

# Active Block Layer Extensions

## Pseudo Block Device Driver & Solid State Device Simulation

Robert Parks

California State University Fresno  
College of Science and Mathematics

Florida International University

Summer 2008 REU

Professor Raju Rangaswami

PhD Student Jorge Guerra

# ABLE Introduction

- For many years, researchers have independently developed intelligence in the storage stack
- Many of these systems have had similarities
- Interactions between these systems are currently handled by administrators

# ABLE Overview

- Self-managing storage systems
  - Adapts to high-level system goal specifications defined by an administrator
- Block extension development framework
  - Provide a development environment to simplify block extension implementation

# ABLE Examples

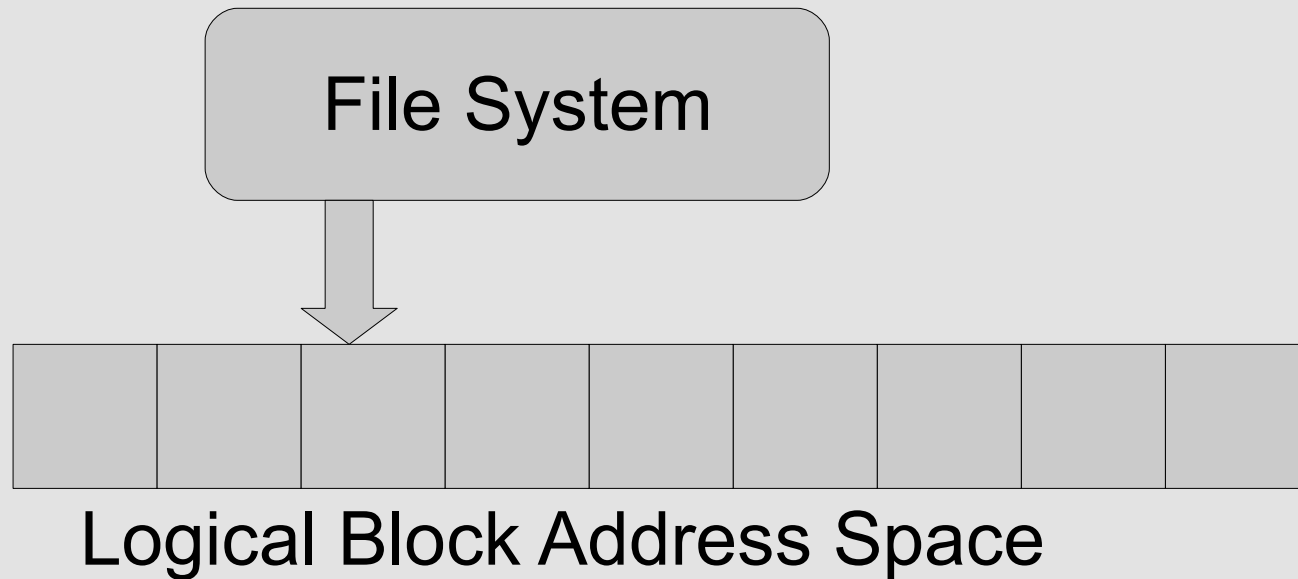
- BORG – Block Reorganization and Self-Optimization
  - Identifies popular block accesses patterns and copies them to a dedicated partition
- EXCES – External Caching in Energy Saving Storage Systems
  - Utilizes low power storage devices as cache for “popular” disk blocks to minimize power consumption

# Block Device Driver Overview

- Block device drivers exist in the block layer
- Provide a Logical Block Address Space for the File System
- Process requests from the File System and transfer them to the I/O scheduler

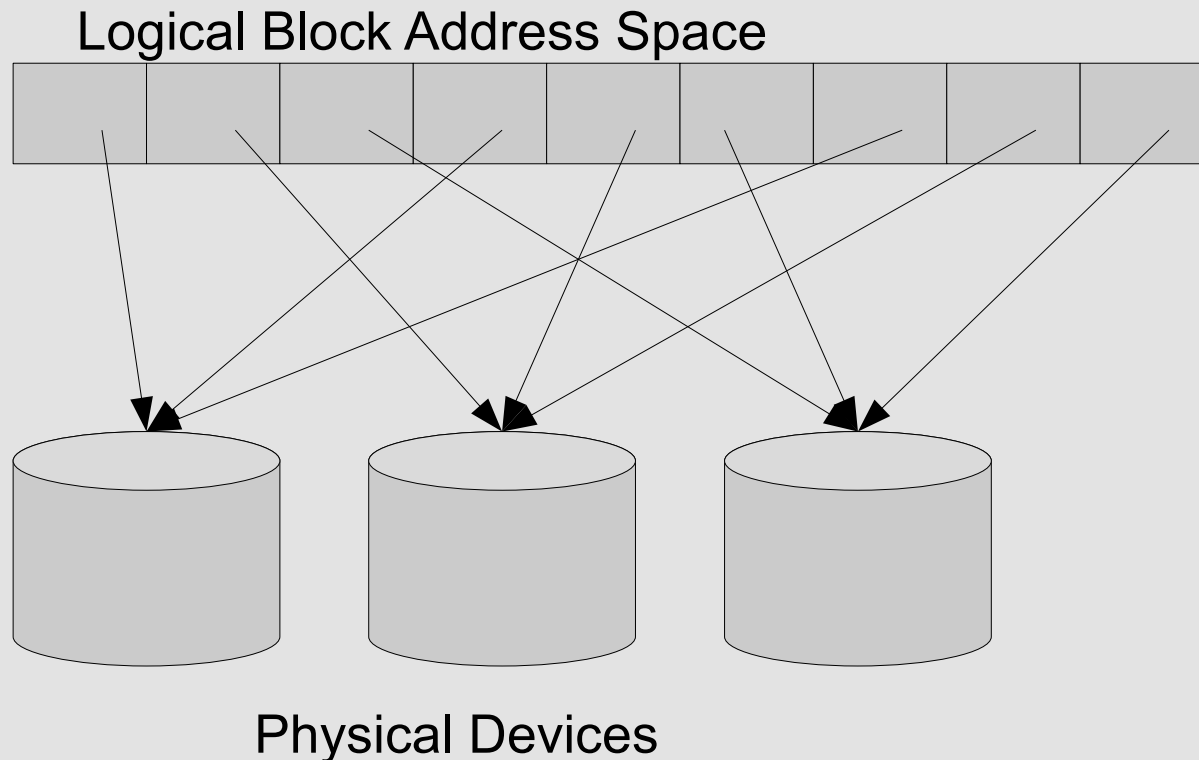
# Logical Block Address Space

- The file system is presented with an array of block addresses to perform I/O operations on



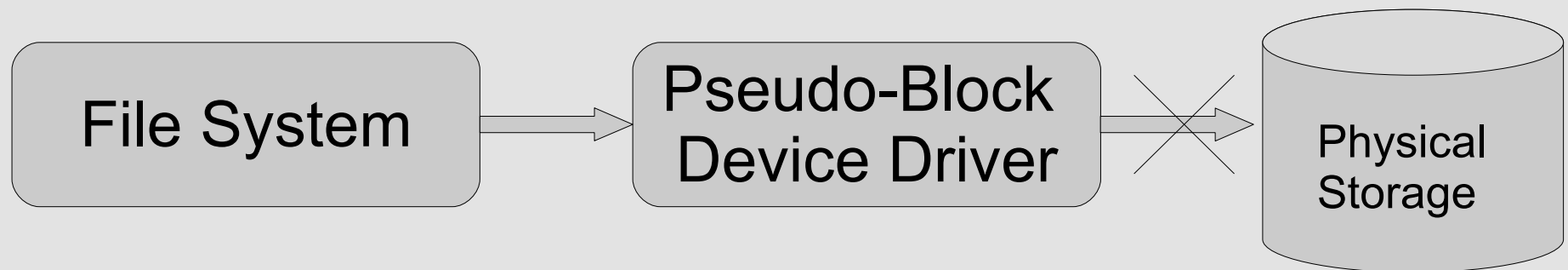
# Logical Block Address Space

- The logical block addresses are mapped to physical device blocks at the block layer



