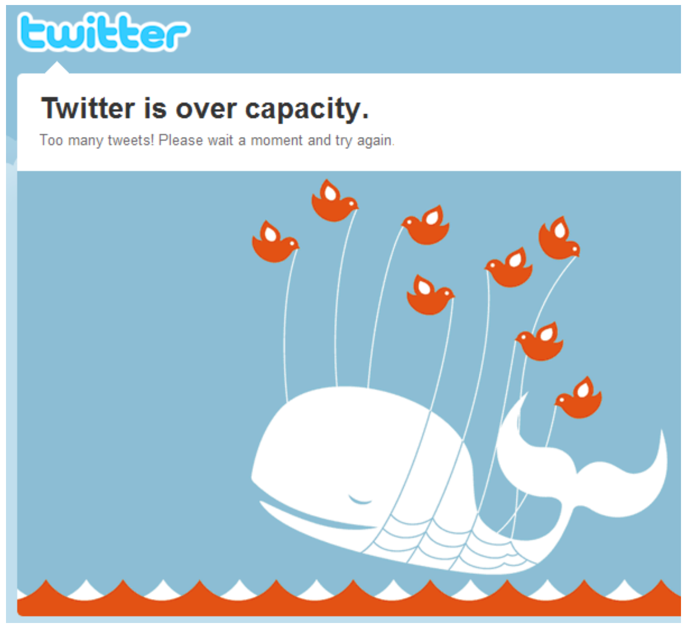
Why does performance matter?



Canada's 'enthusiasm' for census brings down StatsCan website

#Census2016 trends across the country one week before Census Day

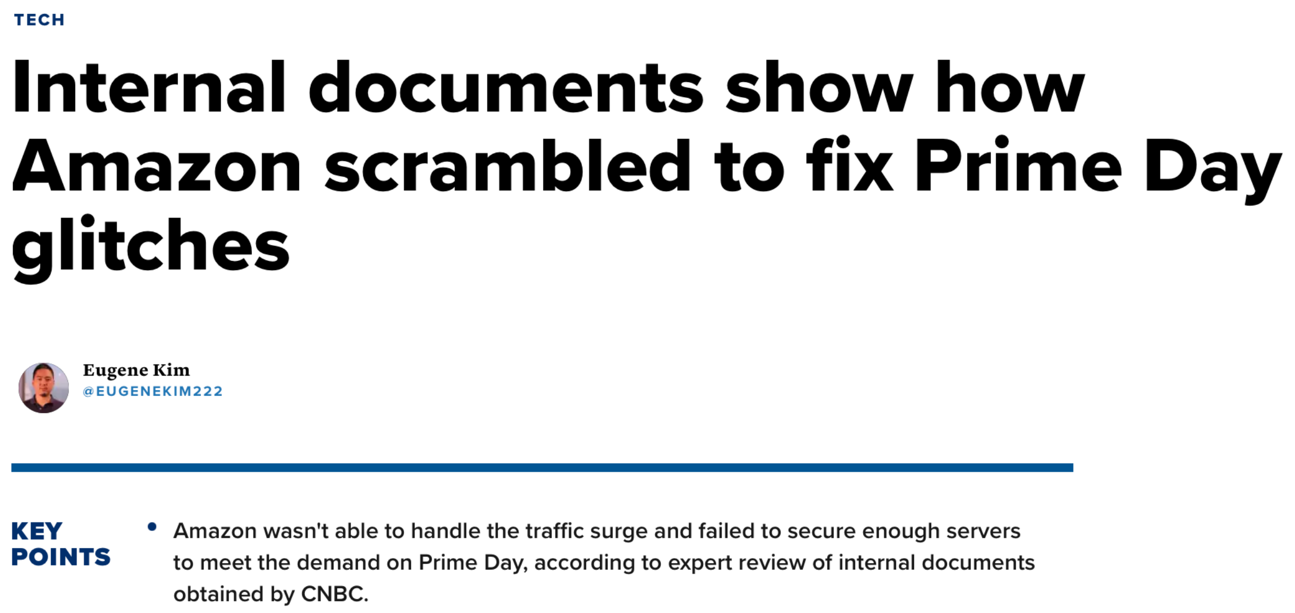
John Bowman · CBC News · Posted: May 03, 2016 7:39 AM MDT | Last Updated: May 4, 2016



twitter

Twitter is over capacity.

Too many tweets! Please wait a moment and try again.



