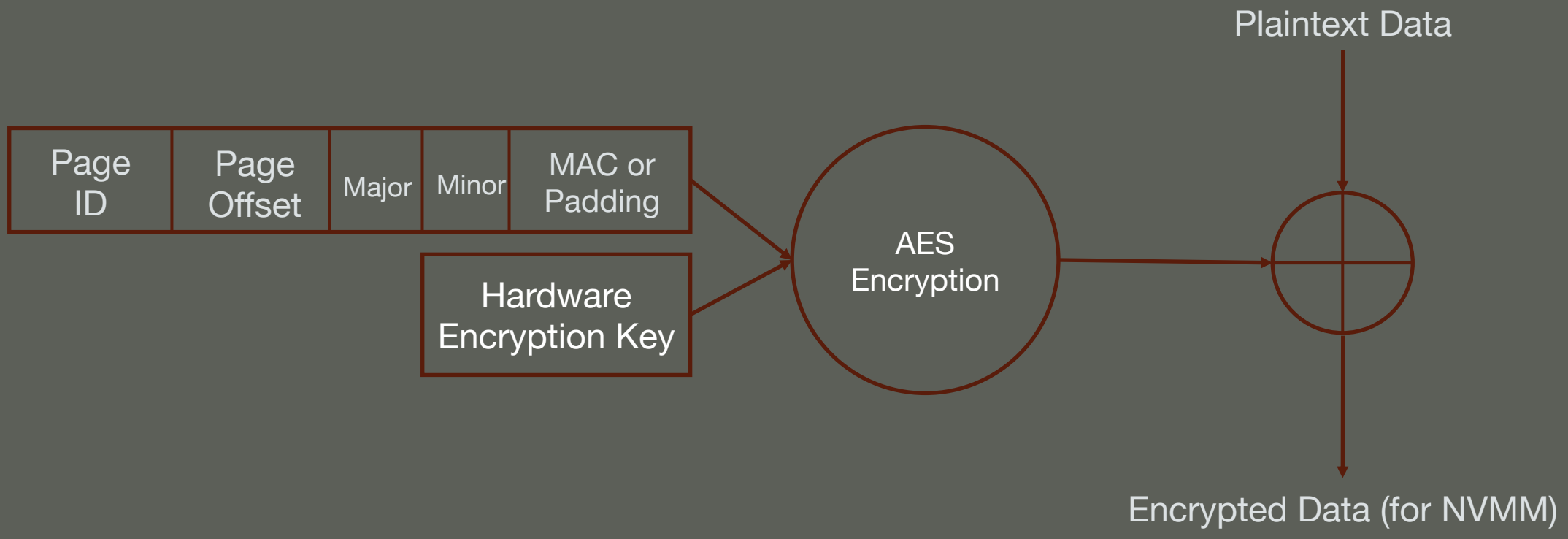
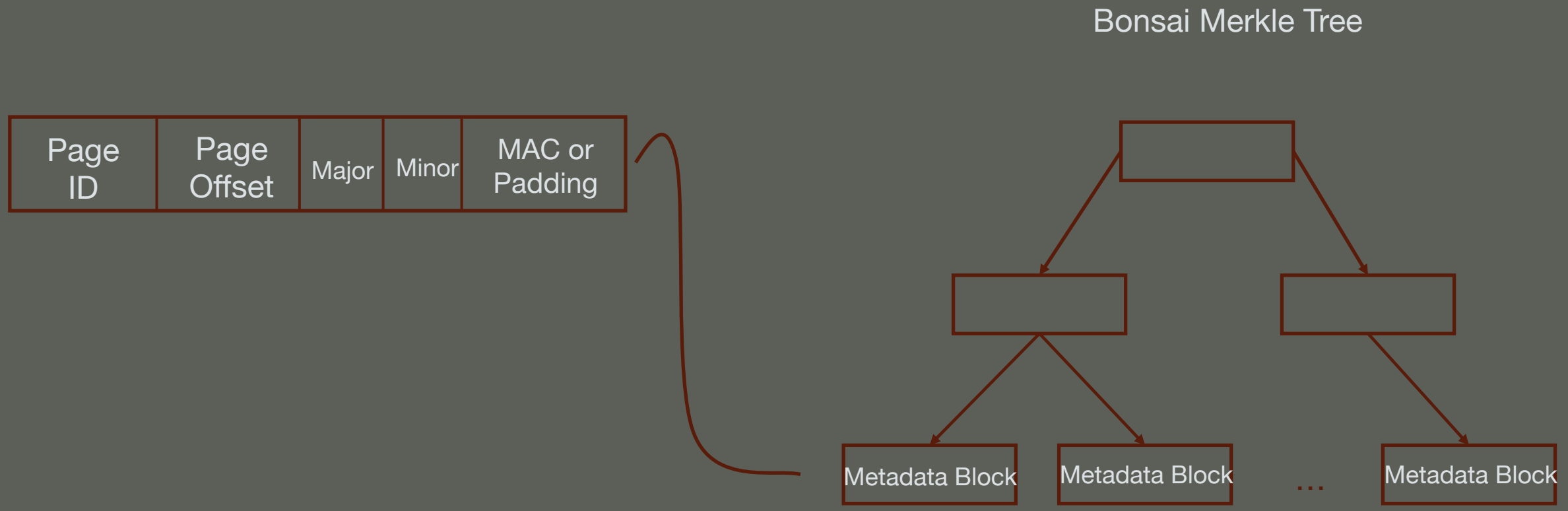
Partial Recovery of Secure Non-Volatile Main Memories

Samuel Thomas, Tamara Lehman, R. Iris Bahar, Joseph Izraelevitz

Background


Page ID
Page Offset
Major
Minor
MAC or Padding
Hardware Encryption Key
AES Encryption
Plaintext Data
Encrypted Data (for NVMM)

Background


Bonsai Merkle Tree
Page ID
Page Offset
Major
Minor
MAC or Padding
Metadata Block
Metadata Block
...
Metadata Block

Background

Threat Model

- Everything off-chip is untrusted
- The memory bus may be snooped
- Memory packets may be replayed
- Memory contents may be scanned
- Memory contents may be tampered with

Integrity-Checking Algorithm

1. If the state of memory is ahead of the Merkle Tree state, use prior work to match the states
2. Reconstruct the Merkle Tree
3. Compare the “computed root” with the root stored on-chip
    1. If the roots are the same, then restore the system
    2. Otherwise, the system is unrecoverable. Panic.

