



ffsck: The Fast File System Checker

*+ **Ao Ma**, + Chris Dragga, + Andrea C. Arpaci-Dusseau, + Remzi H. Arpaci-Dusseau

+ University of Wisconsin, Madison

* Backup Recovery Systems Division, EMC

© Copyright 2013 EMC Corporation. All rights reserved.

Background

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

File system

- Data integrity is critically important
- Should be robust and reliable

Factors can corrupt FS

- Unclean shutdown / system crash
- File system bugs
- Hardware failures

© Copyright 2013 EMC Corporation. All rights reserved.

Existing Solutions



Unclean shutdown System crash	• Journaling • Copy-on-Write • Soft updates
Bugs	• Debugging • Bug finding tools
Hardware failures	• Checksums • Scrubbing

Reduce the probability of faults but can't protect against all of them

© Copyright 2013 EMC Corporation. All rights reserved.

fsck: Last Resort To Repair



Approach

- Scan FS offline
- Check metadata redundancy info
- Restore a damaged FS back to usable state

Dilemma

- Fsck is **slow**, causing **long downtime**
- **Unpredictable** checking time

© Copyright 2013 EMC Corporation. All rights reserved.

fsck Challenges



File system is evolving

- Capacity is increasing
- More complexity (more bugs and hardware failures)

Impact on fsck

- Longer downtime
- Use more frequently

© Copyright 2013 EMC Corporation. All rights reserved.

What we did

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Fast file system checker (ffsck)

- 10 times faster with identical checking policy

A modified version of ext3 (rext3)

- 20% improvement in big writes
- 43% improvement in random reads
- 10% degradation in small reads

© Copyright 2013 EMC Corporation. All rights reserved.

Outline

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

fsck analysis

- Checking approach
- Performance
- FS tradeoffs

ffsck & rext3

- Goals
- Novel features

Evaluation

© Copyright 2013 EMC Corporation. All rights reserved.

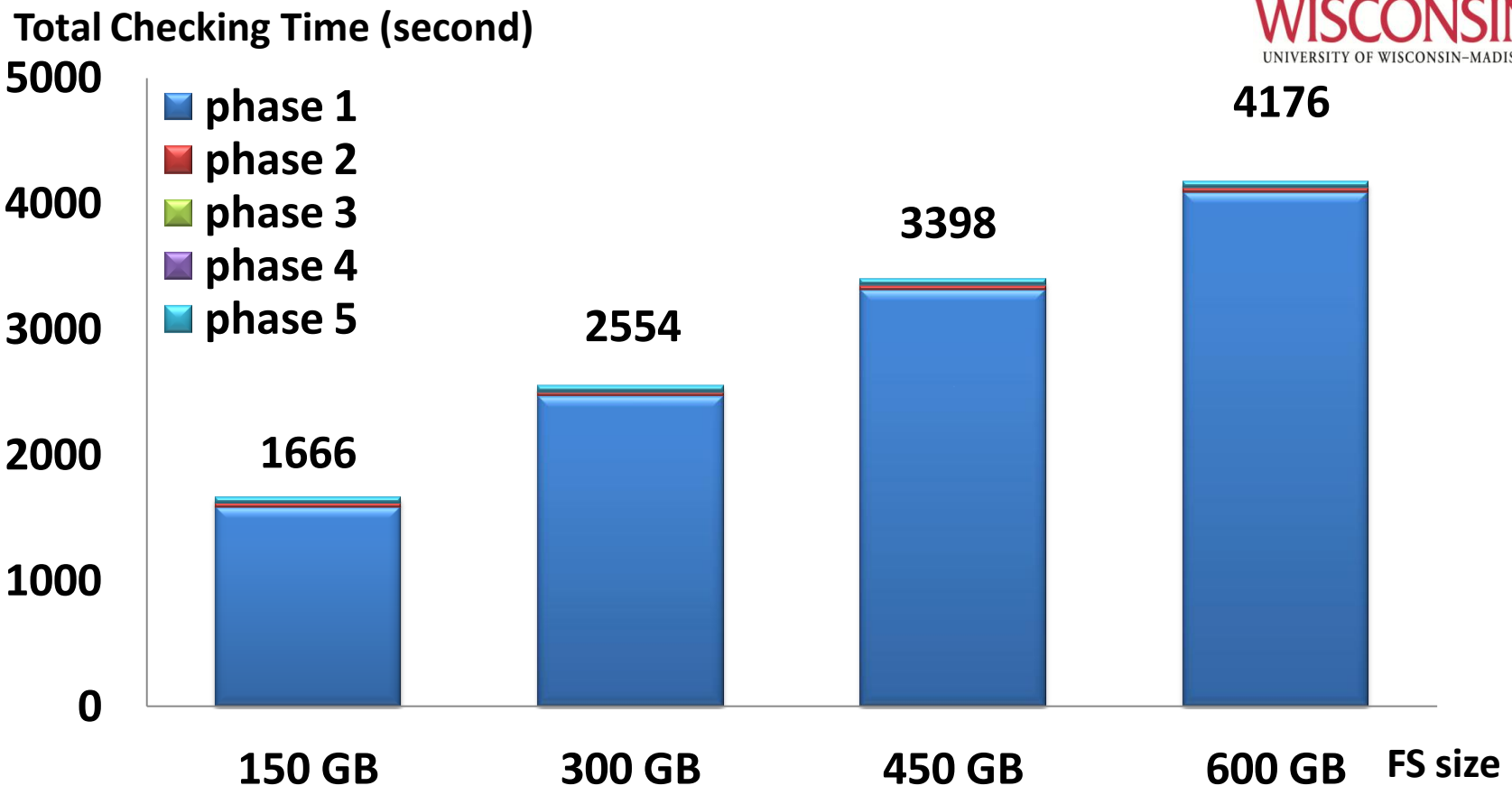
E2fsck Checking Approach



Phase	checking tasks
1	Scan inodes and indirect blocks in a logical order
2	Check each directory individually
3	Check directory connectivity
4	Check inode reference count and remove orphan inodes
5	Update the on-disk copies if necessary

© Copyright 2013 EMC Corporation. All rights reserved.

