**Lecture 15 4/1/09 Building Fault Tolerant Systems from Unreliable Components**

Sam Madden

So far what have we seen:

- Modularity
  - RPC
  - Processes
  - Client / server
- Networking
  - Implements client/server

Seen a few examples of dealing with faults -- e.g., providing exactly once semantics in a best-effort system.

Goal of this part of the course is to understand how to systematically build reliable systems from a collection of unreliable parts.

Basic way this is going to work is to introduce redundancy, e.g.:

- in TCP this takes the form of sending data multiple times;
- today: a general technique -- replication of critical system coponents
- next time: in a file system or database it often takes the form of writing data in more than one place.

Some high level points:

- These techniques improve reliability but building a perfectly reliable system is impossible. Best we can do is to get arbitrarily close to reliable. Our guarantees are going to be essentially probabilisitc.

- Reliability comes at a cost -- hardware, computational resource, disk bandwidth, etc. More reliability usually costs more. Have to decide -- as with TCP -- whether the additional reliability is worth the cost. Always a tradeoff.

This lecture:

Understand how to quantify reliability

Understand the basic strategy we use for tolerating faults

Look at replication as a technique for improving reliability / masking faults

Going to do this in the context of disk systems, but many of these lessons apply to other components.  (Will come back to this at the end.)

Why disks?

Diagram (disk, cable, computer, controller, file system sw)

Where do failures happen?
    (Everywhere.)  Show slide.

Fault anywhere in this system makes compute unusable to user.

Goal is to improve the general availability of the system.

Measuring Availability

    Mean time to failure  (MTTF)

    Mean time to repair (MTTR)

    Availability =  MTTF / (MTTF + MTTR)

    MTBF = MTTF + MTTR

Suppose my OS crashes once every month, and takes 10 minutes to recover
MTTF = 720 hours (43,200 minutes)
MTTR = 10
43200 / 43210 = .9997 ("3 nines") -- 2 hours downtime per year

General fault tolerant design process:

- Quantify probability of failure of each component
- Quantify costs of failure
- Quantify costs of implementing fault tolerance

Probabilities here are high for many components, but

HOWEVER, Unlike other components, disk is the one that if it fails your data is gone -- can always replace the CPU or the cable, or restart the file system.

You can replace the disk, but then your system doesn't work.  So cost of disk failure is high.

Ok, so why do disks fail?

show disk slide

    spinning platters
    magnetic surface
    data arranges in tracks and sectors
    heads "float" on top of surface, move in and out to r/w data

what can go wrong?
    whole disk can stop working (electrical fault, broken wire, etc.)

    sector failures:

        can read or write from the wrong place (e.g., if drive is bumped)
        can fail to read or write because head is too high, or coating too thin
        disk head can contact the drive surface and scratch it

previous slide suggests disks fail often.  what does the manufacturer say?

show Seagate slide

    Mean time between failure : 700,000 hours
    80 years?  80 years ago we were programming with abacuses.

What's going on here?  How did they measure this?

    Ran, say, 1000 disks for say 3000 hours  (3 million hours total)
    Had 4 failures
    So claimed that they 1 failure every 750,000 hours

What does the actual failure rate look like?  Bathtub (5 years)

So what does this 700,000 hours actually tell us?   Probability of failure in the bottom of the tub.  Helpful if I am running a data center with a thousand machines and want to know how frequently I will have to replace a disk.  About 1 % failure rate per disk per year.

Would buying a more expensive disk help?

  not really   (show slide) -- double the MTTR, 5x the price

**Coping with failures:**

1st type of failure -- sector errors

What can I do
  (show arch diagram -- with disk, firmware, driver, FS)

1) Silently return the wrong answer

2) Detect failure --
  each sector has a header with a checksum
  every disk read fetches a sector and its header
  firmware compares checksum to data
  returns error to driver if checksum doesn't match

3) Correct / mask
  Re-read if firmware signals error (if an alignment problem)
  (Many other strategies)

These are general approaches -- e.g., could have checksums at the application level (in files), or a backup to recover / correct from a bad file.

Disks have checksums -- how much do they help?

(Show SIGMETRICS slide)

  1.7 million drives
  10% of drives develop errors over 2 years

These are unrecoverable read failures
Would have resulted in bad data going to file system

Do disks catch all errors? (Show FAST08)

1.5 million drives, 41 months, 400,000 corrupt reads on 1000 disks
(Don't know how many reads were done, but each drive probably read millions of blocks -- so we are talking trillions of reads.)

Very small fraction of uncaught failures.

But they do happen; however, many systems assume they don't (e.g., your desktop file system.)

What about whole disk failures?

Powerful solution: replication

Write everything to two or more disks

If read from one fails, read from the other

Diagram:

```
write (sector, data):
    write(disk1, sector, data)
    write(disk2, sector, data)

read (sector, data)
    data = read (disk1, sector)
    if error
        data = read (disk2, sector)
        if error
            return error
    return data
```

Does this solve all problems? (No)
Uncaught errors
Can do voting / TMR -- diagram

Other parts of the system that fail
    Show Non-stop Slide
    Replicate everything, including operating system -- run N
        copies of every application, compare every memory read / write
        Cost is huge, performance is pport

Assumes failure independence
    (what about a power failure)
Not what app wants -- application is writing large data items that span multiple blocks; what if there is a power failure halfway through those write

Summary
- Disks:
    - lifetime about 5 years
    - 1 %/ year random total failure
    - 10% of disks will develop unreadable sectors w/in 2 years
    - small fraction of uncaught errors
- Replication is a general strategy that can be used to cope

URLs:

http://h20223.www2.hp.com/NonStopComputing/downloads/KernelOpSystem_datasheet.pdf

HP Integrity NonStop NS16000 Server data sheet - PDF (May 2005)

http://www.usenix.org/events/fast08/tech/full_papers/bairavasundaram/bairavasundaram.pdf

http://www.seagate.com/docs/pdf/datasheet/disc/ds_barracuda_7200_10.pdf