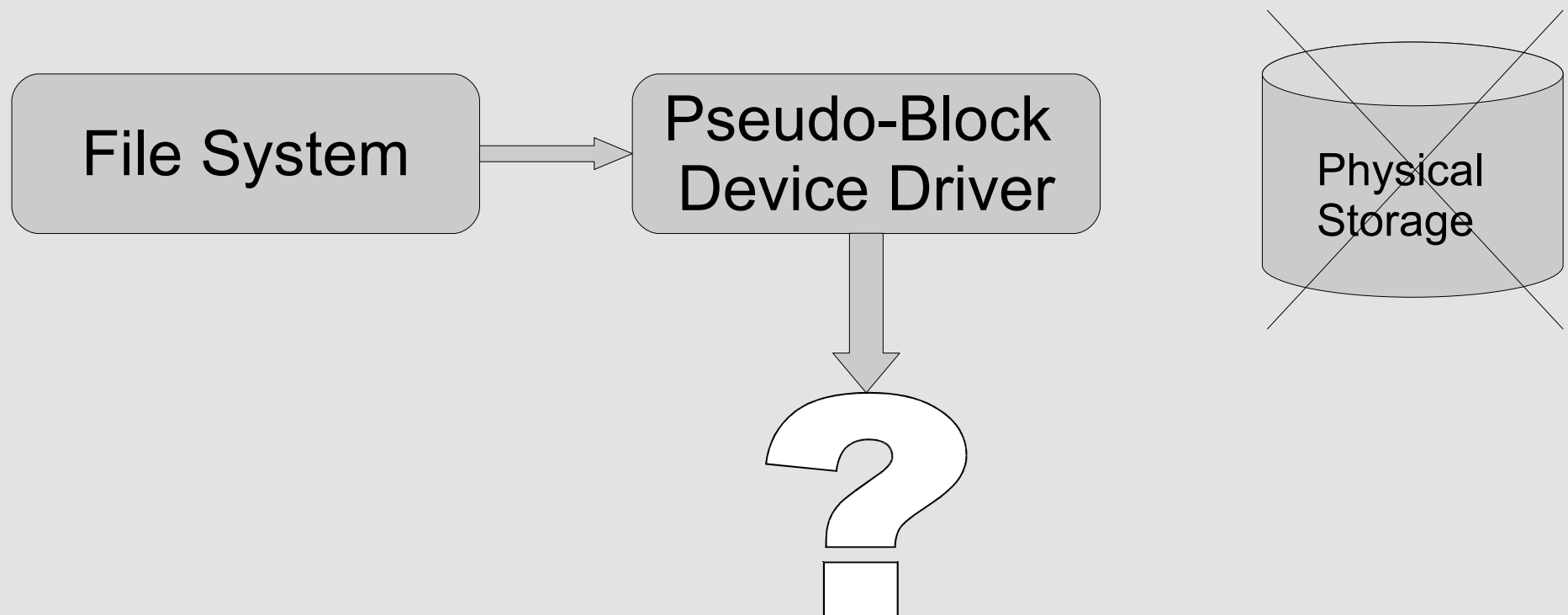
# Pseudo-Block Device Driver

- Performs all the functions of a block device driver, but without a physical block device



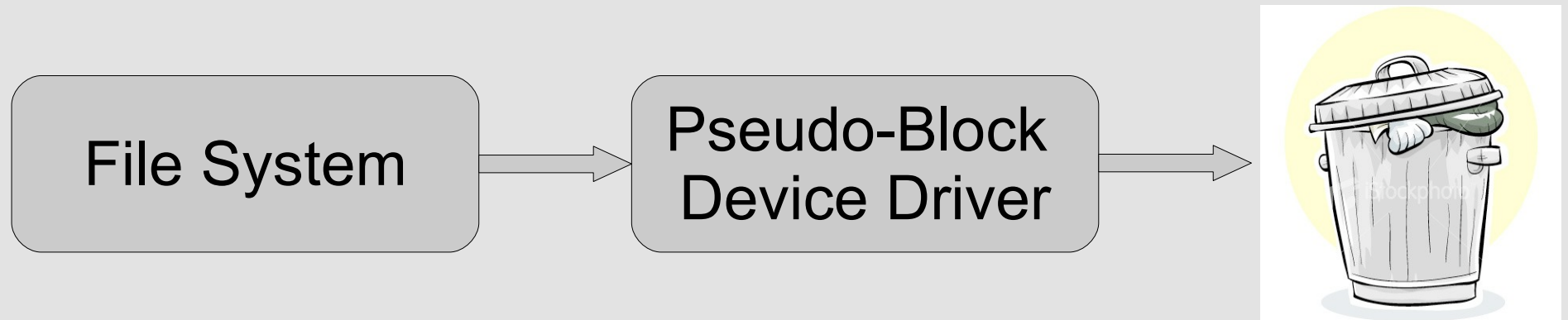
# Pseudo-Block Device Driver

- Performs all the functions of a block device driver, but without a physical block device



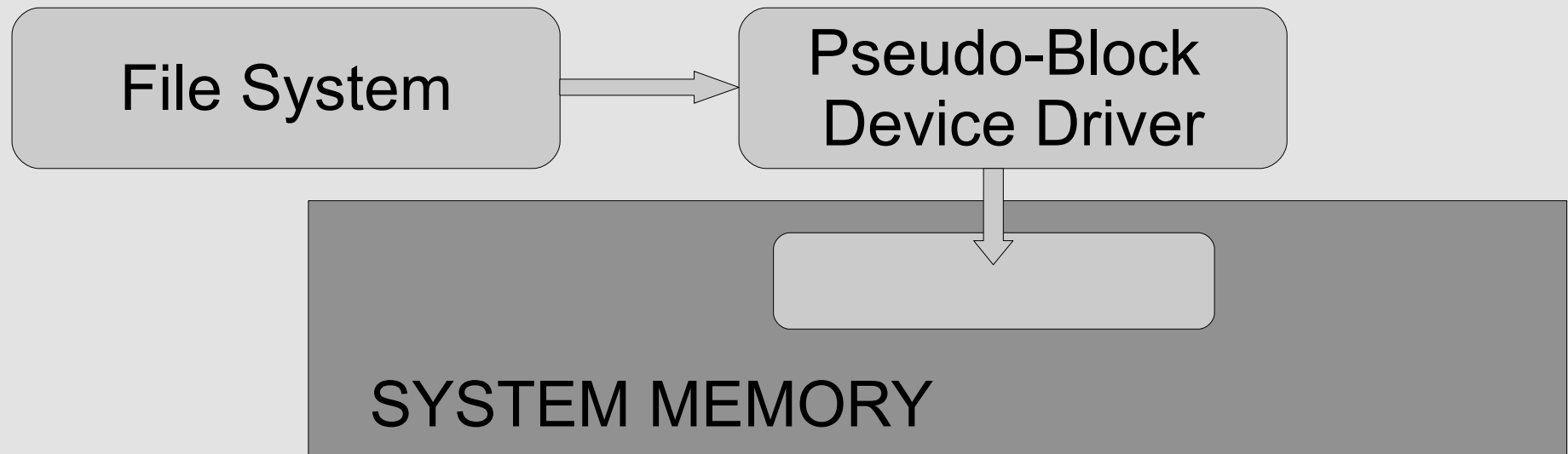
# Pseudo Block Device Driver: Stage 1

- Fully Simulated Block Device Driver
  - Write requests are discarded
  - Read requests return garbage information



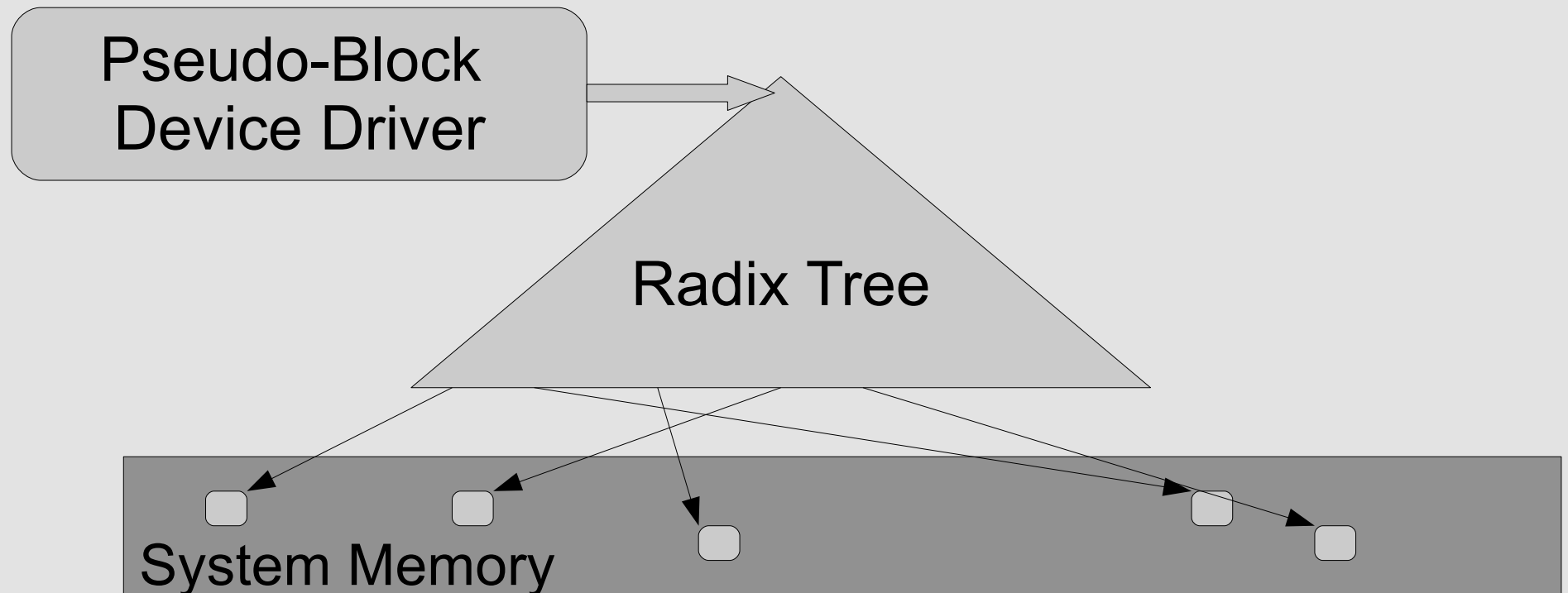
# Pseudo Block Device Driver: Stage 2

- Block Device Simulated With System Memory
  - A virtually contiguous segment of system memory is allocated to the device when it is initialized



