

Emulating Goliath Storage Systems with David

Nitin Agrawal, NEC Labs

Leo Arulraj,

Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau

ADSL Lab, UW Madison



The Storage Researchers' Dilemma

Innovate

Create the future of storage

Measure

Quantify improvement obtained

Dilemma

How to measure future of storage with
devices from present?

David: A Storage Emulator

**Large, fast, multiple disks using
small, slow, single device**

Huge Disks

~1TB disk using 80 GB disk

Multiple Disks

RAID of multiple disks using RAM

Key Idea behind David

**Store metadata, throw away data
(and generate fake data)**

Why is this OK ?

Benchmarks measure performance

Many benchmarks don't care about file content

Some expect valid but not exact content

Outline

~~Intro~~

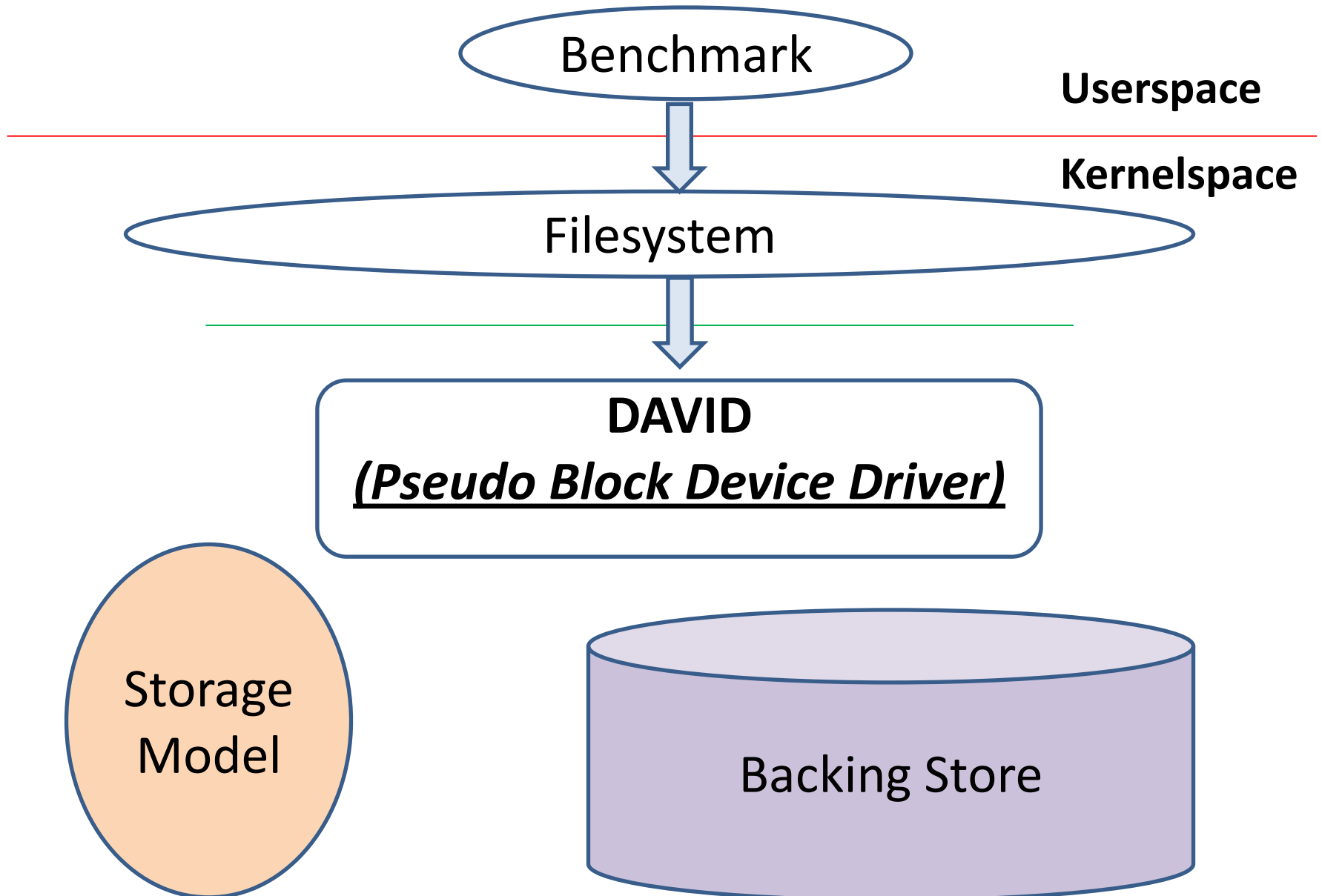
Overview

Design

Results

Conclusion

Overview of how David works



Illustrative Benchmark

Create a File

Write a block of data

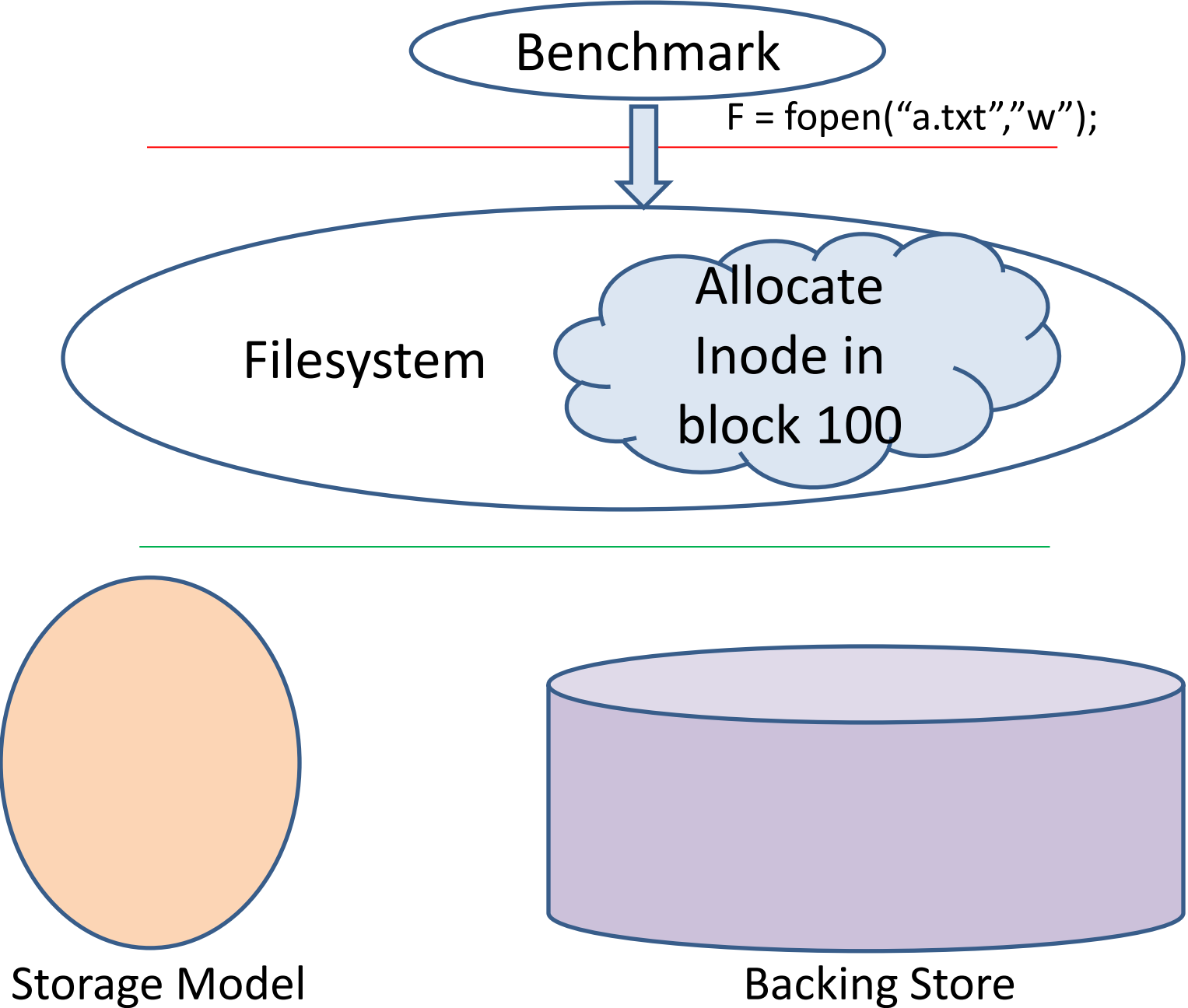
Close the File

Open file in read mode

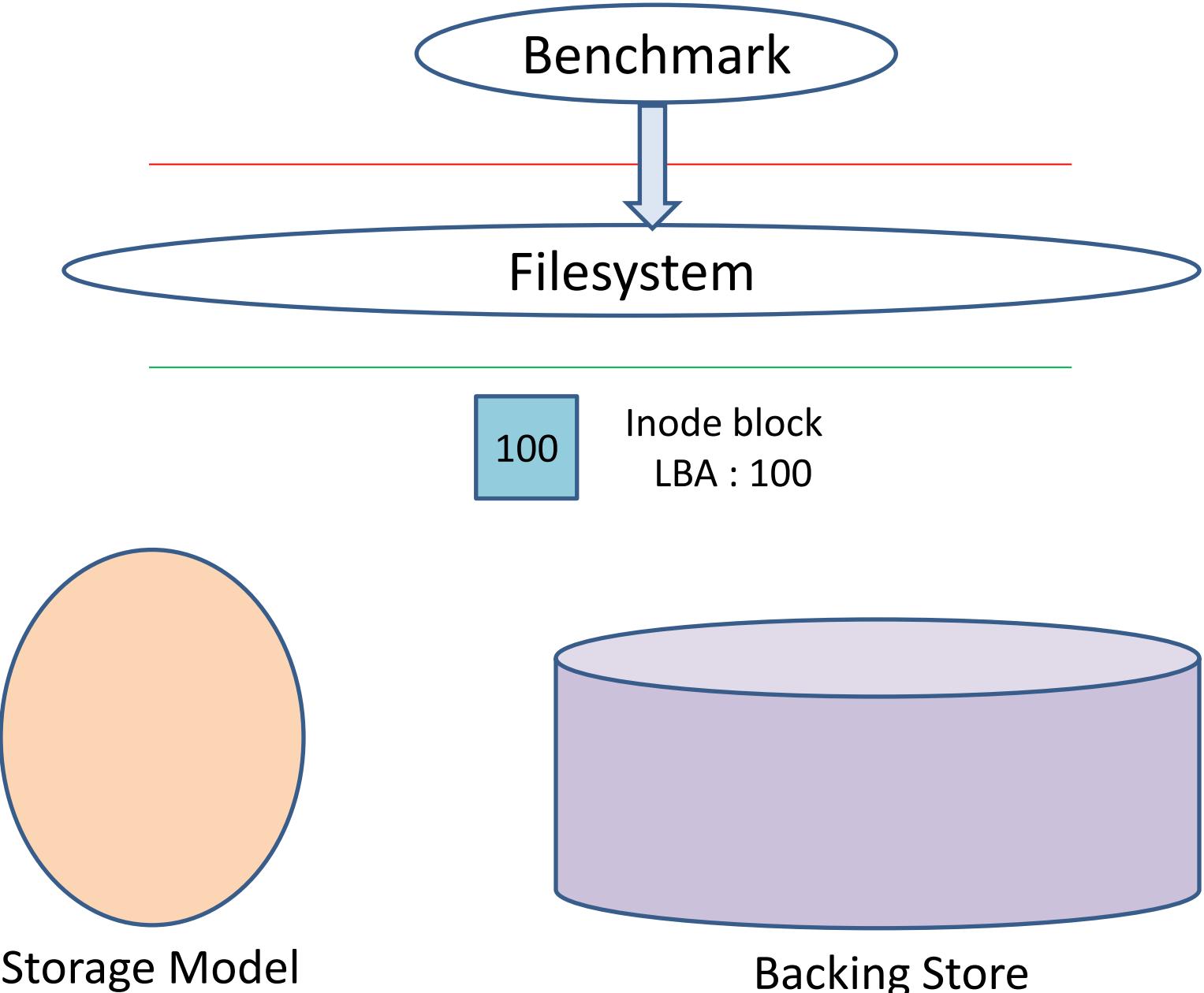
Read back the data

Close the File

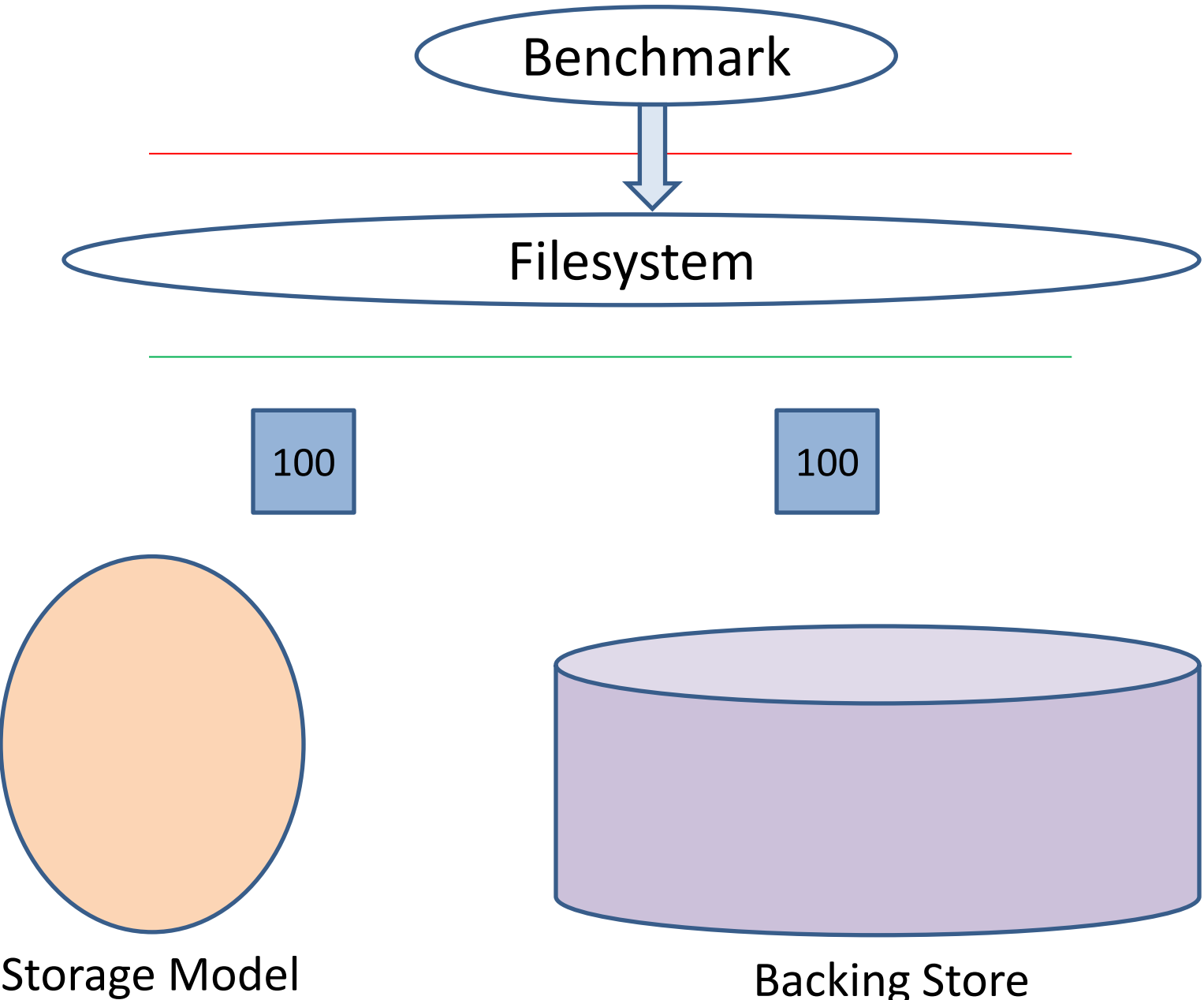
How does David handle **metadata write**?



