

Copyright warning

COMMONWEALTH OF AUSTRALIA
Copyright Regulations 1969
WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (the Act).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

COMP5348

Lecture 10: Fault-tolerance

Adapted with permission from presentations by Alan Fekete

Outline

- Types of Faults and Failures
- Estimating reliability and availability
- A case study: storage failures
- Techniques for dealing with failure

Terminology: Fault and Failure

- Any system or component has specified (desired) behavior, and observed behavior
- Failure: the observed behavior is different from the specified behavior
- Fault (defect): something gone/done wrongly
- Once the fault occurs, there is a "latent failure"; some time later the failure becomes effective (observed)
- Once the failure is effective, it may be reported and eventually repaired; from failure to the end of repair is a "service interruption"

What fails?

- Anything can fail!
 - Dealing with a failure may cause more faults!
- Hardware
- Software
 - OS, DBMS, individual process, language RTE, network stack, etc
- Operations
- Maintenance
- Process
- Environment

How does it fail?

- Failstop
 - System ceases to do any work
- Hard fault
 - Leads to: System continues but keeps doing wrong things
- Intermittent ("soft") fault
 - Leads to: Temporary wrong thing then correct behavior resumes
- Timing failure
 - Correct values, but system responds too late

Repeatability?

- If system is run the same way again, will the same failure behavior be observed?
 - If several systems are run on the same data, will they all fail the same way?
- Heisenbugs: system does not fail the same way in different executions
 - May be due to race conditions
 - Very common in practice
 - Eg “just reboot/powercycle the system”
- “Bohrbugs”: reproducible

Analyzing failure

- Root cause analysis
- Find the fault that led to failure
- Then find why the fault was made
 - And why it wasn't detected/fixd earlier
- See http://en.wikipedia.org/wiki/5_Whys

My car will not start. (the problem)

1. Why? - The battery is dead. (first why)
2. Why? - The alternator is not functioning. (second why)
3. Why? - The alternator belt has broken. (third why)
4. Why? - The alternator belt was well beyond its useful service life and has never been replaced. (fourth why)
5. Why? - I have not been maintaining my car according to the recommended service schedule. (fifth why, root cause)

Attacks or accident?

- Some failures are due to malice: deliberate attempt to cause damage
- Most are due to bad luck, incompetence, or choice to run risks
 - often to several of these together!

Outline

- Types of Faults and Failures
- **Estimating reliability and availability**
- A case study: storage failures
- Techniques for dealing with failure

Memoryless fault model

- Failure in each short period is independent of failure in another short period
 - There is no effect from the lack of failure in the past
 - Contrast with gambler's fallacy

Failure rate

- Express the possibility of failure as AFR (annualized failure rate)
 - Expected ratio of failures to size of population, over a year of operation
- Discuss: how is it possible that $AFR > 1$?
- Can then compute expected number of failures during other periods
 - Recall: approx 30,000,000 sec in 1 yr
 - Almost 10,000 hrs in 1 yr

Warning

- AFR = 1% means: in 100,000 cases, 1000 is the expected number that fail during a year
- It does not say: 1000 will definitely fail during the year
- It is not necessarily saying: there is 50% chance that more than 1000 fail, and 50% chance that fewer fail
- It does not say: my system will fail if I wait for 100 years

Bathtub fault model

- Failure is more likely very early in system lifecycle, and very late in system lifecycle
 - Initially, many construction faults are discovered and fixed
 - Eventually parts wear out
 - Software also has "bit rot" due to shifting uses, other changes etc
- Failure is rare in the extended middle of the system lifecycle

Burstiness and environmental effects

- Many failure types do have occurrences that correlate
 - Often one failure may be fixed, but leave degraded data structures or collateral damage
 - Environmental effects like heat

MTTF

- Express current failure rate through mean-time-to-failure
 - $MTTF = 1/AFR$ years
 - This is often too long to be known through testing
- Quote this at some point in time
 - If failure rate changes through time, this is not the same as true expected time till failure happens
- Many vendors give "best case" MTTF
 - "system is guaranteed to perform like this or worse"

Repair

- After a failure, system or component can be repaired/replaced
- This takes time
- MTTR is average time from failure till repair is completed (and system resumes working correctly)

Availability

- What fraction of a time-period is system functioning correctly?
- $Availability = MTTF / (MTTF + MTTR)$
- Often quoted as "number of nines"
- "3 nines" means 99.9% available
 - Downtime of approx 500 mins each year

Parts and the whole

- When a system is made up of several components
 - How does failure of a component impact the system as a whole?
- Often failure of one component means overall failure
- But some designs use redundancy
 - allow system to keep working until several components all fail

Calculations for systems of components

- For a system whose functioning requires every component to work
 - Assume that component failures are independent!
- $AFR(\text{system}) = \text{Sum}(AFR \text{ of component}_i)$
- $\text{Availability}(\text{system}) = \text{Product}(\text{availability of component}_i)$

Outline

- Types of Faults and Failures
- Estimating reliability and availability
- A case study: storage failures
- Techniques for dealing with failure

Outline

- Types of Faults and Failures
- Estimating reliability and availability
- A case study: storage failures
- Techniques for dealing with failure

Logging

- Record what happens in a stable place
 - Whose failures are independent of the system
 - Especially record all user inputs and responses
- There are frameworks that make this somewhat easy
- Essential to deal with mess left by failure
- Also powerful aid in forensics

Styles of recovery

- Backward: Restore system to a sensible previous state, then perform operations again
- Forward: Make adjustments to weird/corrupted state, to bring it to what it ought to have been

Principles

- Idempotence
 - Make sure that if you perform an operation again, it doesn't have different effects
 - Eg write(5) is idempotent, add(2) is not
 - Create idempotence by knowing what has already been done (store timestamps with the data)
- Causality
 - Make sure that if you perform operation again, the system is in the same state it was in when op was done originally
 - Or at least, in a state with whatever context is necessary

Failfast components

- Design of a larger system from redundant parts, is easier if the parts are failstop
 - And availability is best if they are quick to repair
- So, don't try to keep going when a problem is observed
 - Stop completely
 - Wait for the repair!

N-plex Redundancy

- Have several (n) copies of the same component
- When one fails, use another
 - And start repairing the first
- So system works as long as some component is working
- The hard thing: making the voter work reliably!

Calculating failure rates

- Assume that failures are independent
 - Not very realistic: eg environment can affect all in the same way
- Availability(n-plex)
 $= 1 - (1 - \text{Avail}(\text{component}))^n$
- $\text{MTTF}(\text{n-plex}) = \text{MTTF}^n / n * \text{MTTR}^{(n-1)}$

Summary

- Many varieties of failure
 - Anything can fail!
- Measures of reliability and availability
 - Estimating these
 - Know the assumptions you make!
- Techniques to deal with failures

Further reading

- J. Gray and A. Reuter "Transaction Processing: Concepts and Techniques" (Morgan Kaufmann 1993)
- B. Schroeder, G. Gibson "Understanding Disk Failure Rates: What does a MTTF of 1000000 Hours mean to you?" ACM Trans on Storage 3(3), 2007
- E. Pinheiro, W.-D. Weber, L. A. Barroso "Failure Trends in Large Disk Drive Population" Usenix FAST'07 pp 17-28
- W. Jiang, C. Hu, Y. Zhou, A. Kanevsky "Are Disks the Dominant Contributor for Storage Failures?" Usenix FAST'08 pp111-125