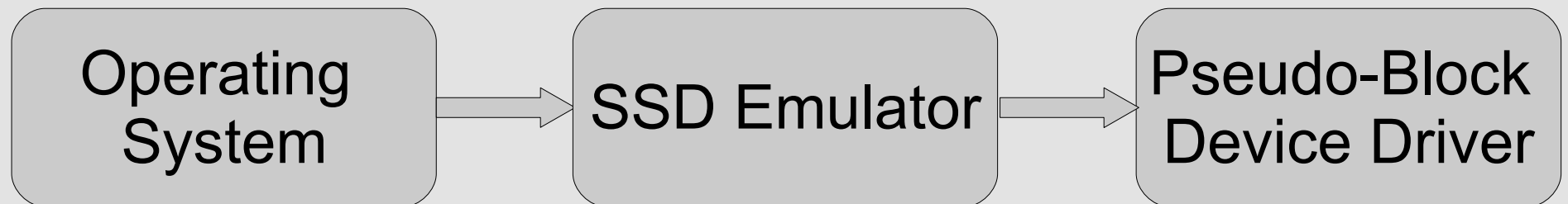
# Pseudo Block Device Driver: Stage 3

- Implementation with Radix Tree
  - Allows arbitrary sized logical block address space
  - Memory allocation is dynamic



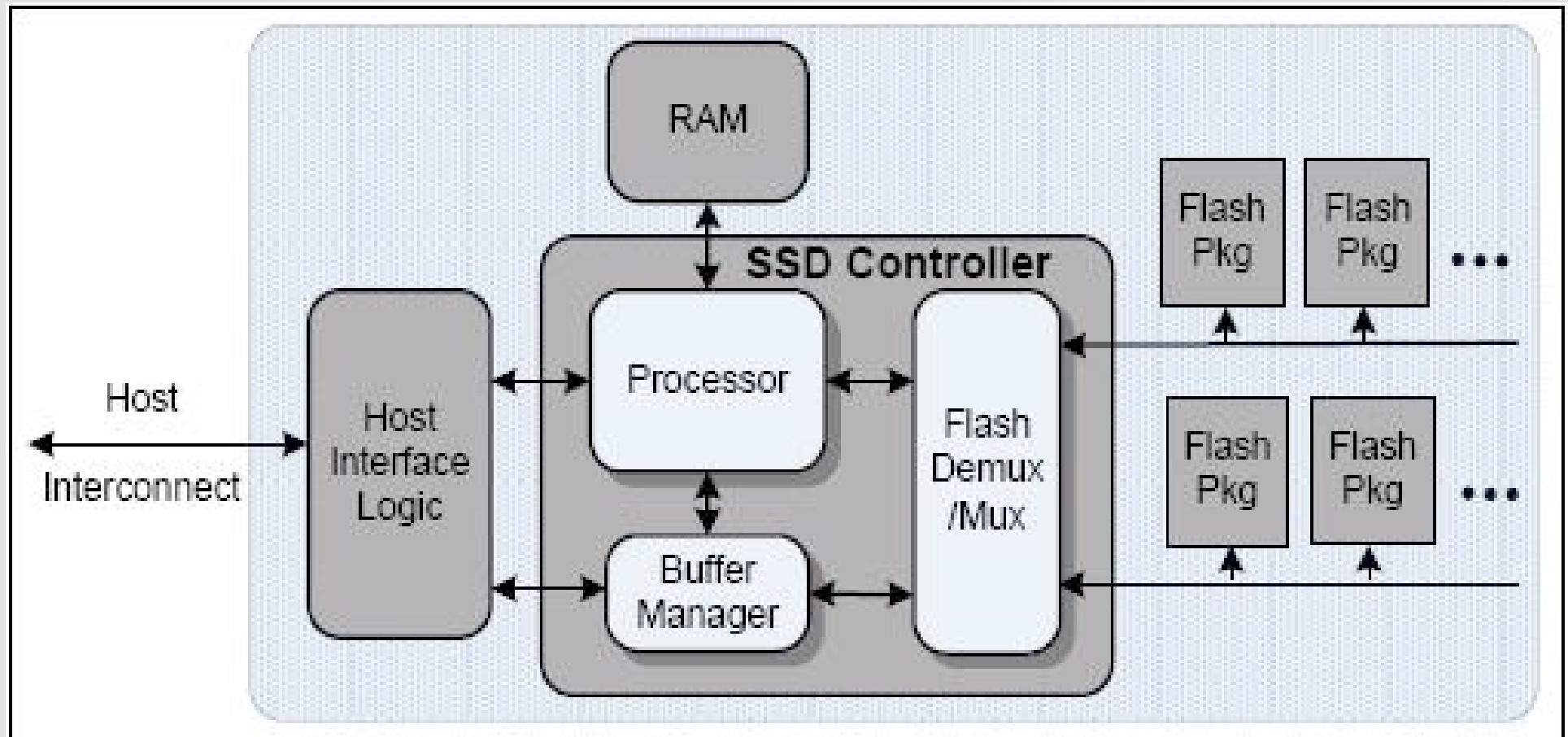
# Solid State Devices and Simulation

- Objective is to create a fully functional solid state device emulator
  - Realistic performance characteristics
  - Storage in system memory using the pseudo block device driver



