高级计算机系统结构

沈海华
shenhh@ucas.ac.cn

第八讲 Availability & Reliability

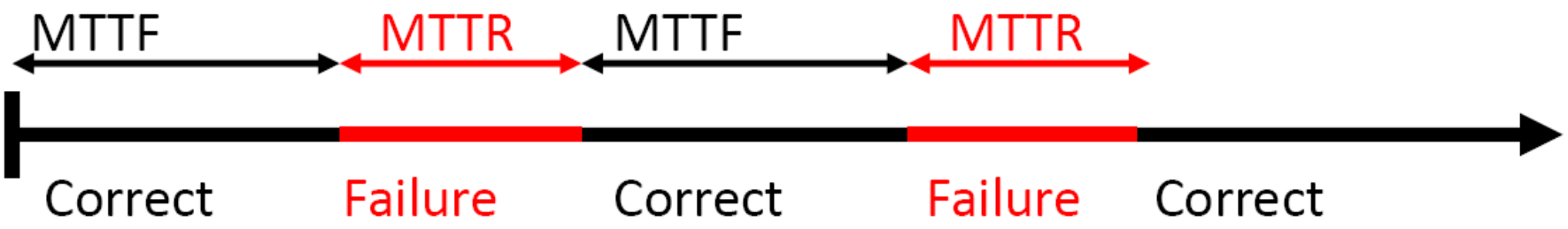
- What is availability and reliability?
- What are the principles of high-availability?
- What are some specific design optimizations?

(Module) Reliability

Reliability: measure of continuous "service"

- MTTF: mean time to failure
  - Time to produce first incorrect output
- MTTR: mean time to repair
  - Time to detect and repair a failure
- MTBF = mean time between failures = MTTF+MTTR
- Failure in time (FIT) = failures per billion hours of operation = 10^9/MTTF

(Service) Availability



Steady state availability

= MTTF / (MTTF + MTTR)

(Service) Availability

- Availability often quoted in "9s"
  - E.g. the telephone system has five 9s availability
  - 99.999% availability or 5 minutes of downtime per year

Uptime	Downtime in one year
99% (two 9's)	87.6 hours
99.9% (three 9's)	8.76 hours
99.99% (four 9's)	53 min
99.999% (five 9's)	5 min
99.9999% (six9's)	32 sec
99.99999% (seven 9's)	3 sec

Why is availability important?

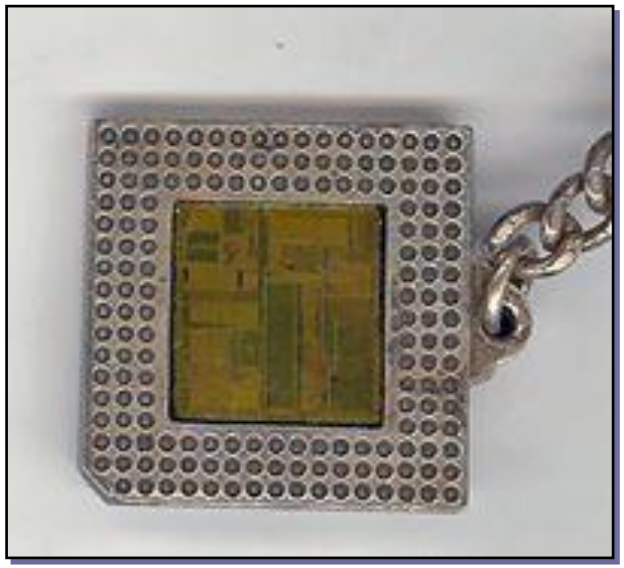
Application	Cost of downtime per hour	Annual losses with downtime of 1% (87.6 hrs/yr)	0.5% (43.8 hrs/yr)	0.1% (8.8 hrs/yr)
Brokerage operations	$6,450,000	$565,000,000	$283,000,000	$56,500,000
Credit card authorization	$2,600,000	$228,000,000	$114,000,000	$22,800,000
Package shipping services	$150,000	$13,000,000	$6,600,000	$1,300,000
Home shopping channel	$113,000	$9,900,000	$4,900,000	$1,000,000
Catalog sales center	$90,000	$7,900,000	$3,900,000	$800,000
Airline reservation center	$89,000	$7,900,000	$3,900,000	$800,000
Cellular service activation	$41,000	$3,600,000	$1,800,000	$400,000
Online network fees	$25,000	$2,200,000	$1,100,000	$200,000
ATM service fees	$14,000	$1,200,000	$600,000	$100,000

Figure 1.3 Costs rounded to nearest $100,000 of an unavailable system is shown by analyzing the cost of downtime (in terms of immediately lost revenue), assuming three different levels of availability, and that downtime is distributed uniformly. These data are from Kembel [2000] and were collected and analyzed by Contingency Planning Research.