How does David handle **metadata write**?



How does David handle **metadata write**?



How does David handle **metadata write**?

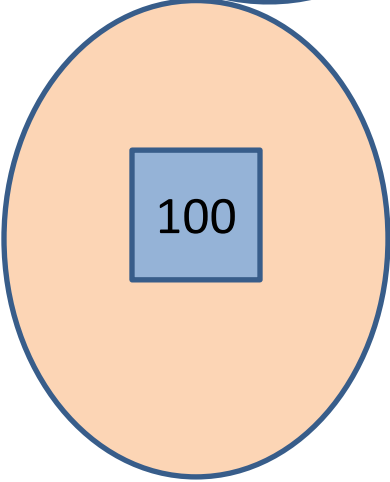
Benchmark



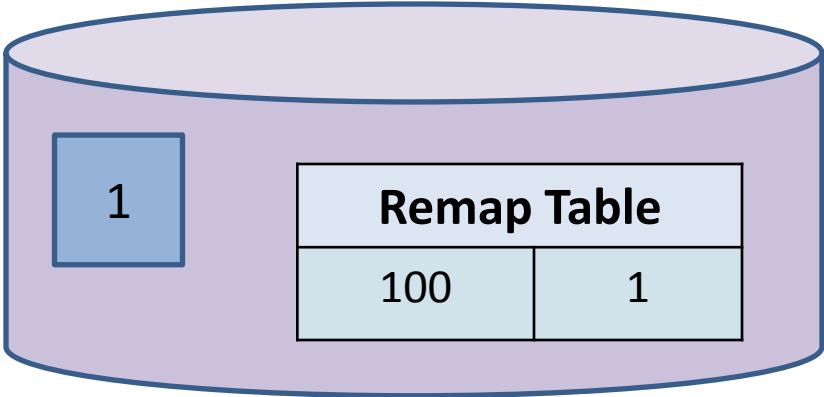
Filesystem

Model calculates response time for write to LBA 100

Metadata block at LBA 100 is remapped to LBA 1

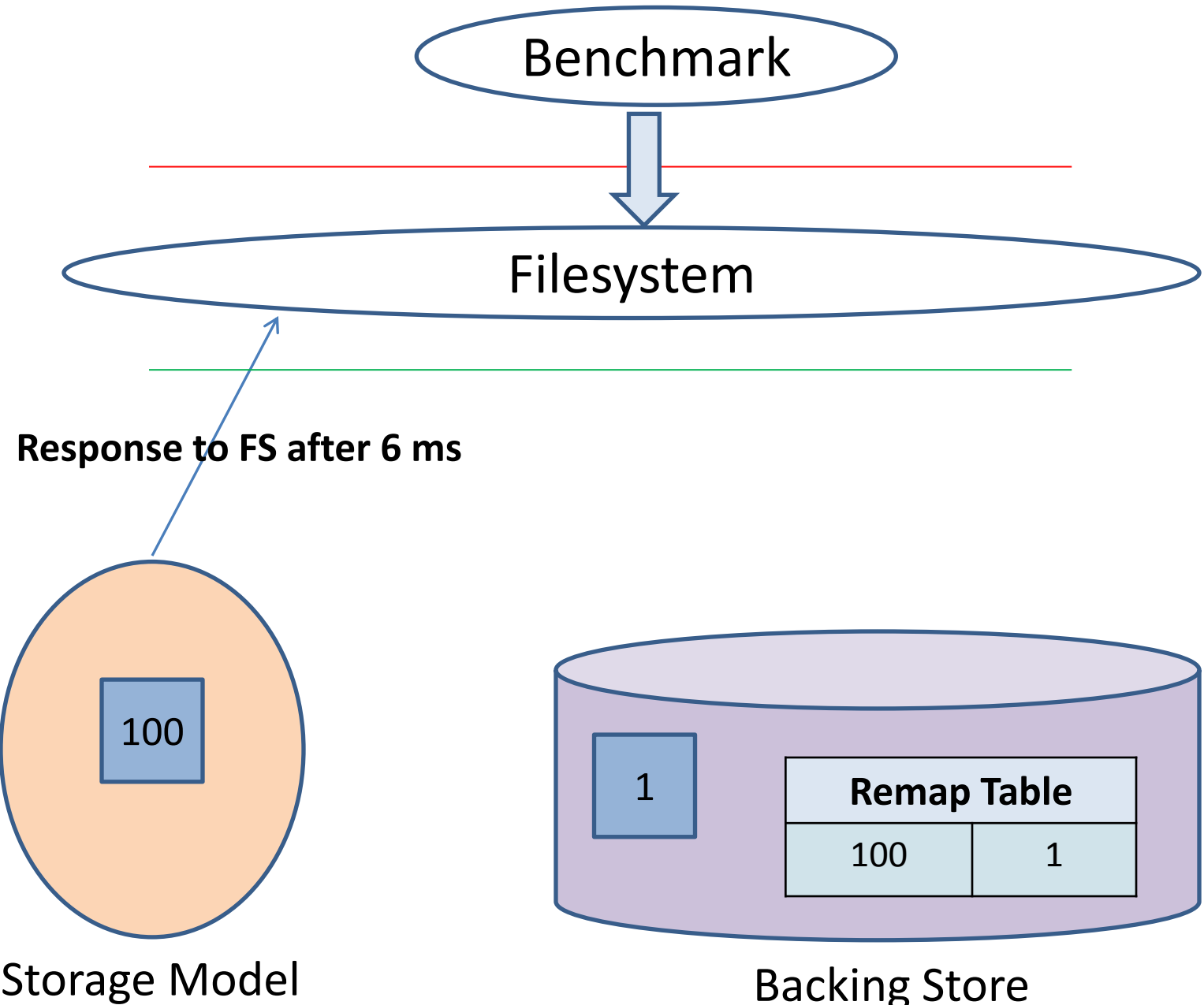


Storage Model

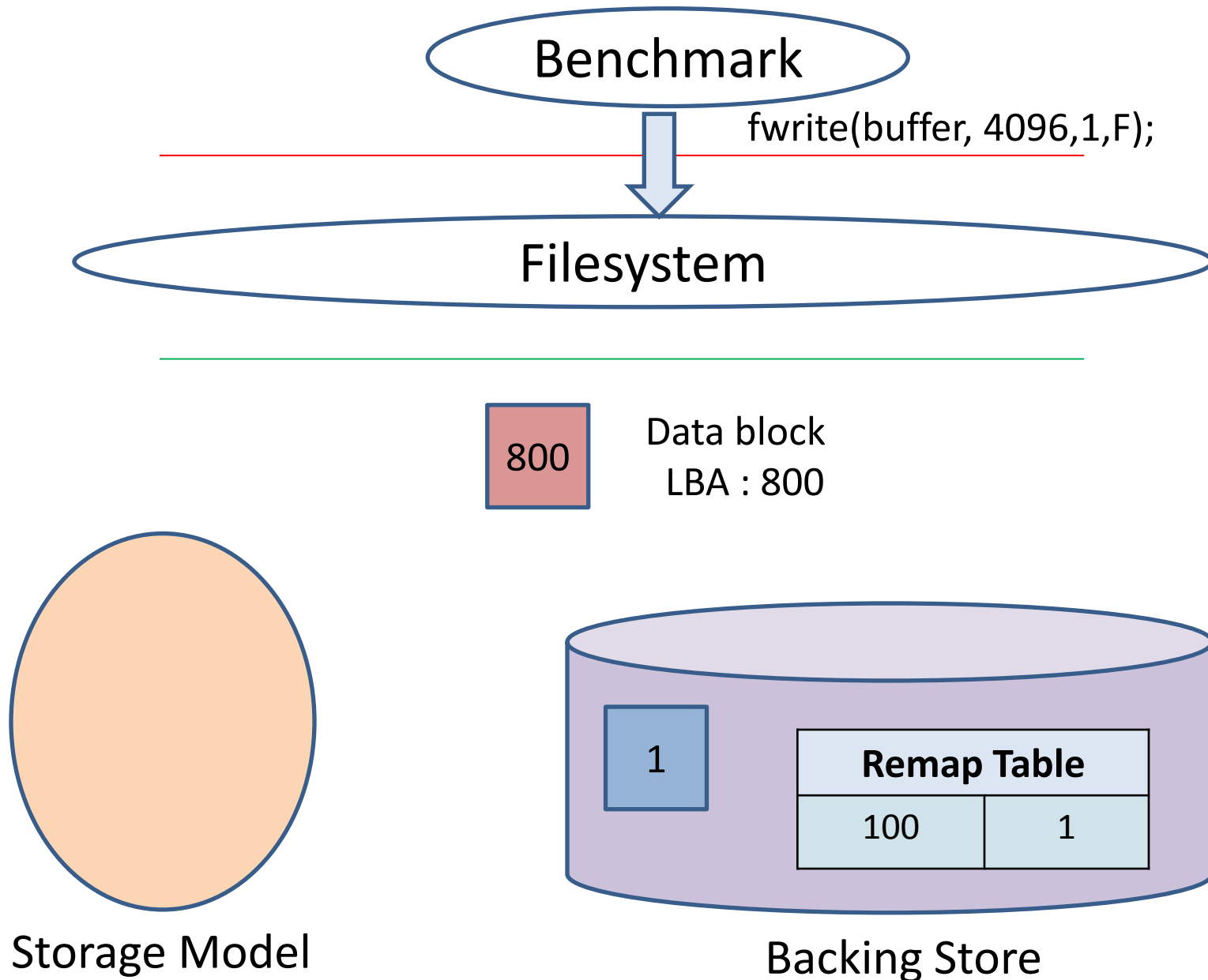


Backing Store

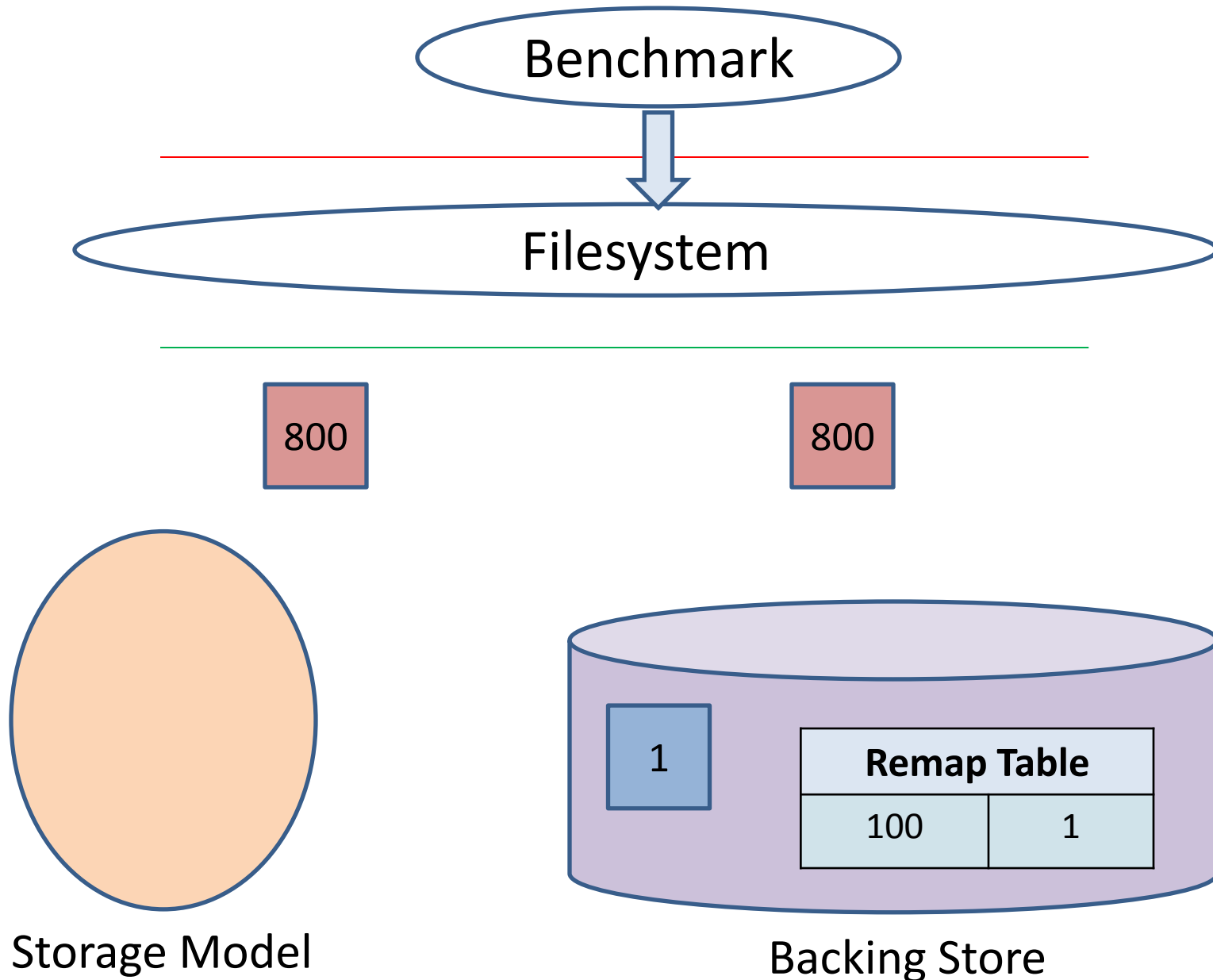
How does David handle **metadata write**?



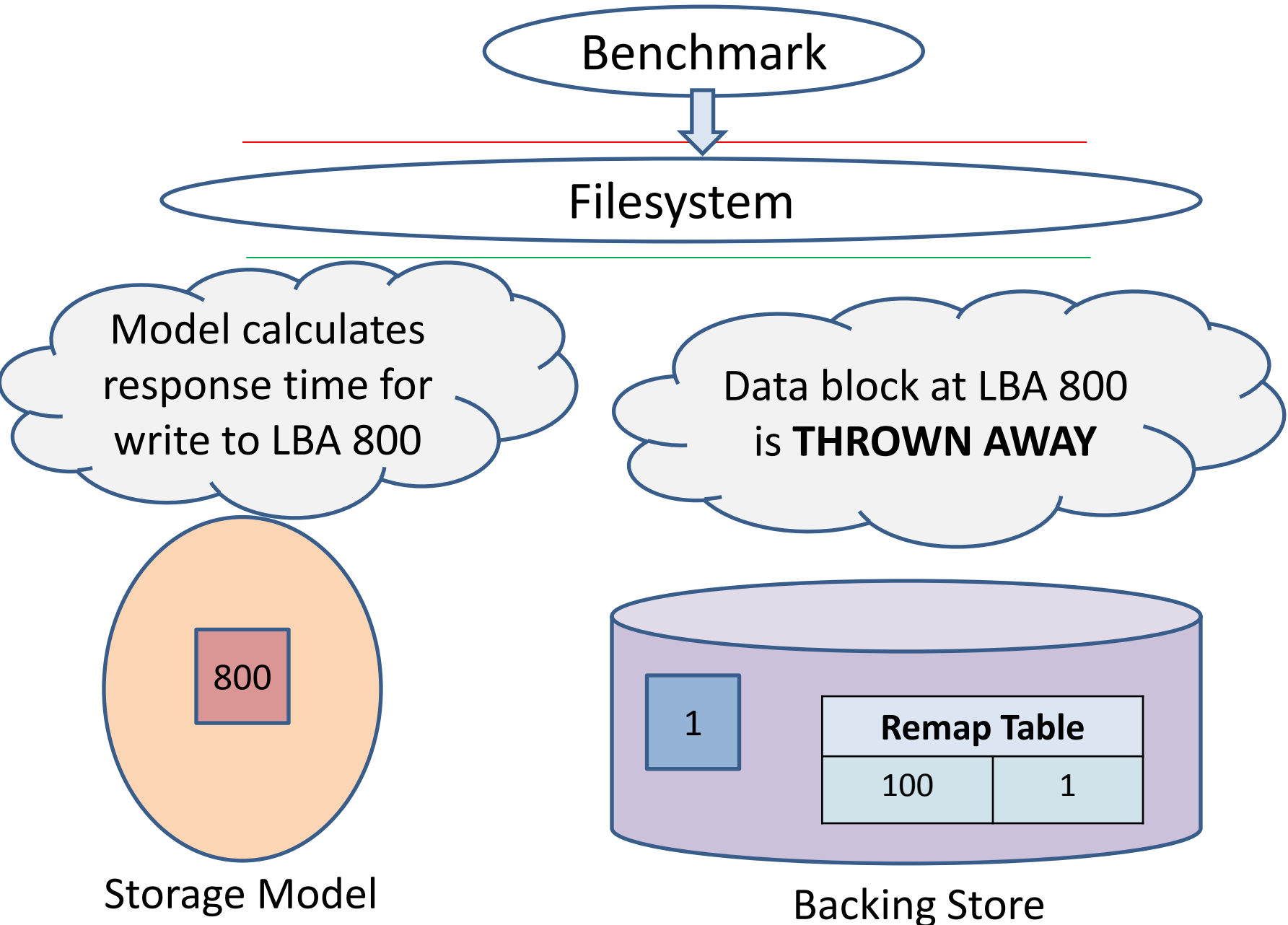
How does David handle **data write**?



