DEPENDABLE SYSTEMS AND CRITICAL INFRASTRUCTURES DESIGN

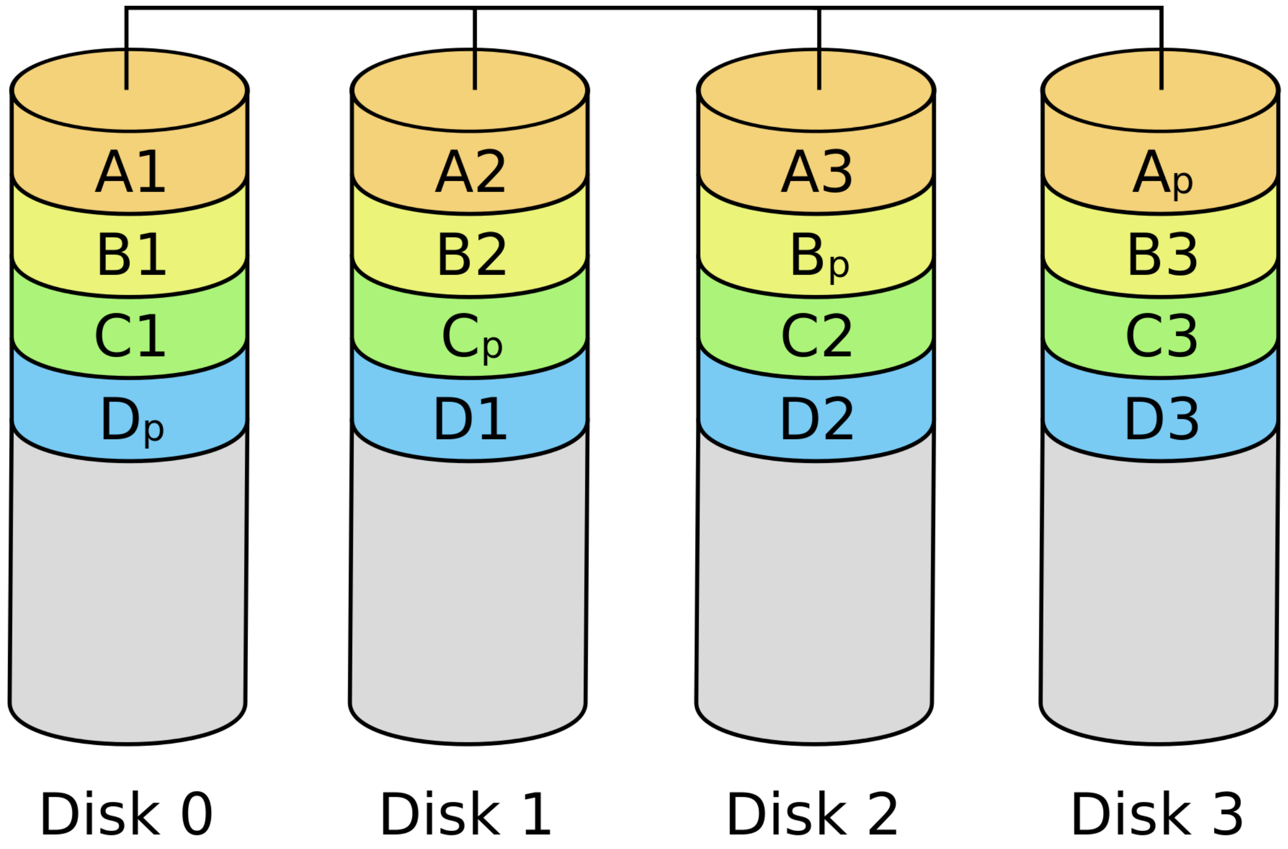
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ Κυβερνοασφαλεια και Επιστημη Δεδομενων
MSc Cybersecurity and Data Science
DEPT OF INFORMATICS
UNIVERSITY OF PIRAEUS

RELIABILITY ENGINEERING AND HARDWARE FAULT-TOLERANCE

DIMITRIS AGIAKATSIKAS, MIHALIS PSARAKIS

OUTLINE

- In our previous lecture we covered
  - HARDWARE Redundancy
  - INFORMATION Redundancy → Example: Extra (redundant) bits are added to the original data bits of a RAM so that an error in the data bits can be detected and corrected. These are usually called Error Correction Codes (ECC).
  - SOFTWARE → Example: Mitigate software faults (bugs) by independently producing (from disjoint teams of programmers) two versions of a software in the hope that the different versions will not fail on the same input.
- In this lecture we will cover the following
  - RAID schemes
  - Non-stop systems
- In our next lecture we will cover
  - Fault tree analysis
  - FMEA



SECTION 1 INFORMATION REDUNDANCY

REDUNDANT ARRAY OF INDEPENDENT (INEXPENDED) DISKS (RAID)

- Data storage virtualization technology that combines multiple physical disk drive components into one or more logical units
- RAID improves fault-tolerance and/or read/write throughput
- There are many types of RAID, each one providing different benefits
- RAID is implemented in hardware (more expensive but better performance) or software

REDUNDANT ARRAY OF INDEPENDENT (INEXPENDED) DISKS (RAID) APPLYING REDUNDANCY AT A HIGHER LEVEL THAN INDIVIDUAL WORDS

The are six RAID structures. RAID Level 0 to 5

RAID 0 (also known as striped volume)

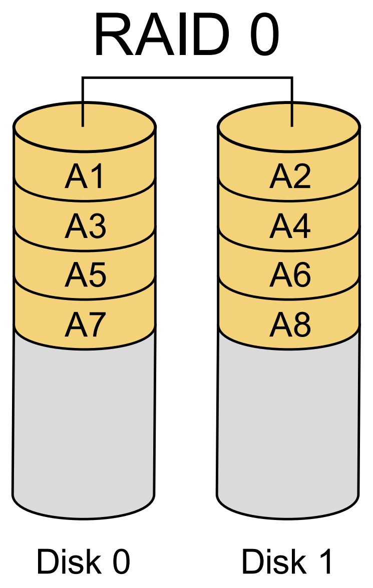
- Stripes ("splits") splits the data evenly across multiple disks (usually 2)
- It does not add any information redundancy and does not provide any fault-tolerance
- **Used to increase read/write speed**
- **Not reliable!**

Types of striping
- Bit
- Byte
- Block

What happens if we create RAID0 with two disks of different capacity?

Can you create the Reliability Markov Chain model of RAID 0?