Performance Analysis



750GB disk, 1GB memory, e2fsprog 1.41.12, Linux 2.6.28

Initialize disk image by creating directories with small files (4KB- 2MB)

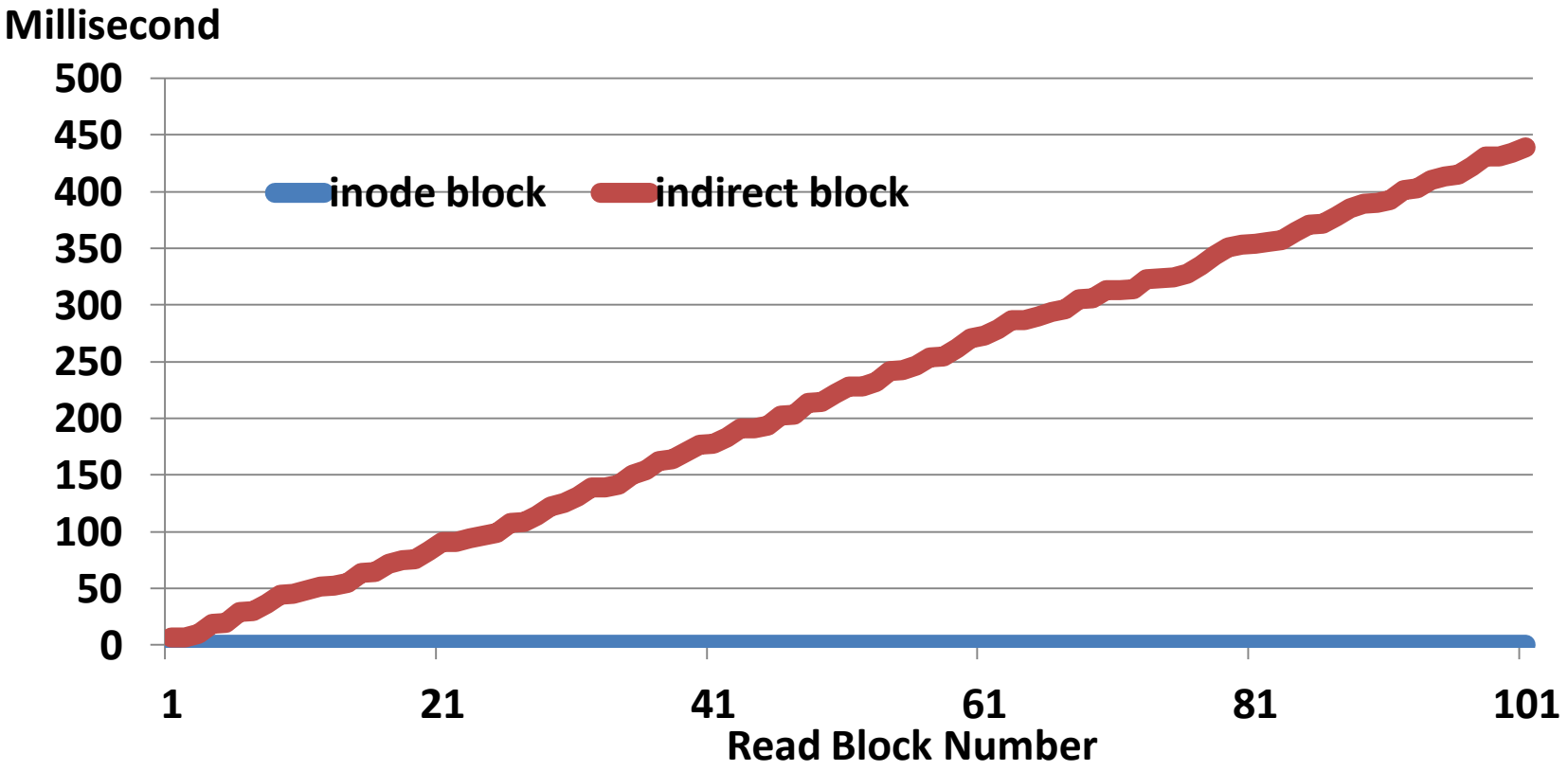
Increase FS size by creating new files and appending data to files (4KB-1MB)

E2fsck doesn't scale well

Phase 1 dominates the checking time

© Copyright 2013 EMC Corporation. All rights reserved.

Cumulative Time Spent on Reading Indirect Blocks and Inode Blocks



Indirect blocks are the bottleneck

© Copyright 2013 EMC Corporation. All rights reserved.

File System Design Tradeoffs



Identical allocation for data and indirect blocks

- Pros: store them contiguously and facilitate sequential access
- Cons: metadata scatters across disk

Rely on tree structure to locate indirect blocks

- Pros: simple and straightforward
- Cons: impose a strict ordering of access

© Copyright 2013 EMC Corporation. All rights reserved.

Fsck: An Afterthought Design



Repairing capability is not prioritized

- File system has limited support for checker
- Checker is developed as a peripheral addition, rather than a tight component

© Copyright 2013 EMC Corporation. All rights reserved.

Outline



fsck analysis

- Checking approach
- Performance
- FS tradeoffs

ffsck & rext3

- Goals
- Novel features

Evaluation

© Copyright 2013 EMC Corporation. All rights reserved.

