CSG2132 Module 3 Lecture Notes

Methods for Predicting Disk Failure

Mean Time Between Failure (MTBF)

Mean time between failure (MTBF) measures the average time computing and network hardware is likely to function before it ceases to be functional. The metric allows technical personnel estimate how long computing or network components are likely to be operate reliably. The likely operational reliability of hardware in terms of duration is known as *uptime*.

MTBF assumes repairability!

MTBF is calculated by dividing a hardware item's total uptime by the number of breakdowns it experiences within any given time frame. Algorithmically speaking, this would be ***MTBF** = Total Uptime (TU) / No of Breakdowns (#Bd)*.

EXAMPLE: Assume that 10 HDDs are tested for 500 hours, and that during this time, two (2) failures occur, then our MTBF calculation would looks something like this:

MTBF= (10[devices]*500[hrs per device])/2[failures]
MTBF = 2,500 hrs between repairs

Mean Time to Failure (MTTF)

Mean Time to Failure (MTTF) measures the average time computing and network hardware is likely to function before it fails completely and is no longer usable.

MTTF assumes non-repairability!

EXAMPLE: Assume five (5) network interface cards (NICs) are tested to failure with the following results:

#	*Hrs to Failure*
NIC1	2000
NIC2	2300
NIC3	1900
NIC4	2020
NIC5	1950
TOTAL	10170

Our MTTF calculation would looks something like this:

MTTF = Total Hours / Devices Tested
MTTF = 10170/5
MTTF = 2034hrs before failure

Self Monitoring, Analysis and Reporting Technology (SMART)

Self-Monitoring, Analysis and Reporting Technology (SMART) is a HDD/SSD monitoring system that detects and reports upon multiple drive reliability indicators to try and predict imminent hardware failure and notify technical staff or users to take action to prevent data loss or initiate replacement.

There are dozens of HDD/SDD operational attributes that SMART uses to predict failure, with some of the most seen being:

HDD/SSD Attribute	Description
SMART 5 Reallocated_Sector_Count	Number of bad sectors that have had to be reallocated
SMART 187 Reported_Uncorrectable_Errors	Number of errors that could not be automatically corrected
SMART 188 Command_Timeout	Number of operations that have been aborted due to the disk timeouts
SMART 197 Current_Pending_Sector_Count	Number of sectors that may be remapped if they are not successfully read
SMART 198 Offline_Uncorrectable	Number of uncorrectable errors when reading or writing a sector

However, SMART has proved to be an inconsistent predictor of HDD/SSD failure at best and cannot be considered effective or reliable for use in large scale datacentres.

Annualized Failure Rate (AFR)

Annualized Failure Rate (AFR) estimates the probability a hardware device will fail during a full year of use. AFR is calculated by dividing the established MTBF of a device and the total number of hours it will be used in a year. For a device that is always in operation, the annual number of hours is **8766**.

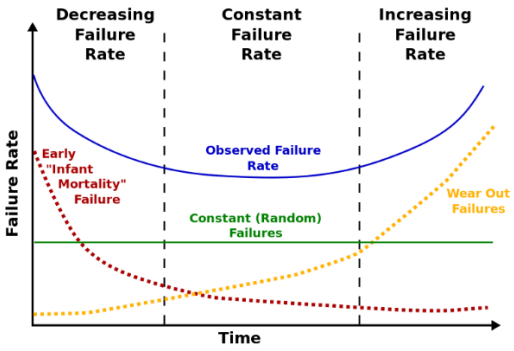
EXAMPLE: If a SATA drive has an MTBF of 300000 hrs, then the AFR calculation would be as follows:

AFR = (8766/300000)*100
AFR = 0.02922 * 100
AFR = 2.92%

However, because MTBF are usually provided by HDD/SSD manufacturers, who often overstate this value for marketing and sales purposes, AFR too is best used as a general guide rather than a hard and reliable metric.

Real World Failure Rates

In the real world, there are increasing and decreasing failure rates based on the stage of life of a hardware device. As shown in the diagram below and the accompanying table, these are the *Decreasing Failure Rate* in early life, *Constant Failure Rate* in useful life, and *Increasing Failure Rate* in end of life. When managing hardware devices in a datacentre, always being aware of what stages of life they are at is crucial to avoiding unanticipated breakdowns and data loss.