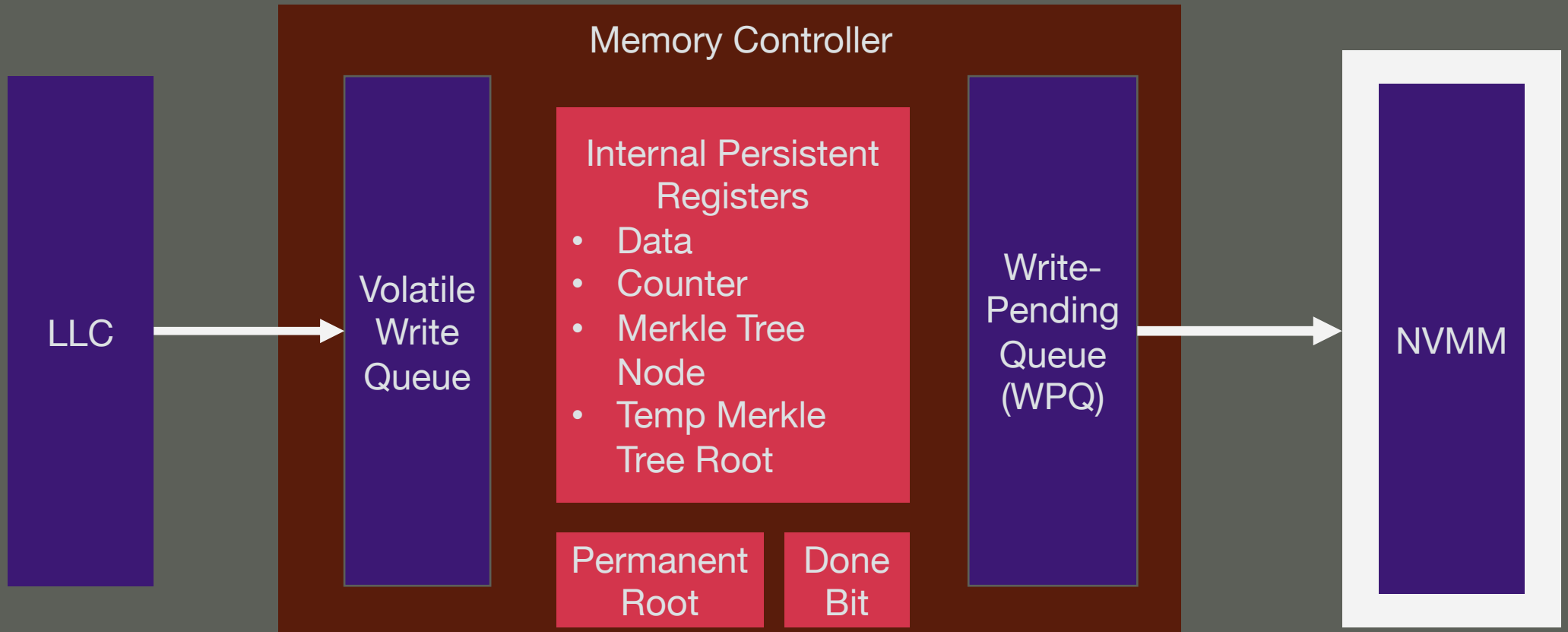
The Problem

- Non-volatile memories are an increasingly likely candidate to replace DRAM as main memory of commodity systems
- By retaining memory state after power-off, NVMMs are subject to a wider array of attacks than traditional DRAMs
- Counter-mode encryption can secure main memory and allow for integrity checking mechanisms, but integrity checks are slow – antithetical to fast power-on