Mission-critical (100% uptime), business-critical (minimal interruptions)

Business-critical

- 1994年，奔腾处理器被发现在执行某个特定的浮点运算时出现错误，这种错误27000年才可能出现一次。对此，Intel付出了巨额代价回收有缺陷的奔腾处理器。事后，Intel将回收的芯片做成了钥匙链 。



Cost = US$ 475,000,000

Mission-critical

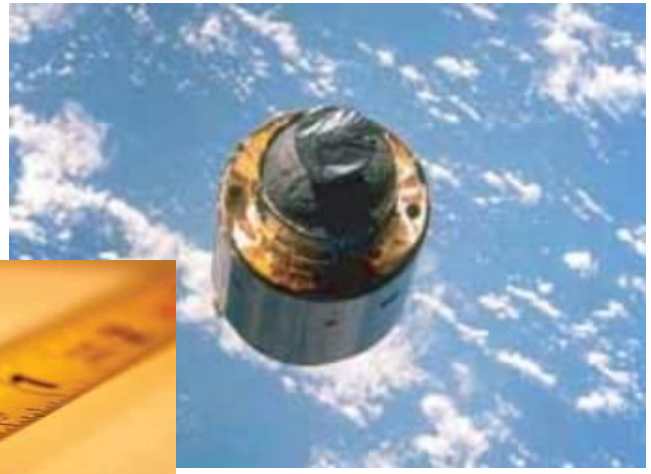
- 1996年6月4日，欧洲航天局研制的阿里亚娜五型火箭在发射后不到40秒爆炸。事后调查发现，错误发生于当一个很大的64位浮点数转换为16位带符号整数时出现异常。细微错误，使得十年的努力毁于一旦。

Left: Lift off: all seemed fine for the missio. Right: Bang: flaming fragments of the Ariane 5 and its payload of scientific equipment fall to earth. Pictures: AFP

Cost = UK£ 5,000,000,000

Mission-critical? Business-critical?

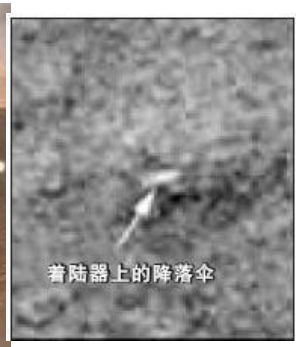
- 火星探测



– 1998 .2，NASA发射的一枚探测火星气象的卫星因未进入预定轨道坠毁。

原因：（承包商洛克希德马丁航天公司）计量单位没有把英制转换成公制。

损失：>1.25亿美元



2015年9月28日，美国航天局宣布火星存在流动水；2018年7月25日，法新社消息称，火星上发现了第一个液态水湖。

– 火星极地着陆者于美国东部时间1999.1.3日下午3时21分“乘坐”一枚德尔塔二型火箭离开地球，1999.12.3顺利到达火星轨道，随后失去联系。2005年美国发现失踪的火星极地着陆者残骸。

可能原因：在着陆前发动机过早关闭，导致飞船坠毁

损失：>1.2亿美元

Mission-critical

- 可能原因：迎角传感器(AOA)存在问题，会反馈错误数据。在飞行员手动飞行时，此问题会导致传感器给出飞机即将失速的错误数据，波音737 Max会自动将机头向下压。（埃塞俄比亚航空公司2019年3月10日）

Types of Faults

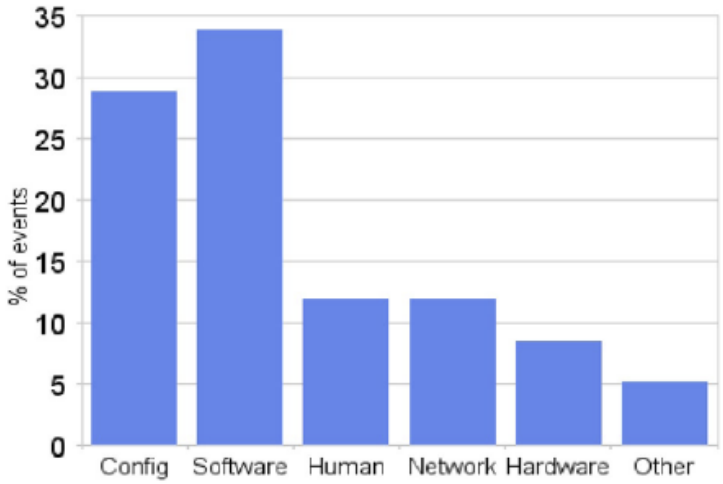
- Hardware faults
  - Radiation, noise, thermal issues, variation, wear-out, faulty equipment
- Software faults
  - OS, applications, drivers: bugs, security attacks
- Operator errors
  - Incorrect configurations, shutting down wrong server, incorrect operations
- Environmental factors
  - Natural disasters, air conditioning, and power grids
  - Wild dogs, sharks, dead horses, thieves, blasphemy, drunk hunters (Barroso'10)
- Security breaches
  - Unauthorized users, malicious behavior (data loss, system down)
- Planned service events
  - Upgrading HW (add memory) or upgrading SW (patch)