- The following example distributes data between two disk into 4 stripes:
  - A1:A2, A3:A4, A5:A6, A7:A8



credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

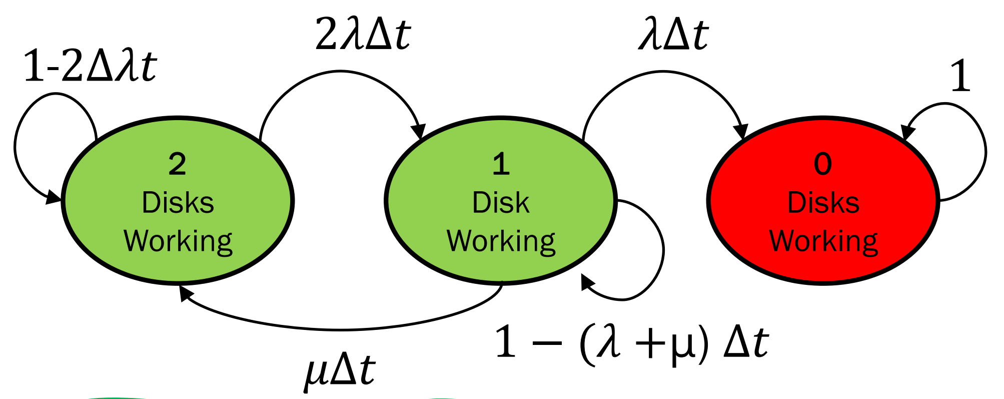
RAID 1 (ALSO KNOW AS "MIRROR" MODE)

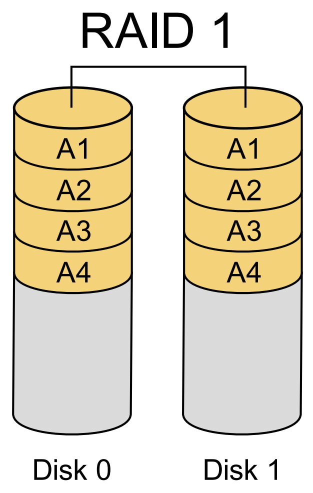
What is the code rate of RAID1?

- Consists of an exact copy of data to multiple disks (usually 2 disks)
- Pros
  - Provides high reliability
  - High read performance
- Cos:
  - Low write performance

- The following example makes two copies of data:
  - A1=A2, A2=A2, A3=A3, A4=A4

Reliability Markov Chain model of RAID 1



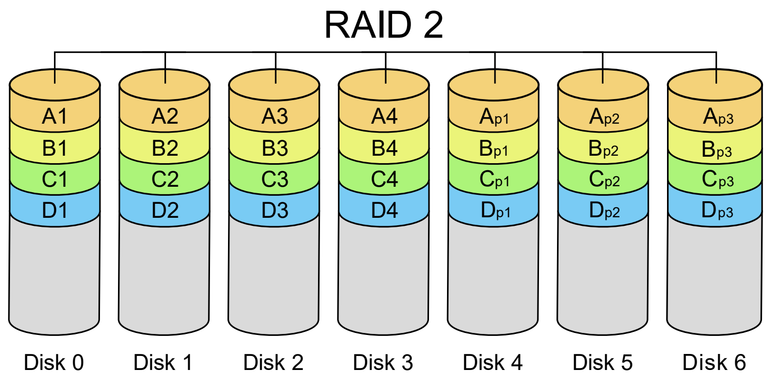


credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

RAID 2

- Level 2 RAID consists of a bank of data disks in parallel with Hamming-coded disks.
- It is rarely used. It stripes (i.e., distributes) data at the **bit level** to ***d*** multiple disks and saves their parity to ***r*** disks.
- Pros:
  - High transfer rates in long sequential reads
  - Random read/write is not ideal
- Cos:
  - Striping data at the bit in not very efficient

- Example:
  - Usable data is distributed in disks 0-4
  - Disks 5-6 are used for storing parity (Hamming codes)

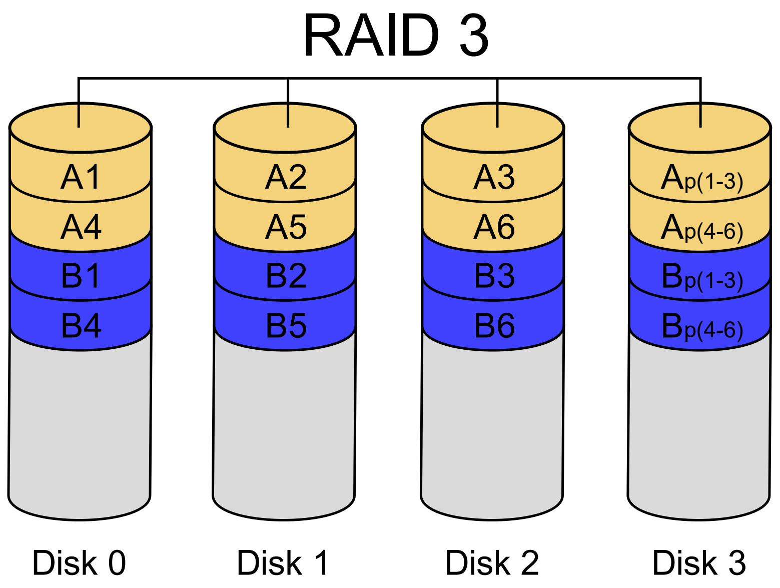


credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

RAID 3

- It is also rarely used. It stripes (i.e., distributes) data at the byte or sector level to multiple disks.
- It consists of *d* data disks + 1 parity disk
- RAID 3 is not commonly used in practice
- Pros
  - High transfer rates in long sequential reads
- Cos:
  - Cannot service multiple requests simultaneously

- Example: The data is distributed across Disks 0-2 and Disk 3 is used for parity checks

