Fault-tolerant design techniques

Key ingredients

- Error processing
- Error detection
- Error diagnosis
- Error recovery
- Backward & forward
- Fault treatment
- Fault diagnosis
- Fault isolation
- Reconfiguration

Strategies

- Tolerance achieved through redundancy by:
  - Fault masking
  - Reconfiguration
    - Fault detection
    - Fault location
    - Fault containment
    - Fault recovery

Redundancy

- Addition of information, resources, time
  - Hardware
    - Static
      - Fault masking: passive redundancy, no action required
    - Dynamic
      - Active redundancy: Detect faults and perform some action
    - Hybrid
      - Combine features of both; NMR with spares; self-purging redundancy
    - Voter
      - Triple modular redundancy
      - Formulas
      - N-Modular redundancy
      - Pros: if NoF =< f faults can be tolerated
      - Cons: higher cost
      - Hardware voting: faster, cost hw
      - Software voting: slow, no addition. Hw
        - Sift-out MR: N modules using comparators (compare each module's output); detectors (determine which disagreements are reported by the comparator produces 0 or 1); collector (produce system's output) etc.
    - Standby sparing
      - One module working, the others in standby; if a fault is detected, that module removed
      - Hot standby sparing
      - Cold standby sparing
      - Key advantage: fewer power tha N parallel modules
    - Pair-and-a-spare
      - Combine features of standby sparing
      - Two modules parallel, results compared to provide the error detection
    - Watchdog timers
      - Concept: lack of an action is an indicative of fault
      - Must be reset
      - Assumptions: system is fault free if it can perform repetitively an action
      - Can be used to detect faults
  - Software
    - Consistency checks: verify the correctness
    - Capability checks: verify possess the expected capabilities
    - ALU Tests: proc. can execute instructions and compare results known in ROM
    - Testing of communication: achieved by sending messages from one processor to another.
      - 2 approaches:
        - N-version prog.
          - Masks fault, relies on voting, concurrent
          - Design diversity: can tolerate hw and sew faults, not correlated ones; important to decide the #versions required
        - Recovery blocks:
          - Backward scheme, relies on acceptance test, serial
          - Multiple alternates (backups); primary task executes first, output checked by acceptance test, if not accept, Another is executed
          - Acceptance test are based on acceptable ranges (as inferential statistics)
          - Is also called primary-backup approach
      - SW rejuvenation
  - Information
    - Guarantee data consistency by exploiting additional info to achieve redundant encoding; it permit to detect and correct bits
    - 2 types of codes: Error detection codes, Error correction code
      - Redundant codes
        - Binary codes: Hamming distance
        - EDC: Intro redundancy in the info; an error generates a word not belonging to the code; error weight: # corrupted bits tolerated; using hamming we will calculate the distance as the number of different positions of bits; with parity code, we calculate the difference between the words obtained adding the parity bit(double/even errors are undetectable)
        - ECC: a code (with minimum distance d) can correct errors with weight <- (d-1)/2 per d(data) code) hamming
        - Circuito di EDAC (Error detection and correction)
  - Time
    - Transient Fault detection: computations repeated in different points in time, no extra hw;
    - Permanent fault detection: 1st comput. operands used; 2nd comp: operands ecoded

RAID (Redundant Array of Inexpensive Disks)

- Architecture
  - Multiple small, inexpensive disks drives into a group; appear as a single virtual drive; support fault-tolerance (redundant in disks); data striping for better performance.
  - Issues: access concurrency; throughput, data striping
- Levels
  - RAID-0
    - No redundancy --> no fault tolerance; high i/o perf; storage efficiency
  - RAID-1
    - Disk mirroring; best read perf; poor write perf; good fault tolerance
  - RAID-2
    - Bit level striping; uses Hamming (ECC); tolerate failures (#red disks= D(log(tot disks))); better stop. Eff. Than 1, high throughput but no conc.; exp. write.
  - RAID-3
    - Subtopic 1
  - RAID-4
    - Block Level striping; intro trade-off: parity disk is a bottleneck in small write; no prob for reads.
  - RAID-5
    - BLS with Distributed parity; reduces parity bottleneck; best small and large read; best large write; costly for small write
    - Limits: most employed; the larger the number of disks, the better perf. But the larger the prob of disk failure
  - RAID-6
    - BLS with Dual Distributed parity; 2sets of parity better fault tolerance; faster data reconstr. Than RAID-5); write worse than 5; due to overhead; better read perf. Cause data and parity are spread into more disks;

Error prop in DS & Rollback Error recovery Techniques

- Systems consist of N processes which comm through messages and cooperate to execute DistApp
- ROLLBACK
  - Treats DS as a coll. of processes that comm. through a net.; fault tolerance achieved by storage saving states; failed process restarts from the saved state (checkpoint)
  - Checkpoints
    - Uncoordinated process' autonomy in deciding checkpoints
    - Coordinated: orchestration of checkpoints in order to form a consistent global state; (coordinator processes communication to, coordinate)
    - Communication-induced: avoids domino; local & forced checkpoints
    - Protocols
      - Global checkpoint: set of N local checkpoints, one from each process, forming a CSS. The most recent is called recovery line.
  - Consistent System state
    - If a process's state reflects a message receipt, then the state of the corr sender reflects sending that message (fundamental goal for rollback-recovery protocol)