Types of Faults

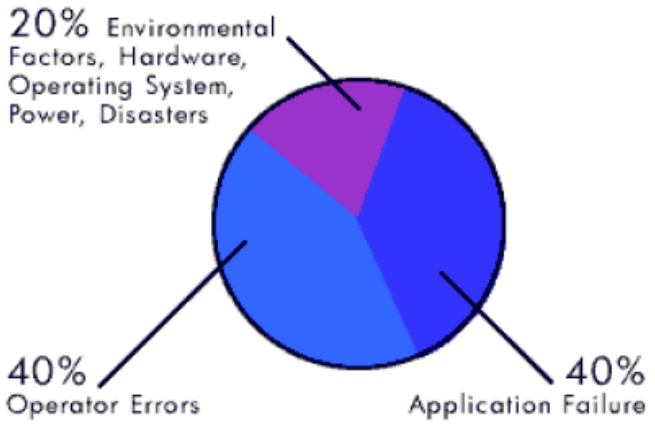
- Permanent
  - Defects, bugs, out-of-range parameters, wear-out, …
- Transient (temporary)
  - Radiation issues, power supply noise, EMI, …
- Intermittent (temporary)
  - Oscillate between faulty & non-faulty operation
  - Operation margins, weak parts, activity, …
  - Sometimes called Bohrbugs and Heisenbugs

Real-world Service Disruptions

Source of "disruptions events" at Google



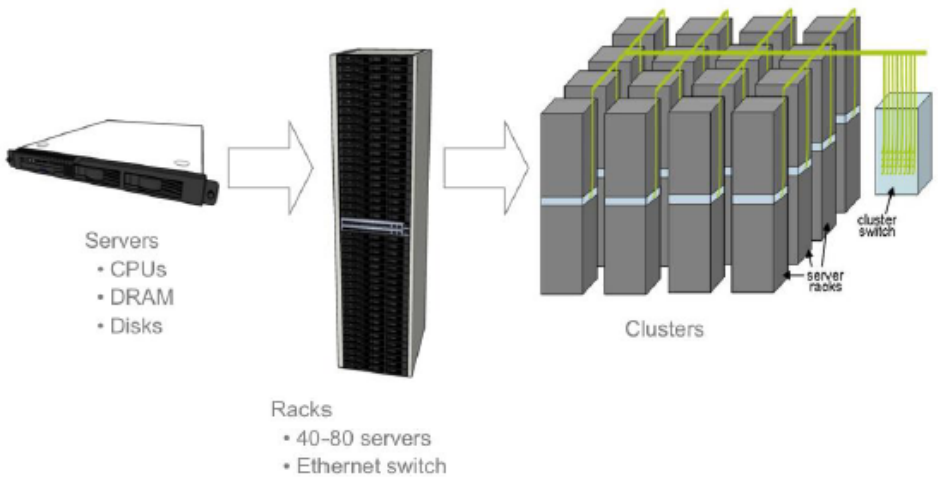
Source of enterprise "disruption events"

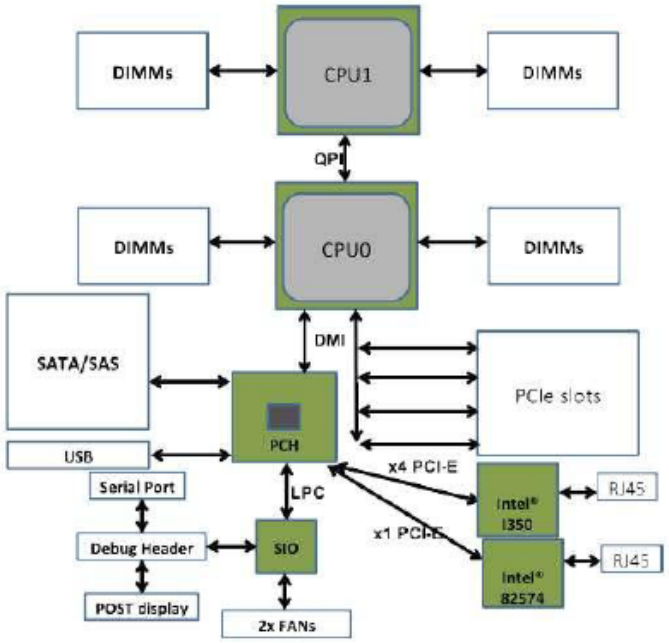


- Large number of techniques on hardware fault-tolerance
- Software, operator, maintenance-induced faults
  - Affect multiple systems at once, correlated failure FT harder