Ffsck and Rext3



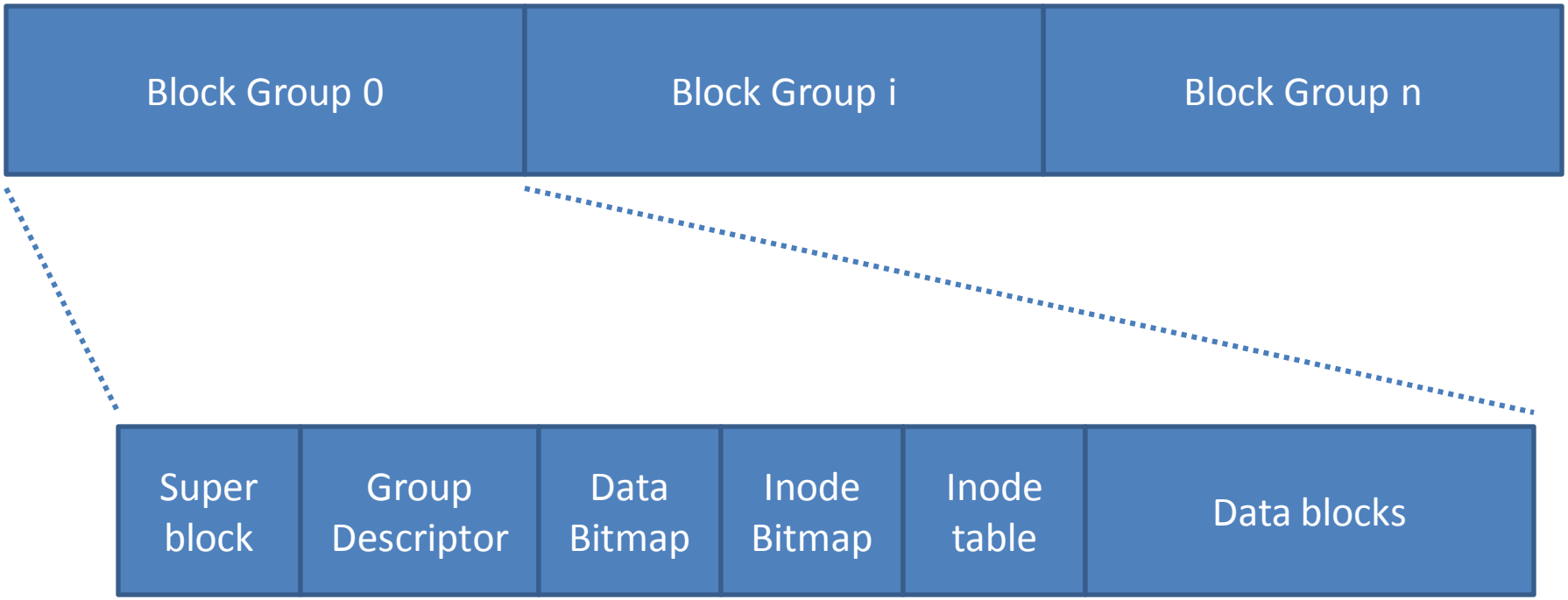
Prioritize fast repair when design FS

- Fast scan
- Robust checking performance
- Competitive FS performance

ffsck and rext3 is based on e2fsck and ext3

© Copyright 2013 EMC Corporation. All rights reserved.

Basic Layout of Ext3



Overview of ffsck and rext3



New disk layout

Disk-order scan

Self-check and Cross-check

Fast recovery with bitmap snapshot

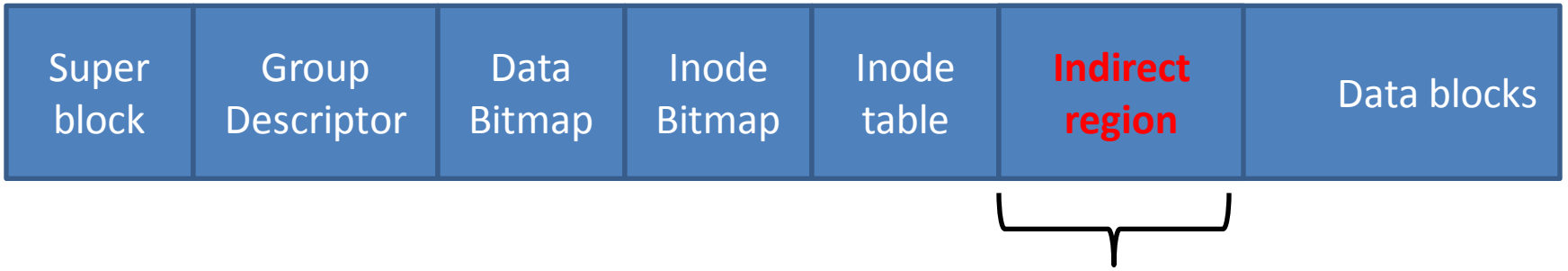
© Copyright 2013 EMC Corporation. All rights reserved.

Rext3 Disk Layout

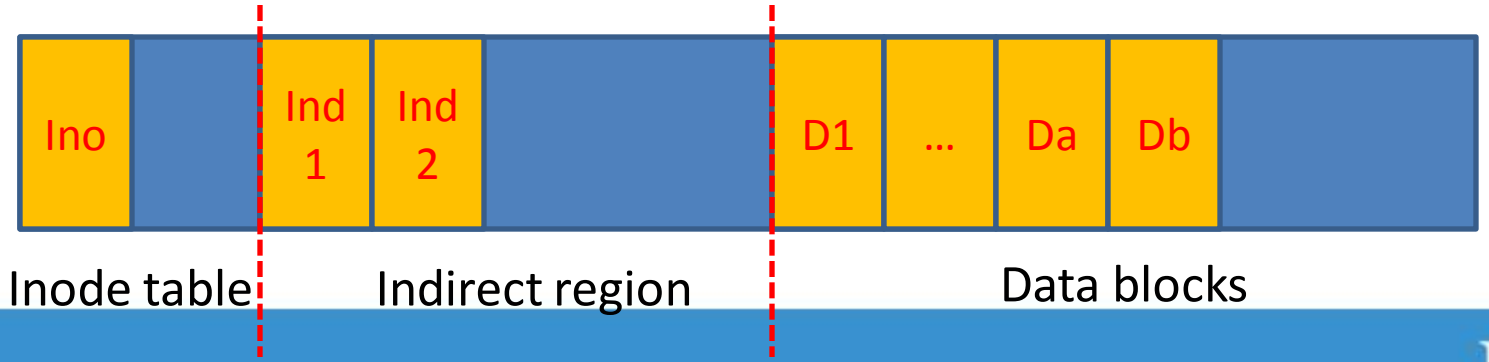
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Decouple allocation

