CS 59000-CAS:
Compiler & Architectural Support for Performance, Reliability & Security

Fall 2019

Changhee Jung
About Me

• Changhee Jung (chjung@purdue.edu)
• Joined CS@Purdue in 2019 Fall
• Before joining Purdue,
  – Was a professor of Computer Science at VT
  – Got Ph.D. at GT under supervision of Prof. Pande
  – Finished 4 years of army duty in ETRI
  – Before this: Master Student of ACRL at SNU
About Me

• Basically, I’m a compiler guy
  – Contributed to GCC 4.5
  – Worked in Google’s Compiler Optimization Team

• But I also do research on
  – Computer architecture, HW and SW interaction

• My top conferences
  – MICRO, ICSE, ASPLOS, HPDC, SC, PLDI
  * Recent papers appeared in the highlighted ones.
About Me

• My research has been supported by

- Google Faculty Research Awards
- AMD
- National Science Foundation

“Compiler and Architectural Techniques for Soft Error Resilience”

“CAREER: Rethinking HPC Resilience in the Exascale Era”
‘BIG MAC’ where My Research Began
‘BIG MAC’ where My Research Began

• A supercomputer out of 1,100 Apple Power Mac G5 built in 2003 at Virginia Tech.
• Big Mac crashed too many times
  • Apps never finish before the crash
  • Sometimes not even boot
• Soft errors corrupted the non-ECC memory.
‘BIG MAC’ where My Research Began

- A supercomputer out of 1,100 Apple Power Mac G5 built in 2003 at Virginia Tech.
- Big Mac crashed too many times
  - Apps never finish before the crash
  - Sometimes not even boot
- Soft errors corrupted the non-ECC memory.
My Research: Soft Error Resilience

- Transient fault mainly caused by high-energy cosmic particles.
- Result in random bit flip.

Application Crash
Silent Data Corruption
Deadlock
...
More Stories …

Bit-flip in a data structure of cruise controller of Toyota Camry killed a driver!
My Research: Soft Error Resilience

One fault per DAY per chip

One fault per DAY per 100 chips

Aggressive voltage scaling (near-threshold computing)

Past
Present
Future

Technology Node (nm)
Triple-Modular Redundancy (TMR)

- Replicate 3 identical modules and use majority voting for the output

3X HW overhead!!!
Idempotent Property

A region of code is idempotent iff it can be re-executed multiple times and still preserves the same, correct result.

\[
\begin{align*}
x &= 1 \\
y &= x
\end{align*}
\]

Output: \(x = y = 1\)
Idempotent Recovery

- Recovery by rollback and re-execution of the faulty region

$\text{Expected Output: } x = y = 1$

$\text{Unexpected Output: } x = 1, y \neq 0$
More to Come

• I have several resilience research problems (not just soft error resilience), you can dive into
  – If you pick one of them and get it done correctly, there would be a high chance for you to publish a research paper!
About You

• Your name and your advisor (if you have)

• Your background
  – Your affiliation/program enrolled
  – Industry/Internship experience (if you have)

• Research interests (past and current)
  – If you’ve published papers, please give an elevator speech
Course Facts

- Class meeting time
  - 4:30 – 5:45pm on Mon/Wed at LWSN 1106
- Office hours: by appointment
  - Please send me an email first
- Course site
  - We’ll use Blackboard and Google Sheets as means of primary communication
- Honor coded
  - All work is conducted under Purdue University Academic Integrity (cs.purdue.edu/homes/chjung/integrity)
Force-Add

• I expect that everybody who is interested in taking this class will be able to do so

• Send an email to chjung@purdue.edu with your name and the last four digits of PUID
About This Class

• Typical seminar class with an emphasis on preparing graduate students for systems research on compilers and computer architecture support for performance, reliability, and security.

• Our goals
  – To be familiar with the state-of-the-art of compiler and computer architecture related to the topics
  – To understand current results in one or more areas of optimizing compilers, microarchitectures, and their interaction related to the topics
  – To identify new ideas for advancing the state-of-the-arts
About This Class

• We’ll read research papers related to advanced topics in compilers and computer architecture support for performance, reliability, and security.

• Possible compiler topics
  – Program analysis, program transformation, and the interaction between the compiler and the rest of the system, …

• Possible architecture topics
  – Processor micro-architecture, memory hierarchy, multi-threading, and the impact of emerging memory technologies …
Background You Should Have

• Programming
  – Good C++ programmer (essential), Linux, gcc, VI/Emacs
  – Debugging experience – hard to debug with printf’s alone – gdb!