(disruption event = service degradation that triggered operations team scrutiny)

DC Hardware Overview





- Rows of rack-mounted servers (typically x86)
  - 2-socket server is the most common configuration these days
  - But other configurations are becoming popular too!
- Ethernet networking (1Gbps or 10Gbps)
  - Hierarchical network with high bandwidth topology

Data Center Example

- Typical first year for a new cluster:

- ~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
- ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
- ~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
- ~1 network rewiring (rolling ~5% of machines down over 2-day span)
- ~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
- ~5 racks go wonky (40-80 machines see 50% packetloss)
- ~8 network maintenances (4 might cause ~30-minute random connectivity losses)
- ~12 router reloads (takes out DNS and external vips for a couple minutes)
- ~3 router failures (have to immediately pull traffic for an hour)
- ~dozens of minor 30-second blips for dns
- ~1000 individual machine failures
- ~thousands of hard drive failures

- slow disks, bad memory, misconfigured machines, flaky machines, etc.

Exercise

- A 2400-server Google cluster has the following events per year:
  - Cluster upgrades: 4 (need to fix)
  - Hard drive failures: 250 (need to fix)
  - Bad memories: 250 (need to fix)
  - Misconfigured machines: 250 (need to fix)
  - Flaky machines: 250 (need to reboot)
  - Server crashes: 5000 (need to reboot)
- Assume: time to reboot SW is 5 minutes, time to fix HW is 1 hour
- What is service availability?

Answer

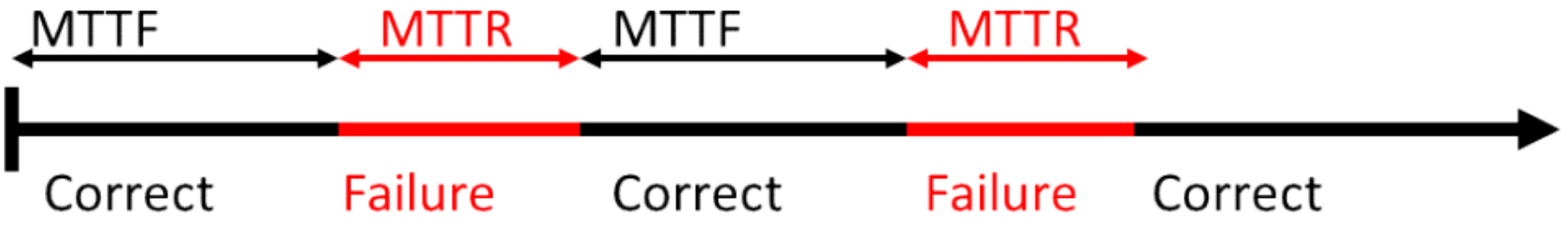
- Hours outage
  - (4 + 250 + 250 + 250) x 1 hour
  - + (250 + 5000) x 5 minutes =
  - 754 + 438 = 1192 hours
- Hours in year = 365 x 24 = 8760
- Availability = 1 – 1192/8760 = 86%

Did Google have only 86% uptime last year?

Faults ≠ Failure

- Failure: service is unavailable or data integrity is lost
  - The user can really tell
- Possible effects of a fault (increasing severity)
  - Masked (examples?)
  - Degraded service
  - Unreachable service (service not available)
  - Data loss or corruption (data are not durable)

Techniques for Availability



- Steady state availability = MTTF / (MTTF + MTTR)