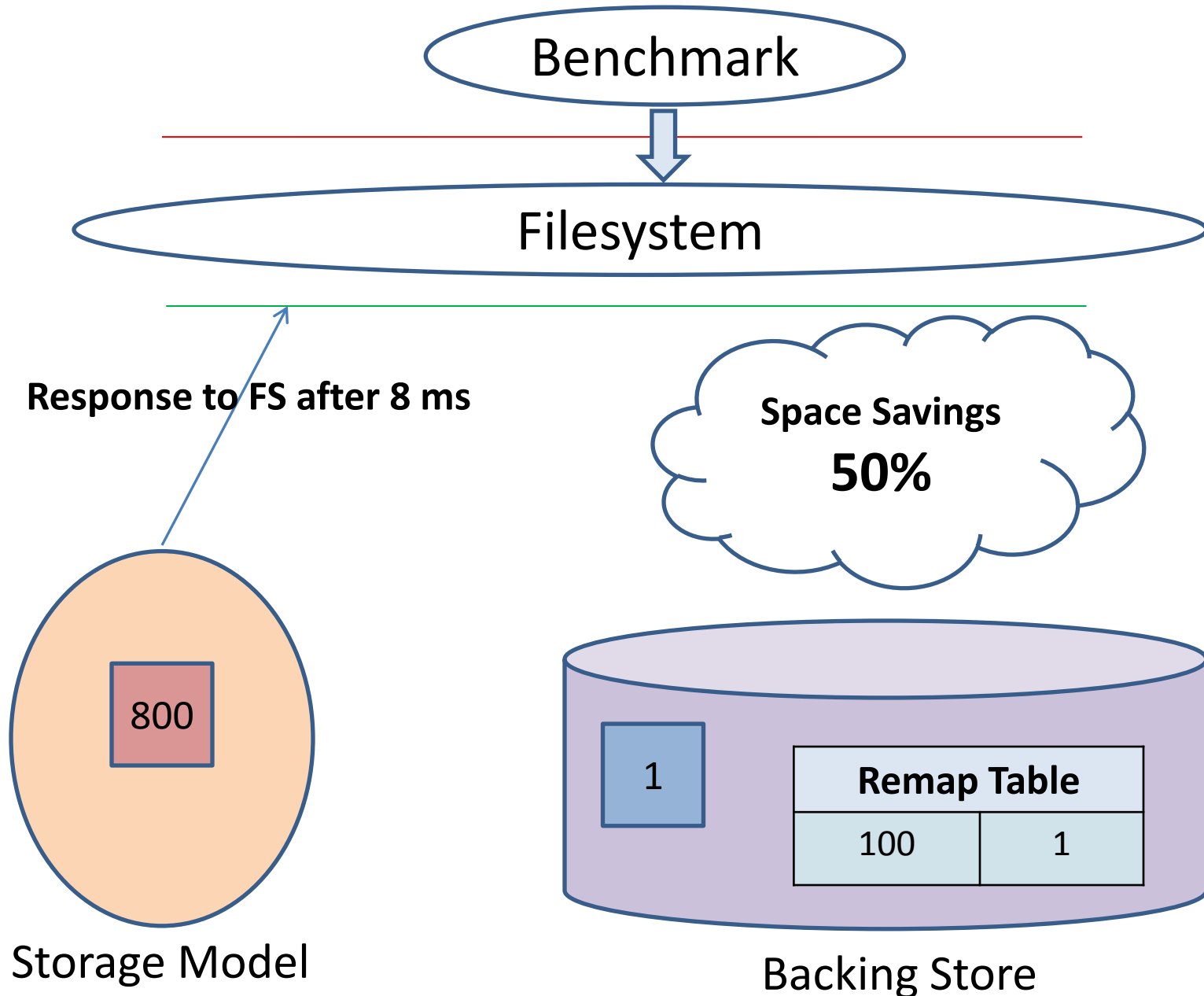
How does David handle **data write**?



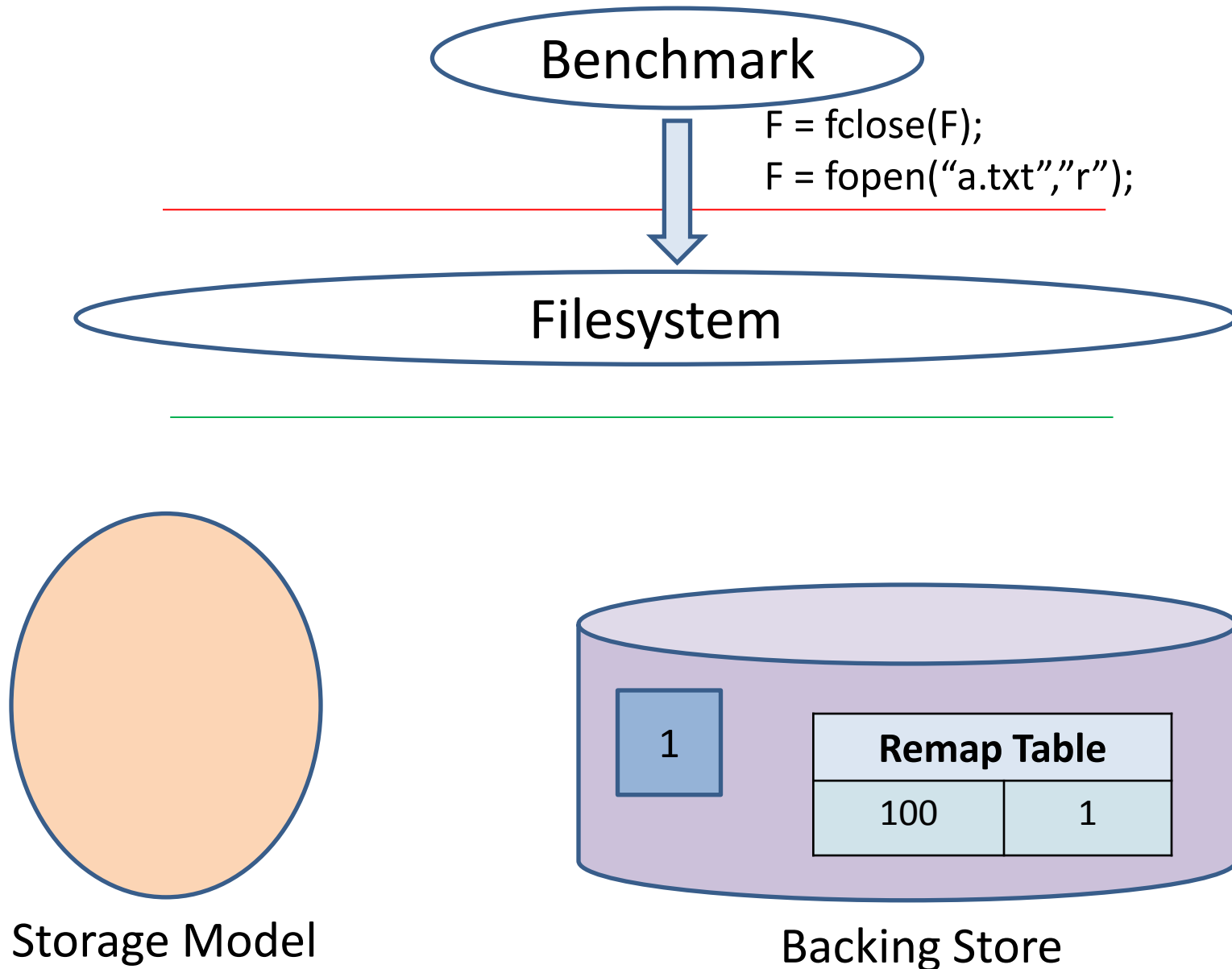
How does David handle **data write**?



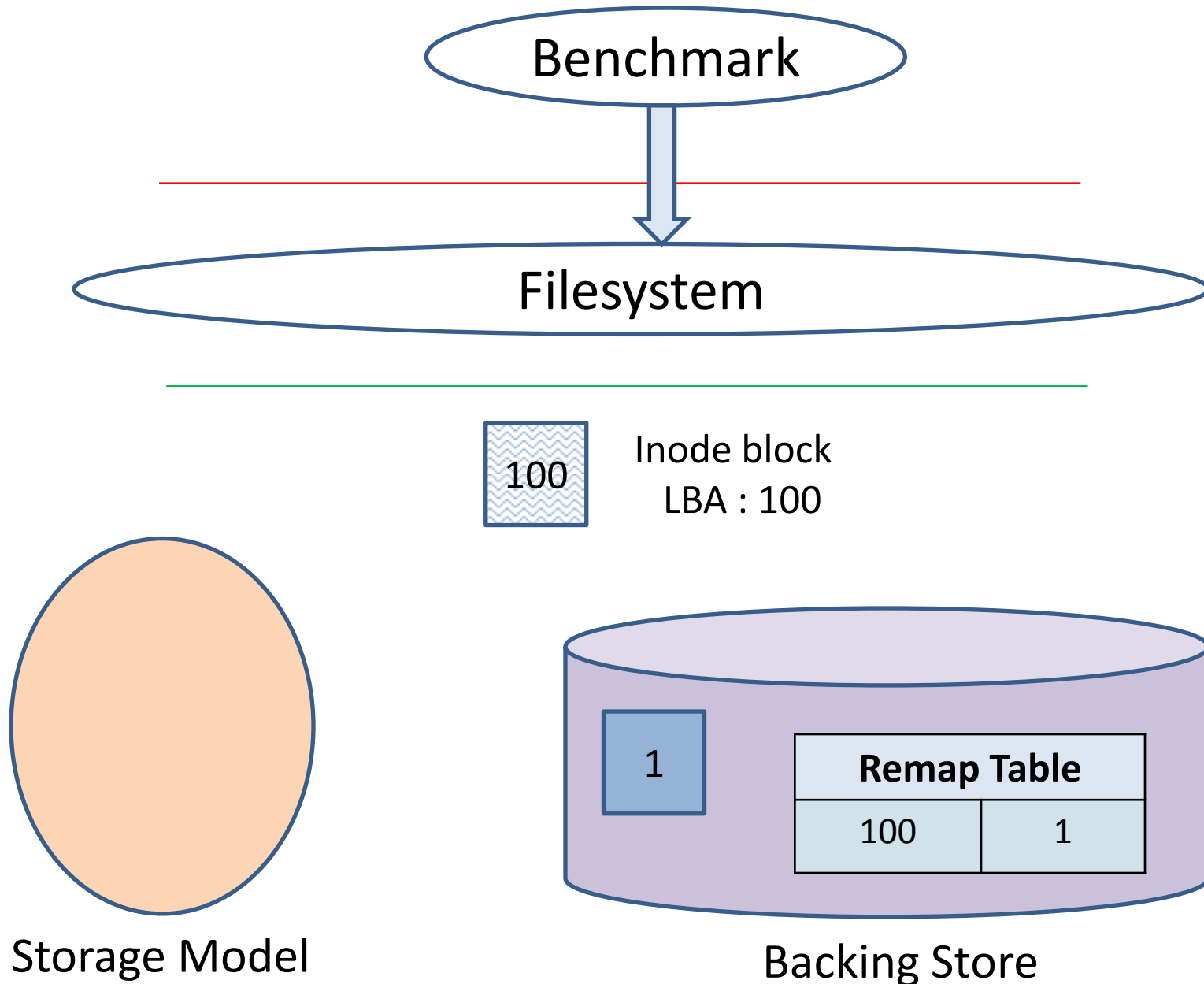
How does David handle **data write**?



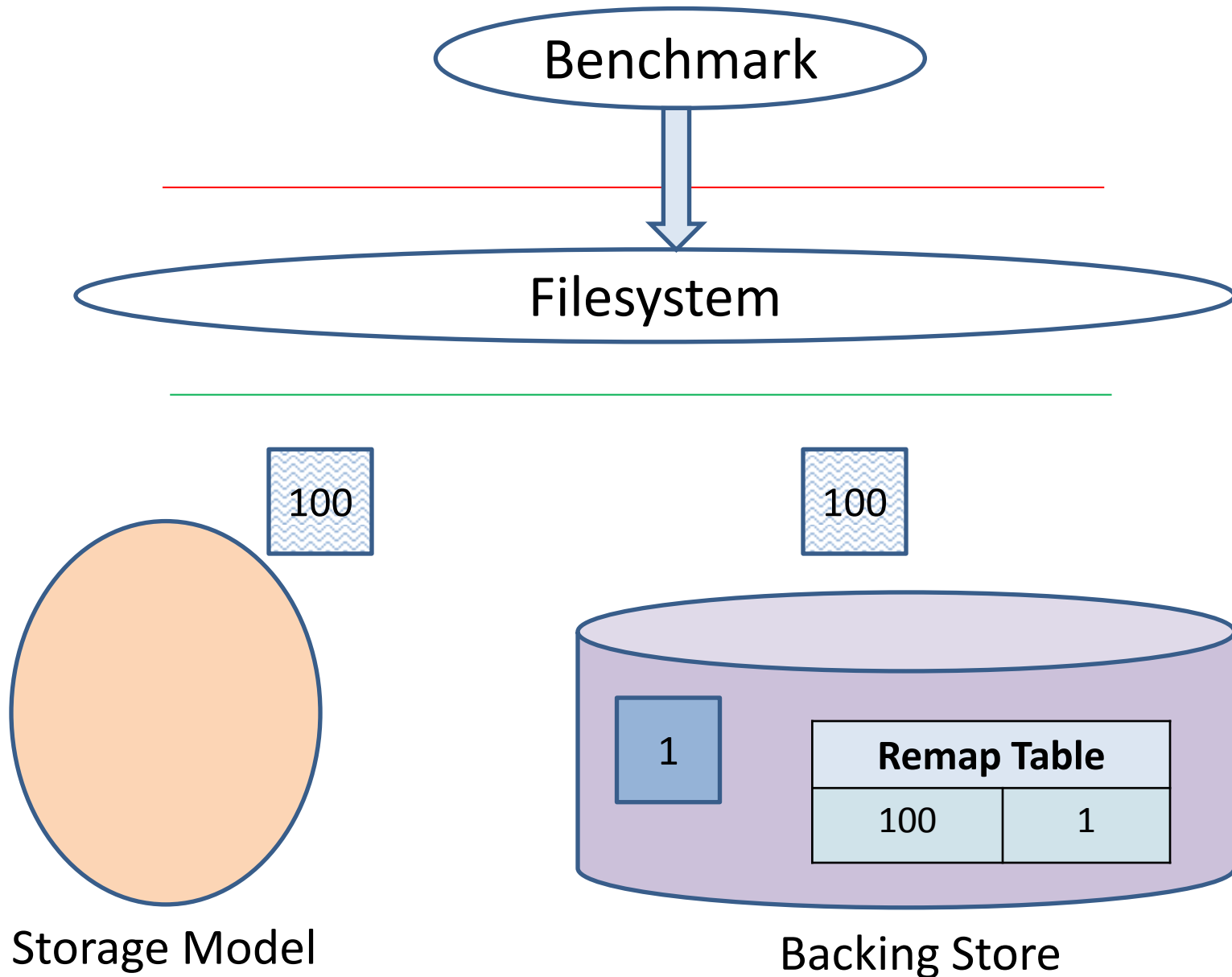
How does David handle **metadata read**?