Failure Type	*AKA*	*Desc.*
Decreasing Failure Rate	Infant mortality Burn-in Early failures	Usually result from manufacturing faults
Constant Failure Rate	Random failures Useful life	Usually result from ongoing stress outside of established operational tolerances
Increasing Failure Rate	Wearout failures	Usually result from corrosion, oxidation, friction, fatigue

Redundancy Using RAID

Redundant Array of Independent Disks (RAID) is a technology that allows the same data to be stored in different places across multiple hard disks or solid-state drives to protect that data in the case of failure. RAID is an essential component of any datacentre.

There are two (2) key elements to RAID, 1) how data is organised across a disk array using techniques such as striping, mirroring, parity, or a combination of the three, and 2) the management of a RAID array at the hardware and software level.

RAID Levels

Raid devices can be set up using different configuration versions known as levels ranging from RAID 0 through to RAID 6. These are known as the *standard* RAID levels. Some of these

standard levels can then be combined to achieve combined performance and redundancy advantages such as RAID 10 and RAID 50. These are knowns as *nested* RAID levels.

All levels of RAID use one or more of three techniques to write data to and read data from a disk array. These techniques are known as *striping*, *mirroring* and *parity*, each briefly explained in the table below:

RAID r/w Technique	Description
Striping	This RAID technique involves splitting the stream of data to be stored into uniformly sized data blocks which are then written one by one ordinally across a RAID controlled disk array. Striping is great for high speed write operations but provides no redundancy benefits.
Mirroring	This RAID technique involves making identical copies of the data to be written disk, and stores each of these on separate RAID controlled disks simultaneously. Mirroring is used when fault tolerance to prevent data loss is important, but because everything is duplicated, disk space is less optimally utilised and write speeds can be impacted.
Parity	This RAID technique involves the combination of striping with checksum methods to generate and store parity blocks from which lost data can be reconstructed. Using parity, data lost due to the failure of a drive can be recalculated from the checksum bits on the parity drive. This method provides an even high degree of fault tolerance and data redundancy advantages but tends to slow write performance significantly.

Each of the standard and nested RAID levels is explained in the table below:

RAID Level	Type	Description
RAID 0	Disk Striping	RAID 0 use a technology called *disk* striping. This technique splits data across multiple disks allowing *higher data throughput*. Individual files are read from these multiple disks, this providing speed and capacity advantages over single disk use. However, RAID 0 does not provide redundancy and fault tolerance of any kind because it does not duplicate data or store any parity information. In RAID 0, all disks in the array appear as a single partition, so if one disk fails, the whole array fails, and data is inevitably lost. RAID 0 is the favoured method for caching applications speed is important and reliability and/or data loss secondary, such as live stream caching.

RAID 1	Disk Mirroring	RAID 1 writes and reads identical data to pairs of drives. This technique is known as data mirroring and it's a key approach to providing effective data redundancy on storage devices. Should any of the disks in a RAID 1 array fail, the data can still be accessed from the remaining disks in the array. Faulty disks can be replaced, with data being copied to them from the still functioning disks in the array once its presence is detected. RAID 1 is used to implement failover data storage solutions due to the fault tolerance and easy data recovery option that it provides. However, it does have the disadvantage of reducing the usable capacity of storage devices on the network, hence resulting in a higher cost-per-megabyte.
RAID 5	Striping w/ Parity	RAID 5 stripes data across multiple disks just like RAID 0 does, however, it provides an effective redundancy option in that it also stores parity information that allows lost data to be recovered should a disk in the array fail. Parity information, in essence, is a small amount of binary data that describes a much larger block of data and provides a mechanism for its retrieval if lost. The advantage of RAID 5 is that it offers both the speed of RAID 0 and effective data redundancy and recovery using parity data stored on a third disk. This means that if any disk in a storage array fails, the data it contained can be rebuilt from the remaining data on the still-operational drives and the parity bits. RAID 5 provides fault tolerance combined with the increased performance of RAID 0, however, it may reduce the performance of servers that need to do voluminous read/write operations due to the existence of parity overhead. RAID 5 is extensively used in file and application servers.
RAID 6	Striping w/ Double Parity	Raid 6 works similarly to RAID 5, with the key difference being that it uses two parity blocks per write operation rather than one. This provides even further reliability advantages because it allows for two drives in the array to fail without breaking it. Thus, RAID 6 provides even greater redundancy than RAID 5 while still providing high read performance. Of course, because RAID 6 servers have to perform a great deal more operations to generate the double-parity values, performance in this regard is somewhat slower than other RAID configurations. However, RAID 6 is ideal for very large file and application server purposes.