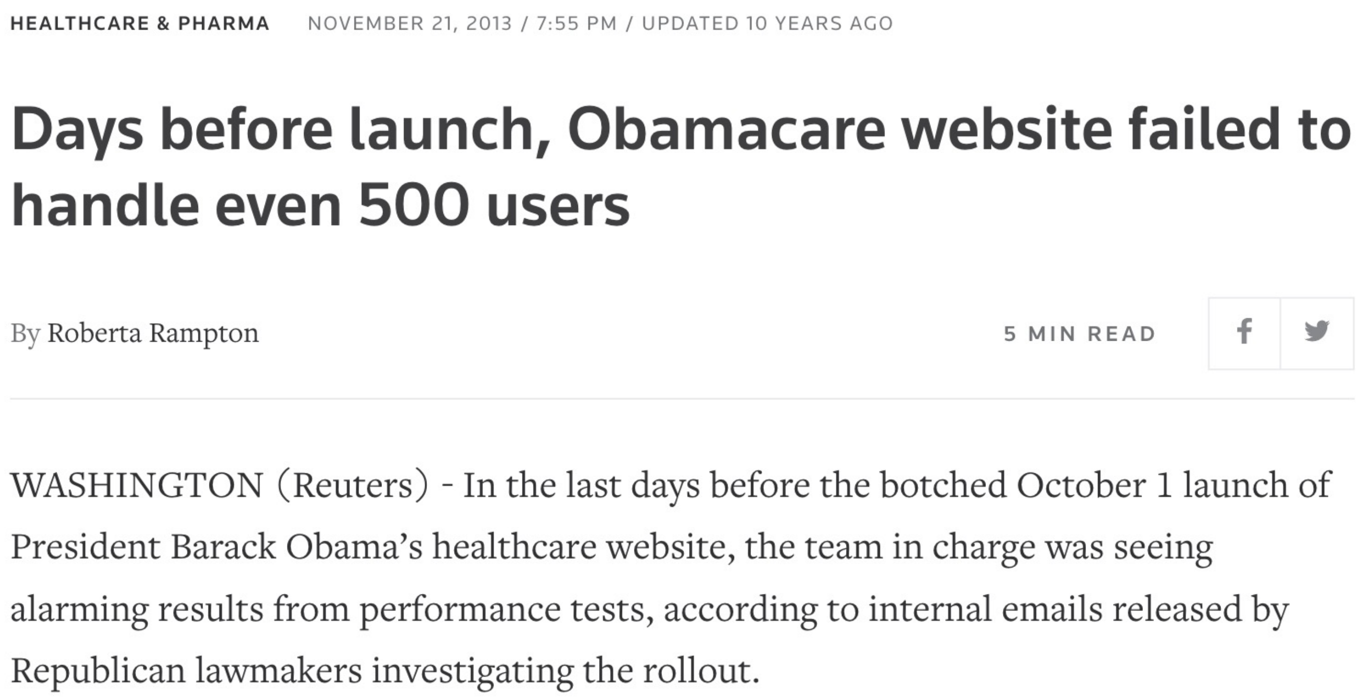
TECH

Internal documents show how Amazon scrambled to fix Prime Day glitches

Eugene Kim
@EUGENEKIM222

KEY POINTS

- Amazon wasn't able to handle the traffic surge and failed to secure enough servers to meet the demand on Prime Day, according to expert review of internal documents obtained by CNBC.