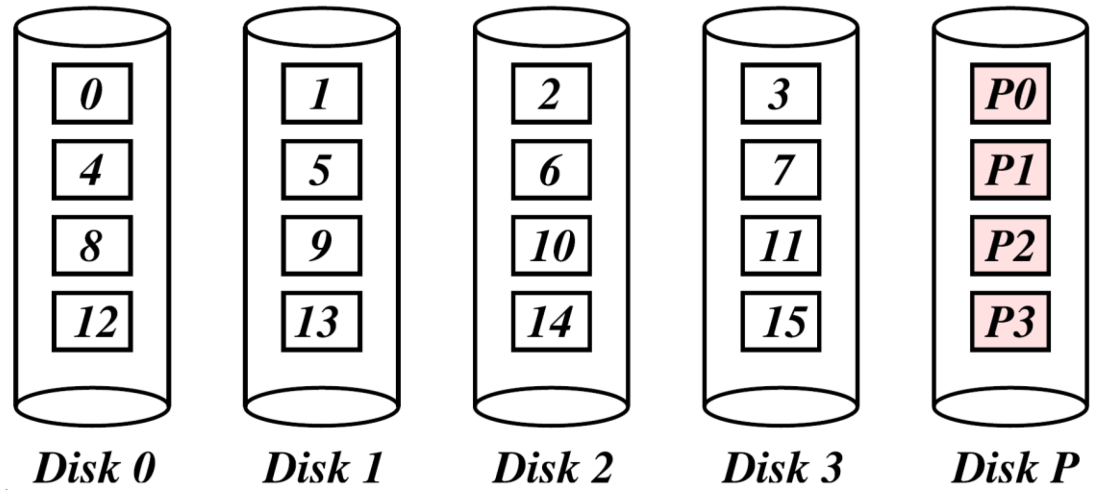


credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

RAID 4

- It is also rarely used. It stripes (i.e., distributes) data at the block level (arbitrary size block – also called stripe) to multiple disks.
- It consists of d data disks + 1 parity disk
- The advantage of RAID4 over RAID3 is that a small read/write operation may be contained in just a single data disk, resulting in faster I/O operations
- Pros
  - Good random reads
  - Read speed: parallel d-1 disks
- Cos:
  - Cannot service multiple requests simultaneously
  - Bad performance random writes
    - Write: Write a data block, calculate its parity, write its parity

- Example: a read request for block 0 would be serviced by disk 0. A simultaneous read request for block 4 would have to wait, but a read request for 13 could be serviced concurrently by disk 1.

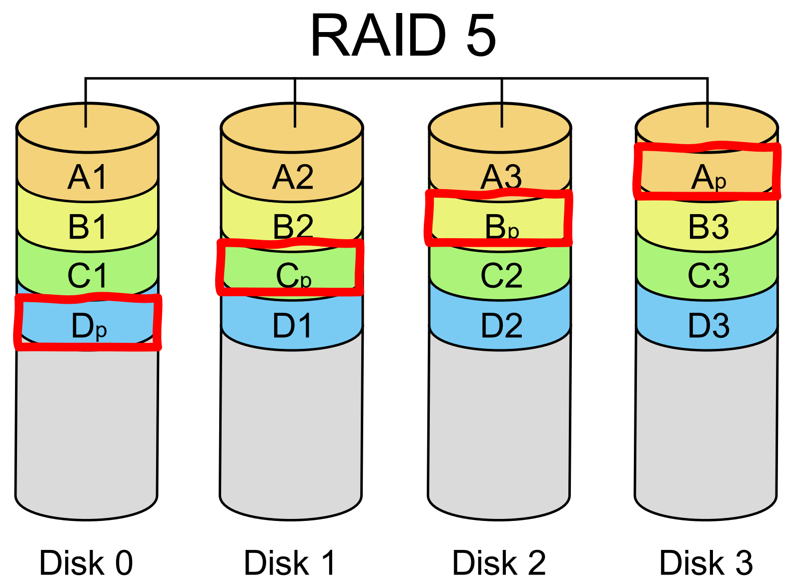


credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

RAID 5 (DISTRIBUTED BLOCK INTERLEAVED PARITY)

- It is the most used. It stripes (i.e., distributes) data at the **block** level to multiple disks.
- It consists of *d* disks containing both data + parity
- The parity bits are distributed across all disks
- Pros
  - Has the best balance of performance-reliability among the RAID schemes
  - In RAID4, the parity disk is accessed in each write operation, which slows down write operations. In RAID5, we simply interleave the parity blocks among the disks, which improves write performance
  - Read speed: parallel ***d*** disks
- Cos:
  - Increased change of data loss during disk rebuild after replacing a failed disk.
  - An extension of RAID5 is RAID6, which solves this problem. The aim with RAID6 is to be able to resist not just one but two disk failures at the same time without losing any data. Additional code blocks are provided to do so.