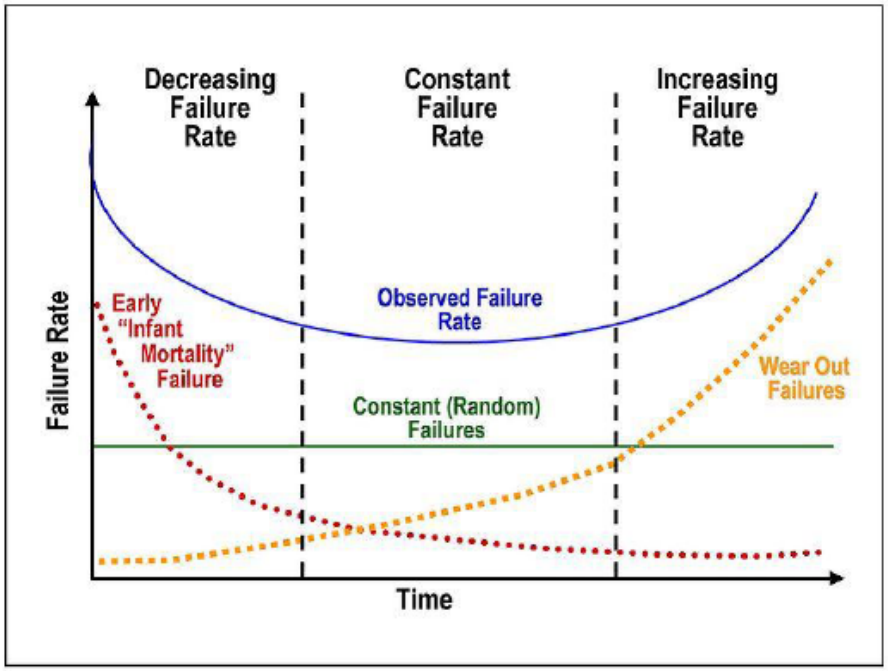
- For higher availability, you can work on
  - Very high MTTF (reliable computing, fault prevention)
  - Very high MTTF (fault-tolerant computing)
  - Very low MTTR (recovery-oriented computing)

Improving MTTF & MTTR

- Two issues: error detection and error correction
- Observations
  - Both are useful (e.g., fail-safe operation after detection)
  - Both add to cost so use carefully
  - Can be done at multiple levels (chip/system/DC, HW/SW)
- Some terminology
  - Fail-fast: either function correctly or stop when error detected
  - Fail-silent: system crashes on failure (not necessarily immediately)
  - Fail-safe: automatically counteracting a failure
- Following slides: example techniques
  - General, disks, memories, networks, processors, system…

General: “Infant Mortality”

- Many failures happen in early stages of use
  - Marginal components, design/SW bugs, etc
- Use “burn-in” to screen such issues
  - E.g., stress test HW or SW before deployment
  - See next slide for some basic approaches

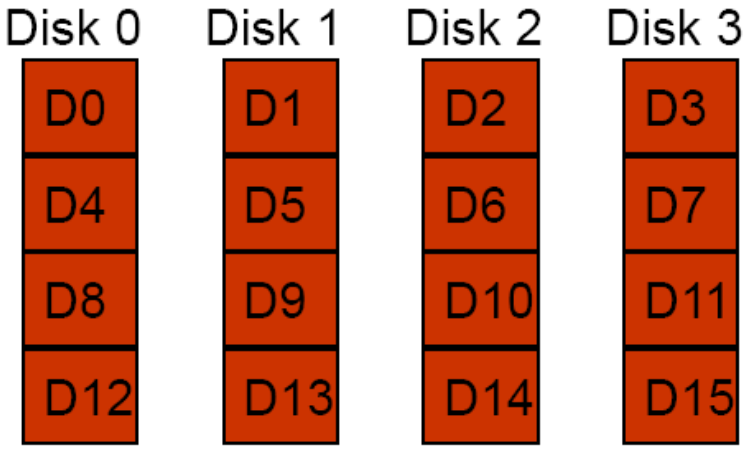


RAID: Dealing with Faults in Storage Systems

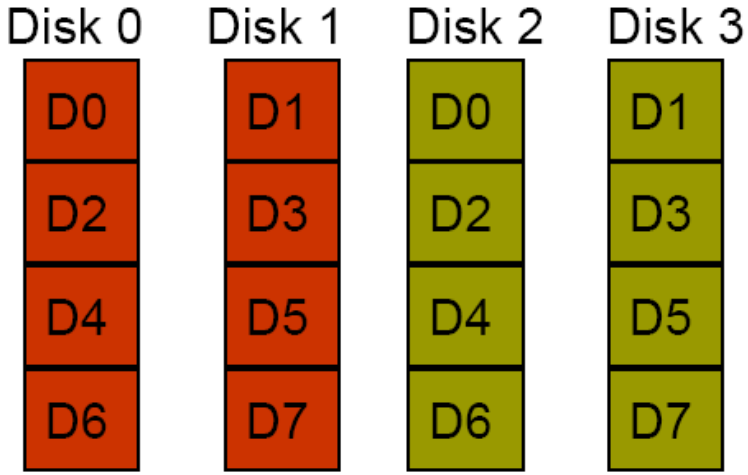
- Redundant arrays of inexpensive disks (RAID)
  - A collection of disks that behaves like a single disk with
    - High capacity, high bandwidth, high reliability disk
  - Key idea in RAID: error correcting information across disks
  - Many organizations; two distinguishing features:
    - The granularity of the interleaving (bit, byte, block)
    - The amount and distribution of redundant information
  - Patterson's classification: RAID levels 0 to 6