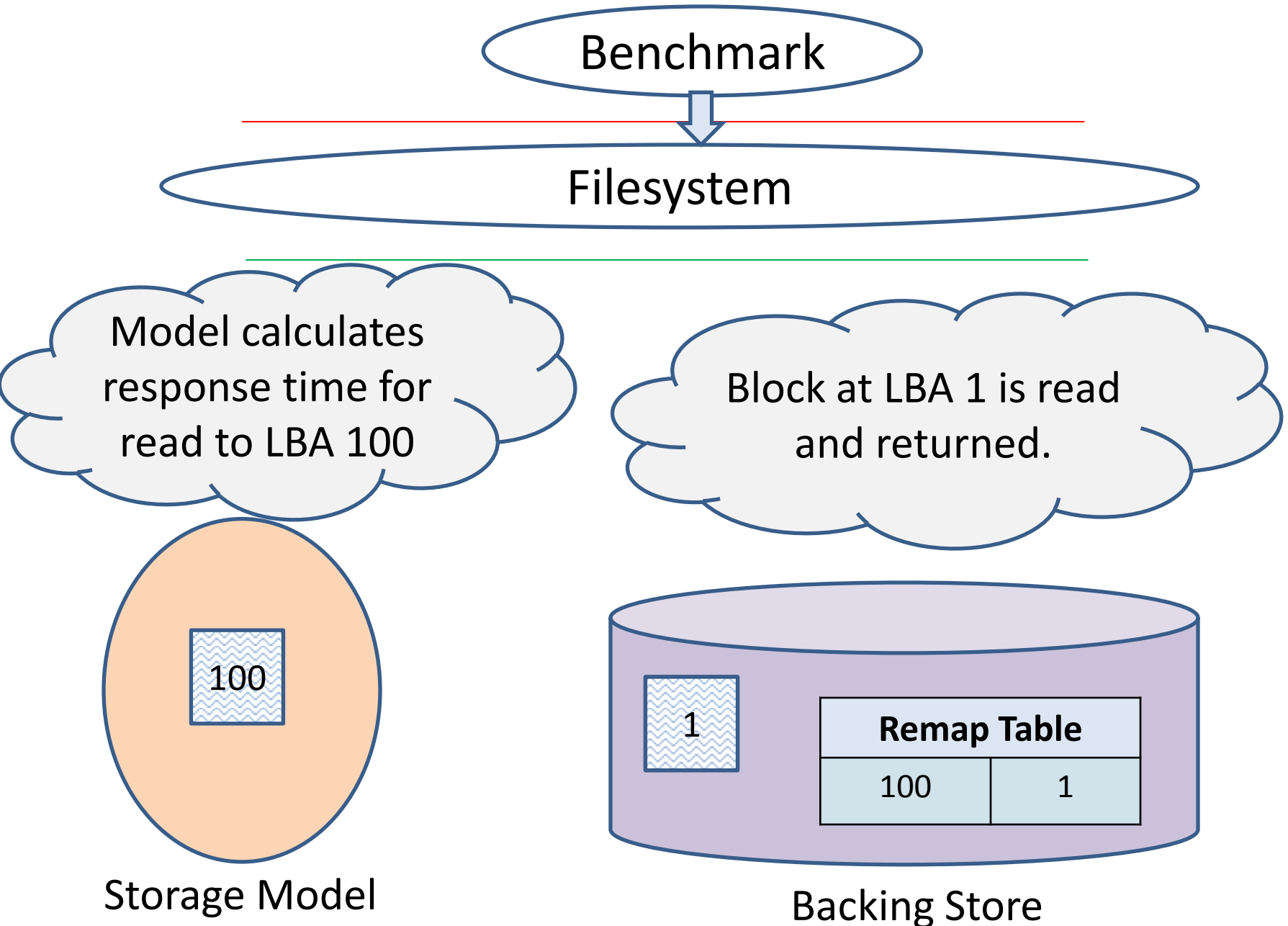
How does David handle **metadata read**?



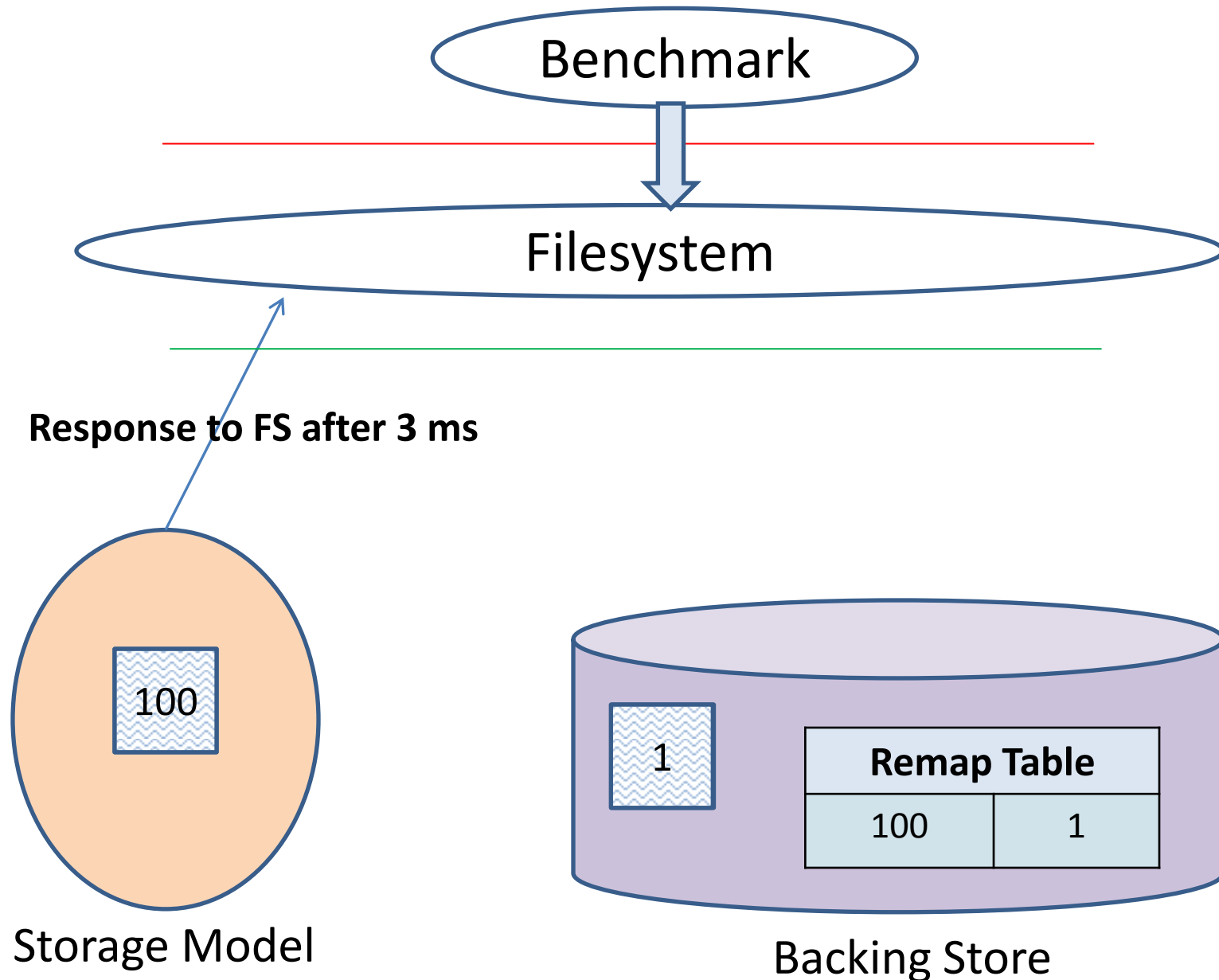
How does David handle **metadata read**?



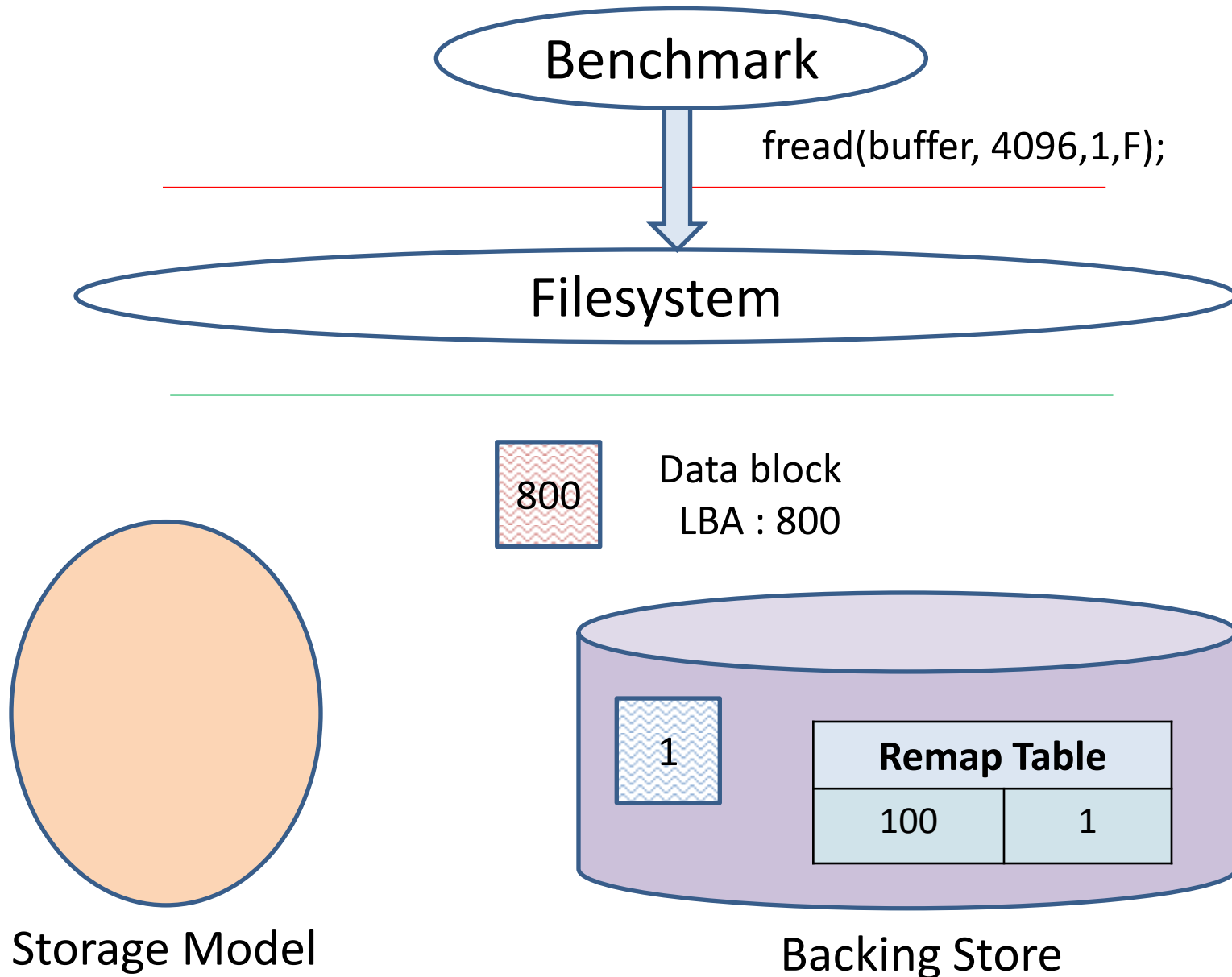
How does David handle **metadata read**?



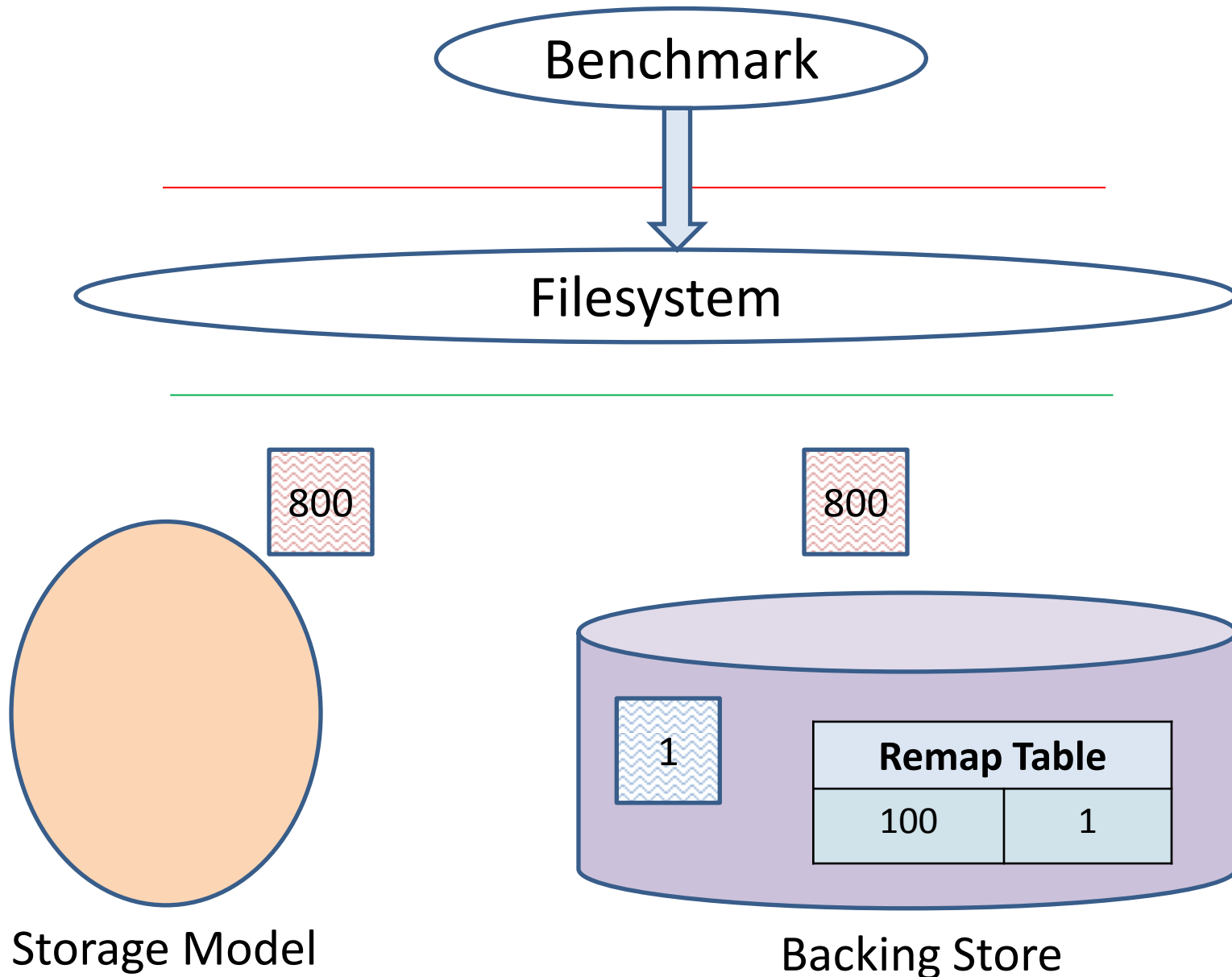
How does David handle **metadata read**?