- The parity is distributed between multiple disks

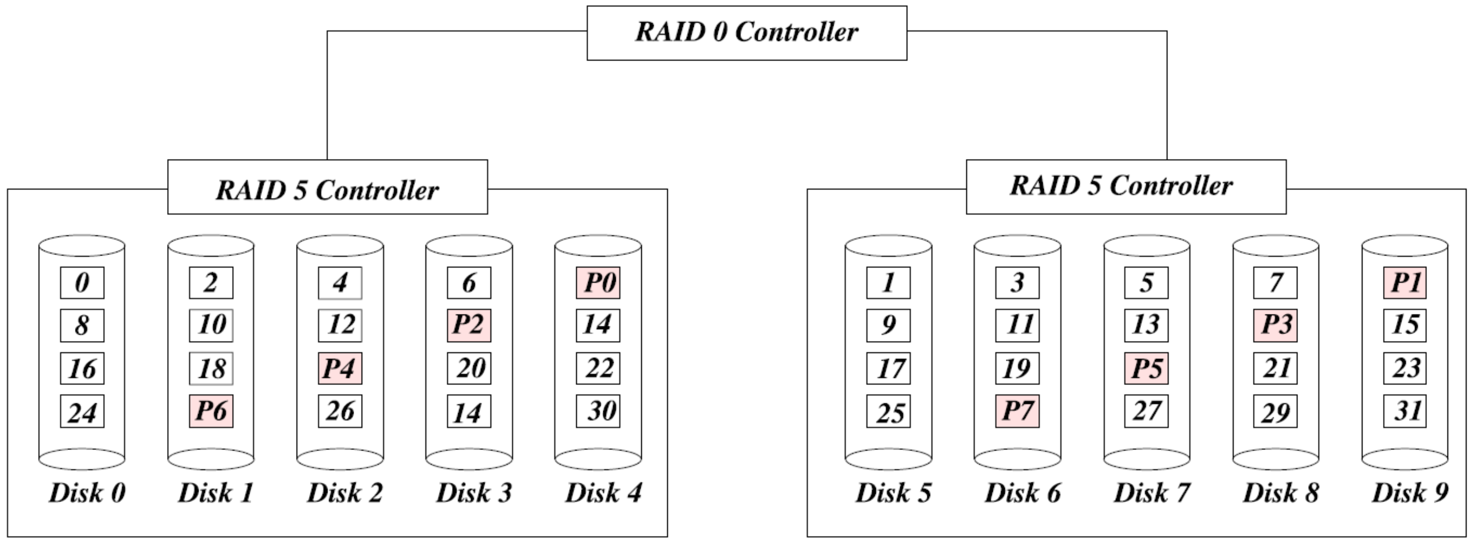


credits: https://en.wikipedia.org/wiki/Standard_RAID_levels, en:User:Cburnett

HIERARCHICAL (NESTED) RAID

Example: RAID 50

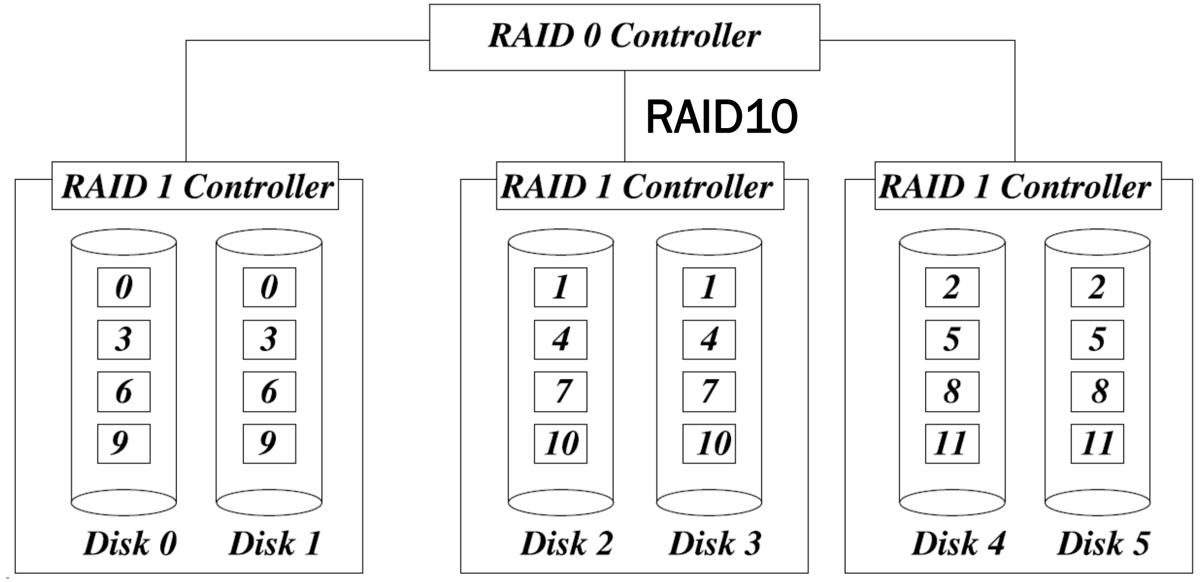
- Top Array: RAID0
- Nested Array: RAID5
- Given a file consisting of segments 0, 1, 2, 3, …. , we would assign segments 0, 2, 4, …. to be stored in one group of disks, whereas 1, 3, 5, …. would be stored in the other. Each of these two groups is organized as a RAID Level 5 structure

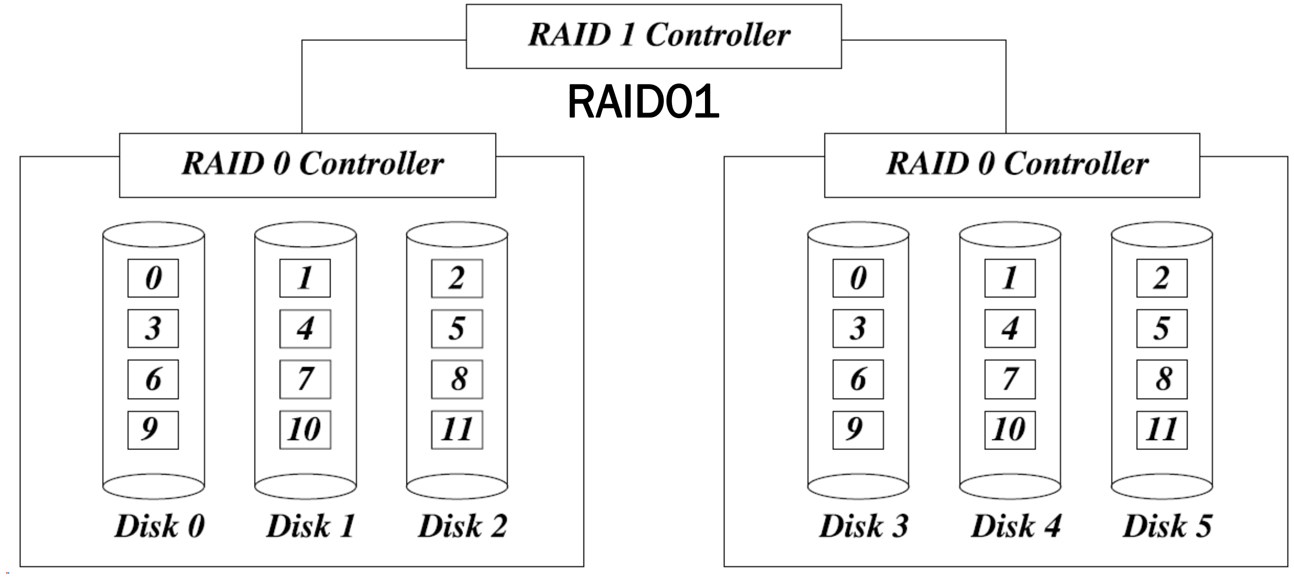


HIERARCHICAL (NESTED) RAID

Example: RAID10 vs RAID01

1) Which hierarchical RAID scheme is more reliable ?



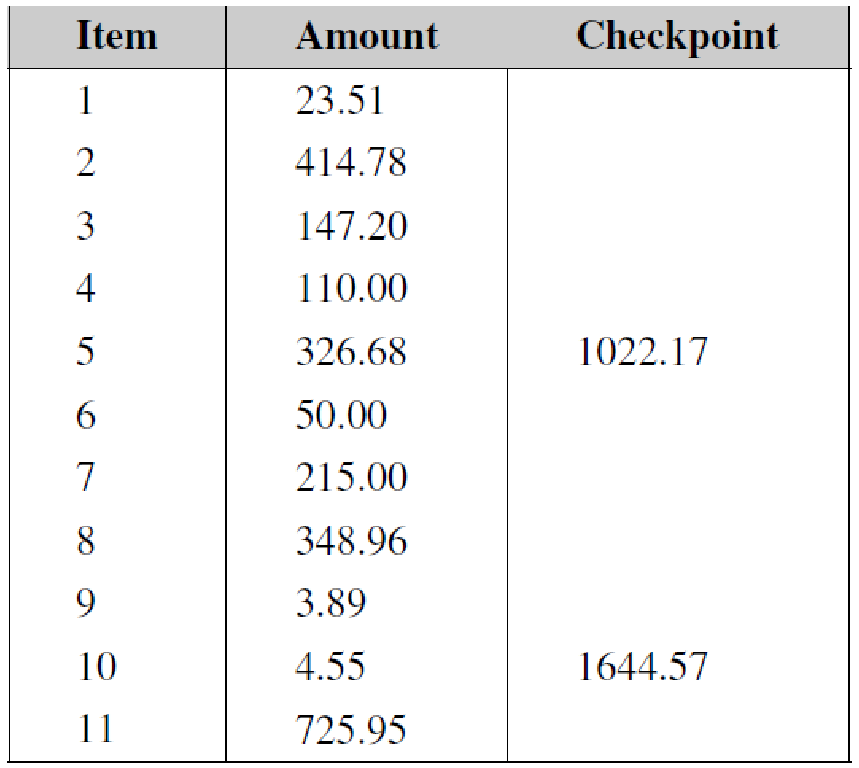


CHECKPOINTING

When a program takes very long to execute, the probability of failure during execution, as well as the cost of such a failure, become significant.