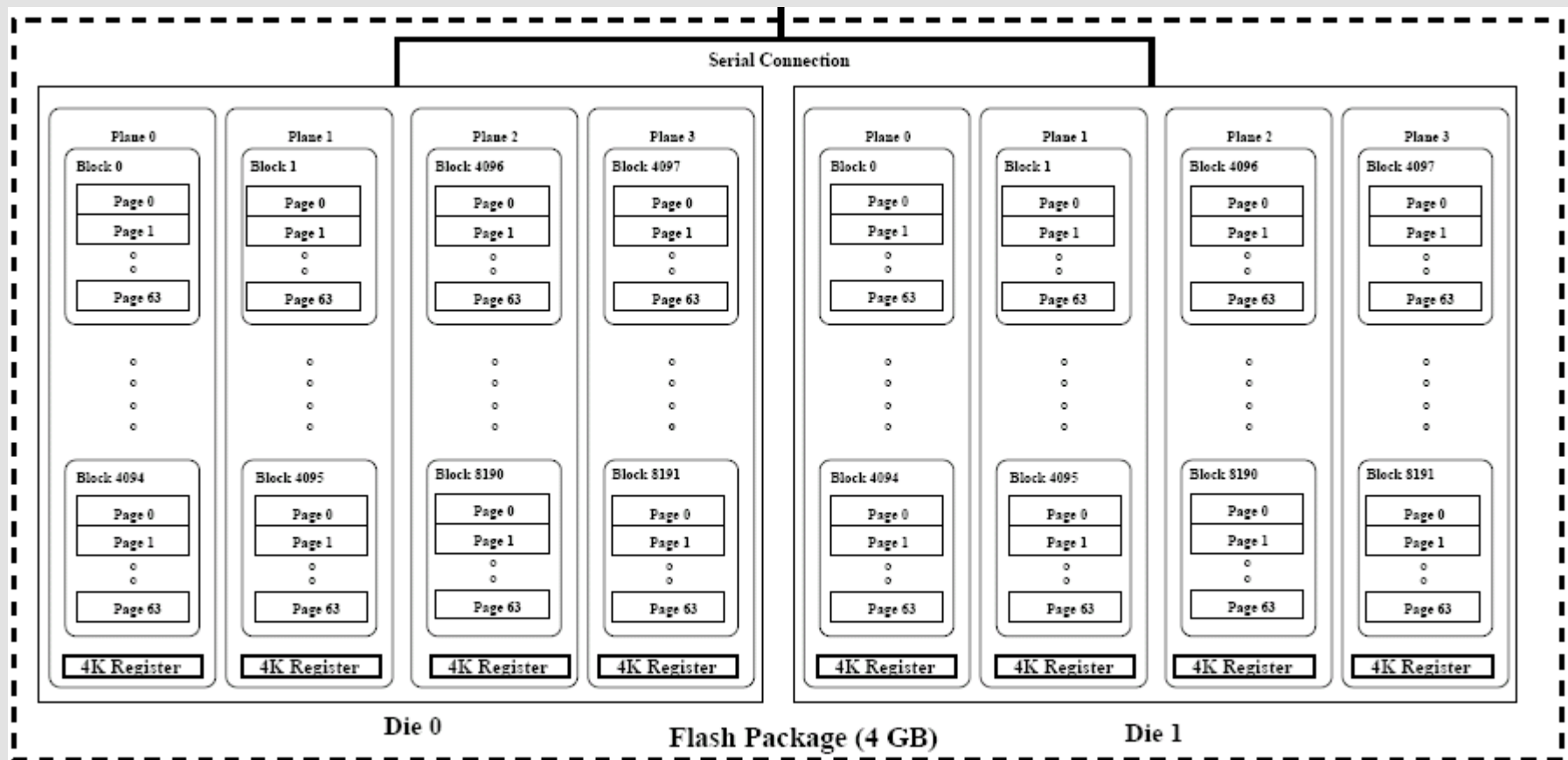
# SSD Architecture

- Controller controls an array of flash packages



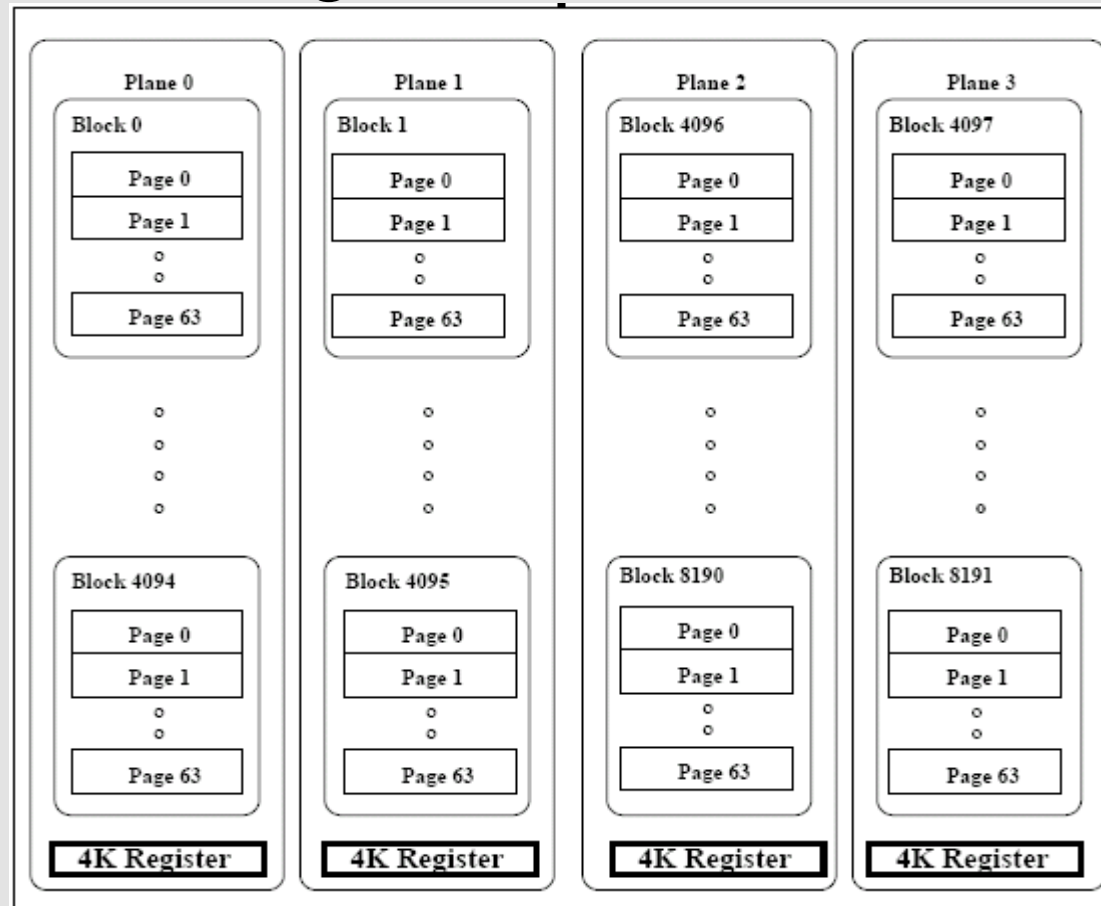
# SSD Architecture

- Flash Packages are made of **Die**, Planes, Blocks, and Pages



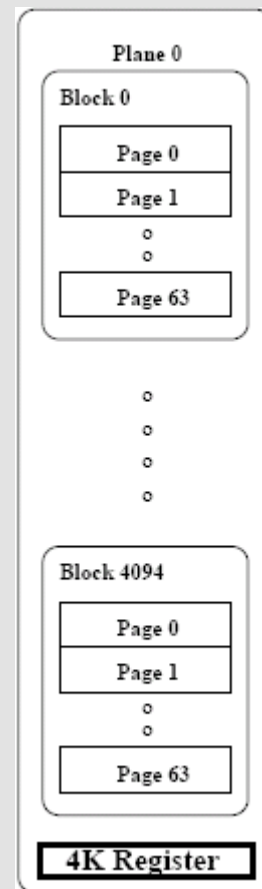
# SSD Architecture

- Flash Packages are made of Die, **Planes**, **Blocks**, and **Pages**



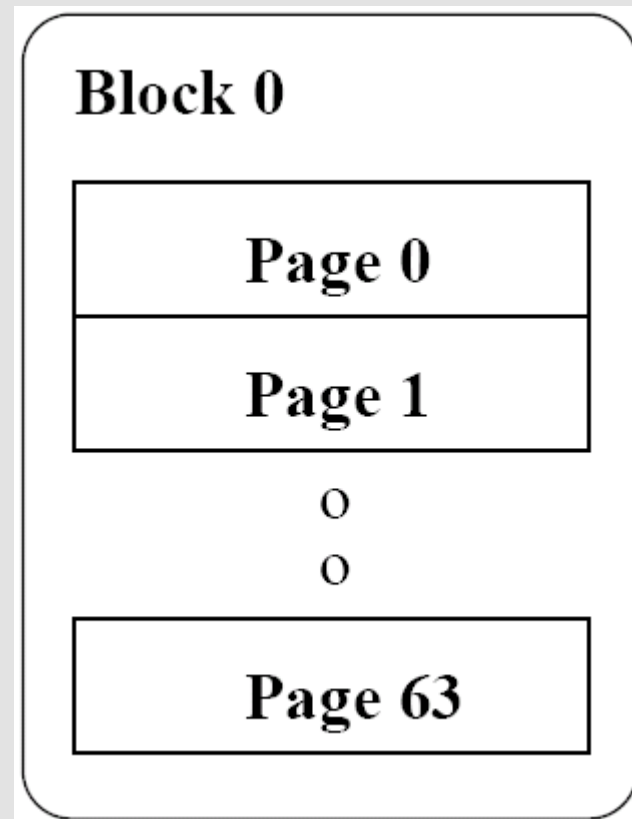
# SSD Architecture

- Flash Packages are made of Die, Planes, **Blocks**, and Pages

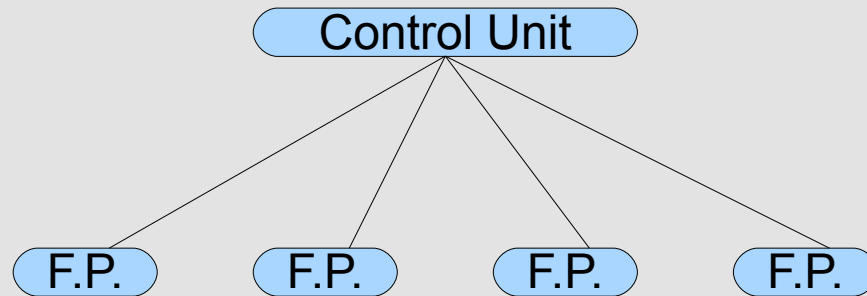


# SSD Architecture

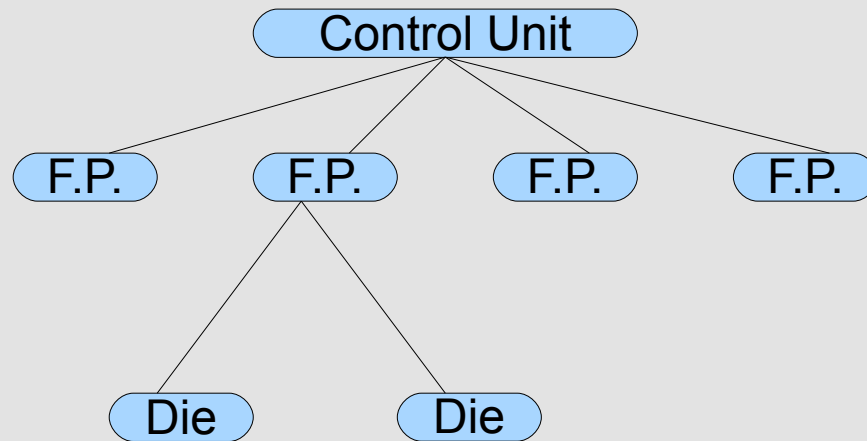
- Flash Packages are made of Die, Planes, Blocks, and **Pages**