Our Goals

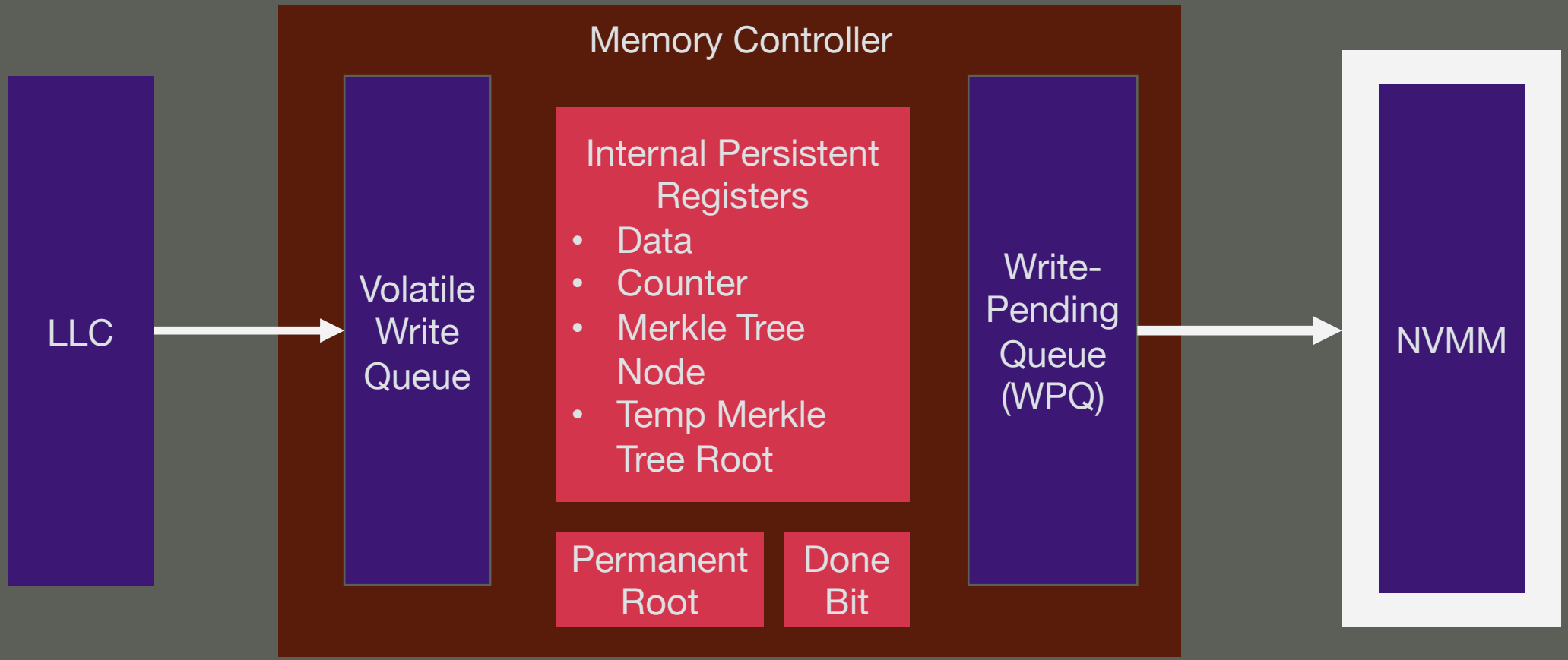
- Partial and lazy recovery of counter-mode encryption-based integrity checking procedures
- Ability to backup non-corrupted memory
- Fast recovery of important code
- Fail-fast of corrupted code