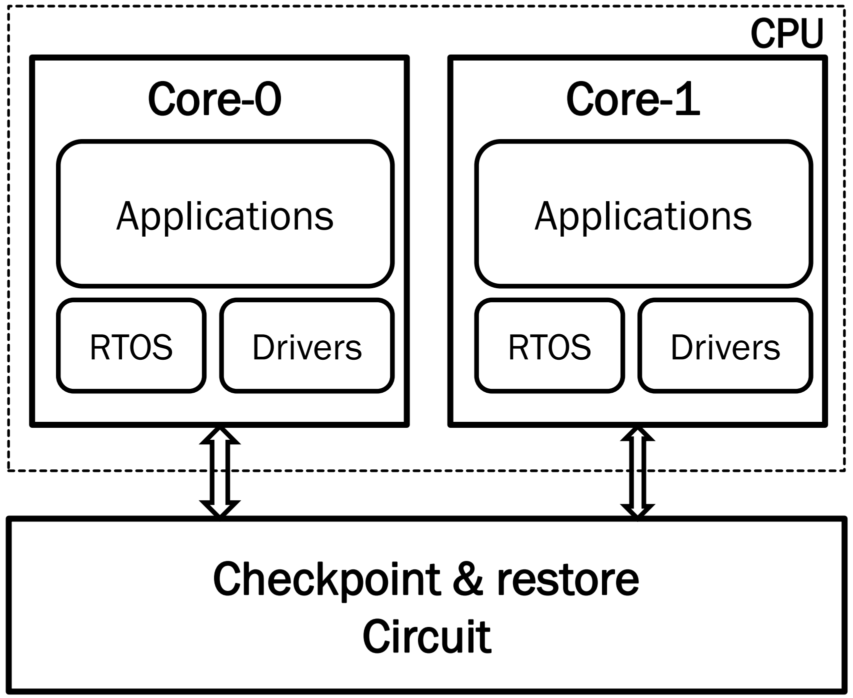
Therefore, we need to take frequent checkpoints to mitigate the cost of such errors

Simple example of checkpointing



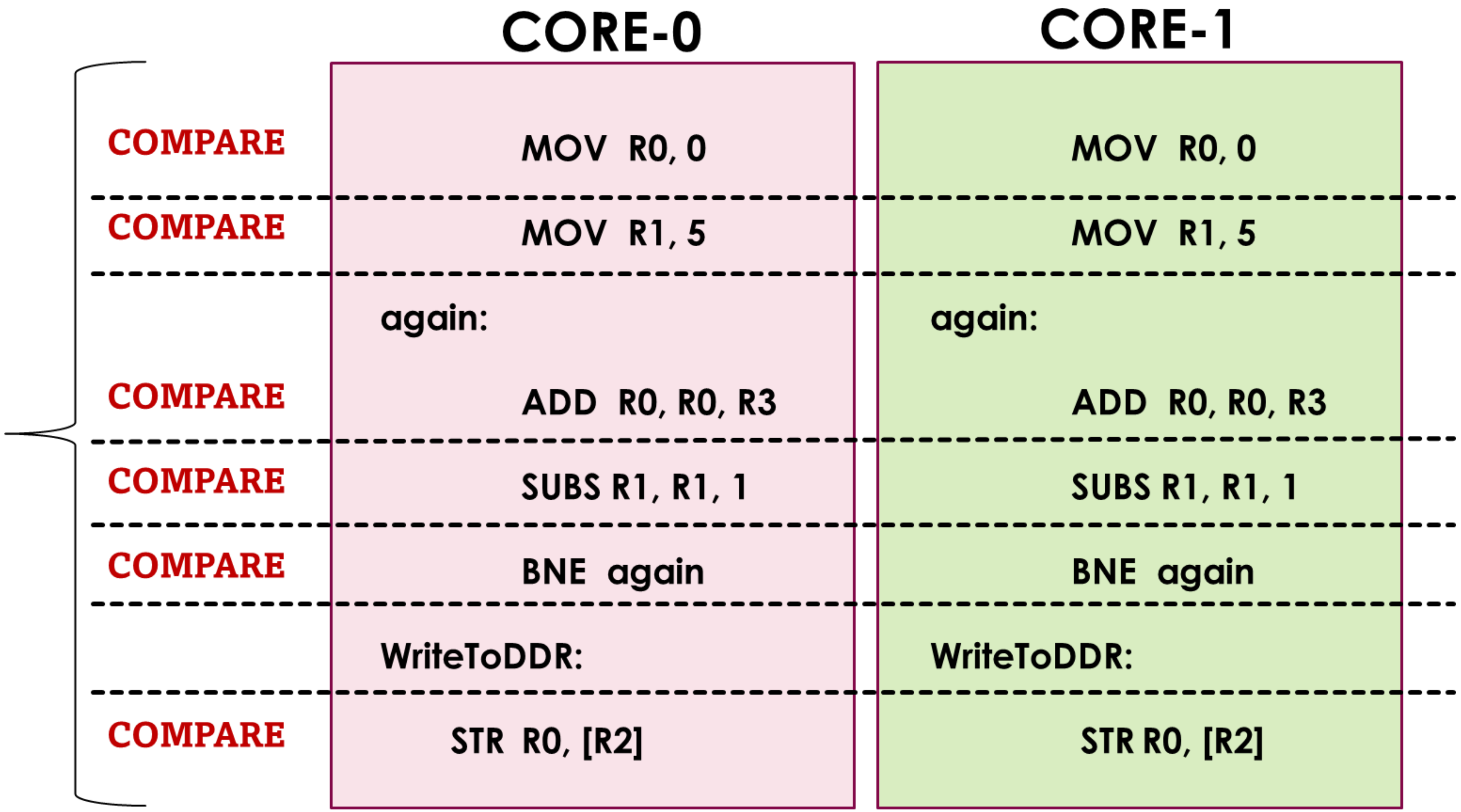
Item	Amount	Checkpoint
1	23.51	
2	414.78	
3	147.20	
4	110.00	
5	326.68	1022.17
6	50.00	
7	215.00	
8	348.96	
9	3.89	
10	4.55	1644.57
11	725.95	