Level	Description
RAID0	Block-level striping without parity mirroring
RAID 1	Mirroring without parity striping
RAID 2	Bit-level striping with dedicated parity
RAID 3	Byte-level striping with dedicated parity
RAID 4	Block-level striping with dedicated parity
RAID 5	Block-level striping with distributed parity
RAID 6	Block-level striping with double-distributed parity
RAID 1+0	Disk mirroring and data striping without parity

RAID 0 & 1: Block Duplication

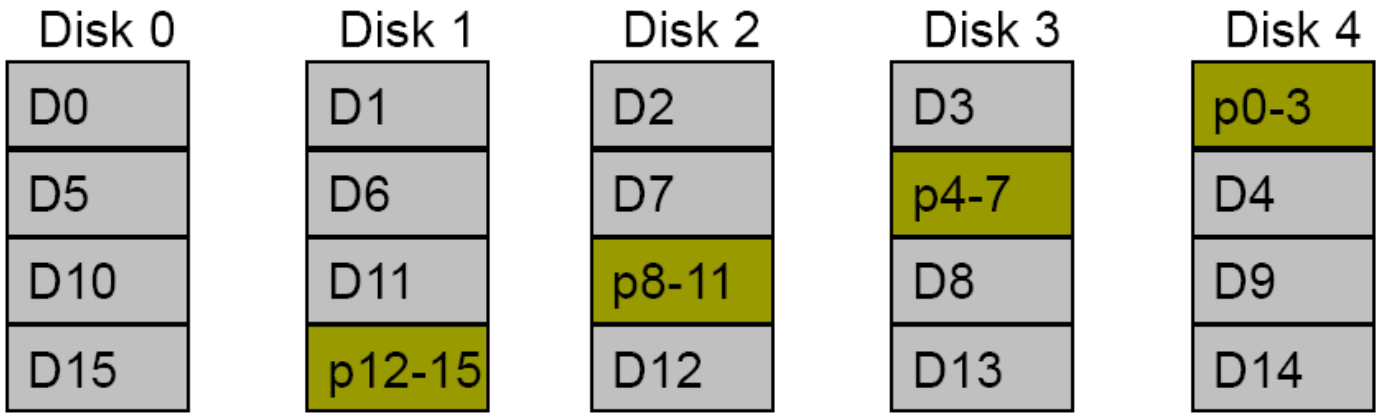


- RAID 0: Non-redundancy (sharding)
  - Best write performance
  - Not necessarily the best read latency!
  - Worst reliability



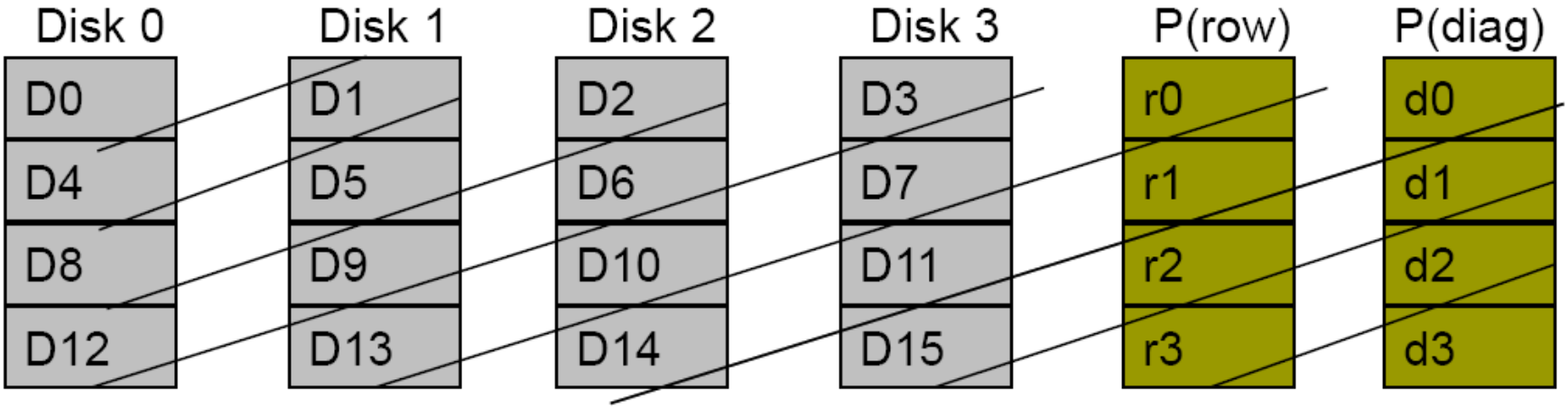
- RAID 1/1+0: Mirroring (shadowing)
  - Half the capacity
  - Faster read latency: schedule the disk with the lowest queuing, seek, and rotational delays