Improve metadata density

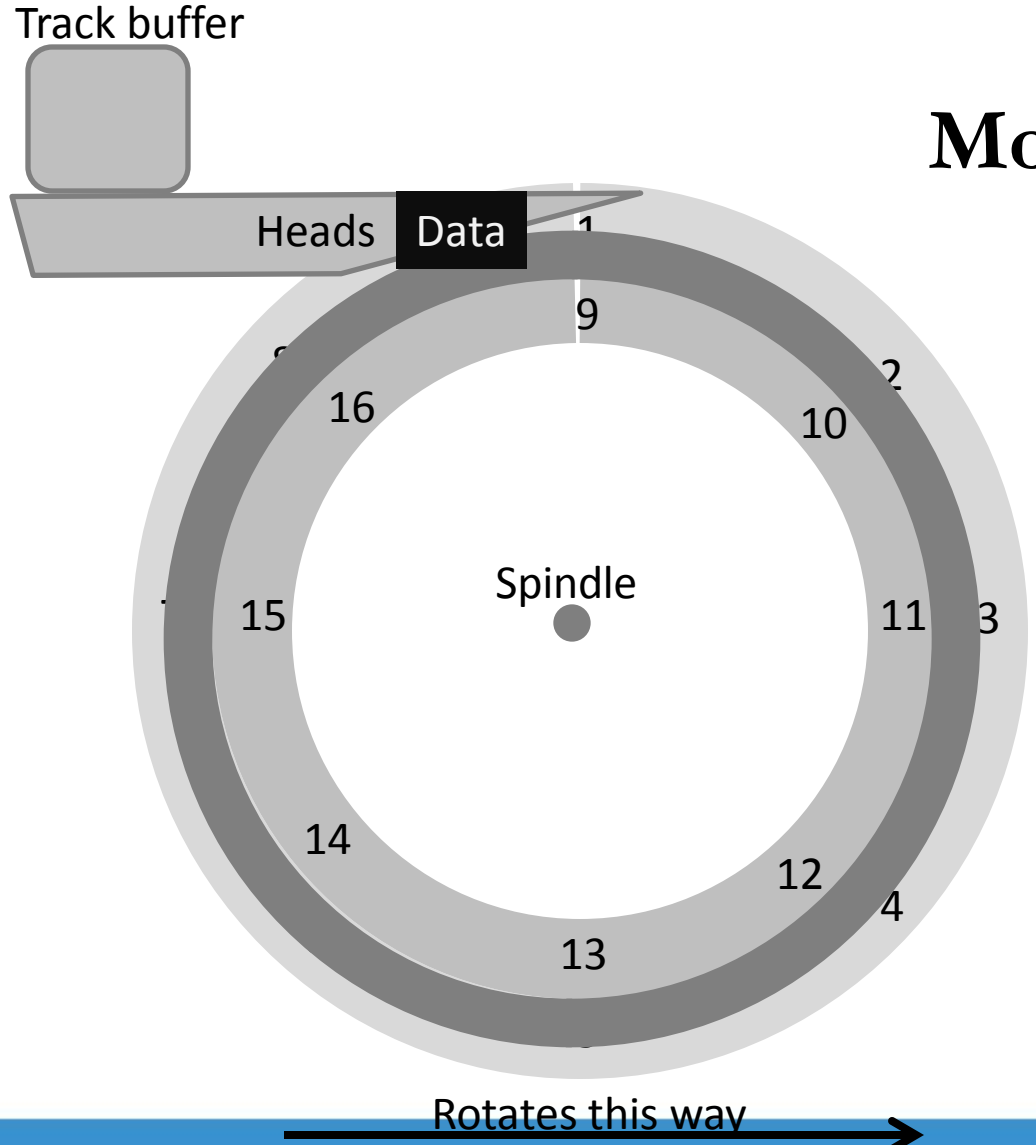


New allocation example



© Copyright 2013 EMC Corporation. All rights reserved.

Rext3 Disk Layout



WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

More additional seeks?

Disk track buffer

16MB

32MB

64MB

128MB

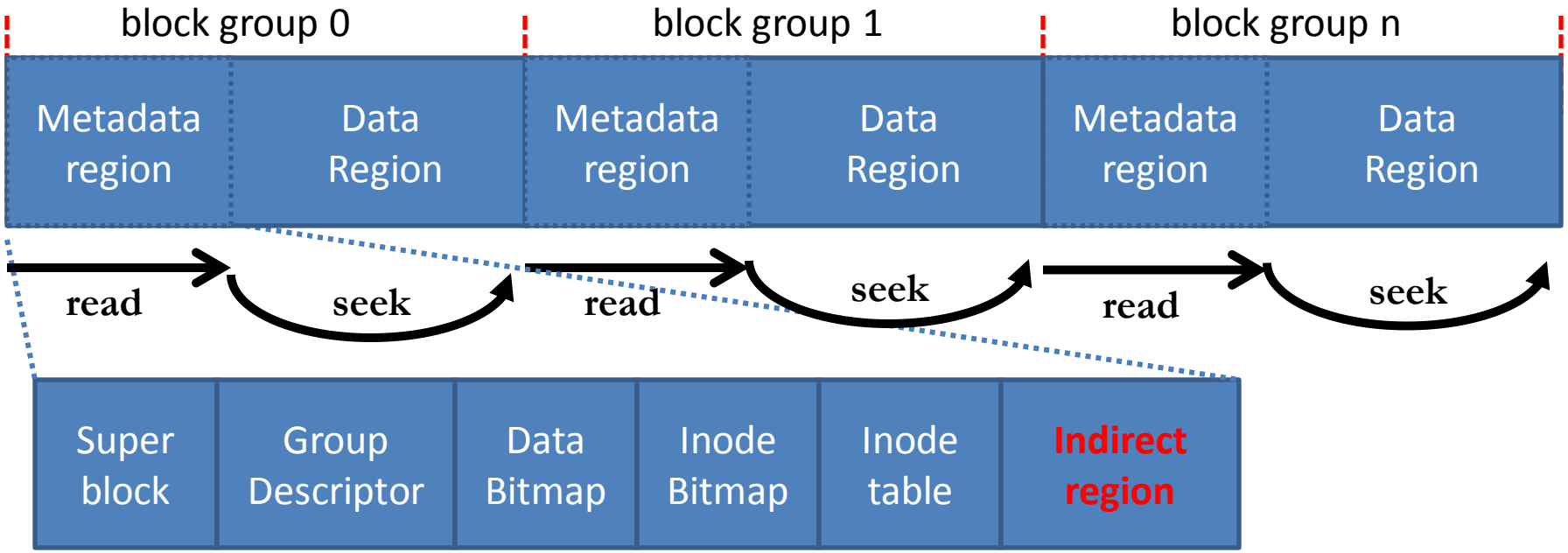
Size keeps increasing

© Copyright 2013 EMC Corporation. All rights reserved.

Disk-order Scan

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Most efficient way to scan metadata
Predictable scanning time



© Copyright 2013 EMC Corporation. All rights reserved.

Memory Pressure



Disk-order scan accesses all the indirect blocks without using the indirect tree

- Can't perform checking until all the related metadata are cached
- Impractical for large-scale FS

© Copyright 2013 EMC Corporation. All rights reserved.