EXAMPLE: LOOSELY-COUPLED DUAL CORE LOCKSTEP WITH CHECKPOINTING



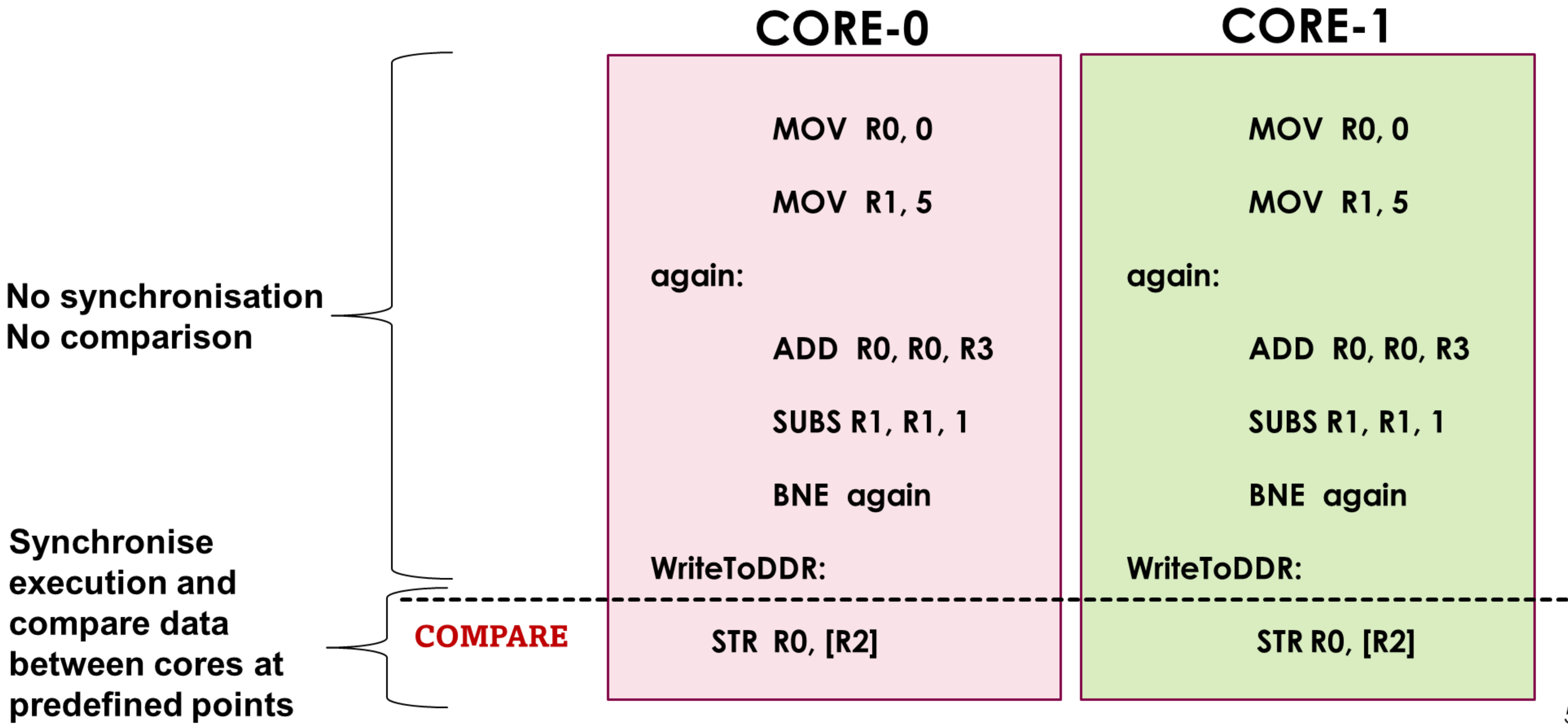
TIGHTLY-COUPLED (MICRO-SYNCHRONISED) LOCKSTEP EXECUTION

Each instruction is synchronised and compared



	CORE-0	CORE-1
COMPARE	MOV R0, 0	MOV R0, 0
COMPARE	MOV R1, 5	MOV R1, 5
	again:	again:
COMPARE	ADD R0, R0, R3	ADD R0, R0, R3
COMPARE	SUBS R1, R1, 1	SUBS R1, R1, 1
COMPARE	BNE again	BNE again
	WriteToDDR:	WriteToDDR:
COMPARE	STR R0, [R2]	STR R0, [R2]

LOOSELY-COUPLED (MACRO-SYNCHRONISED) LOCKSTEP EXECUTION


CORE-0
CORE-1
No synchronisation
No comparison
Synchronise execution and compare data between cores at predefined points
COMPARE