HEALTHCARE & PHARMA NOVEMBER 21, 2013 / 7:55 PM / UPDATED 10 YEARS AGO

Days before launch, Obamacare website failed to handle even 500 users

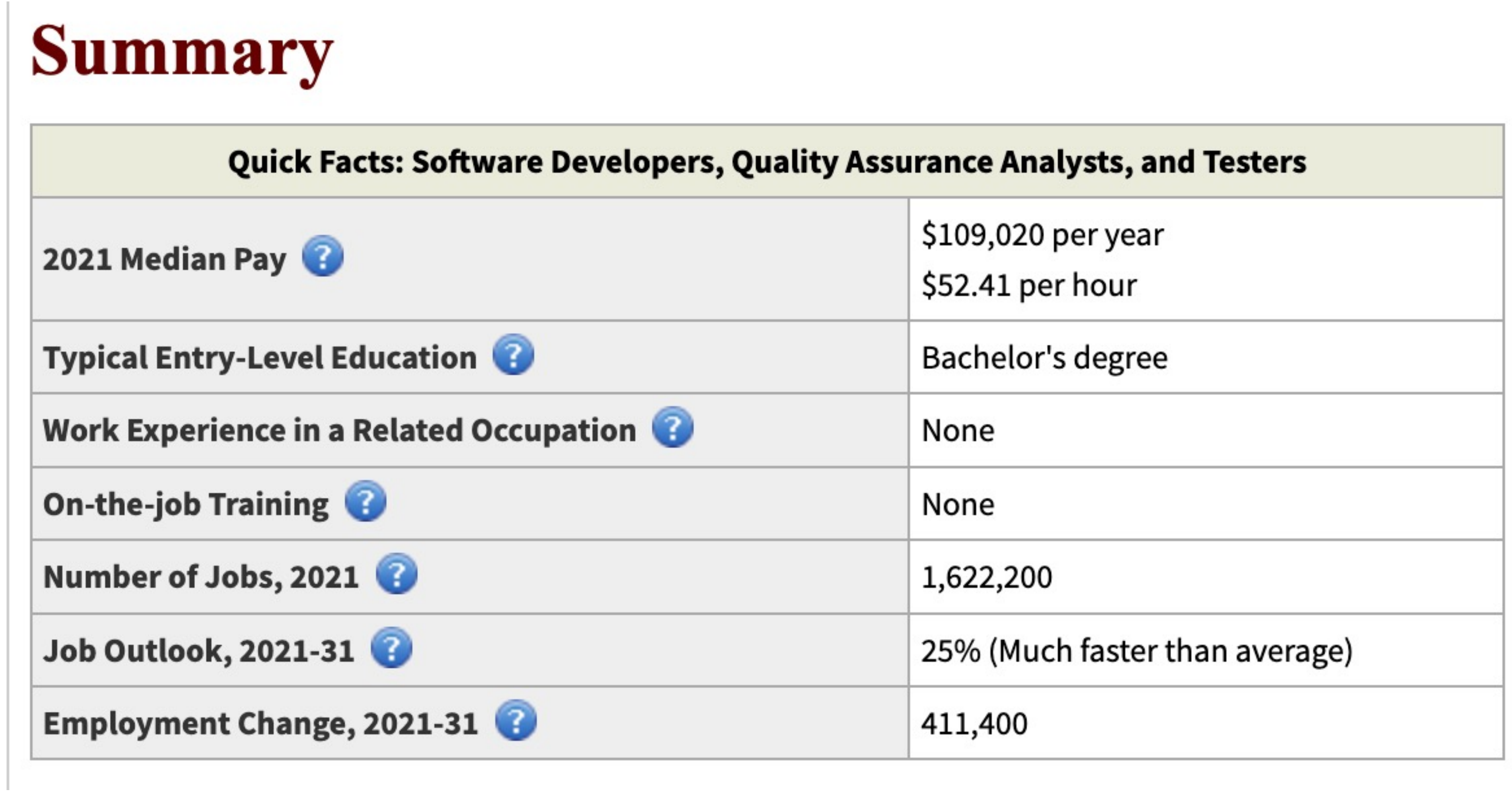
By Roberta Rampton 5 MIN READ

WASHINGTON (Reuters) - In the last days before the botched October 1 launch of President Barack Obama's healthcare website, the team in charge was seeing alarming results from performance tests, according to internal emails released by Republican lawmakers investigating the rollout.

- **Ignoring performance concerns has real-world implications**

Why does performance matter?

- Organizations need engineers with PE skills
- Organizations predominantly hire undergrads as software engineers
- Undergrads don't typically take PE courses
- **Performance training is left to chance!**



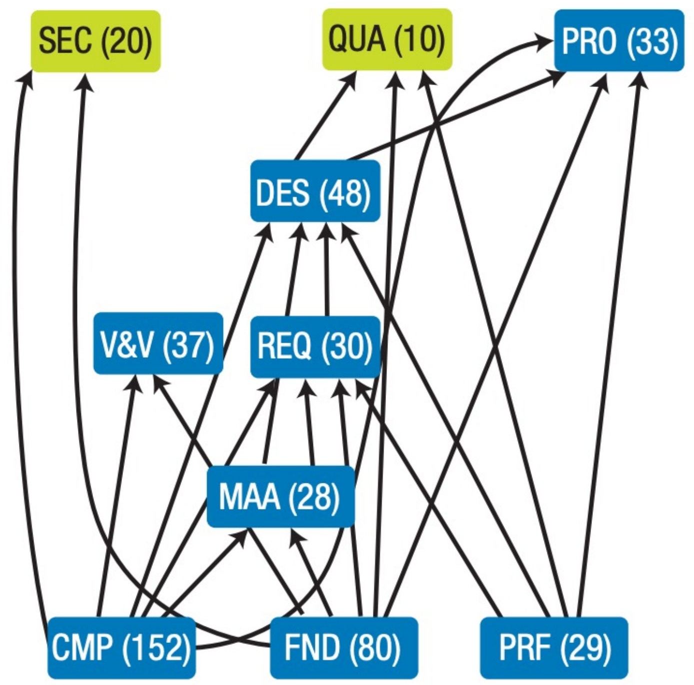
Summary

Quick Facts: Software Developers, Quality Assurance Analysts, and Testers	
2021 Median Pay	$109,020 per year $52.41 per hour
Typical Entry-Level Education	Bachelor's degree
Work Experience in a Related Occupation	None
On-the-job Training	None
Number of Jobs, 2021	1,622,200
Job Outlook, 2021-31	25% (Much faster than average)
Employment Change, 2021-31	411,400

Source: US bureau of labor stats

Why is it hard to get curriculum real estate?

- ”Standard” curricula silent on PE
- ACM/IEEE standard includes ”software quality”
- **No explicit mention of topics such as performance modeling**