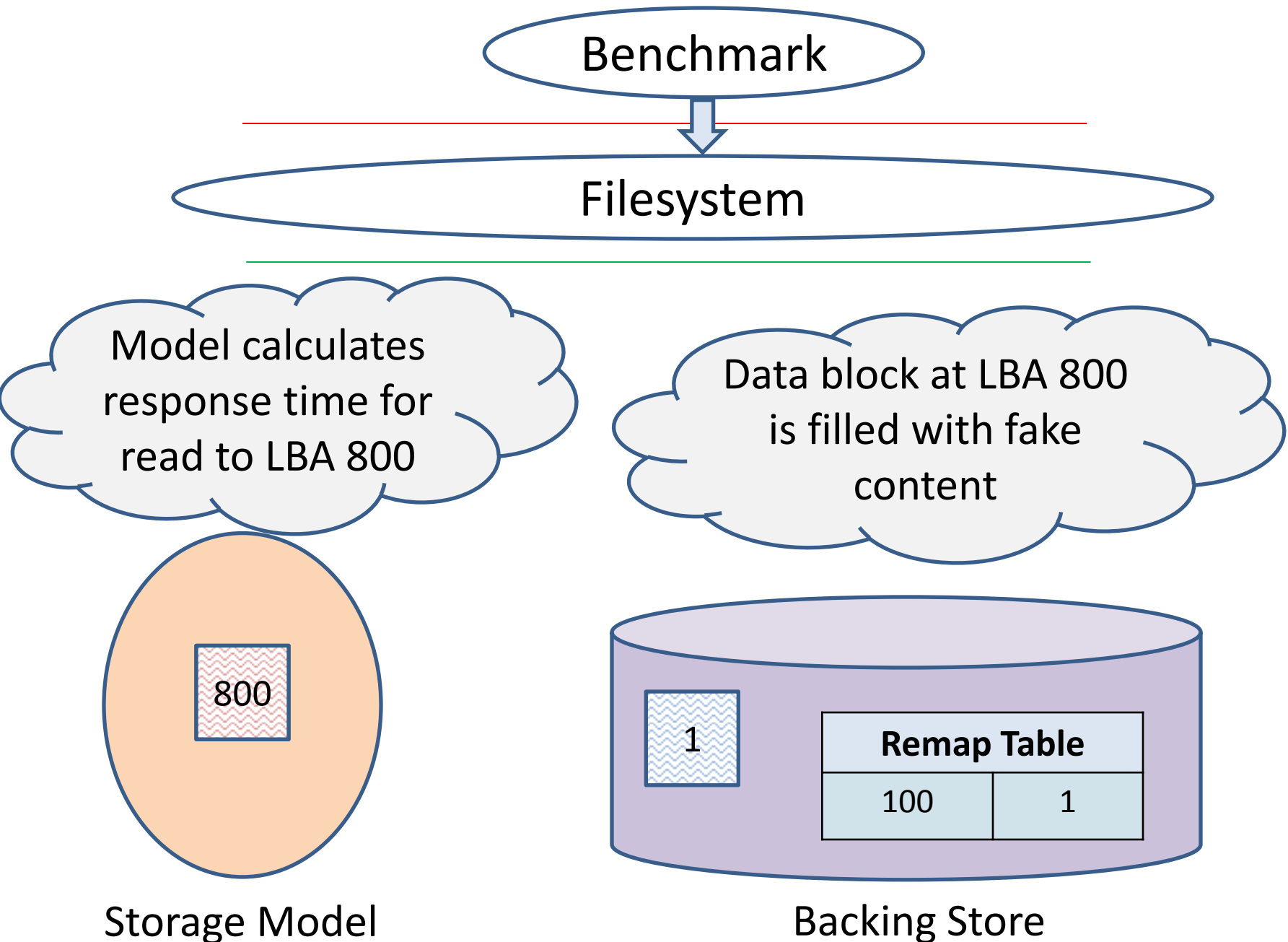
How does David handle **data read**?



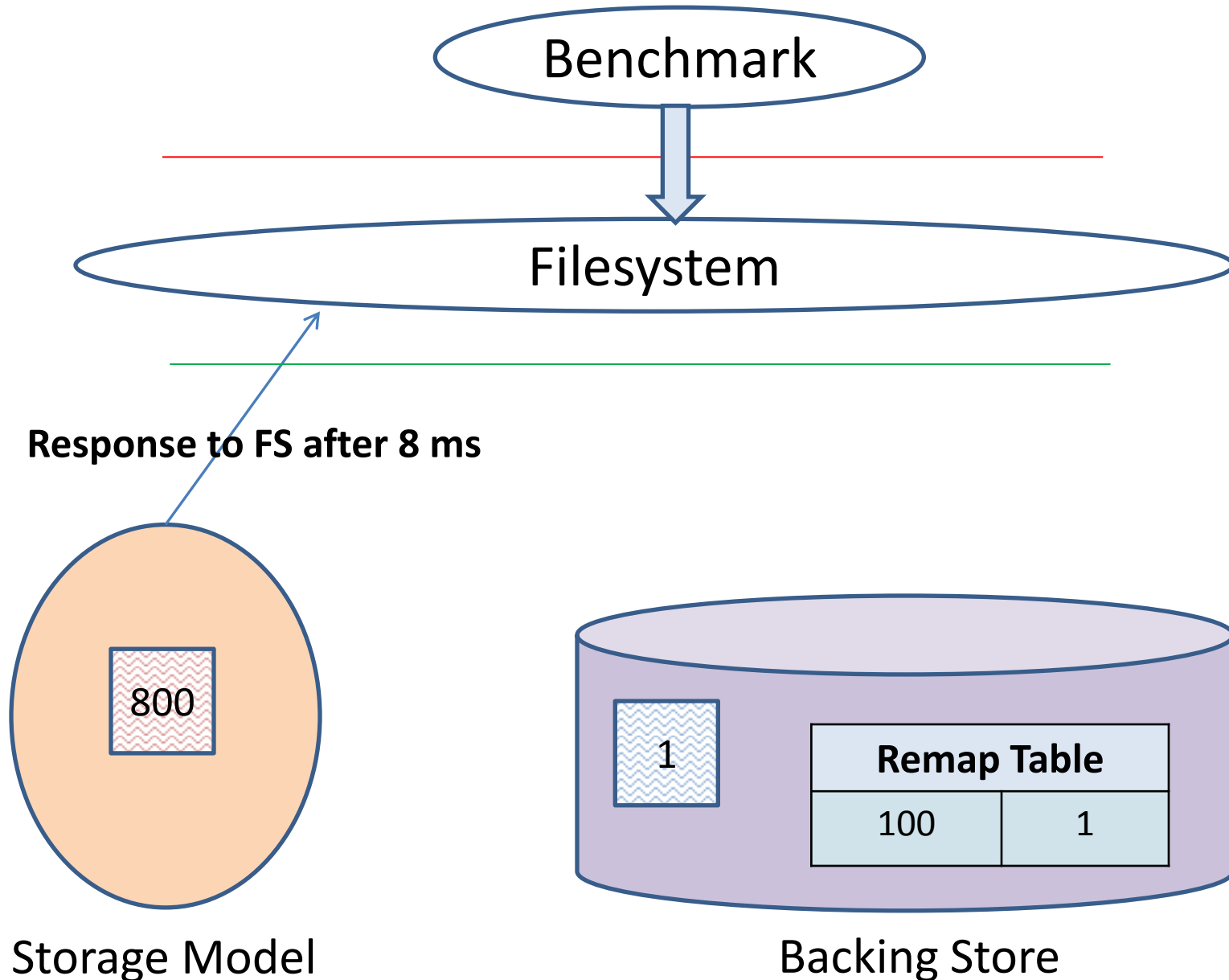
How does David handle **data read**?



How does David handle **data read**?



How does David handle **data read**?



Outline

~~Intro~~

~~Overview~~

Design

Results

Conclusion

Design Goals for David

Accurate

Emulated disk should perform similar to real disk

Scalable

Should be able to emulate large disks

Lightweight

Emulation overhead should not affect accuracy

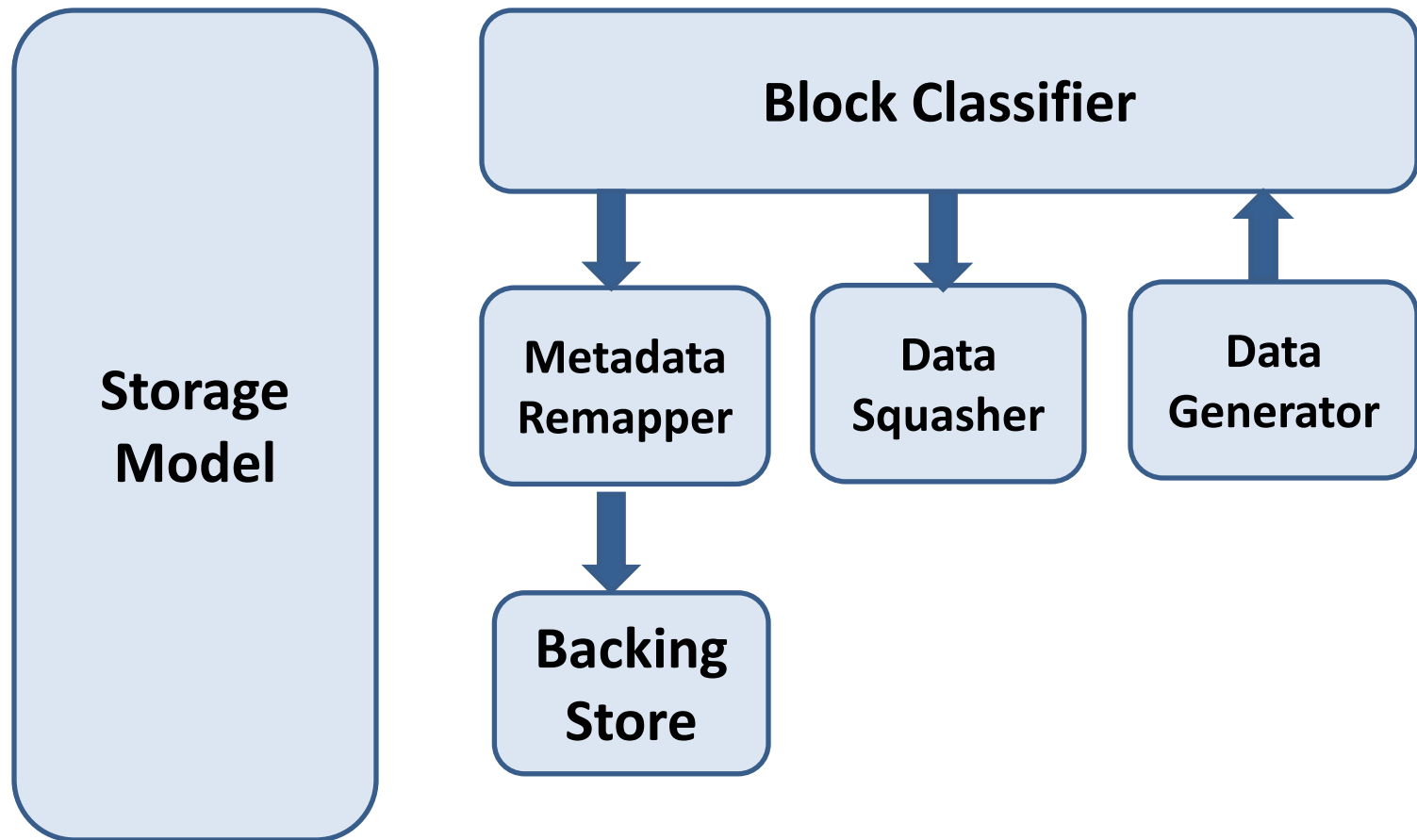
Flexible

Should be able to emulate variety of storage disks

Adoptable

Easy to install and use for benchmarking

Components within David



Block Classification

Data or Metadata?

Distinguish data blocks from metadata blocks
to throw away data blocks

Why difficult?

David is a block-level emulator

Two Approaches

**Implicit Block
Classification**

**(David automatically
infers block
classification)**

**Explicit Block
Classification**

**(Operating System
passes down block
classification)**

Implicit Block Classification

Parse metadata writes using filesystem knowledge to infer data blocks

Implementation for ext3

- Identify inode blocks using ext3 block layout
- Parse inode blocks to infer direct/indirect blocks
- Parse direct/indirect blocks to infer data blocks

Problem

Delay in classification

Ext3 Ordered Journaling Mode (without David)

M

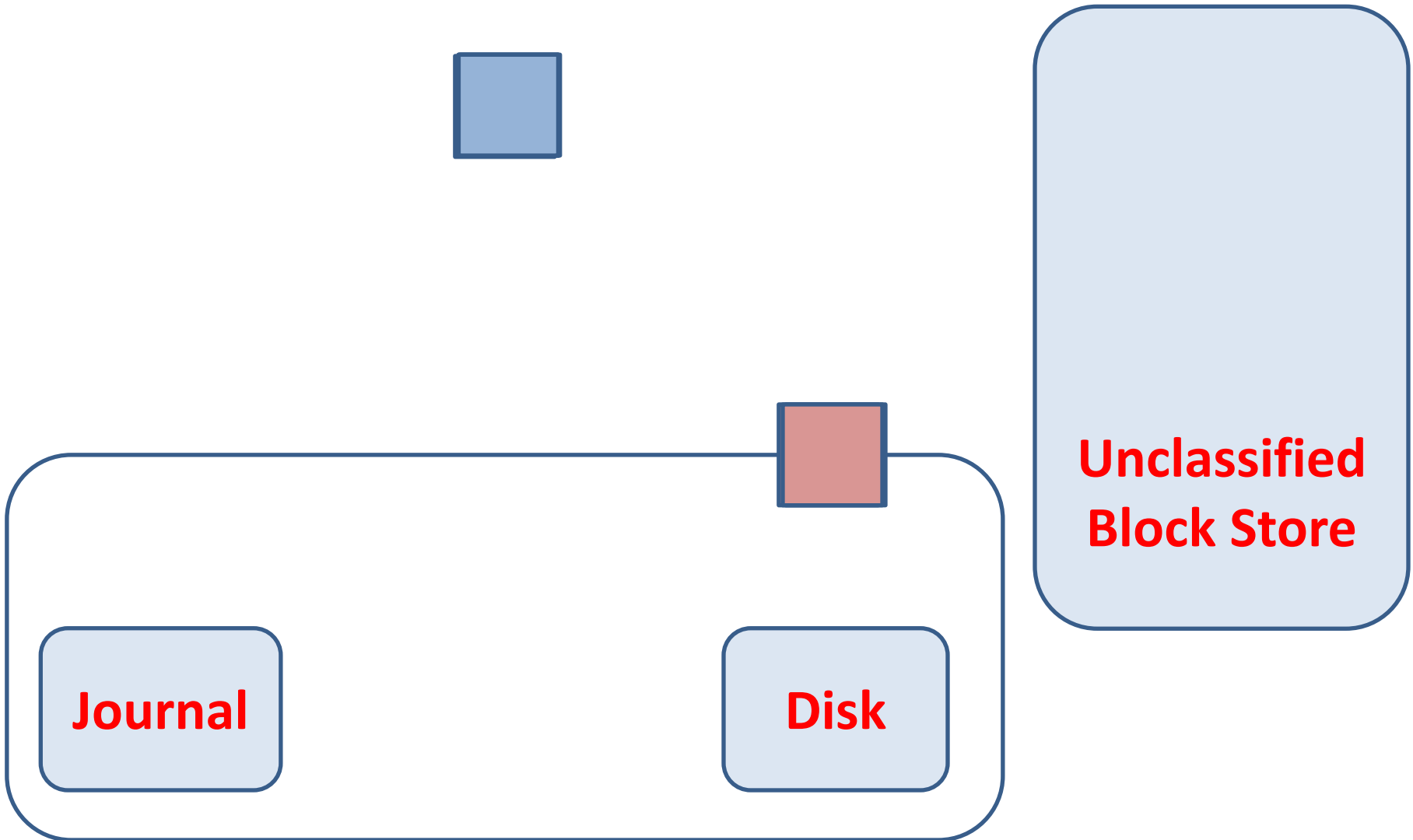
D



Journal

Disk

Ext3 Ordered Journaling Mode (with David)