Self-check, Cross-check



Separate self-check and cross-check

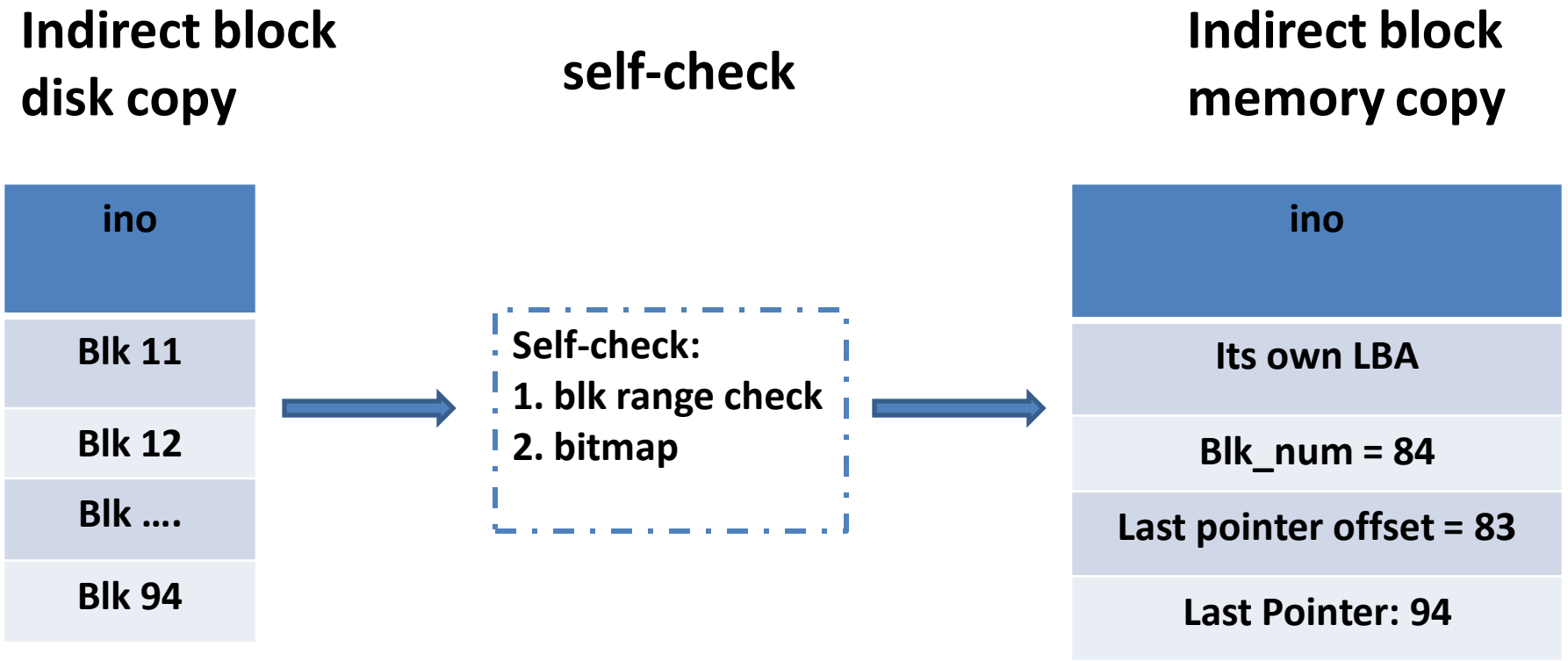
- Self-ID is added
- Finish most checks without referring to other metadata (self-check)

Self-check and discard

- Once self-check is performed, remove unused fields for cross-check

© Copyright 2013 EMC Corporation. All rights reserved.

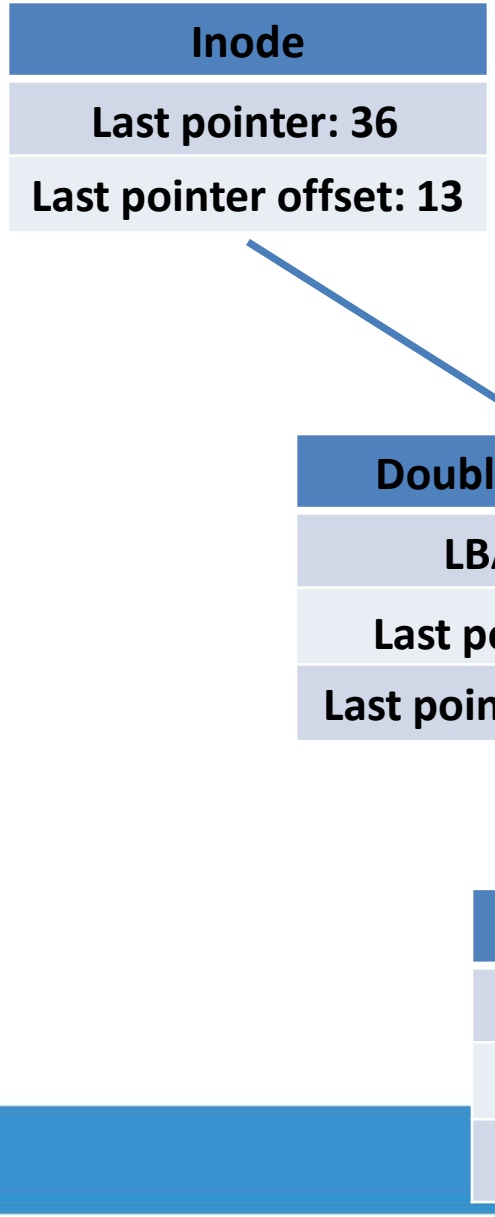
Example: Self-check & Discard



Compression ratio is nearly 250:1

© Copyright 2013 EMC Corporation. All rights reserved.

Example: Cross-check of File Size



1. Partially rebuild the tree structure

2. Calculate the file size using offset

© Copyright 2013 EMC Corporation. All rights reserved.