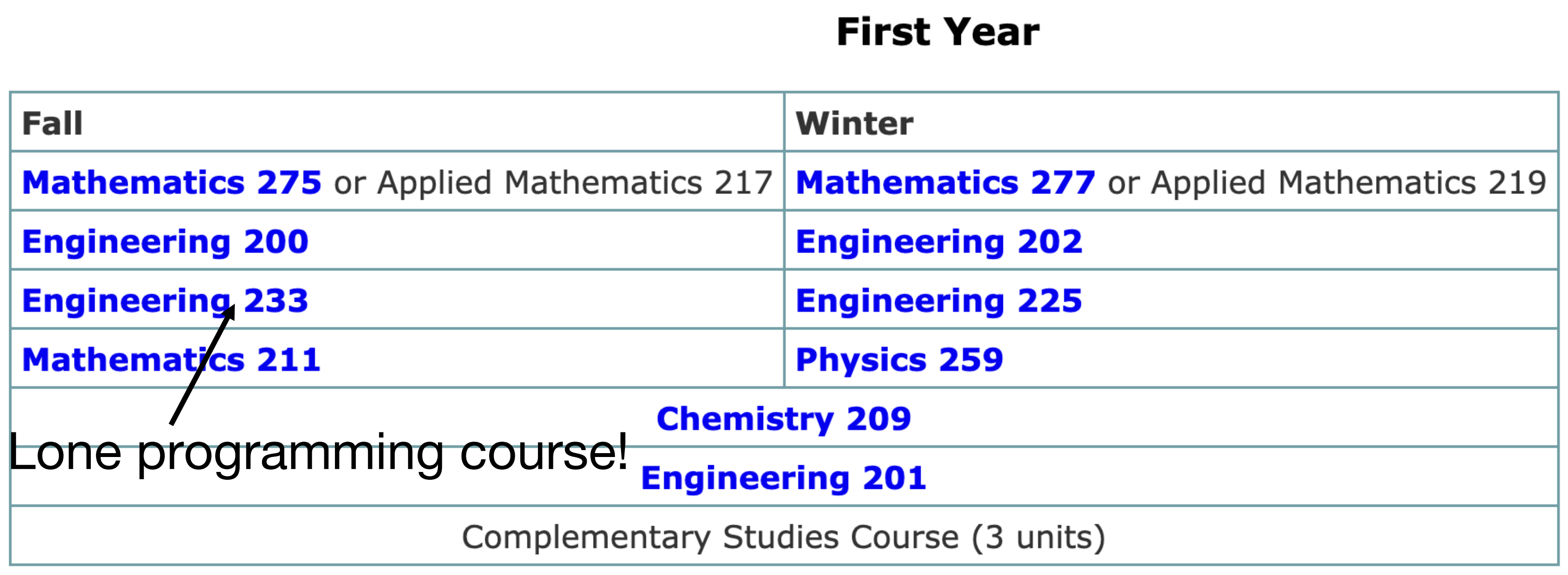
Module key:
CMP—Computing essentials
FND—Mathematical & engineering fundamentals
PRF—Professional practice
MAA—Software modeling & analysis
REQ—Requirements analysis & specification
DES—Software design
VAV—Software verification & validation
PRO—Software process
QUA—Software quality
SEC—Security

Figures in brackets indicate a topic's presentation hours (or lecture hours).

Why is it hard to get curriculum real estate?

- In some jurisdictions, engineering degrees need to be accredited
- E.g., Canadian Engineering Accreditation Board (CEAB) in Canada
  - Stipulates depth in software engineering
  - BUT, also breadth – math, natural sciences, other engineering, professional practice
- **Makes curriculum real estate scarce!**

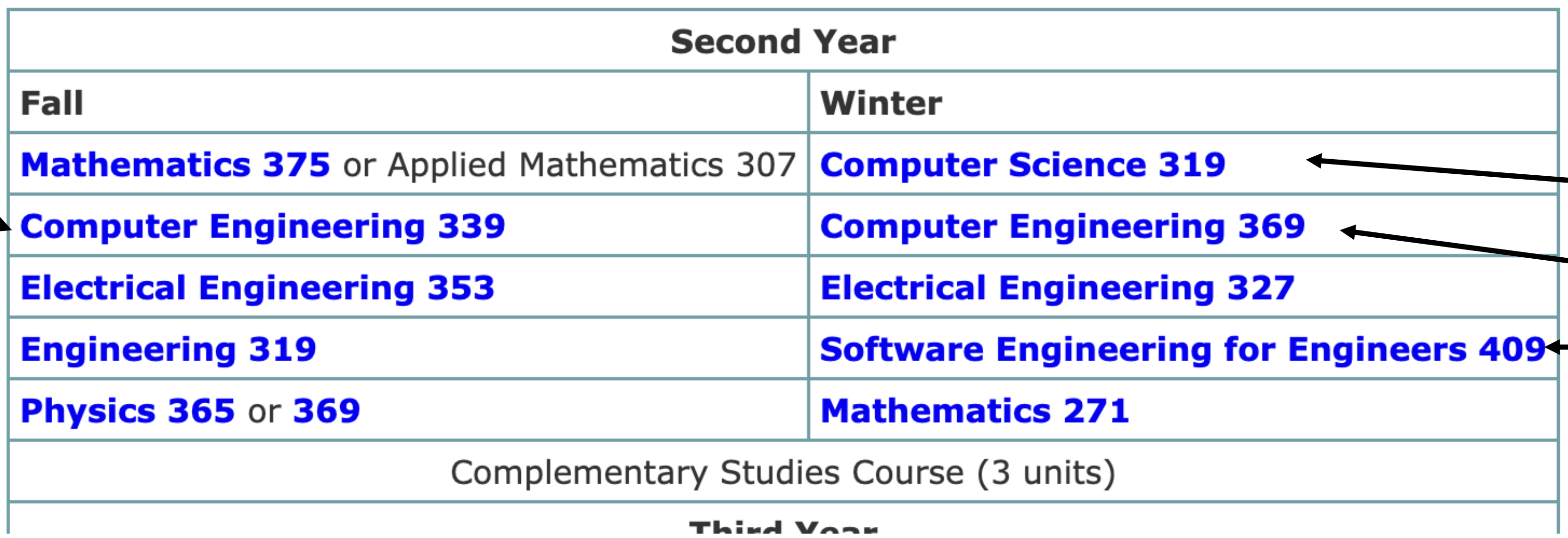
U Calgary Curriculum



First Year

Fall	Winter
Mathematics 275 or Applied Mathematics 217	**Mathematics 277** or Applied Mathematics 219
Engineering 200	**Engineering 202**
Engineering 233	**Engineering 225**
Mathematics 211	**Physics 259**
Chemistry 209	
Engineering 201	
Complementary Studies Course (3 units)	

Lone programming course!