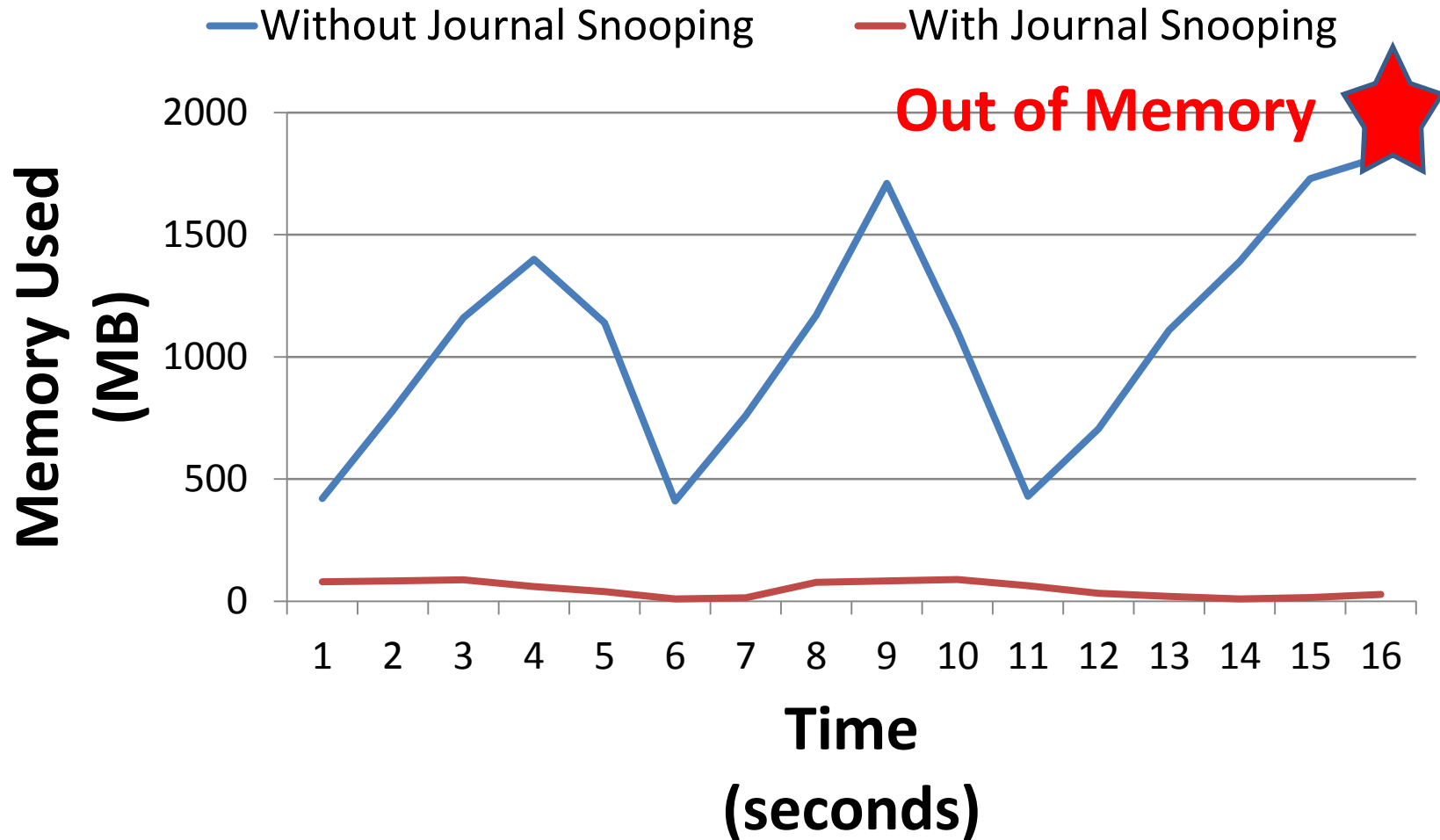
Memory Pressure in Unclassified Block Store

Too many unclassified blocks exhaust memory

Technique: Journal Snooping

Parse metadata writes to journal to infer classification much earlier than usual

Effect of Journal Snooping



Block Classification

Data or Metadata?

Distinguish data blocks from metadata blocks
to throw away data blocks

Why difficult?

David is a block-level emulator

Two Approaches

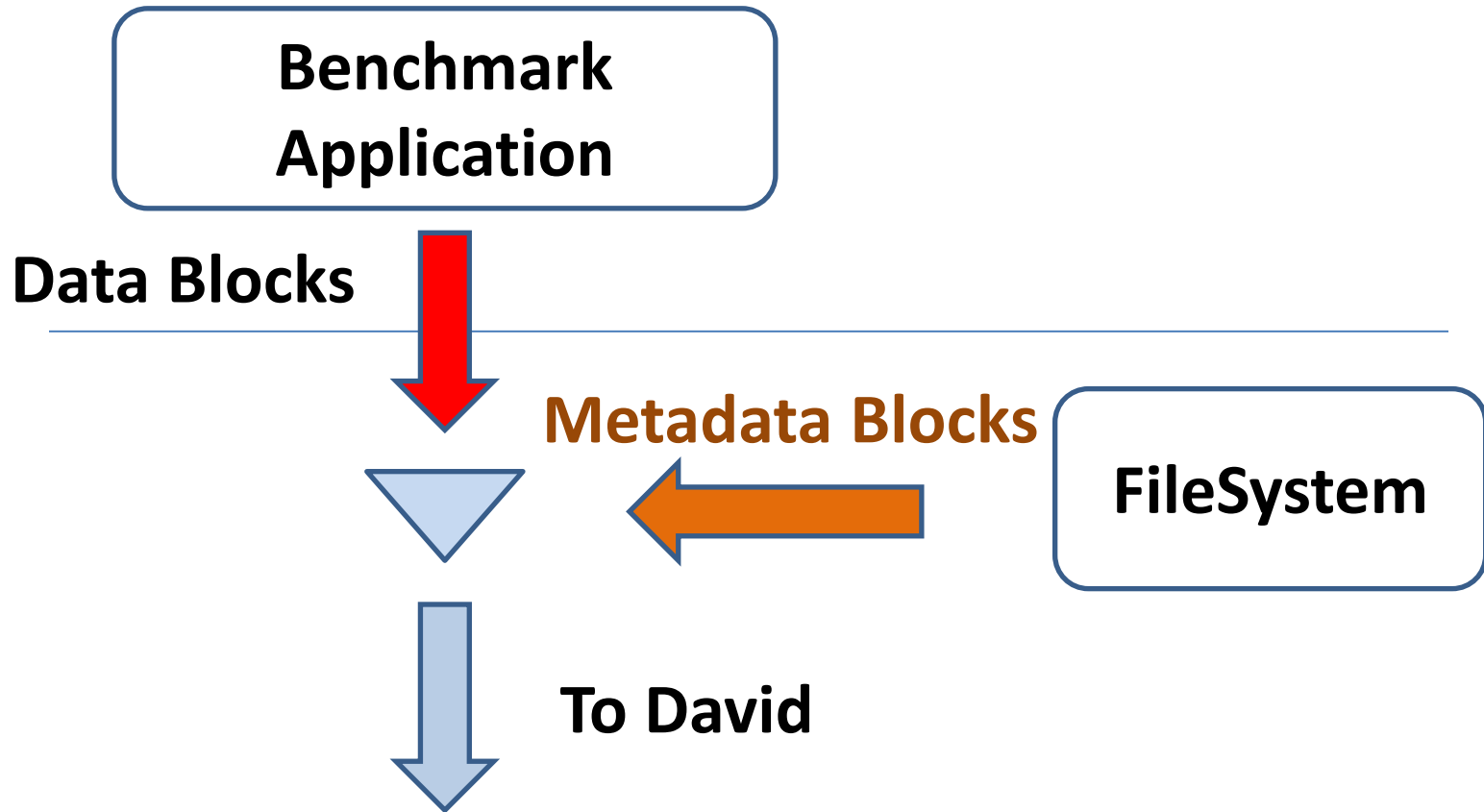
**Implicit Block
Classification**

**(David automatically
infers block
classification)**

**Explicit Block
Classification**

**(Operating System
passes down block
classification)**

Explicit Block Classification

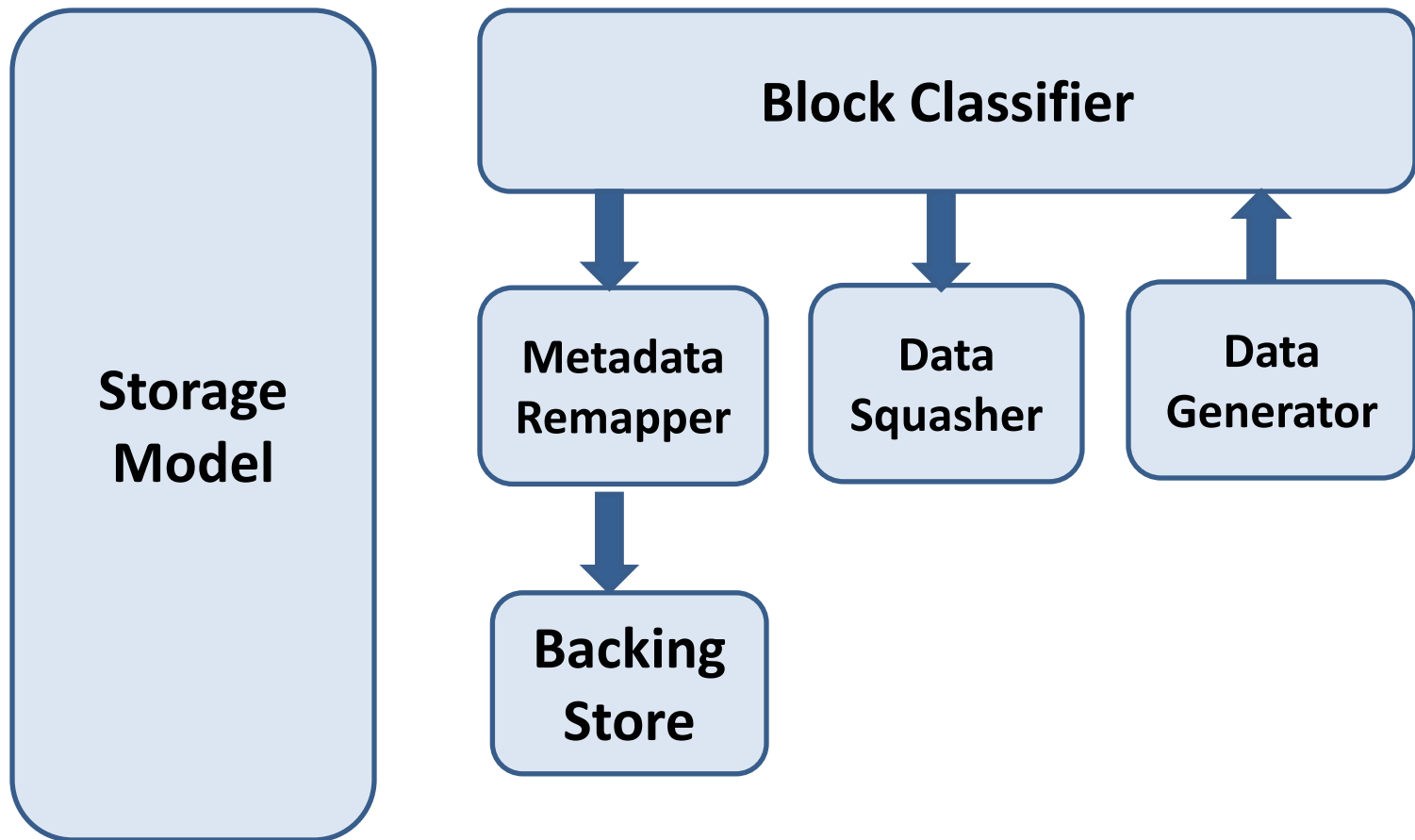


Capture page pointers to data blocks in the *write system call* and pass classification information to David

Block Classification Summary

Implicit Block Classification	Explicit Block Classification
No change to filesystem, benchmark or operating system	Minimal change to operating system
Requires filesystem knowledge	Works for all filesystems
Results with ext3	Results with btrfs

Components within David



David's Storage Model

Actual System

Benchmark

Filesystem

I/O request
queue

Disk

**Storage
Model**

Emulated System

Benchmark

Filesystem

David



I/O Queue Model

Merge sequential I/O requests

- **To improve performance**

When I/O queue is empty

- **Wait for 3 ms anticipating merges**

When I/O queue is full

- **Process is made to sleep and wait**
- **Process is woken up once empty slots open up**
- **Process is given a bonus for the wait period**

I/O queue modeling critical for accuracy

Disk Model

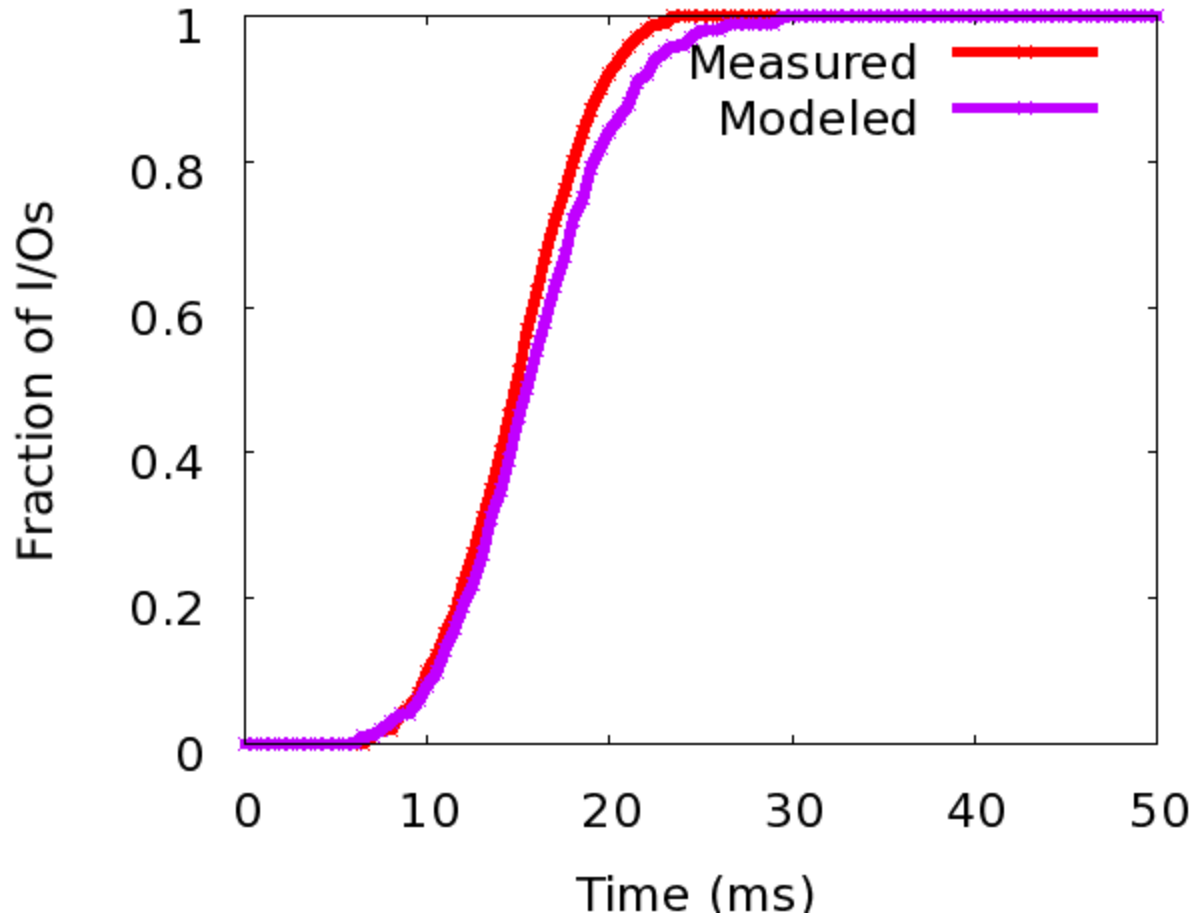
Simple in-kernel disk model

- Based on **Ruemmler and Wilkes** disk model
- Current models: 80GB and 1 TB Hitachi deskstar
- Focus of our work is not disk modeling
(more accurate models are possible)

Disk model parameters

- **Disk properties**
Rotational speed, head seek profile, etc.
- **Current disk state**
Head position, on-disk cache state, etc.