Assumed Architecture

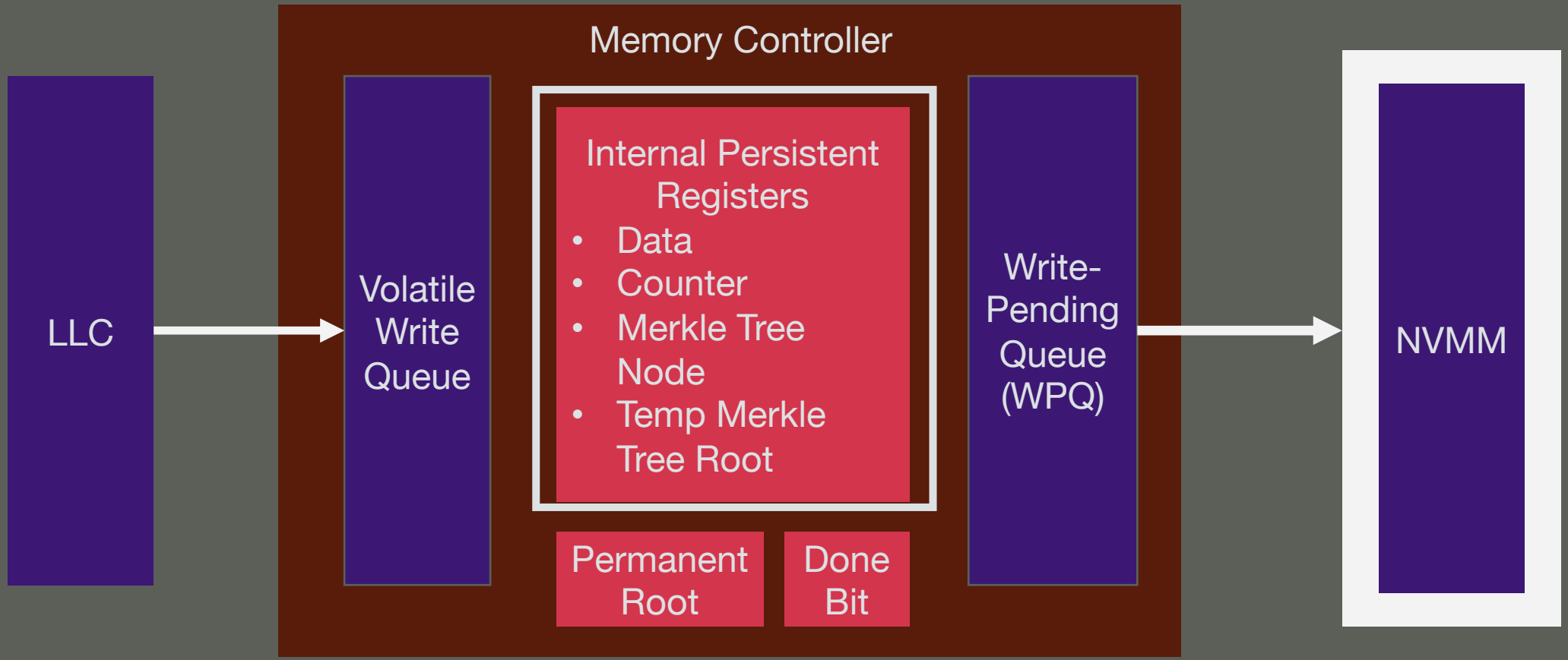


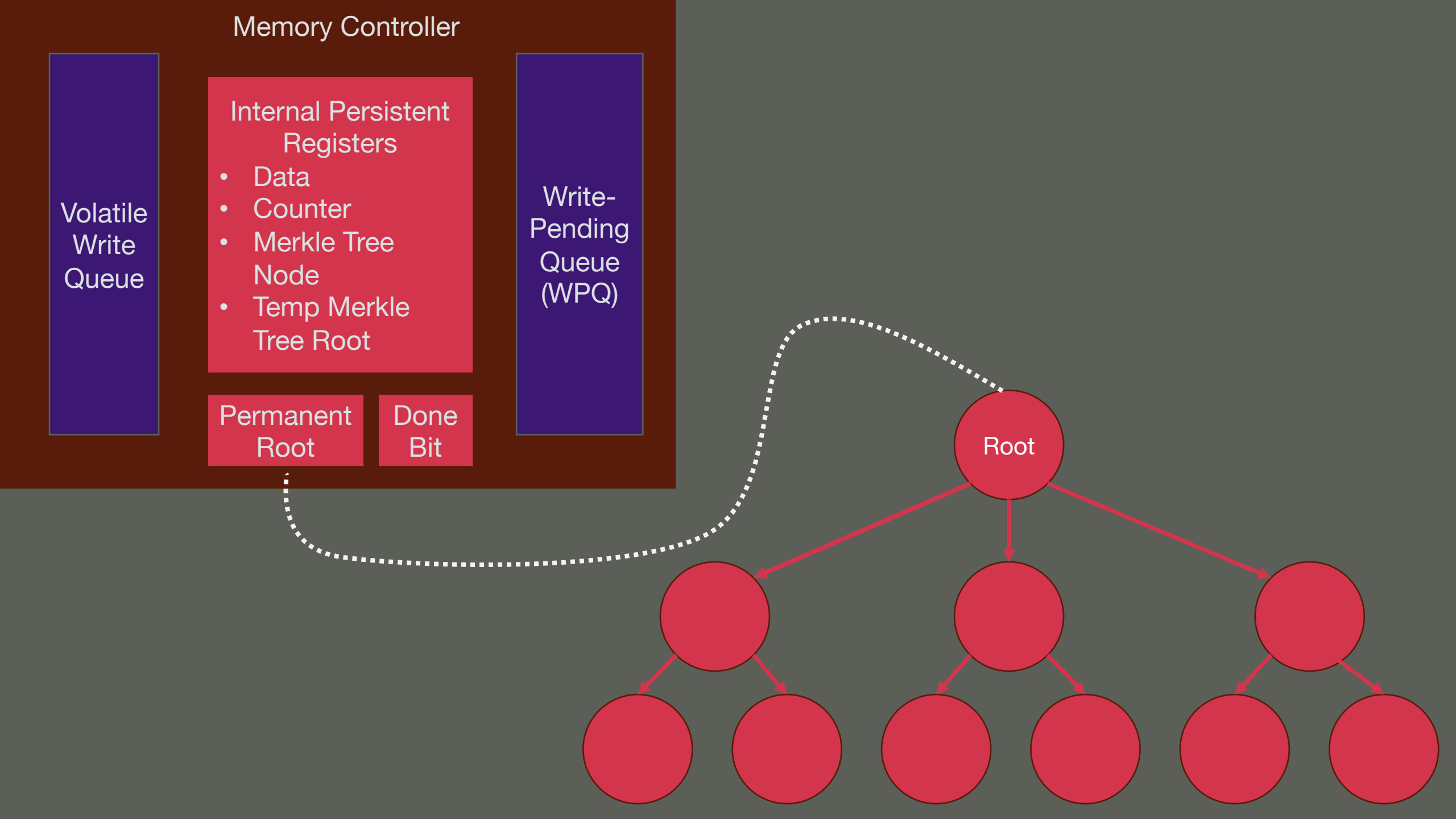
K. Abu Zubair and A. Awad, “Anubis: Low-overhead and practical recov- ery time for secure non-volatile memories,” in *International Symposium on Computer Architecture (ISCA)*, 2019.