Second Year	
Fall	**Winter**
Mathematics 375 or Applied Mathematics 307	**Computer Science 319**
Computer Engineering 339	**Computer Engineering 369**
Electrical Engineering 353	**Electrical Engineering 327**
Engineering 319	**Software Engineering for Engineers 409**
Physics 365 or **369**	**Mathematics 271**
Complementary Studies Course (3 units)	

Third Year

Intro C/C++

Data structures

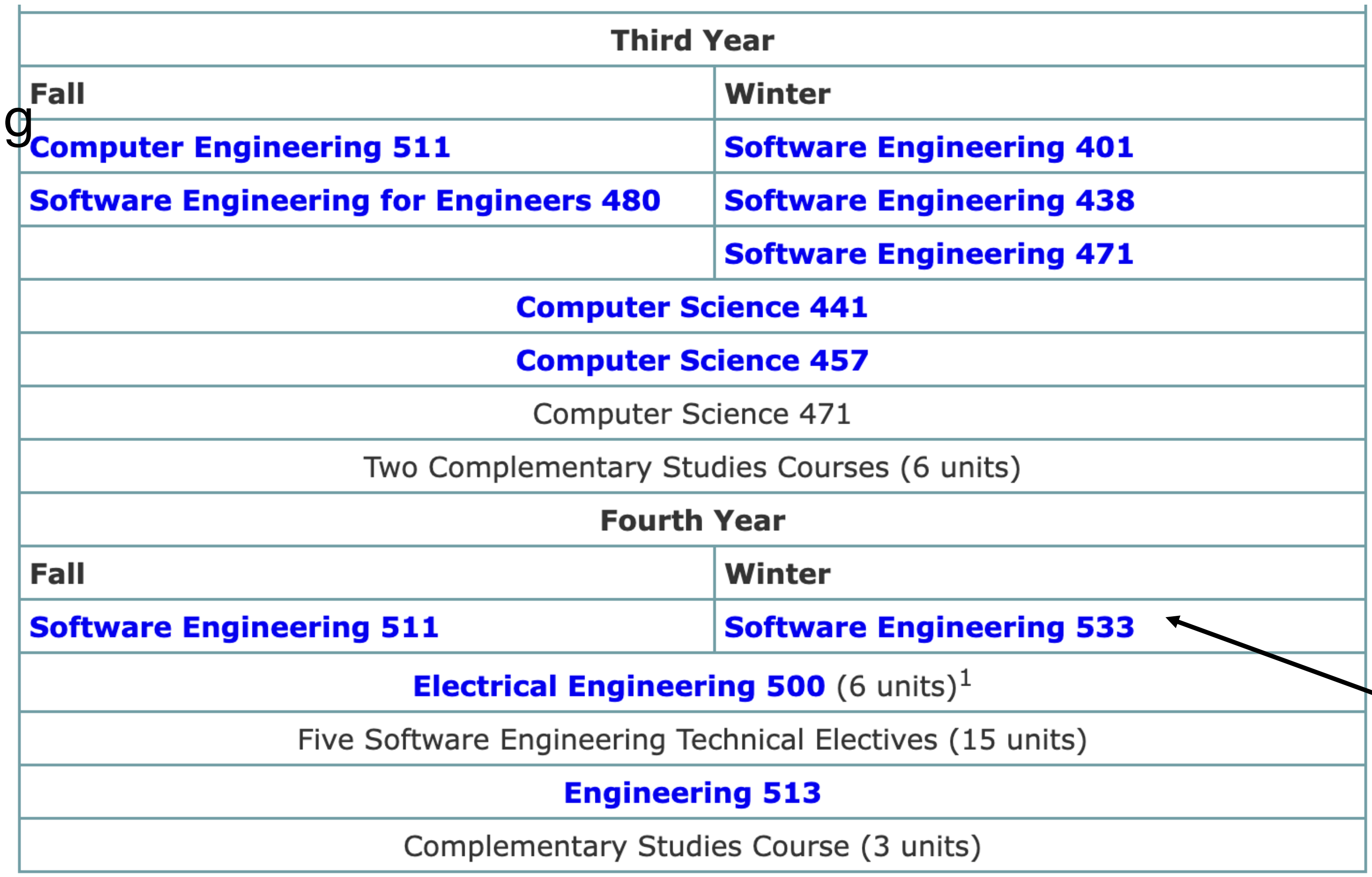
Comp organization

Object orientation

U Calgary Curriculum

Comp interfacing
Design

Processes



Third Year	
Fall	**Winter**
Computer Engineering 511	**Software Engineering 401**
Software Engineering for Engineers 480	**Software Engineering 438**
	Software Engineering 471
Computer Science 441	
Computer Science 457	
Computer Science 471	
Two Complementary Studies Courses (6 units)	
Fourth Year	
Fall	**Winter**
Software Engineering 511	**Software Engineering 533**
Electrical Engineering 500 (6 units)[1]	
Five Software Engineering Technical Electives (15 units)	
Engineering 513	
Complementary Studies Course (3 units)	

Architecture
Testing
Requirements
Networks
Operating systems
Databases

Performance!