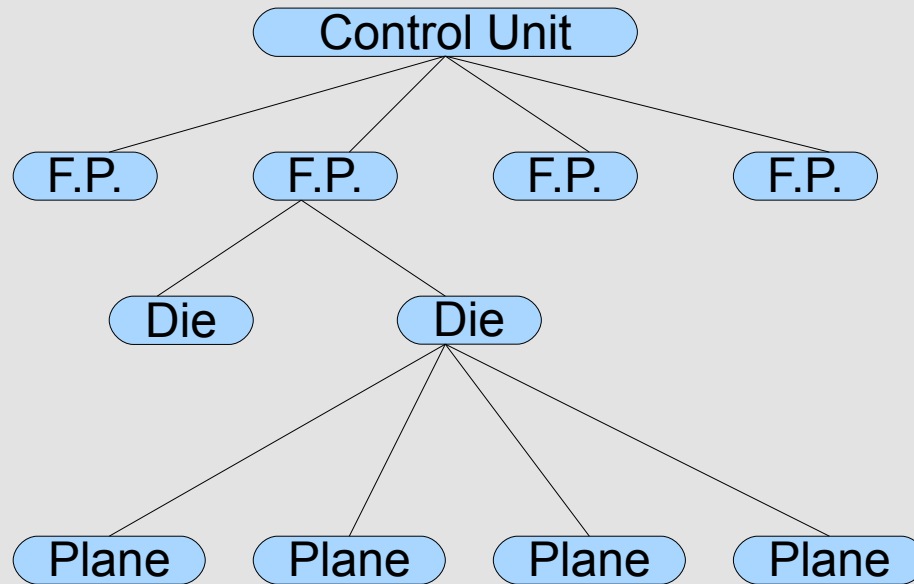
# SSD Architecture



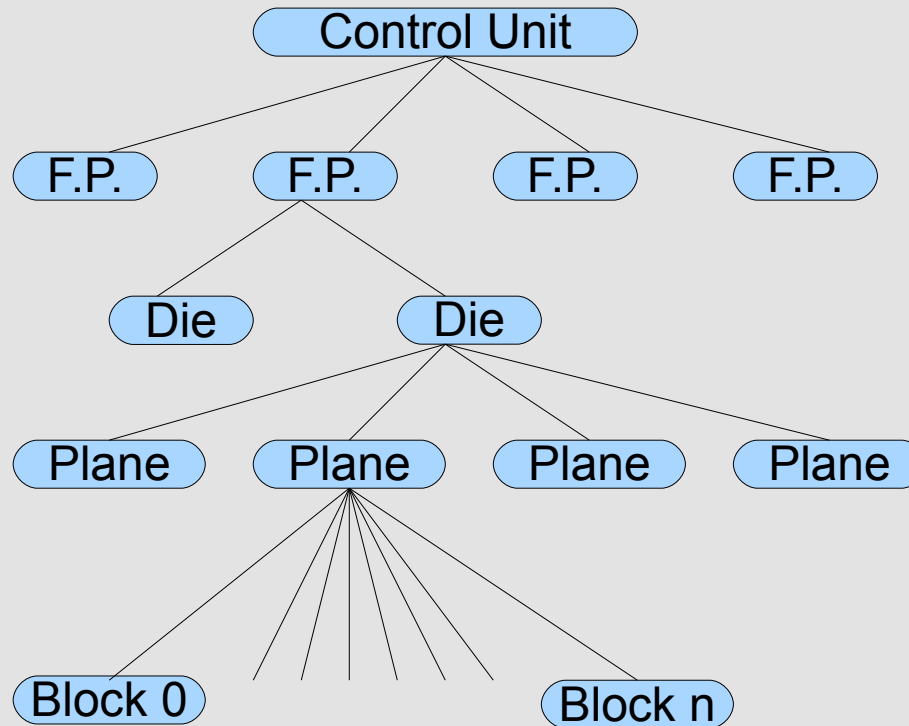
# SSD Architecture



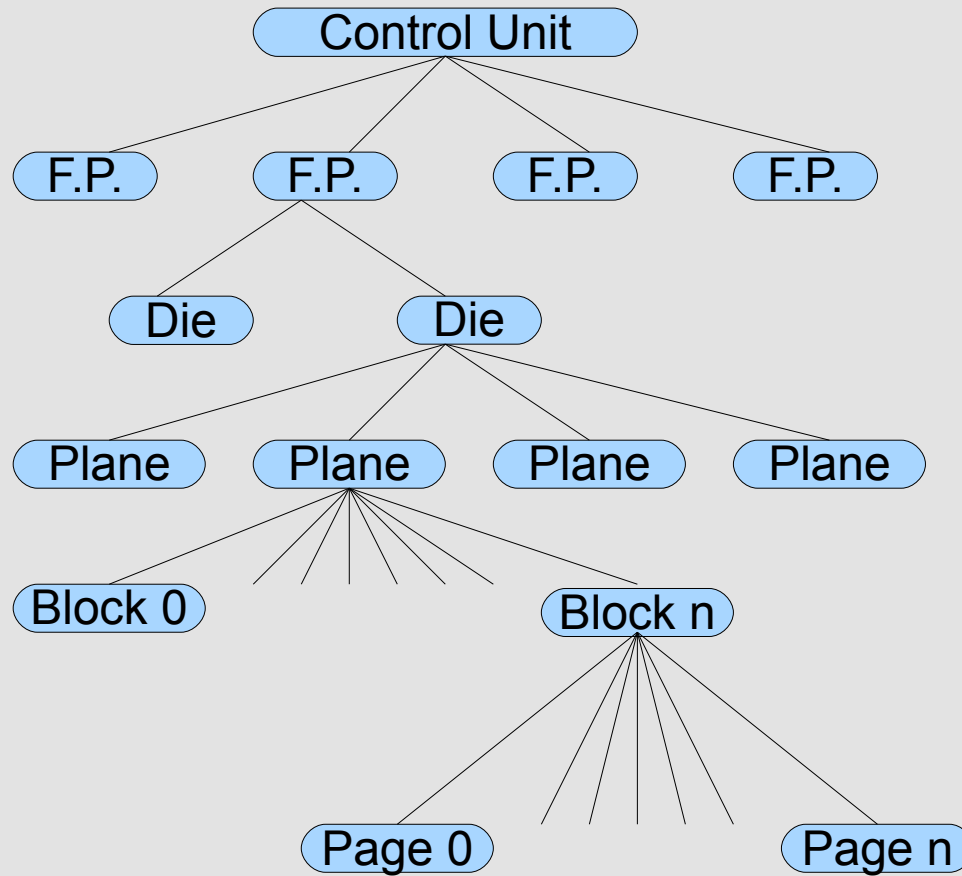
# SSD Architecture



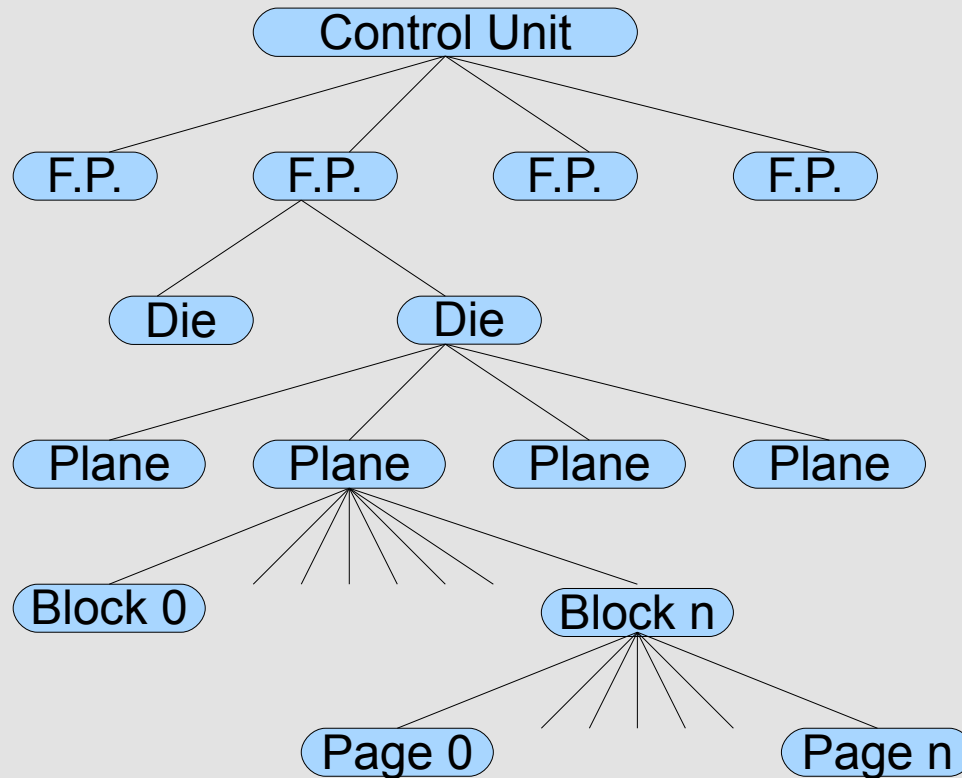
# SSD Architecture



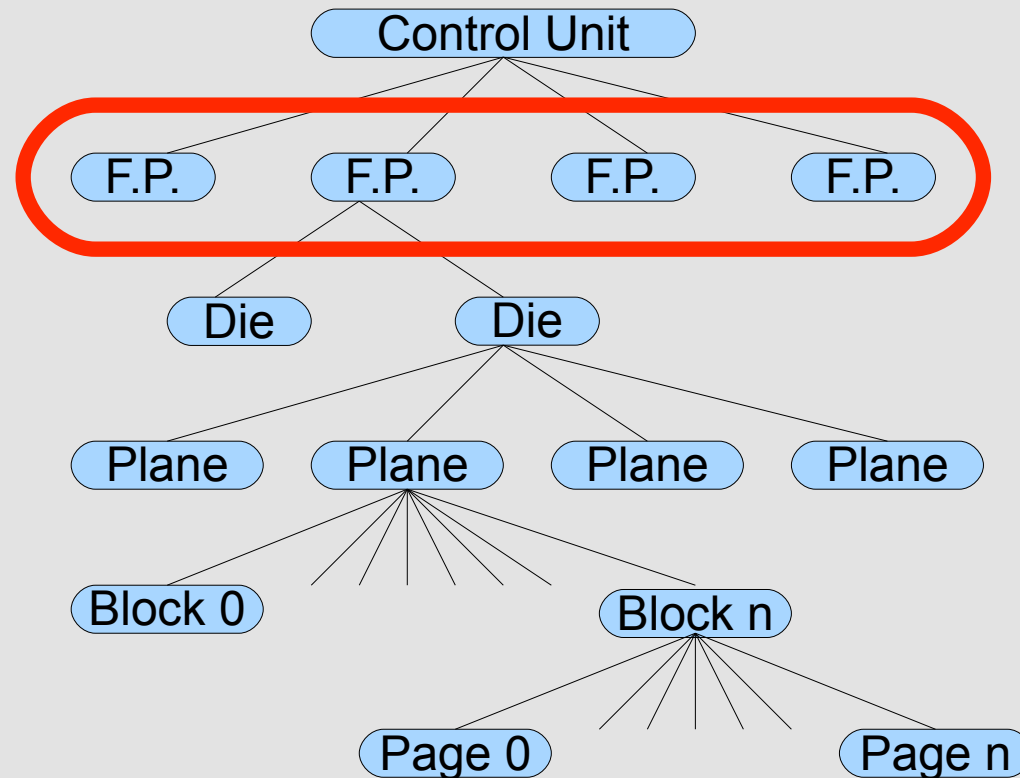