Fast Recovery with Bitmap Snapshot



Costly double scan of inode and indirect blocks

- Detect multiple-claimed blocks in 1st scan
- Detect their owners during 2nd scan

ffsck: 1 full scan + 1 partial rescan

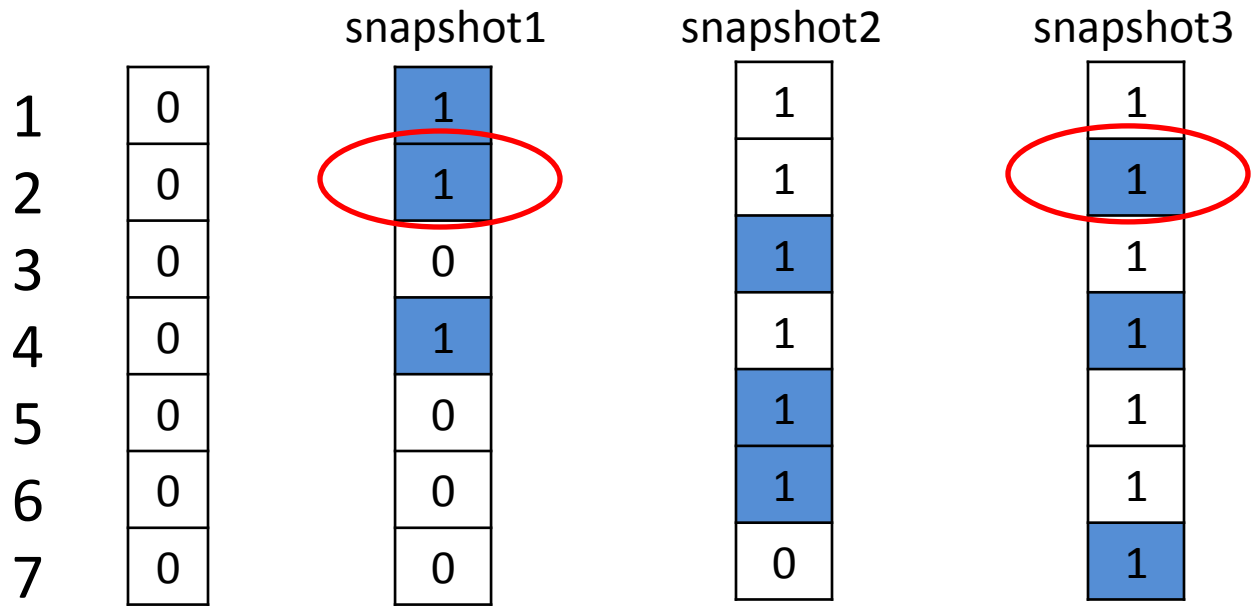
- ffsck builds a list of bitmap snapshots to limit the rescan's scope

© Copyright 2013 EMC Corporation. All rights reserved.

Fast Recovery with Bitmap Snapshot

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Create snapshot for each group of inodes



Only need to rescan
the group of inodes for snapshot1

© Copyright 2013 EMC Corporation. All rights reserved.

Summary



Decouple allocation

- Improve metadata density

Disk-order scan

- Most efficient scan approach

Self-check & cross-check

- Avoid memory saturation

Fast recovery with bitmap snapshot

- One full scan + partial rescan

© Copyright 2013 EMC Corporation. All rights reserved.

Outline



fsck analysis

- Checking approach
- Performance
- FS tradeoffs

ffsck & rext3

- Goals
- Novel features

Evaluation

© Copyright 2013 EMC Corporation. All rights reserved.

Checking Performance Comparison



Question1:

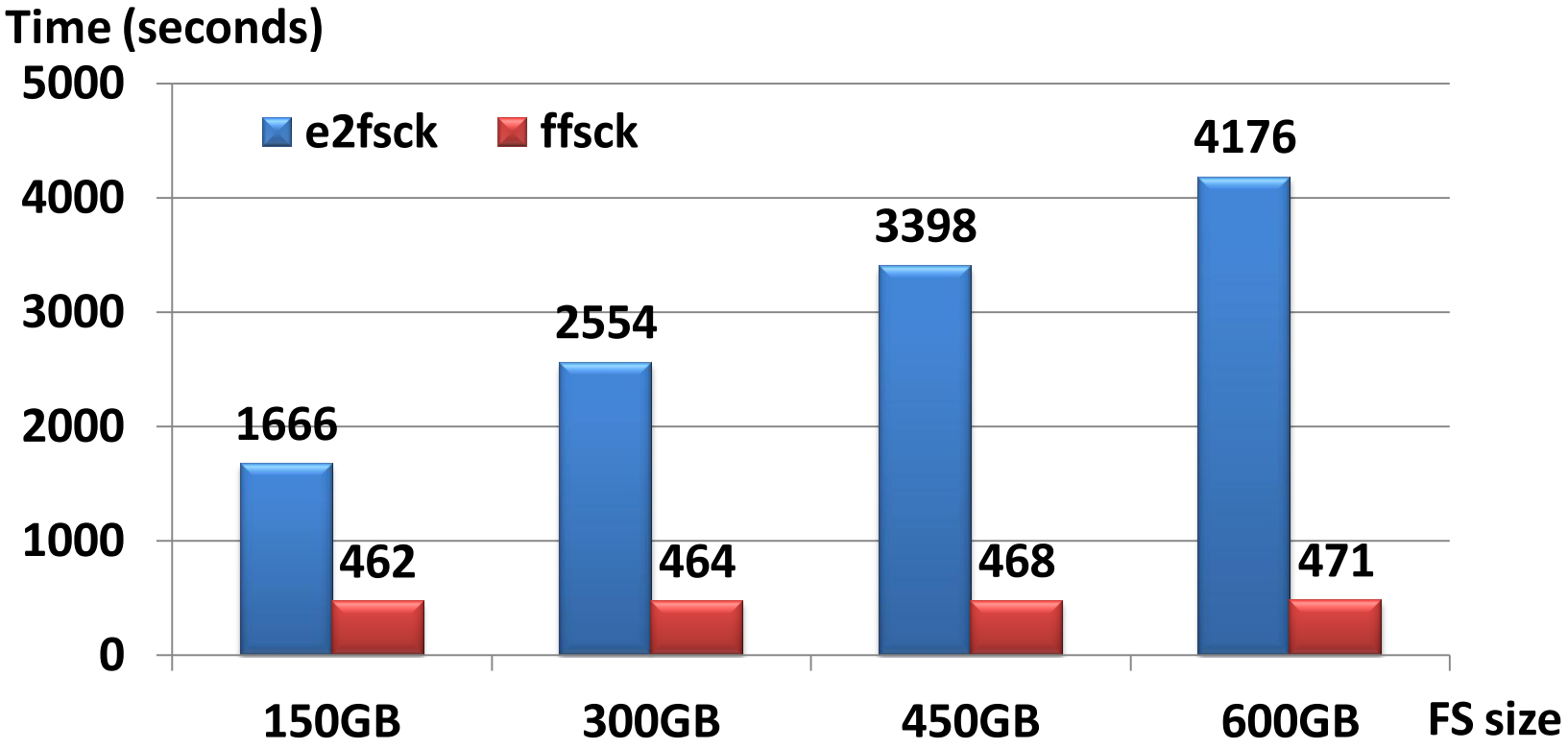
Will ffsck scale well?