What do students want?

- Make up for “irrelevant” breadth courses
- Not particular fans of math!
- Popular opinion on performance analysis
  - “Not relevant to me”
  - “Too much theory – more grad material!”
  - Performance equals throwing latest hardware at systems
- **How do we overcome this challenge?**

SENG 533: Software Performance Evaluation

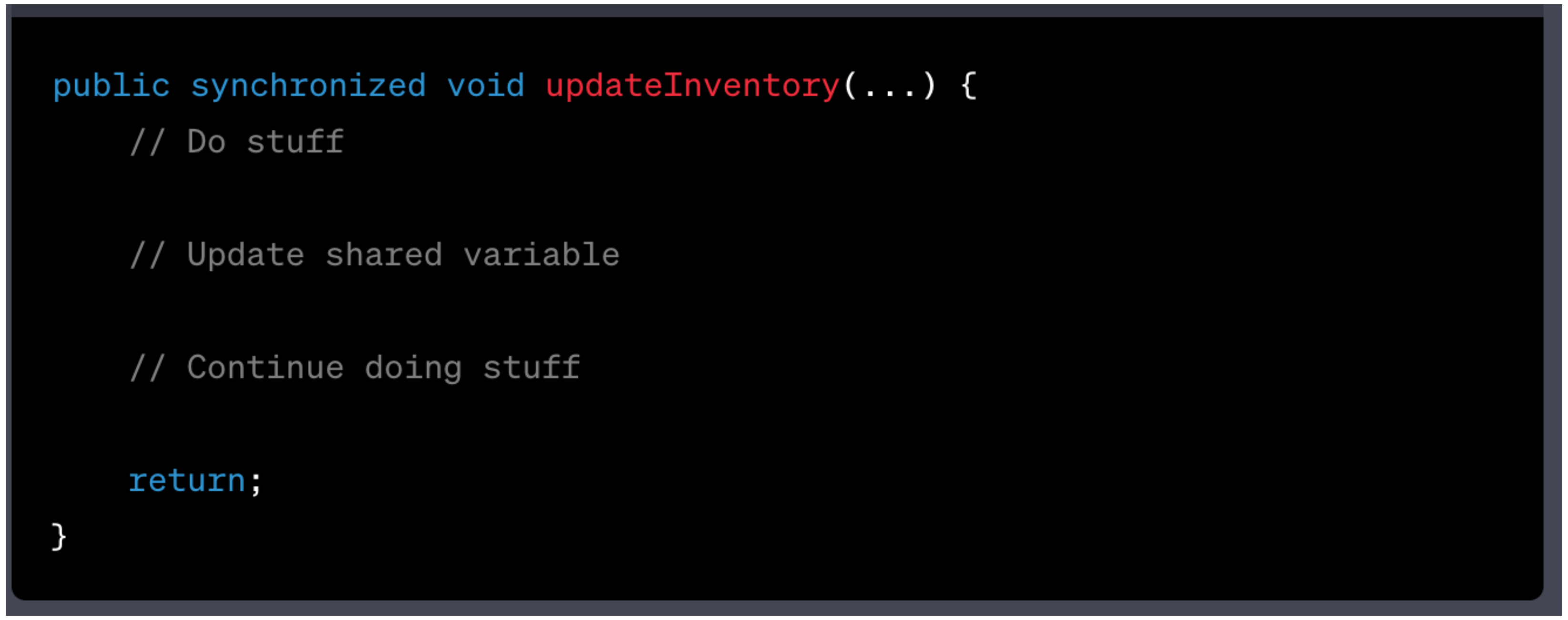
- **Course outline**
  - Motivation
  - Performance oriented review of computer systems
  - Software performance evaluation lifecycle
  - Techniques – Modeling and Measurement
    - Operational laws, MVA, a bit of advanced modeling (LQMs)
    - Load testing
  - Experiment design and analysis
- **What are some key strategies to foster student buy in?**

Strategy 1: Establish linkages to real world

- **Repeated use of appropriate industry/research case studies**
- Example: Fortune 100 company’s retail Web site (discussed in week 1)
- Poor throughput – slow page loads
- Throwing hardware at the solution did not work
- Step 1: Outline the system and ask students to brainstorm
- Step 2: Explain the root cause of problem and the fix