# SSD Architecture



# Parallelism

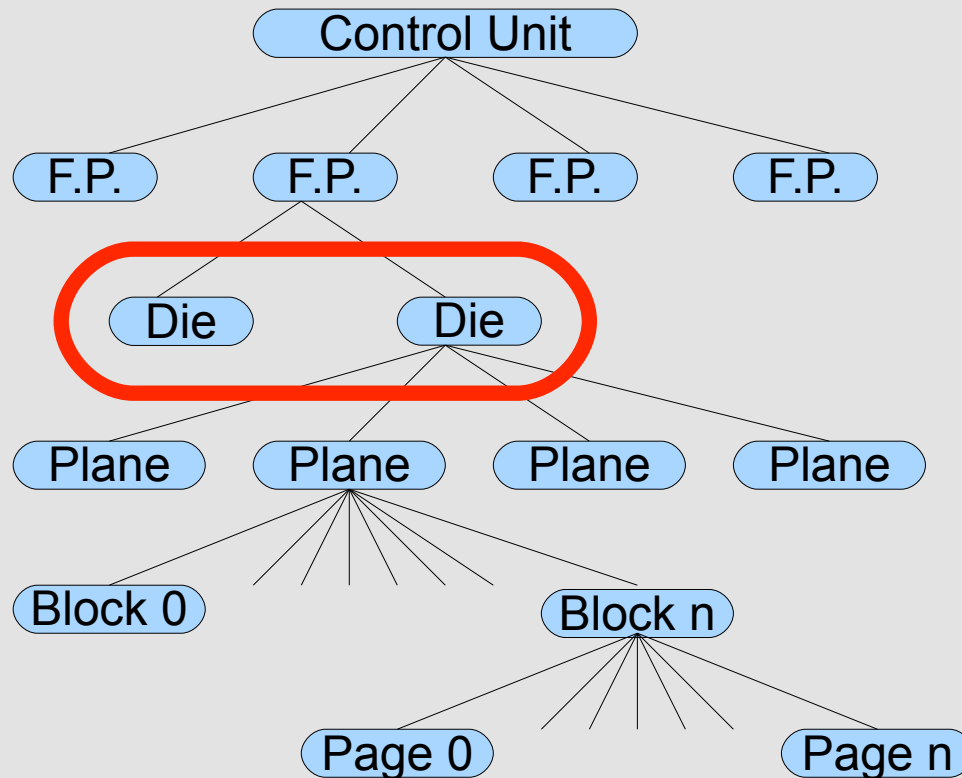


# Parallelism



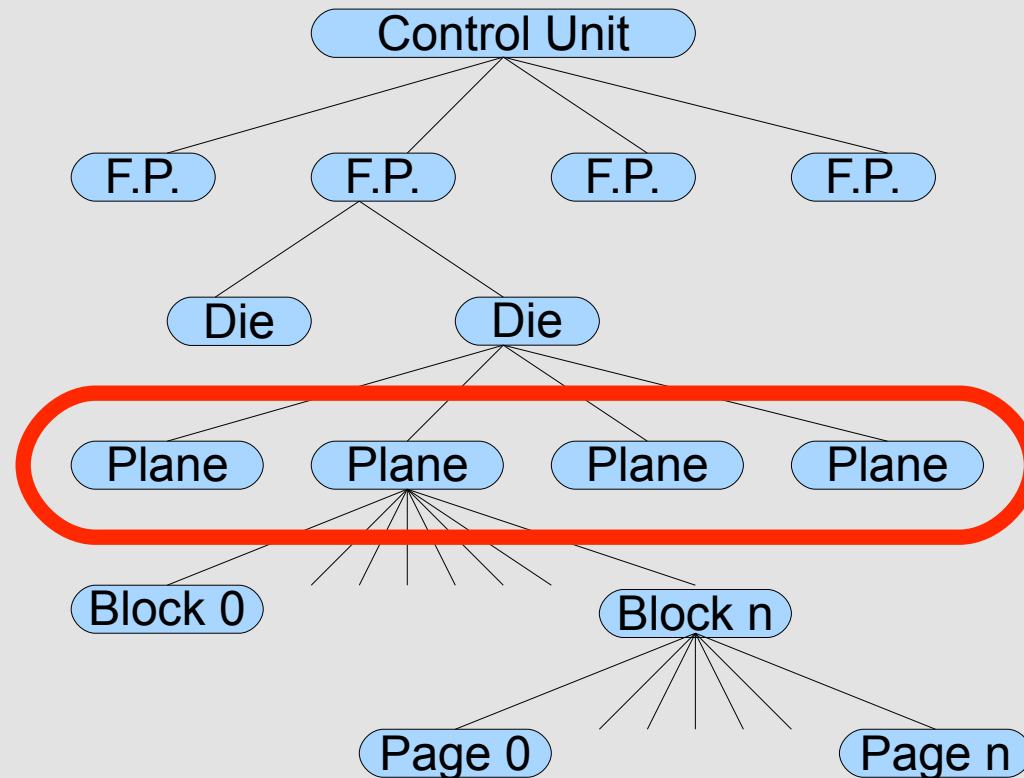
- Parallel units are on the same level and share a parent

# Parallelism



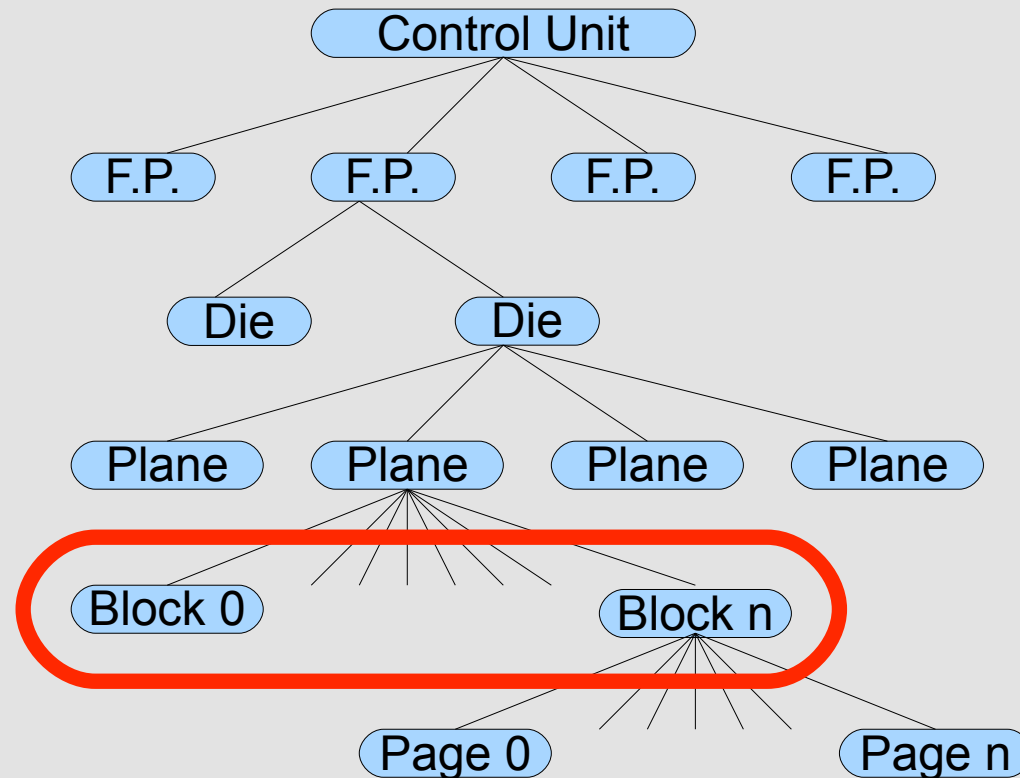
- Parallel units are on the same level and share a parent