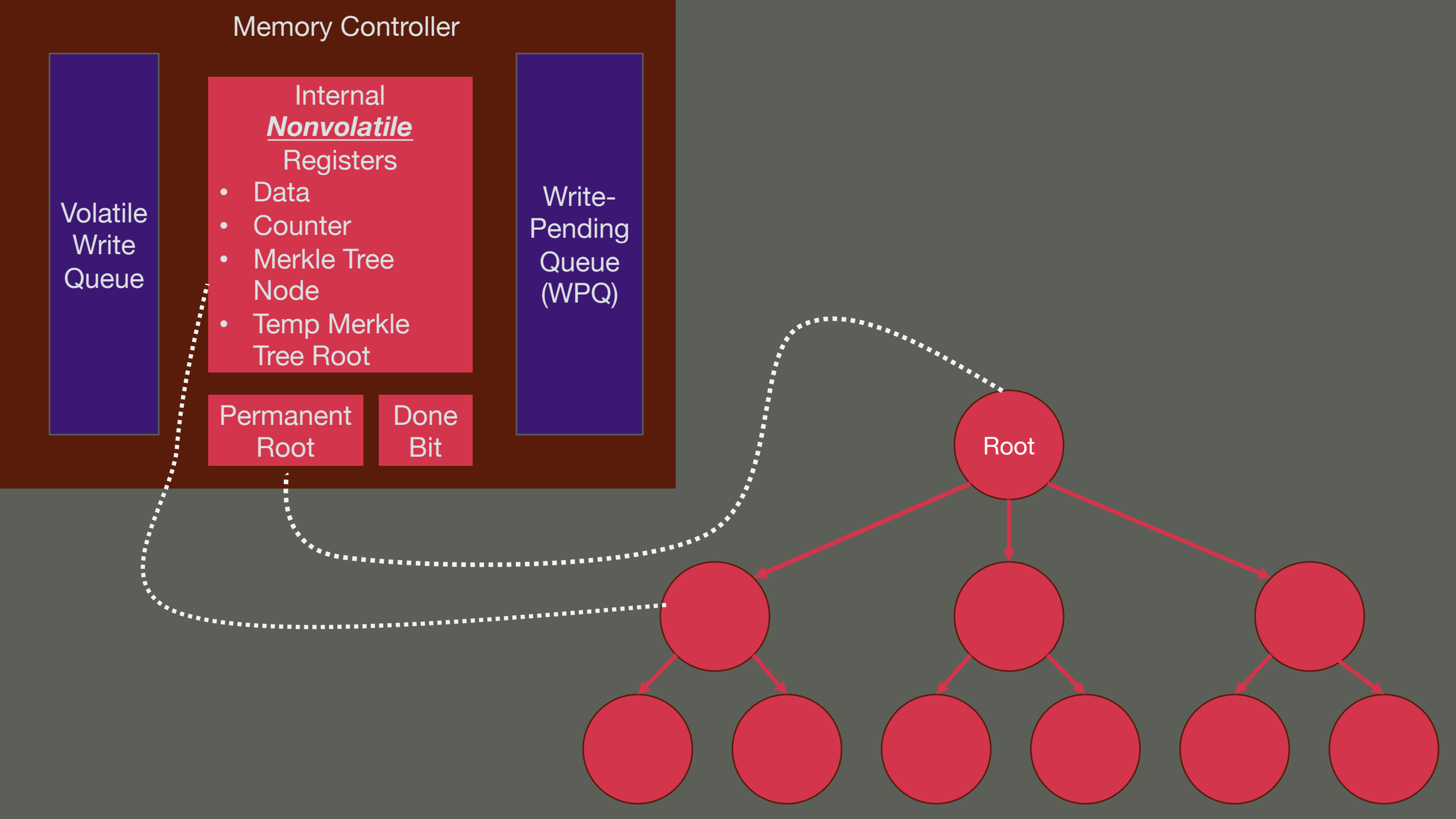
Our Focus

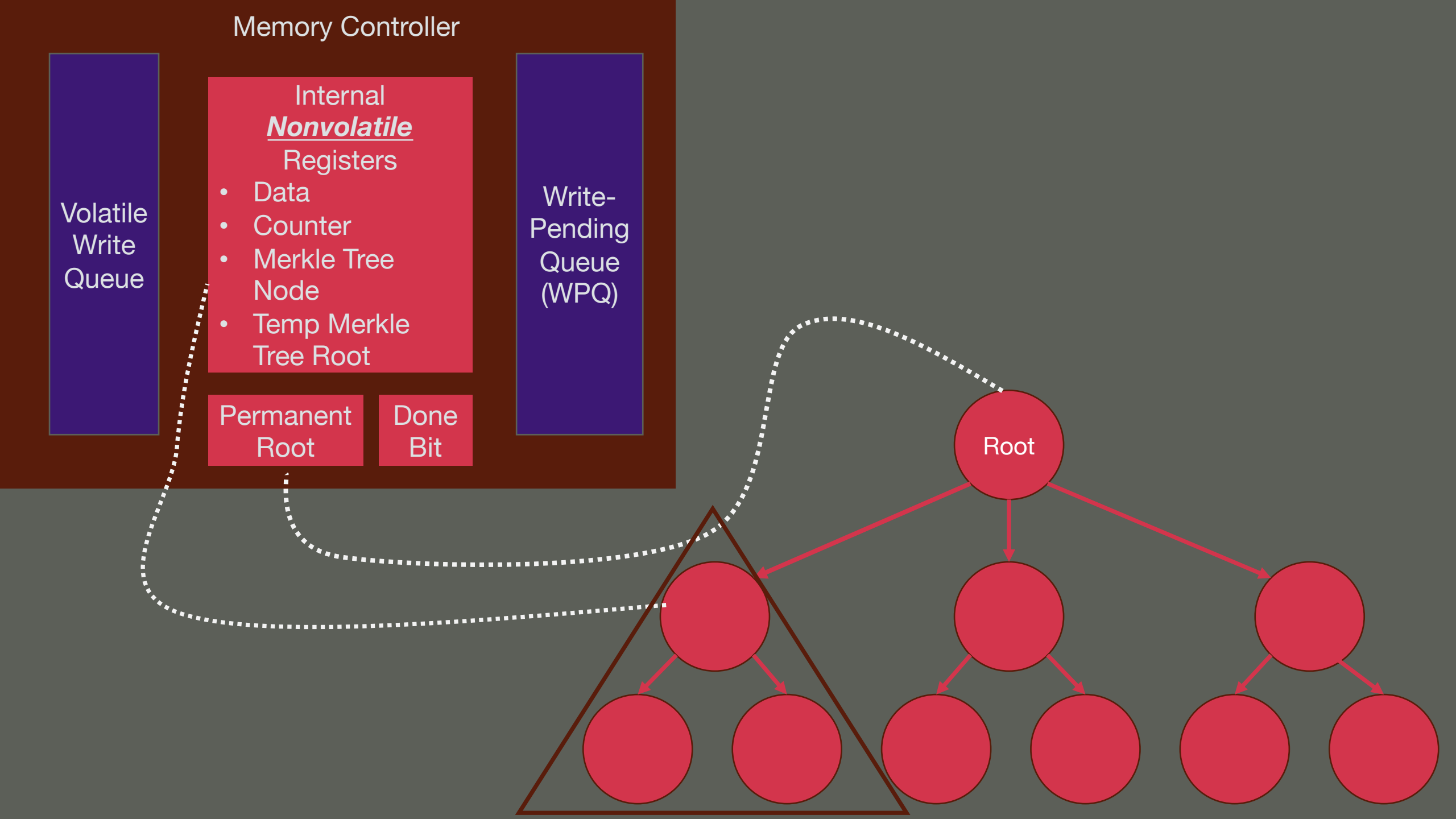

Memory Controller
LLC
Volatile Write Queue
Internal Persistent Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Permanent Root
Done Bit
Write-Pending Queue (WPQ)
NVMM