Strategy 1: Establish linkages to real world

- **Root cause: software bottleneck due to a long critical section**



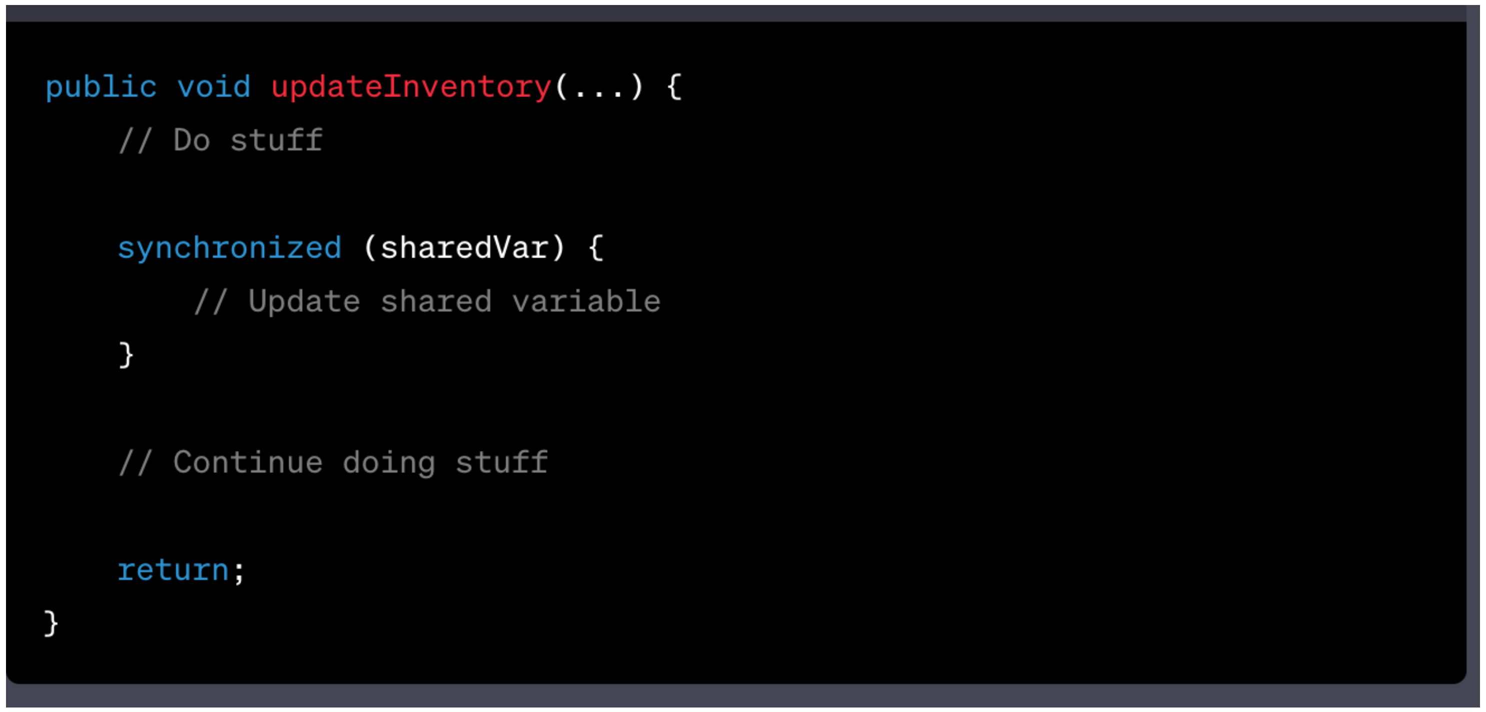
```
public synchronized void updateInventory(...) {
    // Do stuff

    // Update shared variable

    // Continue doing stuff

    return;
}
```

- **Fix: Use finer granularity synchronization**



```
public void updateInventory(...) {
    // Do stuff

    synchronized (sharedVar) {
        // Update shared variable
    }

    // Continue doing stuff

    return;
}
```

Strategy 1: Establish linkages to real world

- **Take home lessons of example**
    - Performance problems can have real implications
    - Functional testing alone is not enough!
    - Can’t just throw hardware at problems

Strategy 2: Active learning

- **Brainstorm realistic performance problems in group**
- Example: Diagnosing scalability problems in a multi-core Web server (Hashemian et al., ICPE 2013)

