Dependable Software Systems

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

ECE 1781, Winter 2016

Topics

- Overview
- What are dependable systems?
  - Why do we care about them?
- Why do systems stop?
  - What can we do about it?
- Topics
- Class format

Overview

- Class website available from my home page
  - http://www.eecg.toronto.edu/~ashvin
- Sign up for class by joining class mailing list
  - Instructions available from class website
- Seminar style course
  - Reading, discussion, presentation
- No assignments
- Project, presentation
- No quizzes or final exams

What are Dependable Systems?

Dependable Systems

- Hard to define, but examples are easy to find
  - Transportation, e.g., cars, airplanes
  - Appliances, e.g., toaster, fridge, TV
  - Medical devices, e.g., MRI, X-rays, prosthesis
- Properties
  - Traditionally, have redundancy, keep running
  - Easily understood operation model
  - Allow monitoring for (well-documented) errors
  - Degrade gracefully
- Bug free? No configuration needed?

Software Systems

- Tightly intertwined with our lives
  - Increased networking, e.g., wireless
  - Cheap devices, e.g., cell phones
- Complex and failure-prone
- Hard to manage
- Dependability issues dominate total cost of ownership

Current Challenges

- “The products of forty years of OS research are sitting in everyone's desktop computer, cell phone, car, etc., — and it is not a pretty picture.”

  — Researchers from Microsoft, 2005.

- Some key problems
  - *Dependability*: frequent unexpected behavior
  - *Security*: systems protect users from each other, not from outside threats
  - *Configuration*: DLL hell

Insight

- Performance is not the only concern today
  - Few applications require all available resources
- Use resources to improve dependability
- Examples
  - Store all data versions to guard against data loss
    - Read “A Conversation with Jim Gray” (`acmqueue.org`)
  - Replicate processes, data
  - Isolate sub-systems to reduce fault propagation
  - Use bug detection, recovery methods

Why Do Systems Stop?

Jim Gray, 1985

Conventional TP Systems

- On average, fail for 90 min every 2 weeks
- Restart time includes
  - Detection time
  - Time to take snapshot for later analysis
  - OS, database, communication n/w reboot
  - Client (e.g, ATM machines) reboot
  - Users take time to refocus on job
- How available is the system?
  - 99.6% availability (2 weeks / (2 weeks + 90 min))
  - Sounds wonderful, isn't!

Highly Fault-Tolerant System

- Analyzed failure reports of 2000 systems running a fault-tolerant Tandem system
- Analysis covered 10M system hours
  - 1300 system years!
- 166 failures reported
- Mean time between failure (MTBF) = 7.8 years!
- Where did the failures occur?

Breakup of Failures

- 59 “infant mortality” failures
  - Recurrent failures due to new software or hardware
  - Bugs should have been fixed before deployment
- Contributors to the other 107 failures

Administration	42%
Software	25%
Hardware	18%
Environment	14%

Maintenance, operations configuration

Fire, flood, >4 hr power loss

Implications

- Reliability requires tolerating *software faults* and *administration errors*
- Hardware becomes more reliable over time
  - Hardware fault tolerance is feasible
- New and changing systems have higher failure
  - If it's not broken, don't fix it
- High % of outages caused by known bugs
  - Install software and hardware fixes ASAP
- Contradiction?

H/W Fault Tolerance

- Modularize hardware to limit faults
- Make each module fail-fast
  - Either it does the right thing or stops
- Detect faults promptly
  - Have module signal failure
- Configure extra backup modules
- Resulting MTBF is in years to decades!

S/W Fault Tolerance

- Use techniques similar to h/w fault tolerance?
- Software modularity via processes and messages
- Fail-fast modules
- Process-pairs to tolerate transient software faults
  - Bohrbug/Heisenbug hypothesis
- Transactions to provide data integrity
- Combine process-pairs and transactions

Administration Errors

- “Dealing with system configuration, operations and maintenance remains an unsolved problem”

  — Jim Gray, 1985.

Topics

Main Topics

- Bugs and race detection
- Testing and debugging
- Fault isolation
- Failure recovery
- Fault tolerance
- Updating software
- System misconfiguration

Class Format

Overview

- Class website available from my home page
  - http://www.eecg.toronto.edu/~ashvin
- Sign up for class by joining class mailing list
  - Instructions available from class website
- Seminar style course
  - Reading, discussion, presentation
- Zero assignments
- Project, presentation
- No quizzes or final exams

Reading and Discussion

- Advanced
- Background in OS, networking, distributed systems
- At least 2 papers per week
  - Unless marked optional, all papers are required reading
- Will take about 4-6 hours per week
- Allows discussion in class
- It will show if you don't do the reading!

Presentation

- For discussion, you *must* prepare five questions
  - One slide for *each* question
  - Then one slide for *each* of your answers
  - That is a total of 10 slides at the end of the presentation
  - The order is Q1, A1, Q2, A2,…,Q5, A5
- Detailed instructions on website
- Please follow carefully
  - E.g., make sure you number slides!
  - Fonts should be reasonably large (>24)
  - Follow this style

Choosing A Paper

- First-come, first served
- Pick 2 papers from website
- Fill form available from website
- Make sure that your choice is not taken

Assignments

- There will be no assignments in this course

Project

- Choose a project based on topics covered
- Sample topics will be posted on website
- Options
  - Implement and evaluate a system
  - Evaluate existing system
  - Write a research paper
- Write up your work
  - 8-10 pages
- Present your work

Grading Policy

- Class presentation: 30%
- Class project: 50%
  - Description: 5%
  - Mid-term report: 10%
  - Final report: 35%
- Class participation: 20%

Please join class mailing list and choose papers at

http://www.eecg.toronto.edu/~ashvin

Thanks!