RAID 5: Block-level, Distributed Parity



- One extra disk to provide storage for parity blocks
  - Given N block + parity, we can recover from 1 disk failure
- Parity distributed across all disks to avoid write bottleneck
  - Best small/large read, and large write performance of any RAID
  - Small write performance worse than, say, RAID-1

RAID 6: Multiple Types of Parity



- Two extra disks protect again 2 disk failures
  - E.g., row and diagonal parities
- Parity blocks can be distributed like in RAID-5
  - Separate parity disks shown for clarity
- Also called RAID/ADG

RAID Discussion

- RAID layers tradeoffs
  - Space, fault-tolerance, read/write performance
- HW-based RAID-5 is becoming less popular
  - RAID1+0 is often used for large disk arrays
- RAID can be done in SW & across servers
  - RAID-5 like approaches are popular

Dealing with Faults in Memories

- Permanent faults (stuck at 0/1 bits)
  - Address with redundant rows/column (spares)
  - Built-in-self-test & fuses to program decoders
  - Done during manufacturing test
- Transient faults
  - Bits flip 0→1 or 1→0
  - Detect using parity (i.e., a 9th bit per byte)
    - Even parity: make the 9th bit 1 if the number of 1s in the byte is odd

ECC for Transient Faults

- Error correcting codes
  - Error correction using Hamming codes
    - E.g., add a 8-bit code to each 64-bit word
  - Common case for DRAM: SECDED (y=2, x=1)
    - Can buy DRAM chips with (64+8)-bit words to use with SECDED (single error correct double error detect)
  - Codes calculated/checked by memory controller

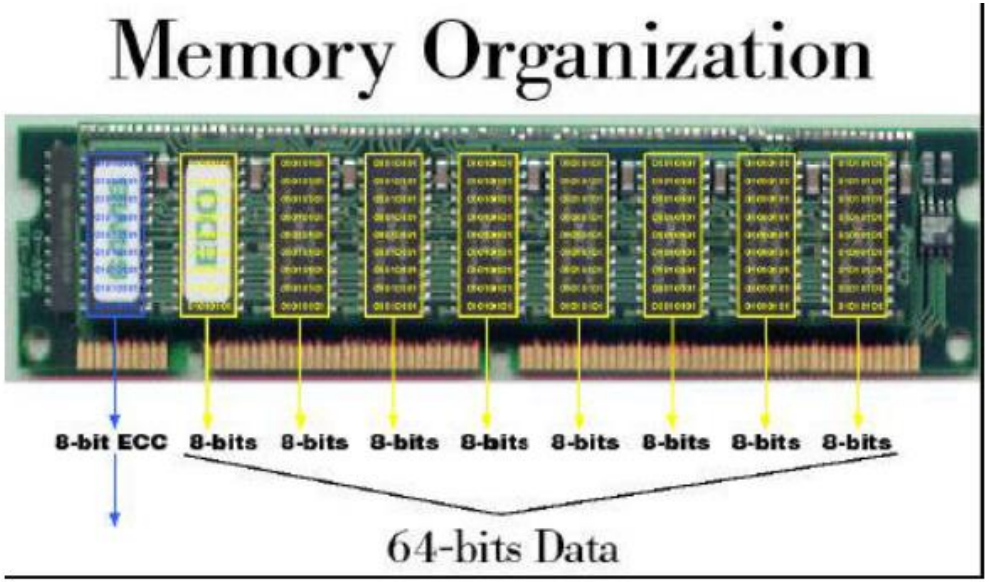
Error Correcting Codes (ECC)

- P extra parity bits for N bits of data
  - Distance: the # of bits that are different between two words
  - Example: 0001 and 1000 have distance D=2
  - We can detect any single-bit error given these two words
- In general, codes were valid words are at least D bits apart
  - We can detect D-1 errors
  - To correct x errors, we need the D >= 2x+1

ECC Issues

- Performance issue
  - Every subword write is a read-modify-write
  - Necessary to update the ECC bits
- Reliability issue: double-bit errors
  - Likely if we take a long time to re-read some data
  - Solution: background scrubbing
- Reliability issue: a whole-chip failure
  - Possible if it affects interface logic of the chip
  - Solutions?