• Prerequisite 1: compilers
  – Basic backend stuff (code generation), e.g., basic block, CFG, etc.
  – Frontend is not very relevant here

• Prerequisite 2: computer architecture
  – Undergrad computer architecture course is good enough
  – Basics – caches, pipelining, functional units, registers, virtual memory, branches, multiple cores, assembly code
Reading Material

- No required text: most of the reading will be assigned papers.
- However, if you wish to brush up on your basics, we recommend the following textbooks as background:
  - “Advanced Compiler Design & Implementation” by Muchnick
  - “Modern Processor Design: Fundamentals of Superscalar Processors” by Shen and Lipasti
Course Format

• Paper reading and discussions
• Paper evaluations
• Student presentations
• Research project
No Exam
Reading and Discussion

• You’ll present research papers along the way
• Everybody reads assigned papers before class
• Everybody should submit a hard copy of your critique
  – To prove you’ve read the paper.
  – To enable you to contribute to discussion
• No late submissions will be accepted
  – Contact me for exceptions in severe circumstances only
Paper Evaluation Form

• What problem does the paper attack? How does it relate to and improve upon previous work in its domain?
• What are the key contributions of the paper?
• Briefly describe how the paper’s experimental methodology supports the paper’s conclusions.
• Write down one question you plan to bring up in the discussion.
• Review form will be provided soon in Blackboard
Your Presentation (2 parts)

• First, present research as if it were your own
  – Giving background (very desired for your friend in class)

• Then, change roles:
  – Evaluate research from your perspective: add insights, criticism, etc.

• Help lead subsequent discussion
Presentation Grading

• Shooting for an hour
  – 45m talk + 15 min Q&A and discussion

• Graded on Comprehension, not Quality
  – To help you in life (and make this semester less painful for everyone)

• Show up for class and ask good questions!
Preparing Your Presentation

• Every student may meet with instructor to discuss slides.
  – It’s your responsibility to schedule a suitable time, early enough such that there’s still time for revisions to your slides
  – You must have your slides ready by that time.
Project: The Most Important

• Design & implement an “interesting” compilation or architectural technique and demonstrate its usefulness for performance, reliability, or security using research compilers or architecture simulators

• Topic/scope/work
  – Individual project
  – You need to pick the topics (but I have to agree)
  – You have to read background material, plan & design, and implement

• Deliverables
  – Working implementation and final presentation
  – Project report: 6 pages in ACM SIGPLAN template
Types of Projects

• New research idea
  – Design & implement small idea, see how it works

• Extend existing idea (most popular)
  – Take an existing paper, implement their technique
  – Then, extend it to do something interesting;
    Generalize strategy, make more efficient/effective

• Implementation
  – Take existing idea, create quality implementation in
    LLVM compiler or Gem5 architecture simulator
  – Get your code released into the community
Types of Projects

• Workload characterization and analysis
  – Evaluate existing compilation or architectural techniques with emerging applications in terms of performance/power/reliability, ...
  • Ex> Evaluate the impact of feedback directed optimization for IoT benchmark applications across different computer architectures
  • Above example is publishable!
Topic Areas (You’re Welcome to Propose Others)

• Repurposing commodity computer architecture feature
  – Transactional memory (Intel TSX)
  – Memory protection (Intel MPX)
  – Processor trace (Intel PT)
  – Performance Monitor (Intel PEBS)

• Memory system performance
  – Instruction prefetching
  – Data prefetching
  – Helper thread prefetching
  – Use of scratchpad memories

• Software Resilience
  – Detect program bugs
  – SW fault tolerance
  – Offer crash consistency
  – Enhance security (or reduce overhead)

• Performance Characterization
  – Real analysis of FDO on emerging workloads
  – Evaluation of pointer analysis impacts on compiler optimization performance
Course Grading

• Components
  – 5% Class Participation
  – 5% Paper evaluation
  – 30% Research paper presentation
  – 20% Final project presentation
  – 40% Final project report

• A >= 93%, A- >= 90%, B+ >= 87%, B >= 83%, B- >= 80%, C+ >= 77%, C >= 73%, C- >= 70%, D+ >= 67%, D >= 63%, D- >= 60%, F < 60% (I may curve)
Course Schedule and Papers to be Discussed

- The list of the papers and the dates of the paper discussion are available in the Google Sheets

https://docs.google.com/spreadsheets/d/1k60xL7eM23OMWg3Xus7xeCJXdh5D3M4LsR WQDZuVsa/edit?usp=sharing