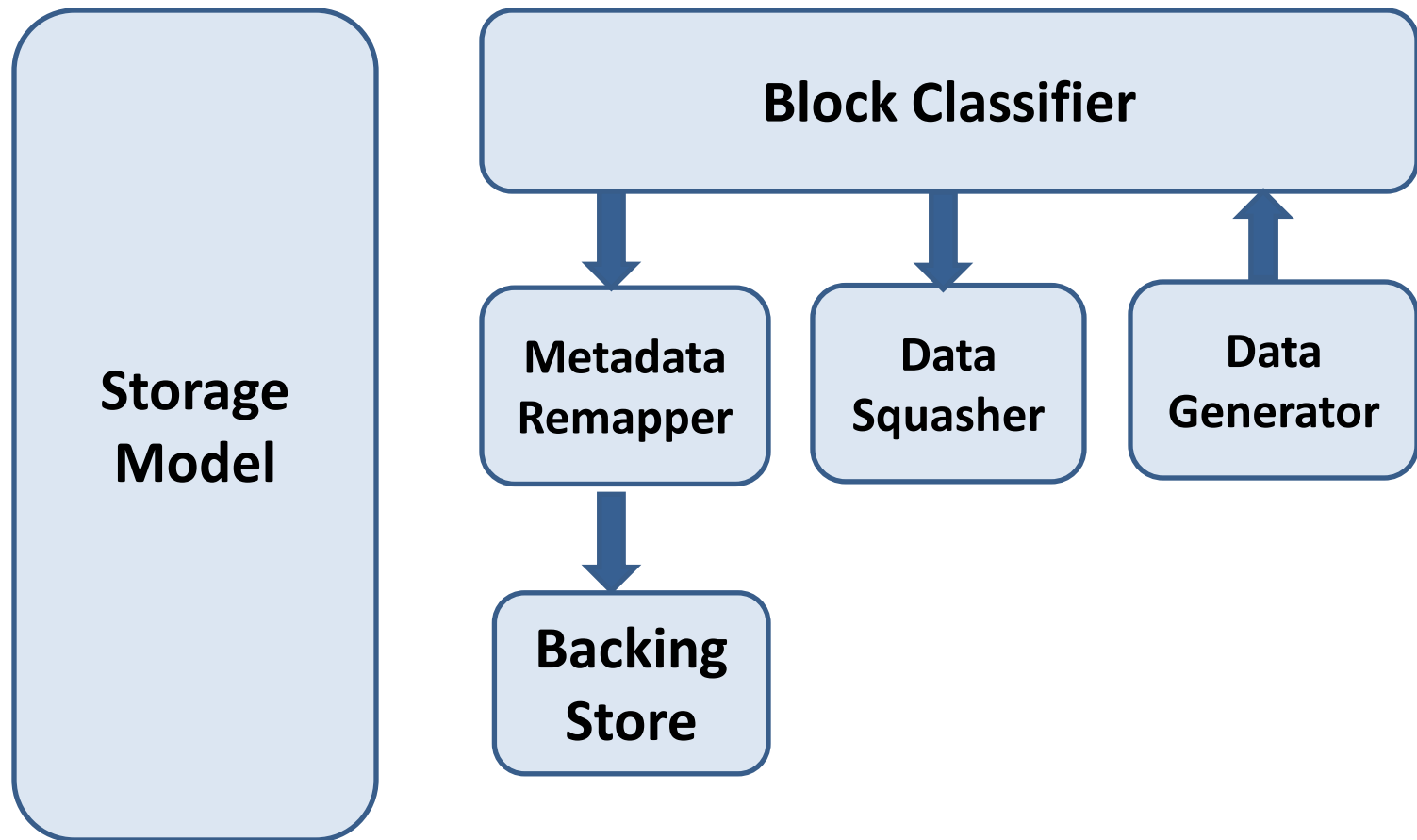
David's Storage Model Accuracy



Reasonable accuracy across many workloads

Many more results in paper

Components within David



Backing Store

Storage space for metadata blocks

Any physical storage can be used

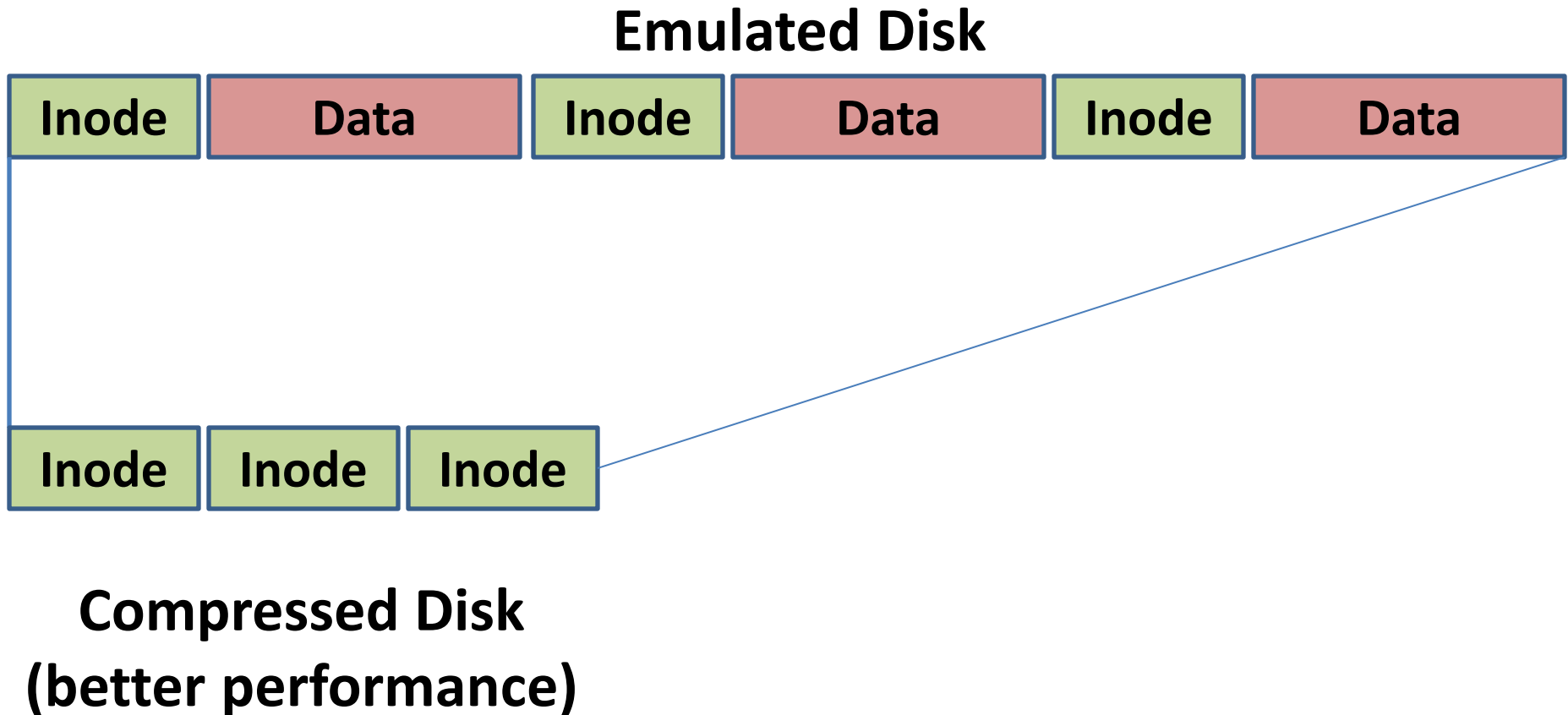
- Must be **large enough** to hold all metadata blocks
- Must be **fast enough** to match emulated disk

Two implementations

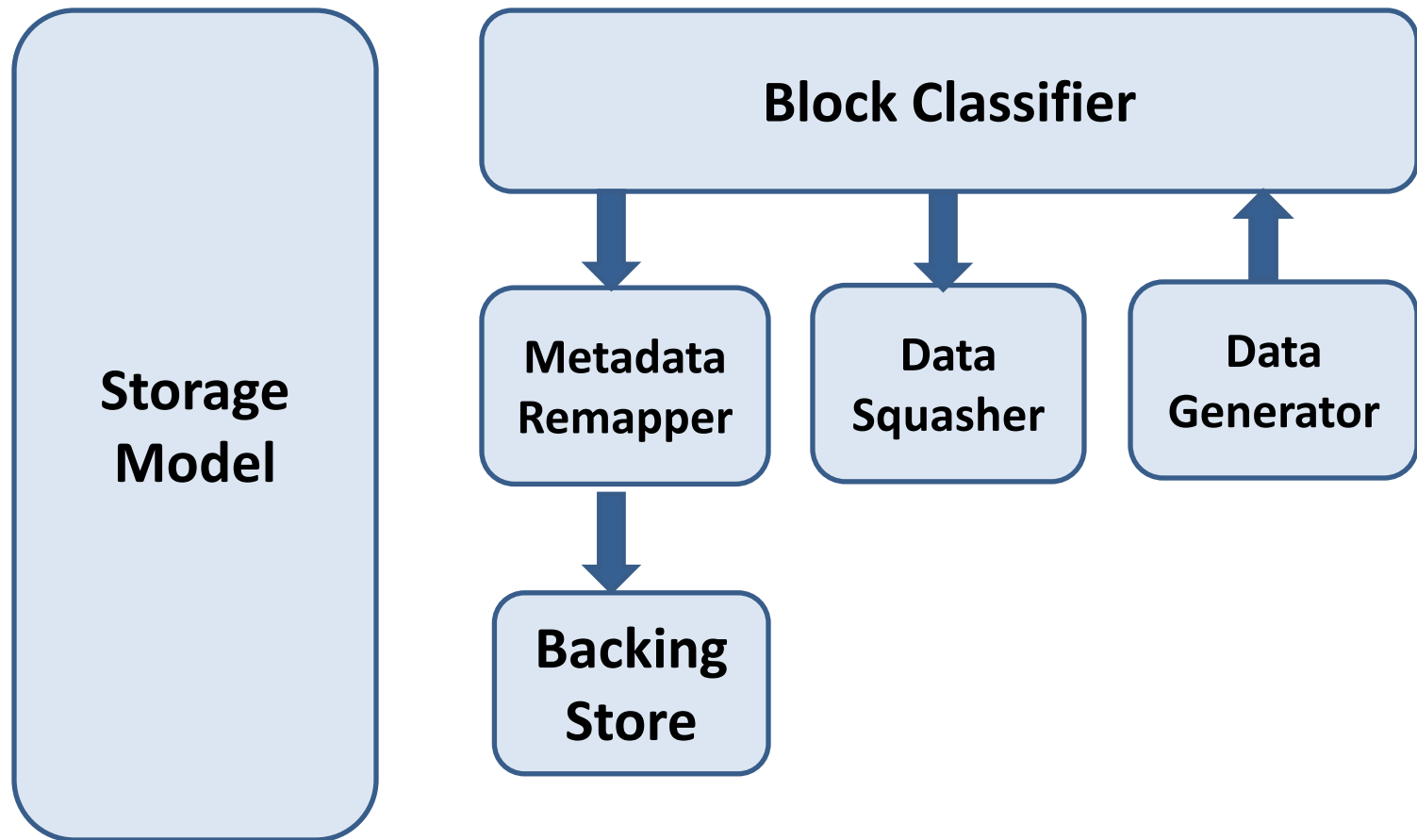
- Memory as backing store
- **Compressed disk** as backing store

Metadata Remapper

Remaps metadata blocks into compressed form



Components within David



Data Squasher and Generator

Data Squasher

Throws away writes to data blocks

Data Generator

Generate content for the reads to data blocks
(currently generates random content)