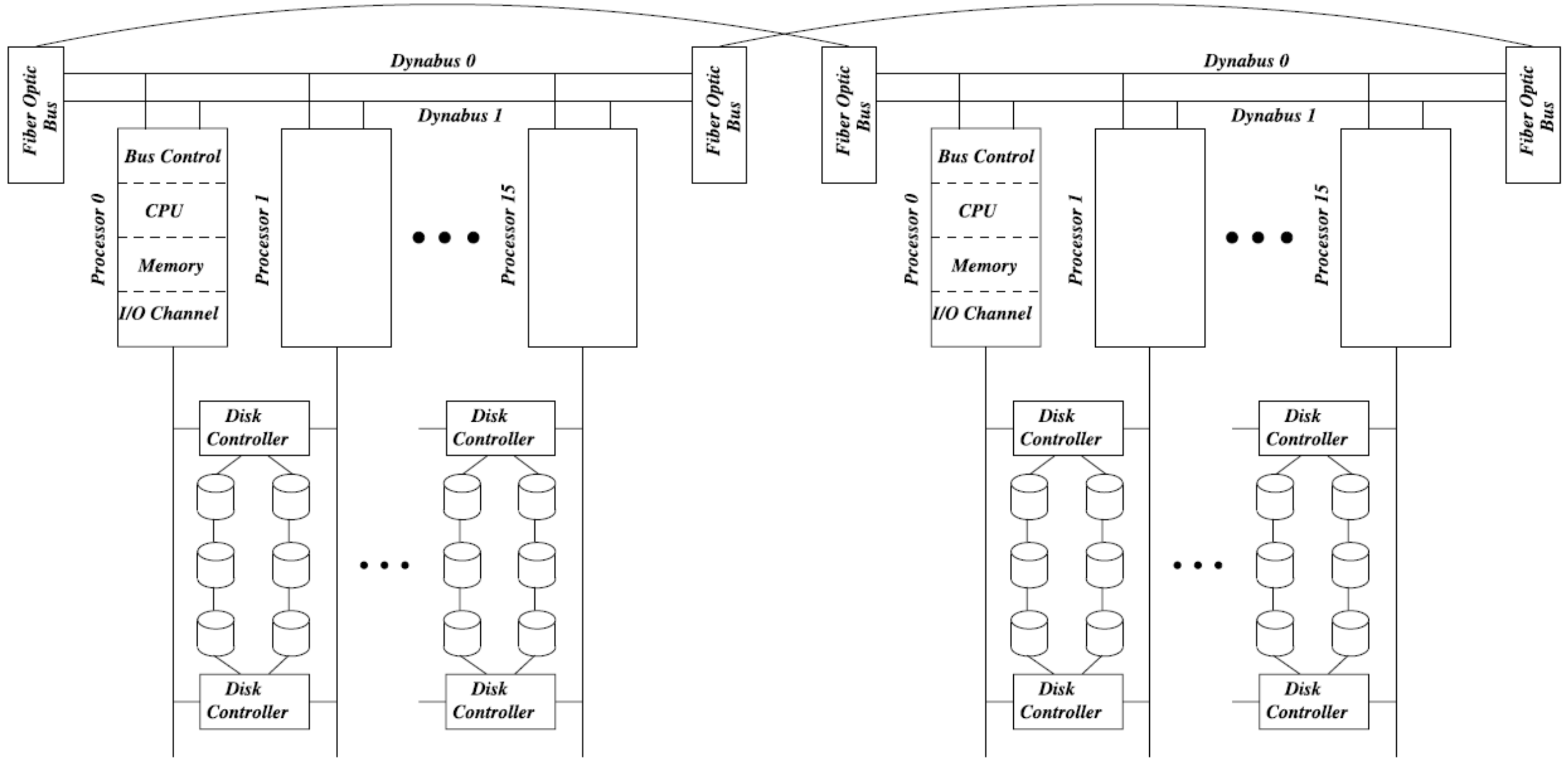
CORE-0

```
        MOV  R0, 0
        MOV  R1, 5
again:
        ADD  R0, R0, R3
        SUBS R1, R1, 1
        BNE  again
WriteToDDR:
        STR  R0, [R2]
```

CORE-1

```
        MOV  R0, 0
        MOV  R1, 5
again:
        ADD  R0, R0, R3
        SUBS R1, R1, 1
        BNE  again
WriteToDDR:
        STR R0, [R2]
```

EXAMPLE: NONSTOP ARCHITECTURE



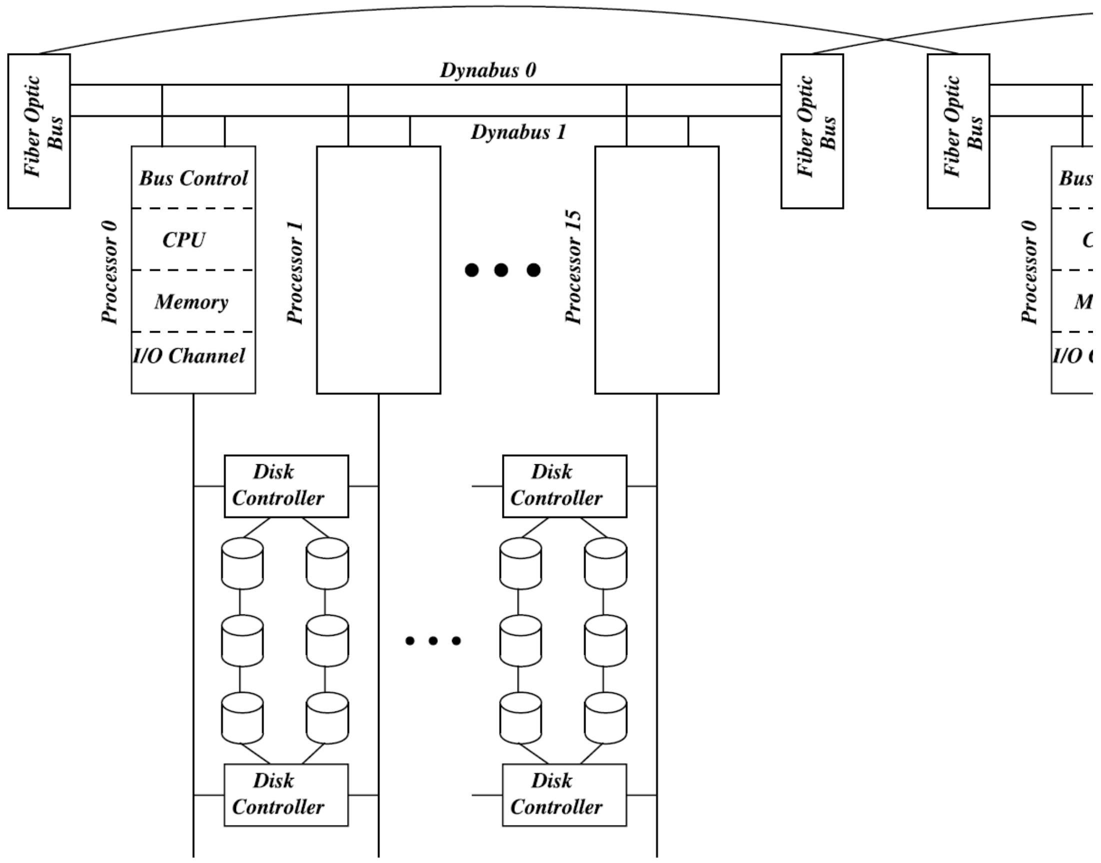
- Developed by Tandem Computers (acquired by HP)
- Used for online transaction processing

NONSTOP ARCHITECTURE – KEY DESIGN PRINCIPLES

- **Modularity:** The hardware and software are constructed of modules of fine granularity. These modules constitute units of failure, diagnosis, service, and repair. **Modules should be isolated so that a fault in one module does not affect another one**
- **Fail-fast operation:** A fail-fast module either works properly or stops. Thus each module is self-checking and stops upon detecting a failure. Hardware checks (through error-detecting codes) and software consistency tests support fail-fast operation.
- **Single-failure tolerance:** When a single module (hardware or software) fails, another module immediately takes over. For processors, this means that a second processor is available. For storage modules, it means that the module and the path to it are duplicated.
- **Online maintenance:** Hardware and software modules can be diagnosed, disconnected for repair, and then reconnected, without disrupting the entire system's operation.