RAID 10 (Nested RAID)	Striping + Mirroring	RAID 10 combines the mirroring of RAID 1 with the striping of RAID 0, which is a good configuration for server storage applications that require the redundancy benefits of RAID 1 and the performance benefits of RAID 0. RAID 10 is often used for applications that require high data security and high performance such as real-time database driven applications, eCommerce, GPS-driven systems and mass scale authentications systems. The main disadvantage of RAID 10 is that it greatly lowers usable disk capacity which inversely raises operational costs.
RAID 50 (Nested RAID)	Striping w/ Distributed Parity	RAID 50 (sometimes referred to RAID 5+0) combines the distributed parity of RAID 5 with the data striping of RAID 0 to improve RAID 5 write performance without reducing data protection through enhanced redundancy.

Hardware vs. Software RAID Controllers

A RAID controller manages a HDD/SSD array so that they work together according to the specifications of the RAID level being used. A RAID controller can either be implemented in the forms of a RAID card, or as software in circumstances where virtualisation is being used.

RAID Controller Type	Description
RAID Card (Hardware)	Hardware RAID controllers tend to be the preferred option in bigger and larger RAID configurations because processing is handled by a dedicated RAID processor rather than a server processor. The latter has latency and processing load issues associated with its use. Further, because its operating at the Network layer if the OSI, OS-specific compatibility issues can be largely avoided. The cost of high-performance RAID cards can be a limiting factor when tight budgets are a reality.
Software RAID Controller	Software RAID controllers are set up and managed in the server operating system being used, be Microsoft, Linux, OpenBSD, FreeBSD, NetBSD or Solaris Unix. They are a much less expensive option than hardware RAID controllers, and because they are configured with the OS, are much easier to work with in many cases. However, software RAID controllers are often software specific, which does pose compatibility issues, and will often restrict the RAID levels that can be used.

ZFS

ZFS is a filesystem that is designed to aid in the efficient and safe storing of data created by Sun Microsystems (now owned by Oracle) and released for OpenSolaris in November 2005. In 2008, ZFS was ported to FreeBSD. It can also be used with Linux but is not included in the Linux kernel due to licence compatibility issues. However, most Linux distros do provide methods by which to install ZFS. The open source versions of ZFS is maintained by the OpenZFS project, and are available for FreeBSD, Linux, macOS and Windows.

Most file systems are separate from the lower level management of physical devices, focusing on providing file level access. ZFS combines block level management of physical devices with the filesystem providing features like software RAID, pooled storage, Copy-on-write, snapshots, data integrity verification and automatic repair (scrubbing), RAID-Z, a maximum 16 Exabyte file size, and a maximum 256 Quadrillion Zettabytes storage with no limit on number of filesystems (datasets) or files. An overview of each of these is provided in the table below:

ZFS Feature	Description
Pooled Storage	ZFS combines the features of a file system and a volume manager. This means it can create a file system that spans across a series of drives, known as a pool. When adding more drives to the pool, ZFS automatically handles partitioning and formatting.
Copy-on-write	Instead of overwriting an existing block of information on a drive, ZFS writes new information to a different block and updates file system metadata to point to the latest block. This ensures old data is preserved in the event of a system error.
Snapshots	Snapshots are copies of the original version of an existing file ZFS maintains in the files system that then exists as an earlier version of an updated version. It's sort of a file system *track changes* function. If the current version of a file is deleted, so to are any earlier snapshots linked to it.

RAID-Z	RAID-Z is the ZFS implementation of RAID-5, which it is able to implement without the need for additional hardware or software. RAID-Z offers three (3) versions, these being RAID-Z1, RAID-Z2 and RAID-Z3. RAID-Z1 requires at least two (2) disks for storage and one (1) for parity. RAID-Z2 requires at least two (2) storage drives and two (2) for parity. RAID-Z3 requires at least two (2) storage drives and three (3) for parity. Drives have to be added in multiples of two (2) to ZFS controlled drive pools.
Huge Storage Potential	ZFS is built on a 128-bit file system which means that it can handle 16bn times the capacity of 32 or 64-bit systems. This makes ZFS a highly attracted option in a world where the outer boundaries of data storage so badly need extending.

ZFS can be used with hardware RAID but this is inadvisable. ZFS is designed to work on JBOD (Just a Bunch of Disks) and apply its own parity and data distribution. RAID-Z behaves similarly to RAID 5 but the stripes are different sizes considering the file boundaries. It uses checksums to detect which parts of the array have been corrupted and is capable of rebuilding or repairing only the parts required.

ZFS can also support hot spares to remove downtime when repairing failed disks. Disks do not even need to be the same size to be managed by ZFS and arbitrary storage devices can be added at any time. It provides powerful snapshotting capabilities to facilitate backups or to efficiently restore storage to a previous state.