Appendix D: Storage Systems (Cont)

Instructor: Josep Torrellas
CS433
Reliability, Availability, Dependability

- Dependability: deliver service such that reliance can be placed on this service
- Actual vs specified behavior
- System failure: actual and specified behavior deviate
- A fault creates a latent error, which becomes effective when it is activated
- When the error actually affects the delivered service, a failure occurs
- Time between the occurrence of an error and the resulting failure is the error latency.

THEREFORE: given a fault:
- its manifestation in the system: an error
- manifestation of the error on the service: failure
Examples

• Example of a fault: a programming mistake
• Its consequence: a latent error (or an effective error)
• When the effective error causes erroneous data that affects service: a failure

• Example of a fault: alpha particle hitting a DRAM
• If it changes the memory: creates an error
• If the memory word is read and the error affects the delivered service: failure (if EEC corrected the error, a failure would not occur).

• Example of a fault: a mistake by a human operator
Notes

• An effective error often propagates from one component to another, creating new errors

• Service can be in two states:
  – Service accomplishment
  – Service interruption

• Transitions between these two states:
  – failures
  – restoration

• Module reliability: measure of the continuous service accomplishment. Often given as Mean Time To Failure (MTTF).

• Reciprocal of MTT: failure rate

• Service interruption: measured as Mean Time To Repair (MTTR).
Module Availability

- **Availability**: measure of the service accomplished with respect to the alternation between the two states

  - **Module Availability**: \( \frac{MTTF}{MTTF + MTTR} \)
  - **Mean time Between Failure (MTBF)** = \( MTTF + MTTR \)
Example: find system MTTF

- 10 disks, each 1,000,000 hour MTTF
- 1 SCSI controller: 500,000 hour MTTF
- 1 power supply: 200,000 hour MTTF
- 1 fan: 200,000 MTTF
- 1 SCSI cable: 1,000,000 MTTF

Assume: component lifetimes are exponentially distributed (age of the component is not important in the prob. Failure)

Assume: failures are indep

\[ \text{Failure rate system} = 10 \times \frac{1}{1M} + \frac{1}{0.5M} + \frac{1}{0.2M} + \frac{1}{0.2M} + \frac{1}{1M} \]

\[ = \frac{23}{1M} \]

\[ \text{MTTF} = \frac{1}{\text{Failure rate}} = \frac{1M \text{ hours}}{23} = 43,500 \text{ hours} \]
Classifications

- Faults:
  - Hardware faults: devices that fail
  - Design faults: faults in the software or hardware design
  - Operation faults: mistakes by maintenance personnel
  - Environmental faults: fire, flood…

- Faults:
  - Transient: exist for a limited time and are not recurring
  - Intermittent: system oscillate between faulty and fault-free
  - Permanent: remain
Classifications

• Error Recovery:
  – Backward: returns to a previous correct state, such as with checkpoint and restart
  – Forward: constructs a new correct state: TRM (triple module redundancy)

• Reliability Improvements:
  – Fault avoidance: prevents the occurrence of the fault
  – Fault tolerance: provides service with redundancy
Benchmarks of Disk Performance

1. Transaction processing benchmarks (TP or OLTP)
   - Concerned w/ I/O rate: disk accesses/second
   - Involves change to a large body of info (database) from many terminals
   - Need to guarantee proper behavior on a failure
e.g. bank transactions from ATM
   airline reservation systems
   - Several benchmarks: TPC-A, TPC-B, TPC-C, TPC-D
   measure # transactions per second (TPS) or per minute
Example: TPC-C

- Simulates an order-entry environment for a wholesale supplier
- Includes transactions to enter and deliver orders, record payments, check the status of orders, etc
- Runs 5 concurrent transactions of varying complexity
- Measured in transactions per minute (tpmC) and price of the system
- TPC benchmarks (Figure 6.12):
  - the higher the throughput, the better, but price included in benchmark results
  - however, benchmark requires that for throughput ↑
    the size of the files ↑
  - this scaling is necessary to ensure that benchmark measures I/O; else large memory w/small files
  - Throughput is the performance metric, but response times are limited
  - Benchmark results are audited.
Benchmarks of Disk Performance

2. Spec system level File Server, Mail and Web Benchmarks

• System Level File Server:
  – synthetic benchmark to evaluate NFS performance
  – contains a mix of rd/wr/file ops
  – scales the size of the file system according to the reported throughput: for every 100 NFS ops/second, capacity must increase by 1GB

• SpecMail: evaluate mail servers
• SpecWeb: evaluate web servers
Queuing Theory

- Since equilibrium $\Rightarrow$ input rate = output rate

-Little’s Law:

\[
\text{Mean Number of tasks in system} = \frac{\text{Arrival rate of tasks}}{\text{Mean response time}}
\]
**Queuing Theory**

The system is

arrivals \[\rightarrow\] queue \[\rightarrow\] Server

- $\text{Time}_{\text{server}} = \text{avg time to service a task}$
- Service rate $= 1/\text{Time}_{\text{server}} = \mu$
- $\text{Time}_{\text{queue}} = \text{avg time per task in the queue}$
- $\text{Time}_{\text{system}} = \text{T}_{\text{queue}} + \text{T}_{\text{server}} = \text{response time}$
- Arrival rate $= \# \text{Tasks arriving} / \text{second} = \lambda$
- $\text{Length}_{\text{server}} = \text{Avg} \# \text{tasks in service}$
- $\text{Length}_{\text{queue}} = \text{Avg length queue}$
**Queuing Theory**

Little’s law applied to each component

\[
\text{Length}_{\text{queue}} = \text{Arrival rate} \times \text{Time}_{\text{queue}}
\]
\[
\text{Length}_{\text{server}} = \text{Arrival rate} \times \text{Time}_{\text{server}}
\]

Example: time to service a disk request 50 ms

system requires 200 I/O req/second

On average, how many I/O req at the server?

\[
\text{Length server} = \text{Arrival rate} \times \text{Time}_{\text{server}}
\]

\[
200 \text{ req/sec} \times 0.05 \text{ sec} = 10 \text{ req}
\]

10 req on average at the server


**Queuing Theory**

Length\textsubscript{system} = Length\textsubscript{queue} + Length\textsubscript{server} = \#tasks in system

Therefore, Little’s law:

\[
\text{Length\textsubscript{system}} = \text{Arrival Rate} \times \text{Time\textsubscript{system}}
\]

Server utilization:

\[
\rho = \frac{\text{Arrival rate}}{\text{Service Rate}}
\]

Needs to be between 0 and 1

Also called *traffic intensity*

Example: Disk gets 10 I/O req/second

\[
\text{time to service 1 request} = 50\text{ms}
\]

\[
\text{server util} = \frac{\text{Arrival rate}}{\text{Service Rate}} = \frac{10 \text{ IOPS}}{1/0.05} = 0.5
\]