NONSTOP ARCHITECTURE

- Consists of clusters of computers
- Each cluster includes up to 16 custom-designed processors
- Each custom processor has a CPU, local memory (containing its own copy of OS), bus control unit, and I/O channels
- The CPU incorporates many error detection capabilities for fail-fast mode of operation
- The CPU datapath is protected with parity checking
- The CPU control-path is protected with parity checking, BIST and detection of illegal CPU states



NONSTOP ARCHITECTURE

- The working memory protected with a Hamming code with SECDED capability
- The address bus of the working memory is protected with single error detection parity code
- CPU caches are protected from transient faults invalidating cache entries that report parity errors so that data is refeched from the main/working memory
- Spare modules of various processor modules exist to recover any permanent damaged modules
- Parity checking is also used in all memory units (e.g., register file) and processing elements (ALU, counters etc.) of the CPU, and into buses of the processor.
- There is not shared memory between the processors since it is a single point of failure. ICP communications occure through messages
- Power supplies and cabling are also fully redundant to eliminate any single point of failure
- Backup batteries are used to save the state of the system in case of power failure
- All controllers operate in lockstep and support BIST features.

NONSTOP ARCHITECTURE

- The disks work in RAID1
- The disk data is protected end-to-end with checksums
- For each data block, the processor calculates a checksum and appends it to the data written to the disk. This checksum is verified by the processor when the data block is read from the disk.
- The checksum is used for error detection, whereas the disk mirroring is used for data recovery.

NONSTOP ARCHITECTURE - MAINTENANCE AND REPAIR AIDS

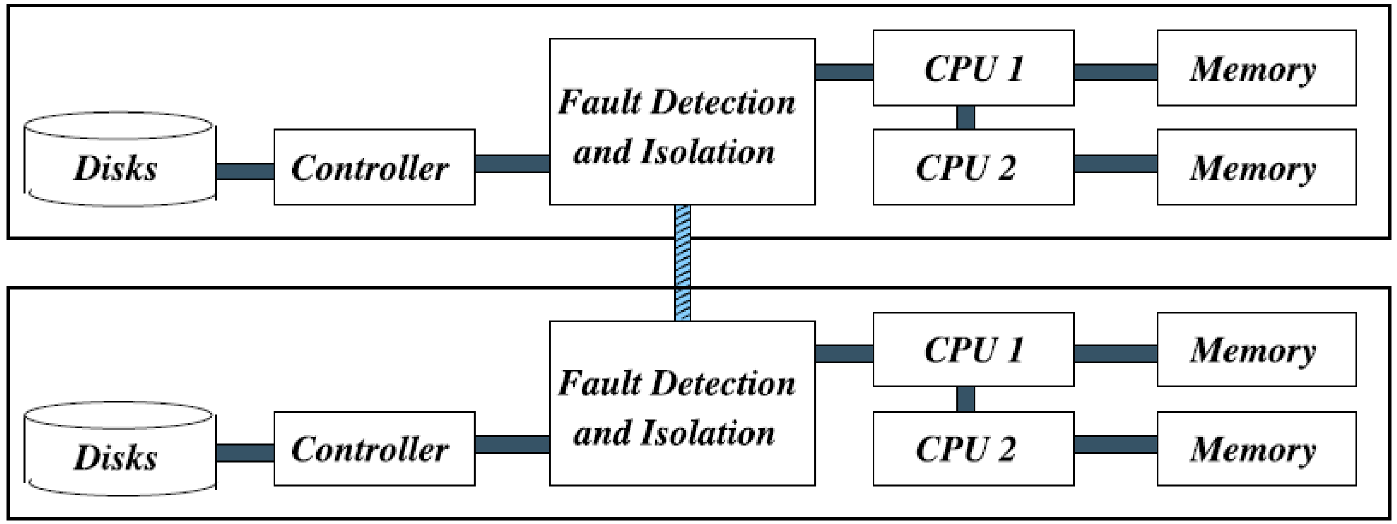
- Detected errors are automatically analysed and reported to remote support centers which track related repair actions.
- Includes a maintenance processor and diagnostic units:
  - Communicates with all the processors and a remote service center.
  - Collects failure-related information
  - Inform engineers at the remote center to run diagnostic tests.
  - Reconfigured the system in response to detected faults.
- Each processor module has a diagnostic unit
  - Monitors the status of the computing processor, working memory, Dynabus interface, and the I/O channel.
  - Reports to the central maintenance processor any detected errors
  - Upon a request received from the remote service center (through the central maintenance processor) it can perform diagnostic tests at the module level.
- The central maintenance processor condact automatic fault diagnosis through the use of a knowledge database that includes a large number of known error values. It also controls and monitors a large number of sensors for power supply voltages, intake and outlet air temperatures, and fan rotation.

NONSTOP ARCHITECTURE - SOFWARE

- Most of system fault tolerance is done by the OS. The OS detects failures of processors or I/O channels and performs the necessary recovery.
- The OS manages the process pairs that constitute the primary fault-tolerance scheme used in NonStop.
- A process pair includes a primary process and a passive backup process that is ready to become active when the primary process fails.
- When a new process starts, the OS generates a clone of this process on another processor. This backup process goes immediately into passive mode and waits for messages from either its corresponding primary, or the OS.
- At certain points during the execution of the primary process, checkpoints are taken and a checkpointing message containing the process state is sent by the primary to the backup. The process state of the backup is updated by the OS, while the backup process itself remains passive. If the primary process fails, the OS orders the backup to start execution from the last checkpoint.

EXAMPLE: XEON

- The Xeon processor was designed to support continuous self-monitoring and self-healing.
- The processor actively monitors for errors, all the interconnects, data buffers, and data paths.
- Memory is protected SECDED Hamming code. Memory address is protected with parity codes.
- If a memory chip fails completely (or has exceeded a threshold of bit errors that have been corrected), it is replaced by a spare memory chip.
- Memory is scrubbed



Lockstepped CPUs; Multipath I/O; Mirrored Disks

EXAMPLE: XEON

- At the CPU level, the Xeon processor uses error-correcting codes to protect the registers from transient faults. The execution units include error-detection circuits using residue and parity codes. If an error is detected, the instruction is retried. If the retry fails, a fatal error signal is generated.
- At the highest level, the processor interacts with the operating system (OS), virtual machine manager, and application software to support recovery from errors that the hardware was unable to correct.

BIBLIOGRAPHY

- Koren, Israel, and C. Mani Krishna. *Fault-tolerant systems*. Morgan Kaufmann, 2020.