# Parallelism



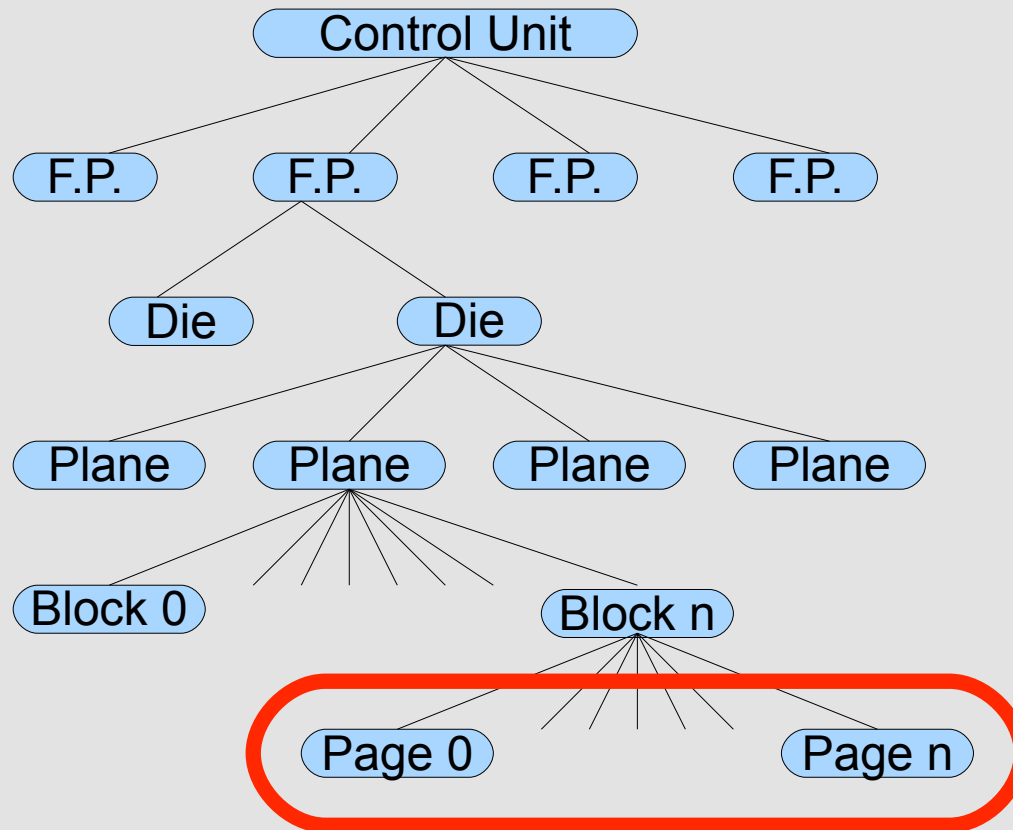
- Parallel units are on the same level and share a parent

# Parallelism



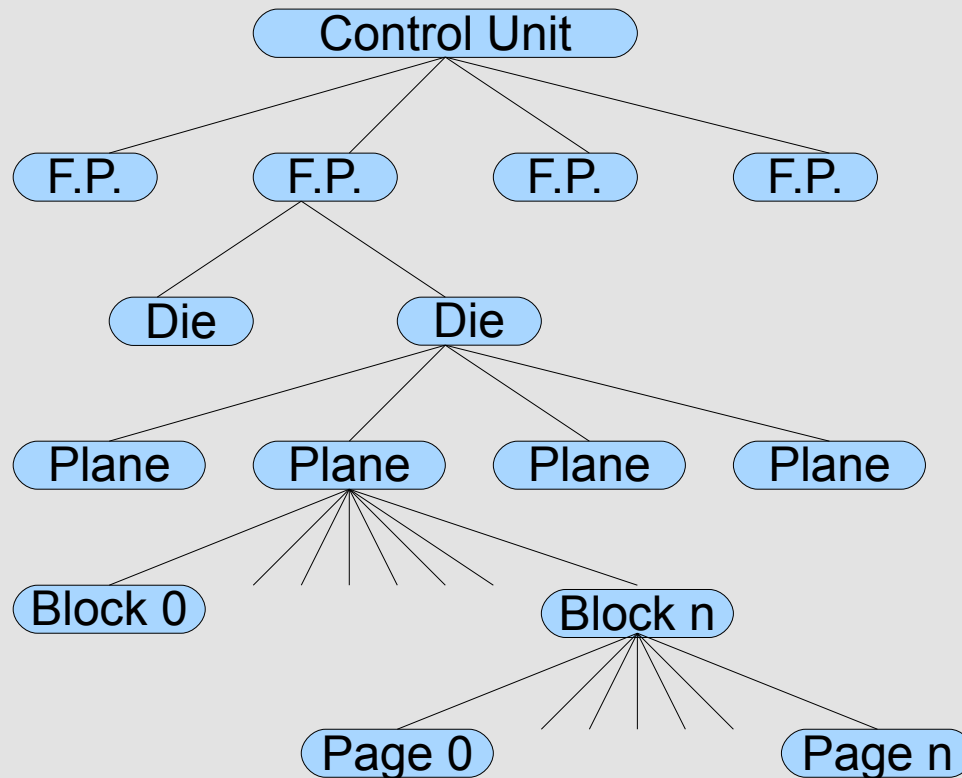
- Parallel units are on the same level and share a parent

# Parallelism



- Parallel units are on the same level and share a parent

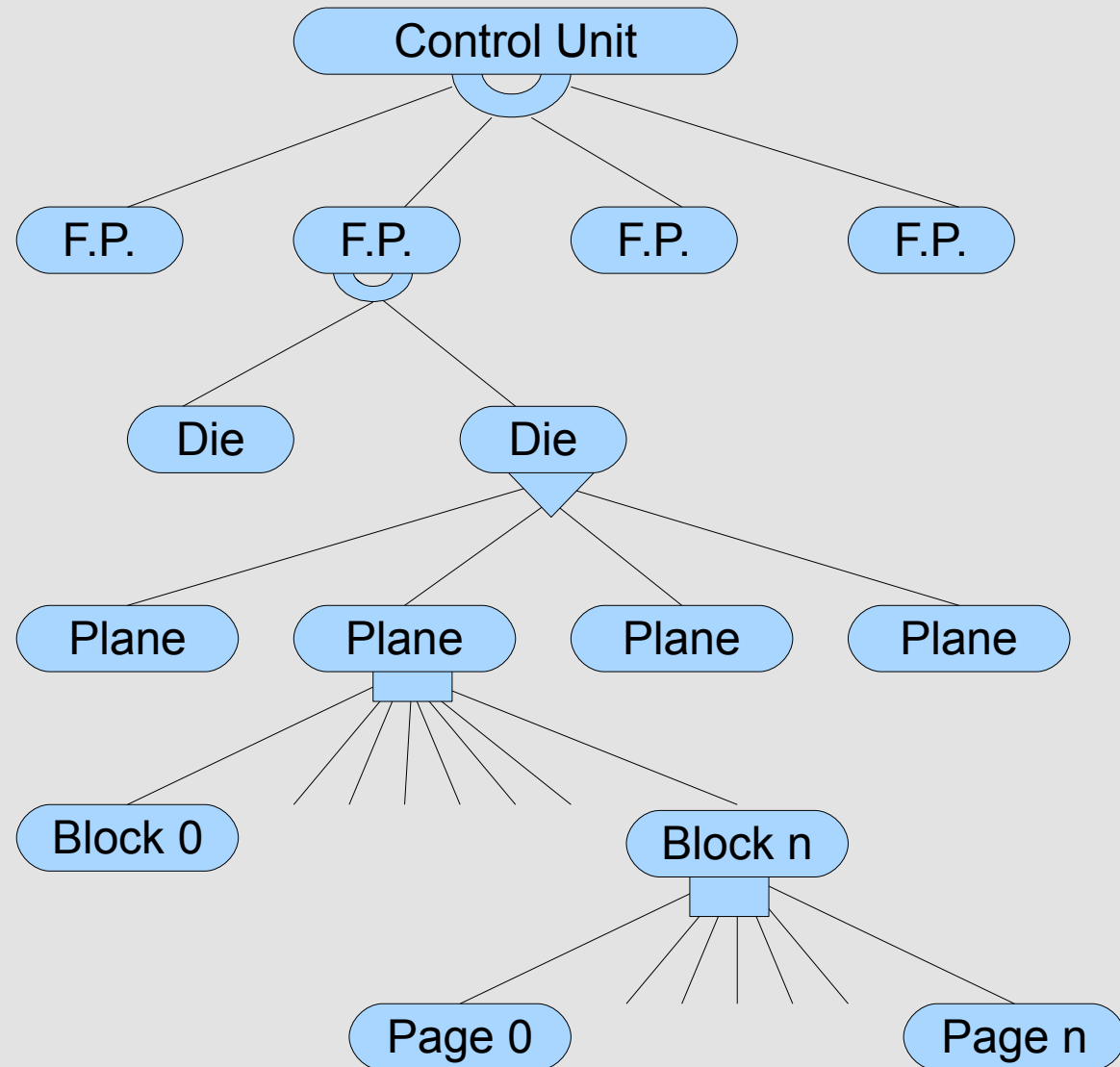
# Parallelism



- Depending on the vendor, parallel units may be accessed in parallel, semi parallel, or sequentially