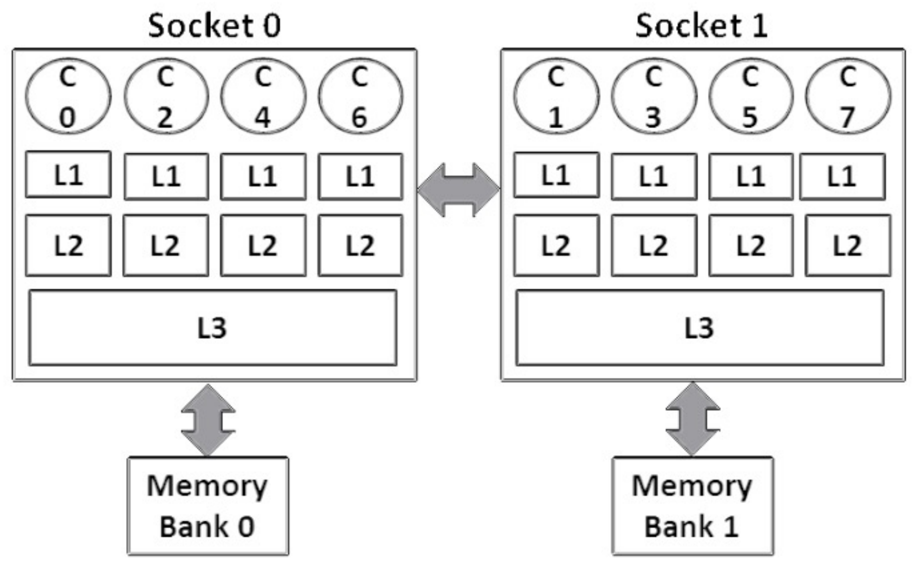
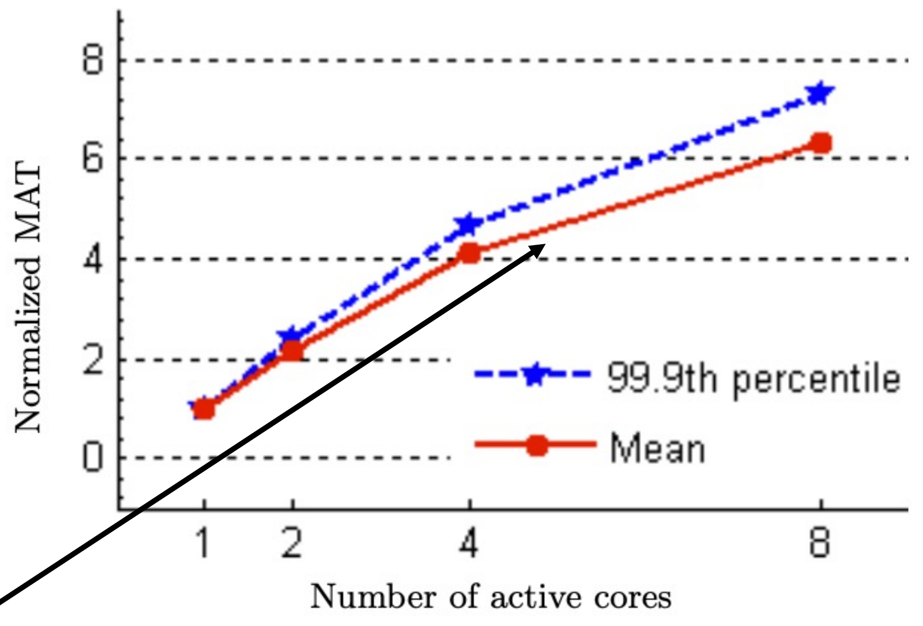


Figure 1: Server Architecture



Why does the throughput scaling degrade beyond 4 cores?

Strategy 2: Active learning

- **Root cause**: Context switching behaviour/cold cache/interconnect bottleneck
- **Fix**: Two web server replicas – each affined to their own socket
- **Take home lessons of example**
  - Performance analysis is fun – similar to detective work
  - Even production grade software (Apache) can have scalability problems
  - Throwing hardware (cores) at a system is not enough.

Strategy 3: Prioritizing application over theory

- Focused on **how to build models** as opposed to **how to solve them**
    - How to represent a given scenario with a model?
    - What type of model is appropriate for a situation? (E.g., closed vs open)
    - What are model assumptions? Are they likely to be valid?
    - Allow formula/cheat sheet!
- **Trade mathematical sophistication for student buy in**

Strategy 3: Prioritizing application over theory

- **Wording problems to emphasize application/model building**

Emphasis on model solution

Consider a 2 class system. The first class is interactive with 20 customers each having an average think time of 30 seconds. The CPU and network demands of the class are 0.15 secs and 0.95 secs, respectively. The second class is batch having 10 customers. The CPU and network demand of this class are 1 sec and 0.08 sec, respectively. Calculate the response time of the interactive class.

Emphasis on model building

A cloud provider is exploring the option of consolidating two applications from two different cloud subscribers on to a physical machine having a single CPU core. The first application is a short movie clip service with 20 customers each having an average think time of 30 seconds. On average, the transfer of a movie clip demands 0.15 secs from the CPU and 0.95 secs from the network. The second application is a scientific computing batch program. There are 10 instances of this program running concurrently. The program places a demand of 1 secs on the CPU and 0.08 seconds on the network. The subscriber owning the first application stipulates that the mean response time per clip should not exceed 2 secs. Is consolidating these 2 apps a good idea?

Strategy 4: Linking labs with lectures

- Simulate performance engineering practice with a 13-week project
- **Show how to use analytic models in the process**
- Example – Performance analysis of a microservice system
  - Synthesize workloads – integrate with a load generator
  - Validate load generator – e.g., **is $X = N/(R+Z)$?**
  - Obtain service utilizations - **calculate demands using utilization law**
  - **Build a LQM using estimated demands and service dependencies**
  - Validate model predictions – discuss how assumptions impact predictions
  - Use model to conduct “what-if” analyses, e.g., impact of different scaling strategies

Open challenges

- Course has been generally well received
- However, several open challenges remain
- Challenge 1: Not ideal to teach PE in a silo
  - Need integration with other aspects of software engineering
  - Demands team teaching type effort
- Challenge 2: How do we teach advanced models without scaring students?
- Challenge 3: Is teaching queuing theory irrelevant given rise of ML/AI models?
- ……….

Thank you!