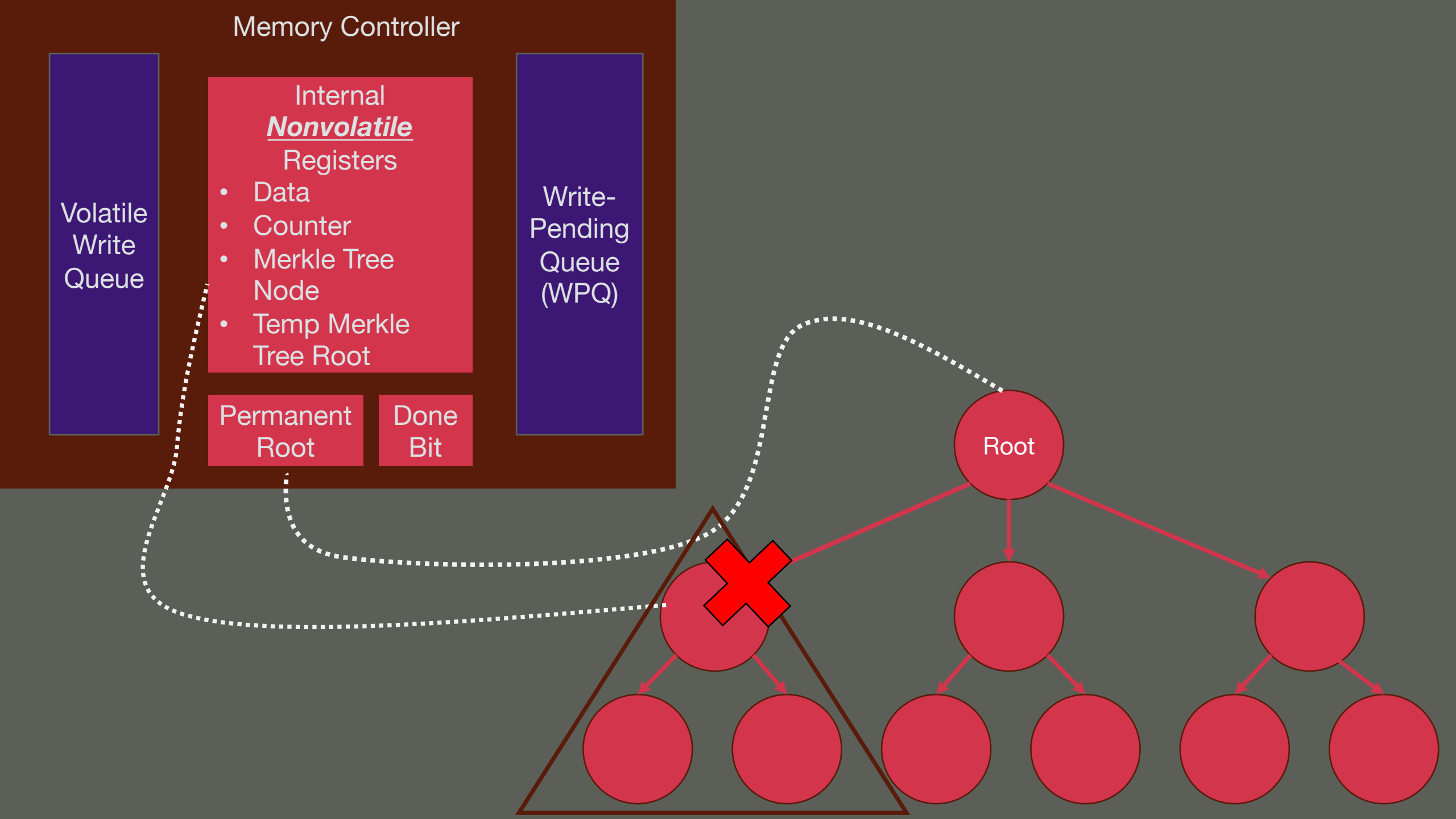
Our Focus

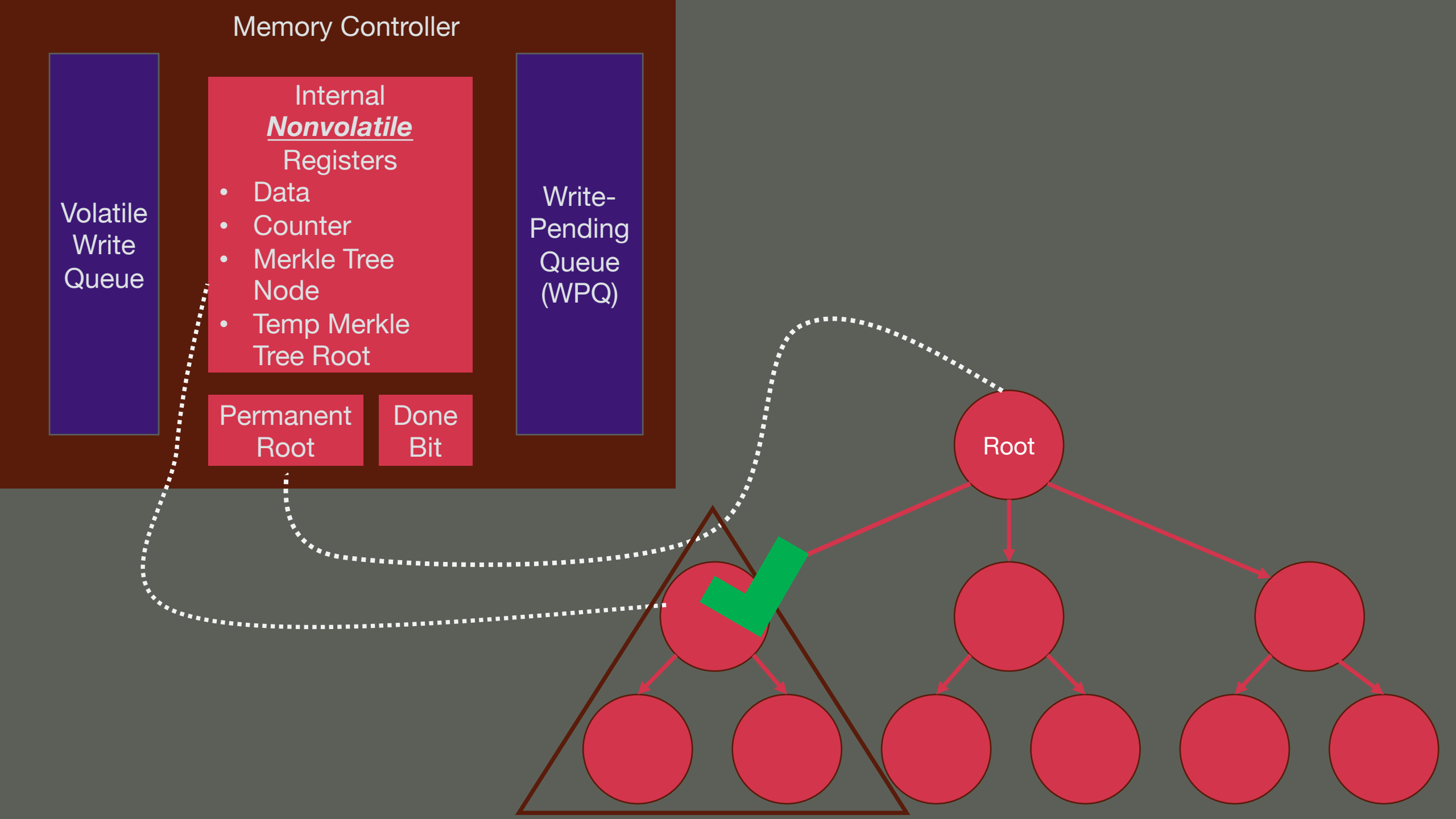

Memory Controller
LLC
Volatile Write Queue
Internal Persistent Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Permanent Root
Done Bit
Write-Pending Queue (WPQ)
NVMM