ChipKill: RAID for DRAM



- Chipkill DIMMs: ECC across chips (or even DIMMS)
  - Instead 8 use 9 regular DRAM chips (64-bit words)
  - Can tolerate errors both within and across chips
  - Disadvantage: coding overhead
    - A chip failure affects 8 bits now!

Failure Numbers [IBM Study]

- 10K processors with 4GB per server → following rates of unrecoverable errors in 3 years of operation
  - Parity only: about 90K; 1 unrecoverable failure every 17 minutes
  - ECC only: about 3.5K; 1 unrecoverable or undetected failure every 7.5 hours
  - Chipkill: about 6; one unrecoverable/undetected failure every 2 months
- 10K server chipkill = same error rate as a 17-server ECC system

Dealing with Network Faults

- Use error detecting codes and retransmissions
  - CRC: cyclic redundancy code
  - Receiver detects error and requests retransmission
    - Requires buffering at the sender side
  - An ack/nack protocol is typically used
    - To indicate when the receiver received correct data (or not)
  - Time-outs to deal with the case of lost messages
    - Error in control signals or with acknowledgements
- Permanent faults
  - Use network with path diversity

Dealing with Faults in Logic

- Triple module redundancy (TMR)
  - 3 copies of compute unit + voter
  - Issues: synchronization & common mode errors
- Dual modular redundancy (DMR)
  - 2 copies of compute unit + comparator
  - Can use simpler 2nd copy (e.g., parity predictor)
- Checkpoint & restore
  - Periodic checkpoints of state
  - On error detection, rollback & re-execute from checkpoint
  - Issues: checkpoint interval, detection speed, # of checkpoints, recovery time, …

Data Center Availability

- Mostly system-level, SW-based techniques
  - Using clusters for high-availability
  - Active/standby; active/active
- Reasons
  - High cost of server-level techniques
    - Cost of failures vs cost of more reliable servers
  - Cannot rely on all servers working reliably anyway!
- But components need to be reliable enough…
  - ECC based memory used (detection is important)

DC Availability Techniques

Technique	Performance	Availability
Replication	✔	✔
Partitioning (sharding)	✔	✔
Load-balancing	✔	
Watchdog timers		✔
Integrity checks		✔
App-specific compression	✔	
Eventual consistency	✔	✔

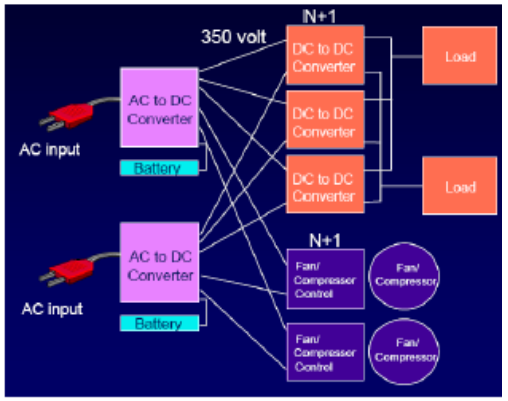
- Other techniques
  - Fail-safe behavior, admission control, spare capacity
  - Use monitoring/deployment mgmt system to handle failures as well

Discussion

- Availability in GFS / HDFS
- Availability in MapReduce / Hadoop

Is There More?

- Reliability in power supply & cooling
  - Tier I: no power/cooling redundancy
  - Tier II: N+1 redundancy for availability
  - Tier III: N+2 redundancy for availability
  - Tier IV: multiple active/redundant paths
- Other server design features
  - Hot pluggability, redundancy, monitoring, remote management, diagnostics, battery-backup



Other Lessons & Advice

- In general, fault prediction is difficult
  - Exception: common mode errors
- Repairs
  - Given some spares, repairs can be delayed
  - Must compare cost of repair vs cost of spare (TCO)
- **If it is not tested, don't rely on it**
- **Check for single points of failure**

Key principles: Modularity, fail-fast, independence of failure modes, redundancy and repair, single-point of failures

Performance & Availability

- Graceful degradation under faults
  - E.g., search quality results with loss of systems
  - E.g., delay in access to email
  - E.g., corruption of data of web (really bad)
  - Internet itself has only two nines of availability
- More appropriate availability metrics
  - Yield = fraction of requests satisfied by service/total number of requests made by users
  - Performability: composite measure of performance and dependability

致谢：

本讲内容参考了M.I.T. Daniel Sanchez教授的课程讲义，特此感谢。