# Parallelism

- Parallelism at each level is vendor specific
- Values will be set by the user before run time in a configuration file



# SSD Characteristics

- Fast random reads
- Slow deletion time
- Only clean blocks may be written to
- Writes must occur sequentially within blocks
- Read and Write to the page level
- Deletion on the block level

Page Read to Register	$25\mu\text{s}$
Page Program (Write) from Register	$200\mu\text{s}$
Block Erase	1.5ms
Serial Access to Register (Data bus)	$100\mu\text{s}$

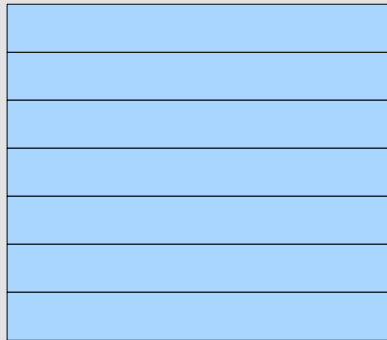
# SSD Characteristics

- Dynamic Block Mapping, Cleaning, and Wear Leveling
  - All handled by the control unit
  - Mapping of logical blocks to physical pages is dynamic
  - Each page has a limited number of erase operations in its lifetime
  - Cleaning mechanism is needed to keep a supply of clean blocks on hand for writes

# SSD Characteristics

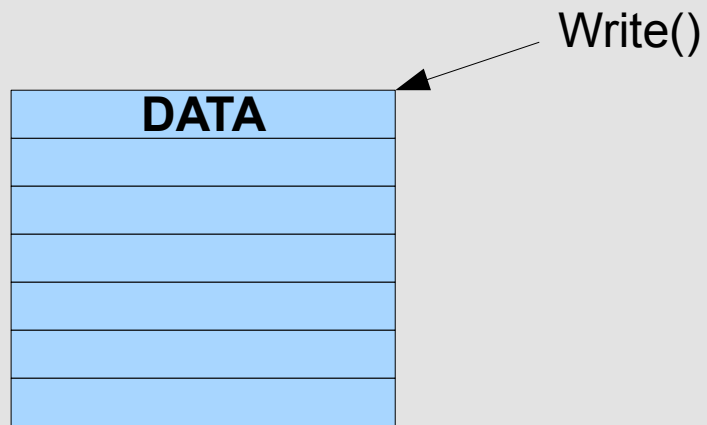
- Merging

CLEAN BLOCK



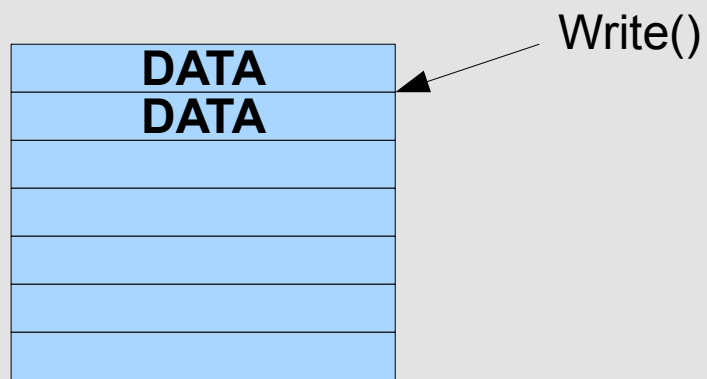
# SSD Characteristics

- Merging



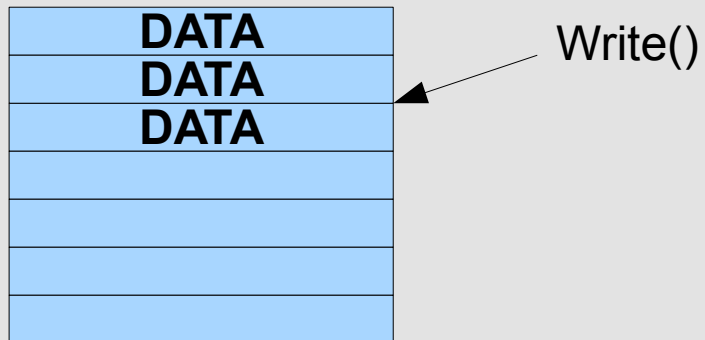
# SSD Characteristics

- Merging



# SSD Characteristics

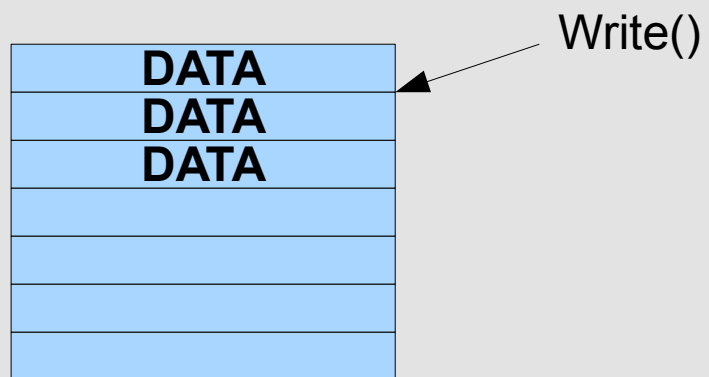
- Merging



# SSD Characteristics

- Merging

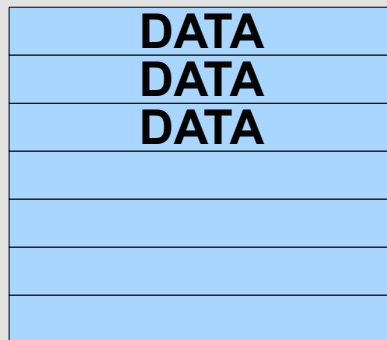
INVALID OPERATION



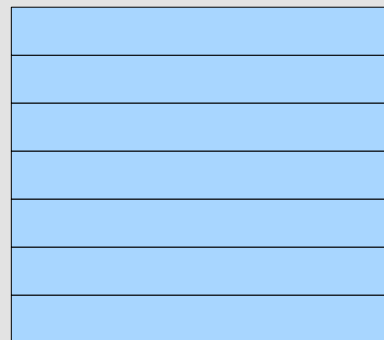
# SSD Characteristics

- Merging

CURRENT BLOCK

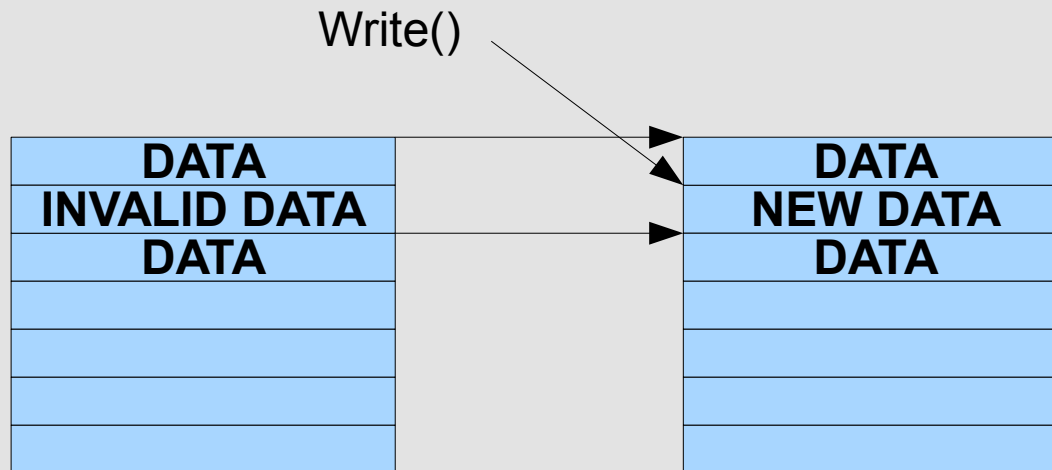


CLEAN BLOCK



# SSD Characteristics

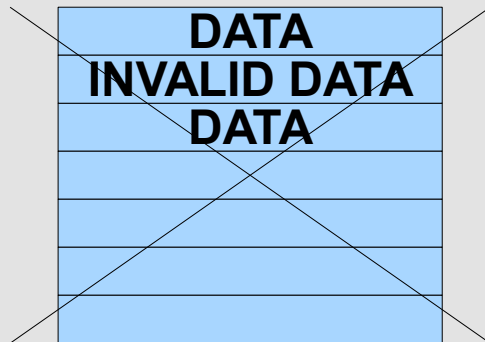
- Merging



# SSD Characteristics

- Merging

DIRTY BLOCK



NEW BLOCK



# Conclusion

- Pseudo Block Device Driver stages 1, 2, and 3 are complete
- SSD Emulator will utilize the stage 3 pseudo block device driver
- SSD Emulator will allow testing of ABLE in a SSD environment with storage and performance feedback

# Any Questions?