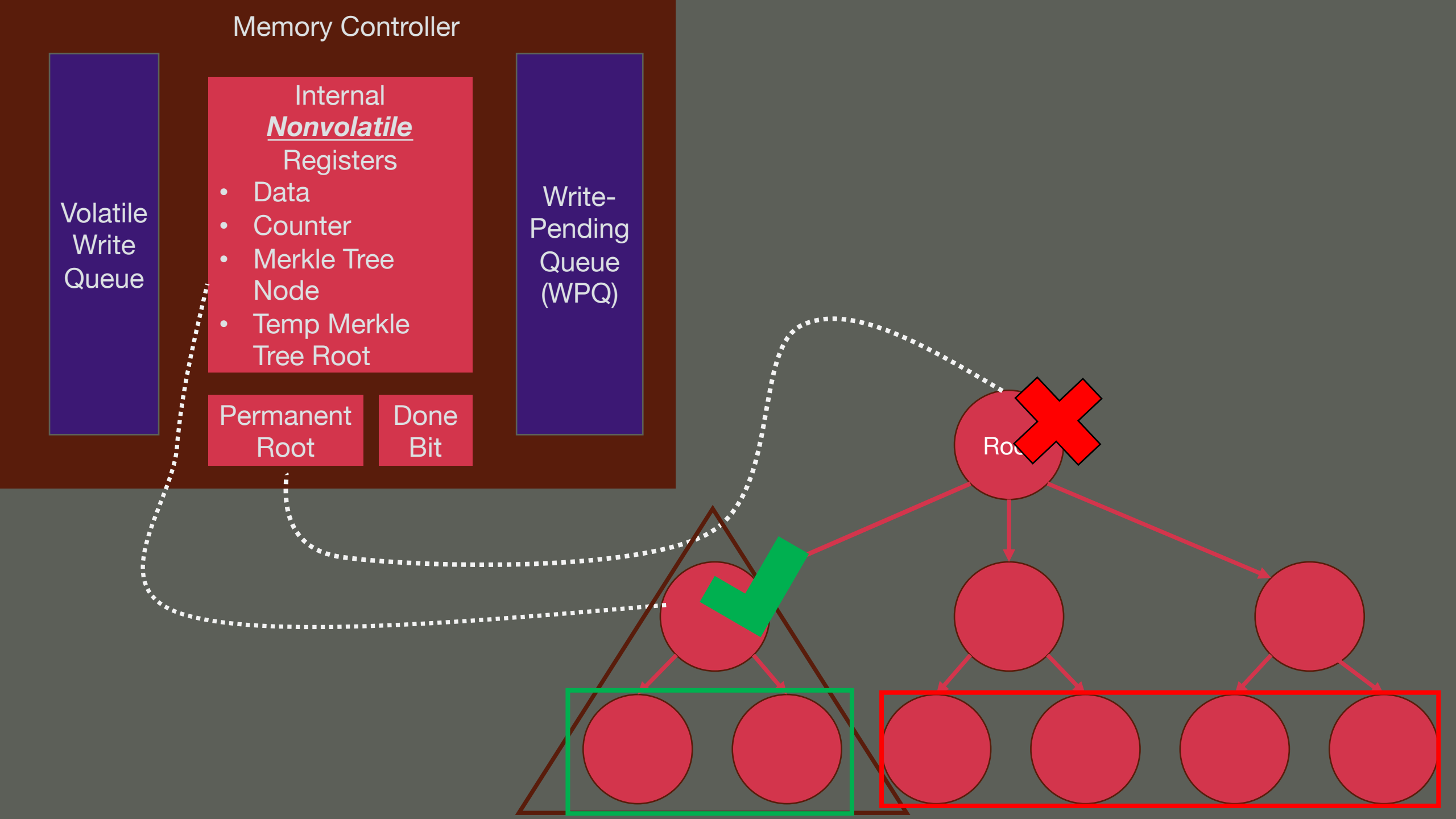

Memory Controller
Volatile Write Queue
Internal Persistent Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Root


Memory Controller
Volatile Write Queue
Internal Nonvolatile Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Root


Memory Controller
Volatile Write Queue
Internal Nonvolatile Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Root


Memory Controller
Volatile Write Queue
Internal Nonvolatile Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Root


Memory Controller
Volatile Write Queue
Internal Nonvolatile Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Root

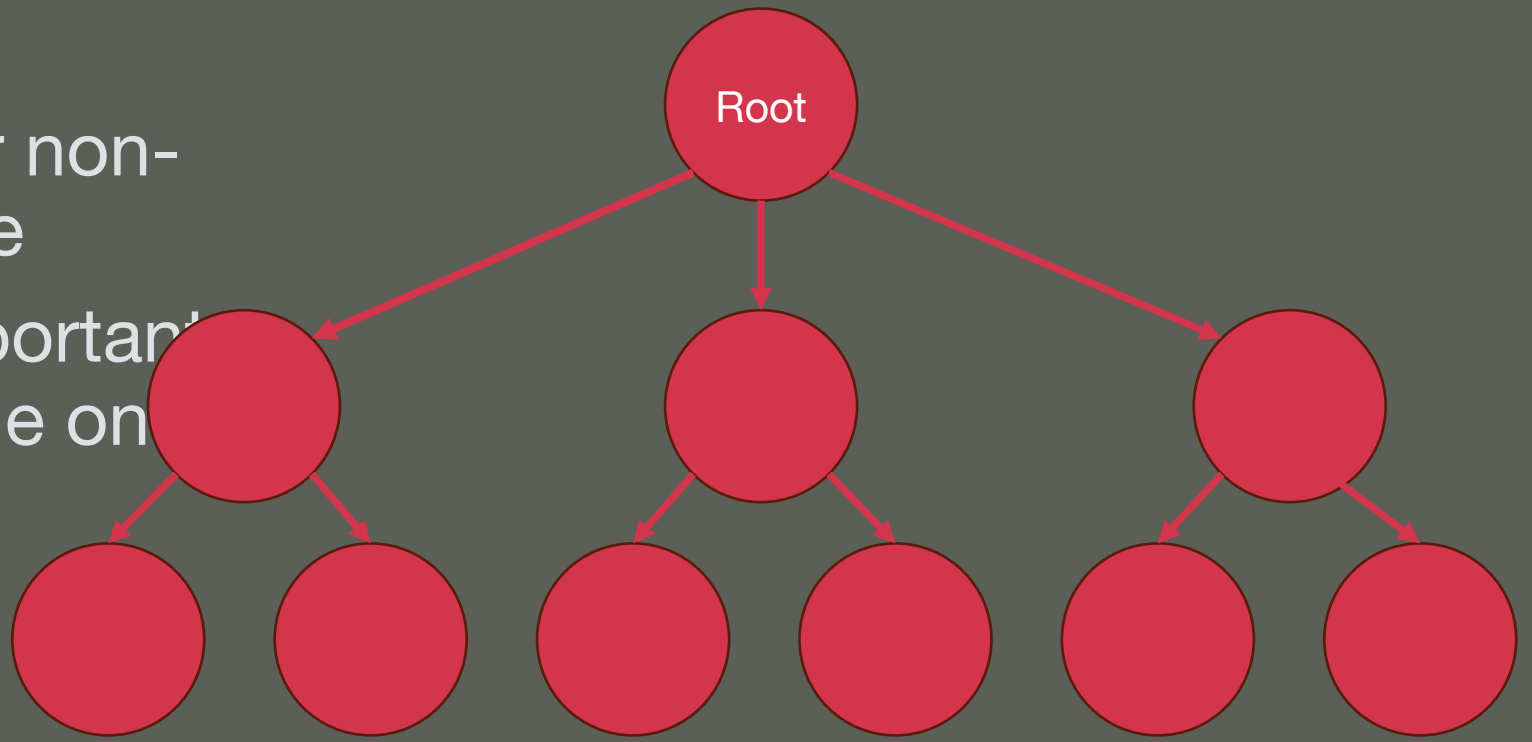

Memory Controller
Volatile Write Queue
Internal
Nonvolatile
Registers
• Data
• Counter
• Merkle Tree Node
• Temp Merkle Tree Root
Write-Pending Queue (WPQ)
Permanent Root
Done Bit
Ro

Methodology

- We are going to extend the memory controller device class in gem5
- We want to ensure that the runtime doesn’t suffer too much overhead
- We will measure recovery time after crash with a variety of implementations

Questions Raised

- Tradeoff between recoverable memory versus time to recover
- On-chip transaction for non-volatile tree node cache
- How to ensure that important software is always in the on-chip “node domain”



Thank you for your time!