Outline

~~Intro~~

~~Overview~~

~~Design~~

Results

Conclusion

Experiments

Emulation accuracy

Test emulation accuracy across benchmarks

Emulation scalability

Test space savings for large device emulation

Multiple disk emulation

Test accuracy of multiple device emulation

Emulation Accuracy Experiment

Experimental details

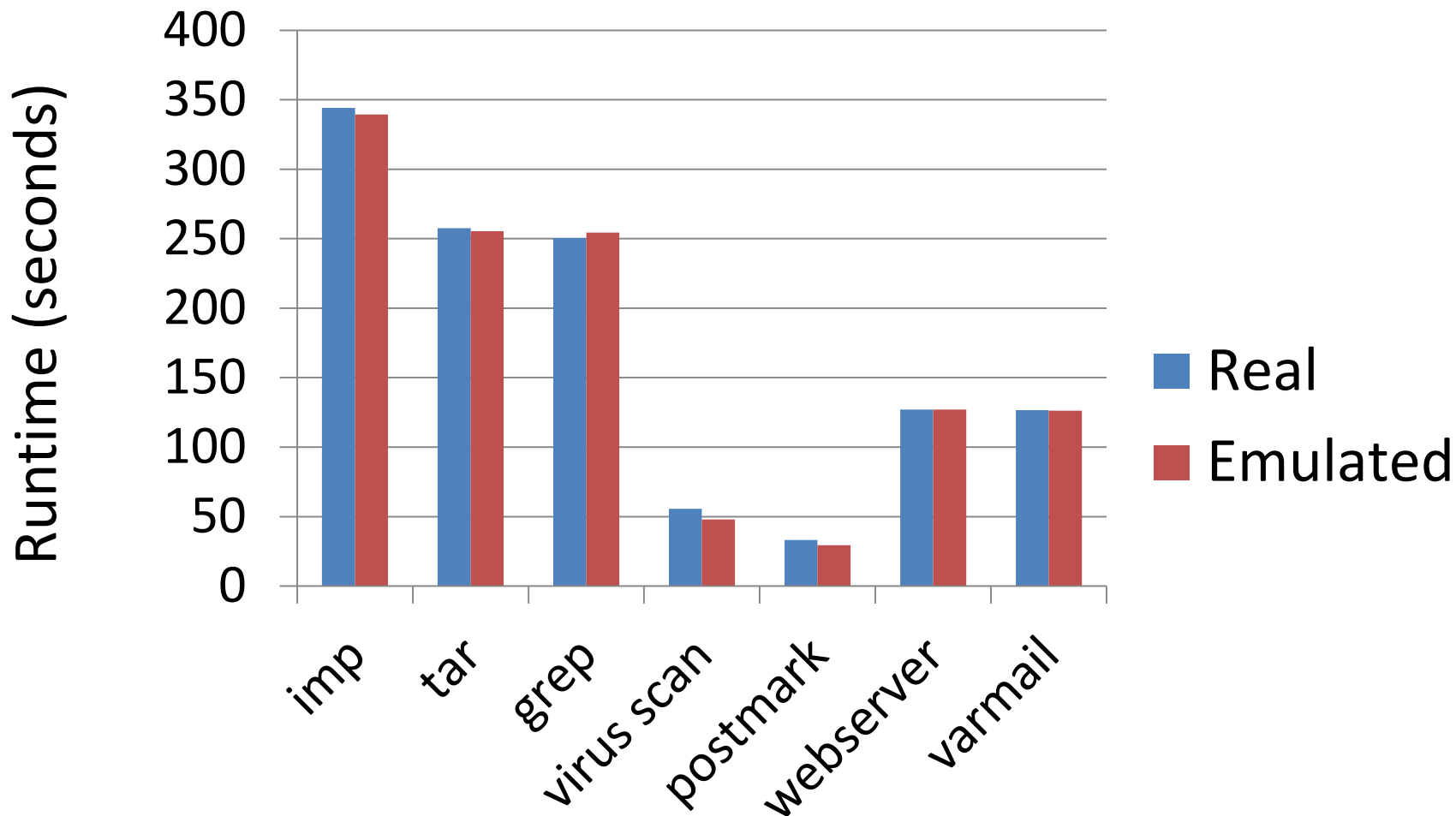
Emulated ~1 TB disk with 80 GB disk

Ran a variety of benchmarks

Validated by using a real 1 TB disk

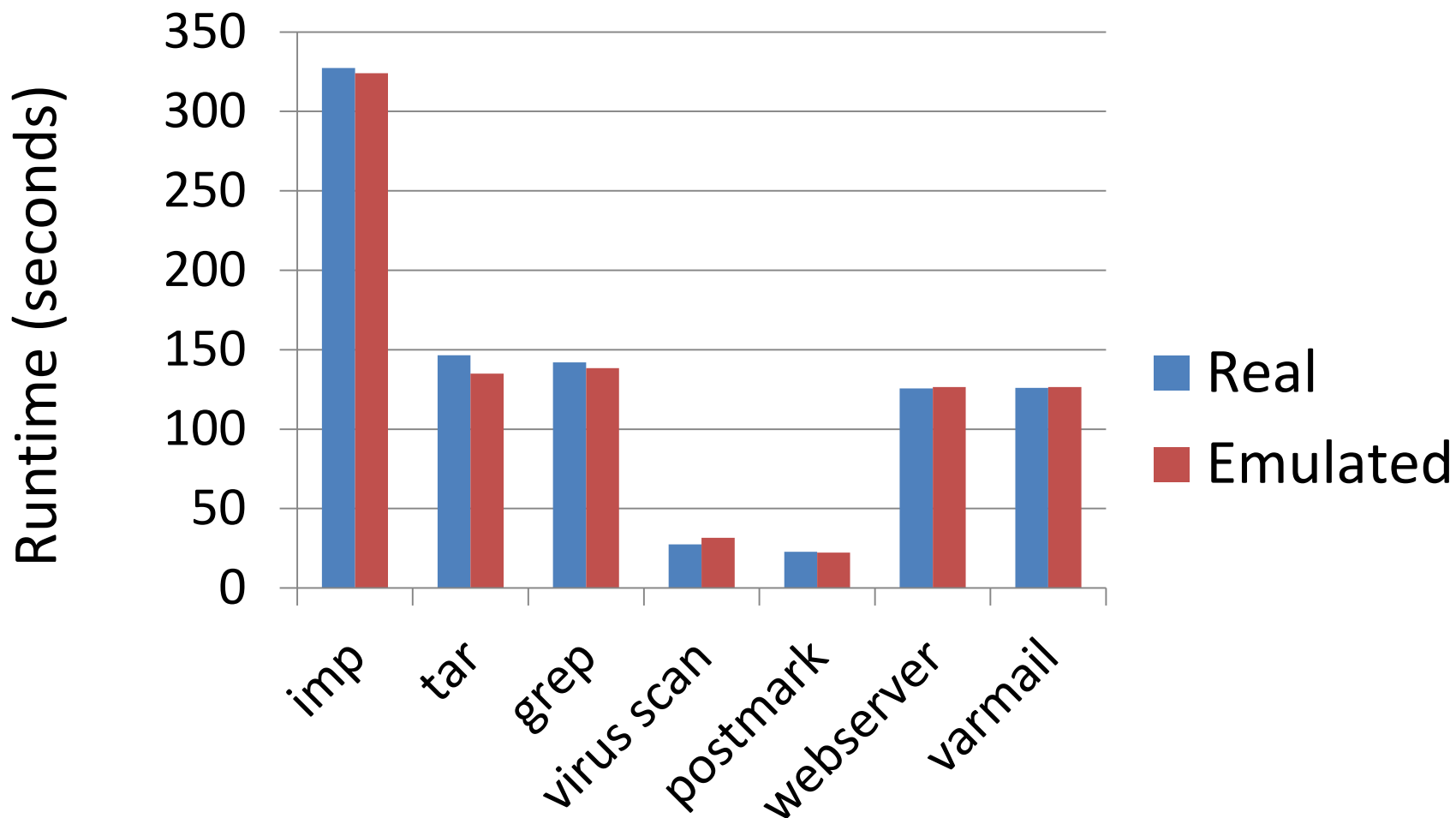
Emulation Accuracy Results

(Ext3 with Implicit Block Classification)



Emulation Accuracy Results

(Btrfs with Explicit Block Classification)

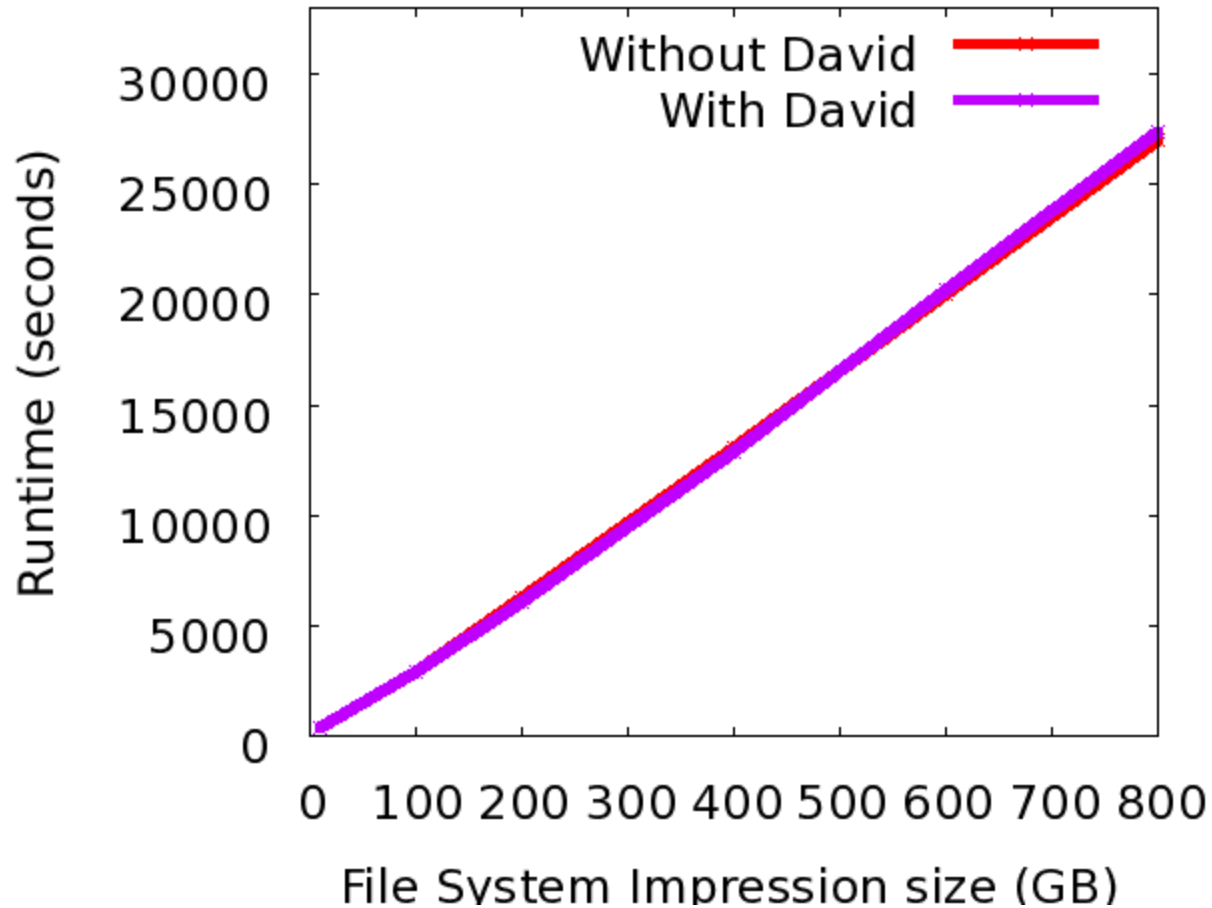


Emulation Scale Experiment

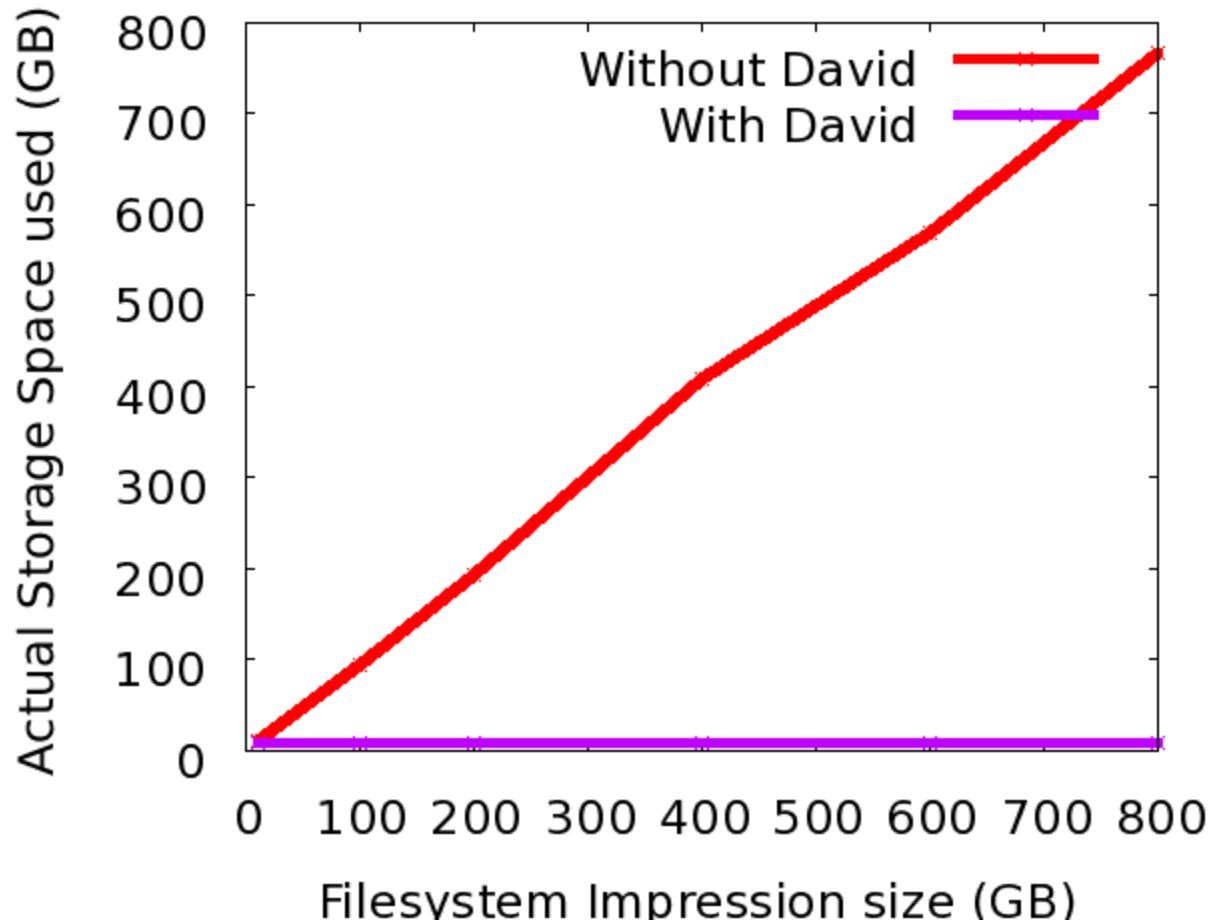
Experimental details

Emulated ~1 TB disk using a 80 GB disk
Created filesystem images using Impressions
Validated by using a real disk

Emulation Scale: Accuracy



Emulation Scale: Space Savings



Multiple Disks Experiment

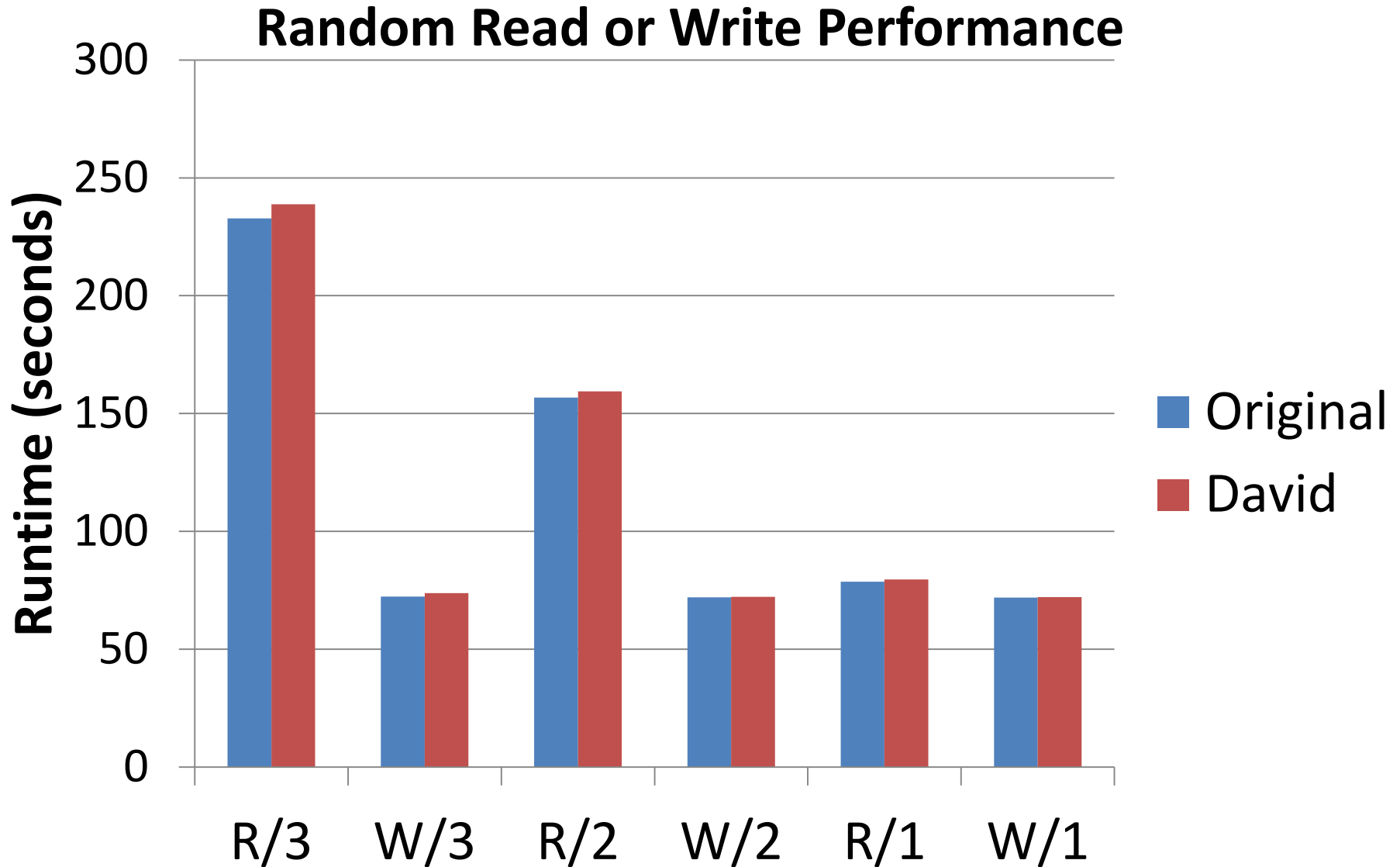
Experimental details

Emulated multiple disks using RAM

Measured micro-benchmark performance on RAID-1

Validated our results against real disks

Simple RAID-1 Emulation



Outline

~~Intro~~

~~Overview~~

~~Design~~

~~Results~~

Conclusion

Conclusion

David:

Emulate large devices with limited means

Key idea:

Throw away data

Results:

Accurate emulation of large and multiple disks

Future:

Emulating storage cluster with few machines

Thank You

www.cs.wisc.edu/adsl

Questions?

Measuring Innovation

*Thorough measurement
is Hard and Costly*

**Time, Money, Effort needed
to measure performance on a variety
of storage devices**

Tiny benchmarks are easy to run

Implicit Block Classification

Unclassified block store

- Unclassifiable blocks are temporarily stored in Unclassified Block Store which is in RAM
- Journal checkpoint frequency determines the delay in classification
- Upon classification, data blocks are squashed and metadata blocks are persisted