Question2:

Can ffsck perform consistently as the FS ages?

© Copyright 2013 EMC Corporation. All rights reserved.

Checking Time Comparison

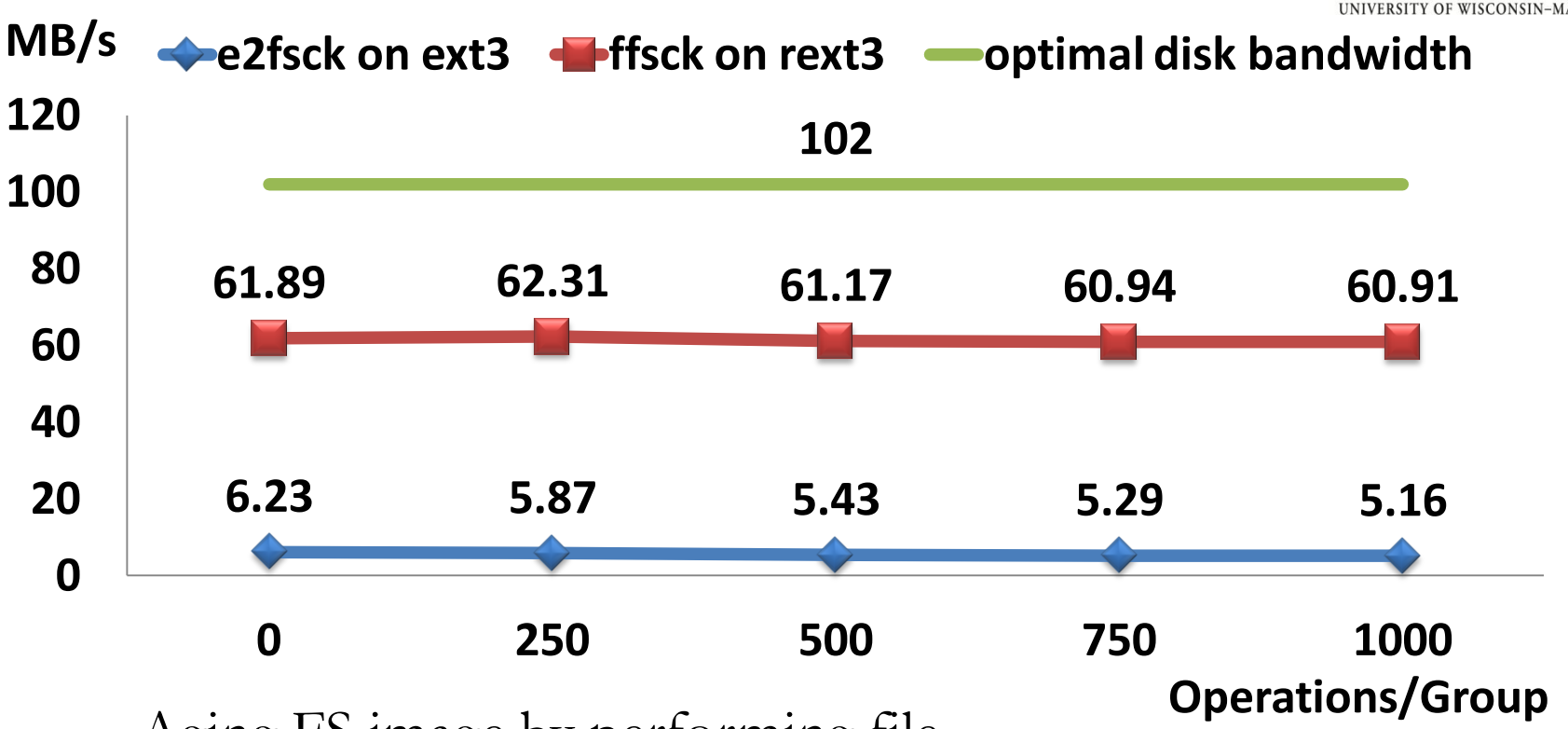


750GB disk, 1GB memory, e2fsprog 1.41.12, Linux 2.6.28
Initialize disk image by creating directories with small files (4KB- 2MB)
Increase FS size by creating new files and appending data to files(4KB-1MB)

Ffsck checking time is determined when the file system is created

© Copyright 2013 EMC Corporation. All rights reserved.

Checking Speed Comparison on Aging FS Image



Aging FS image by performing file creations, appends, truncations and deletions (750GB partition, roughly 95% utilization)

Much faster and robust

© Copyright 2013 EMC Corporation. All rights reserved.

File System Comparison



Question1:

Can rext3 compete with ext3 in sequential reads?

Question2:

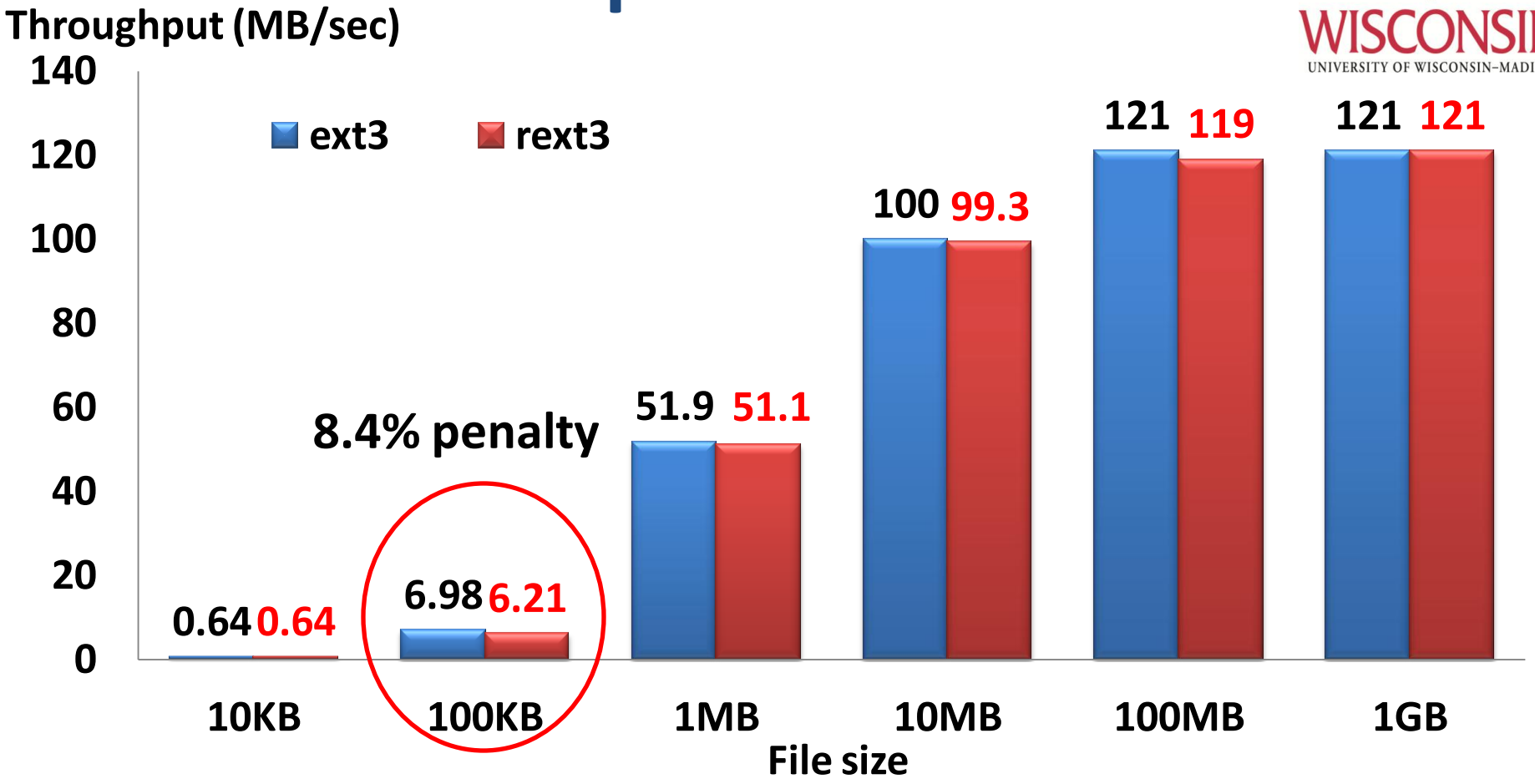
What is its impact on sequential writes?

Question3 and 4:

What about random reads and macro-benchmark?

© Copyright 2013 EMC Corporation. All rights reserved.

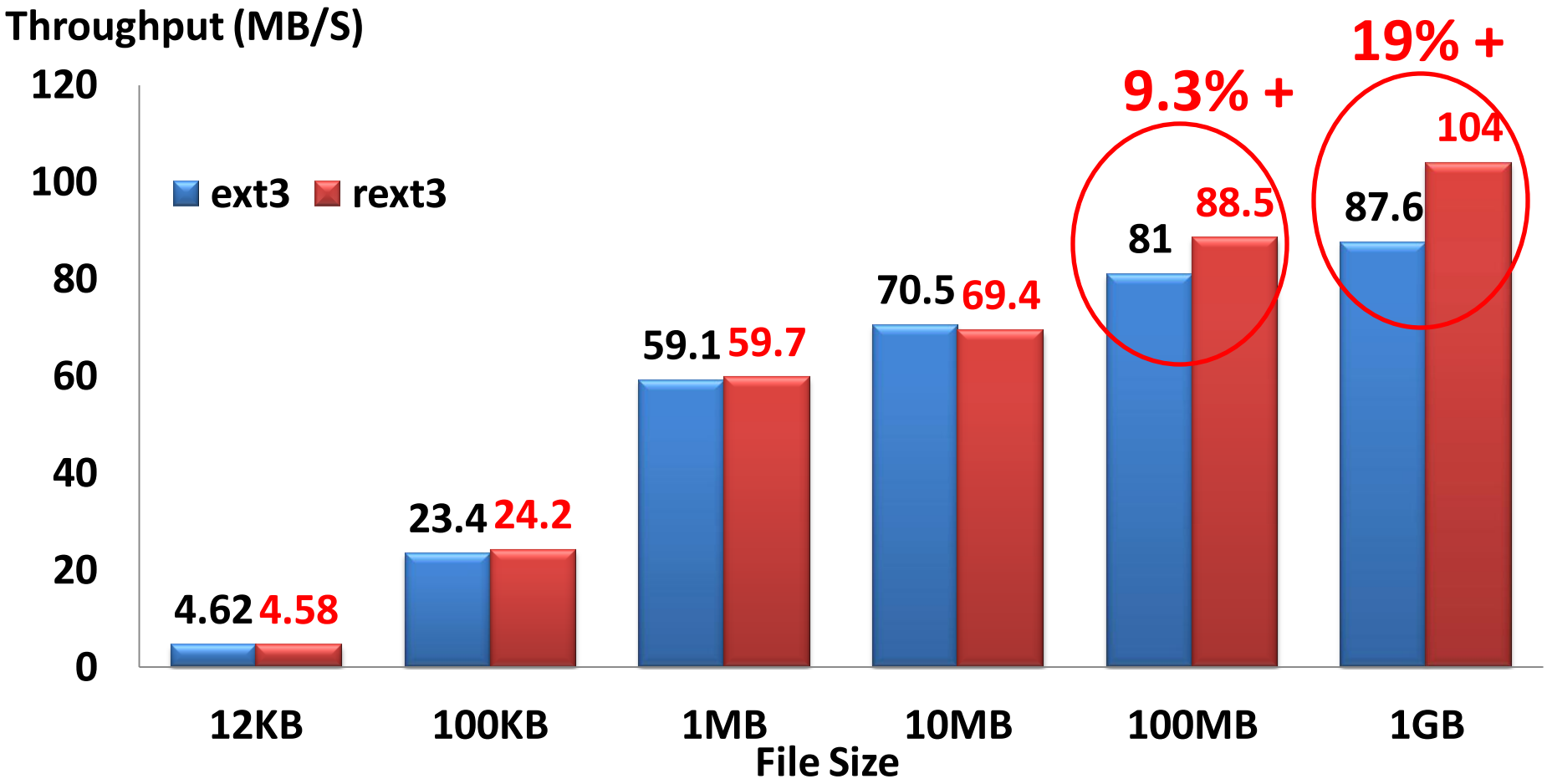
Sequential Read



The disk track buffer allows rext3 to match ext3's performance, except for small reads

© Copyright 2013 EMC Corporation. All rights reserved.

Sequential Write



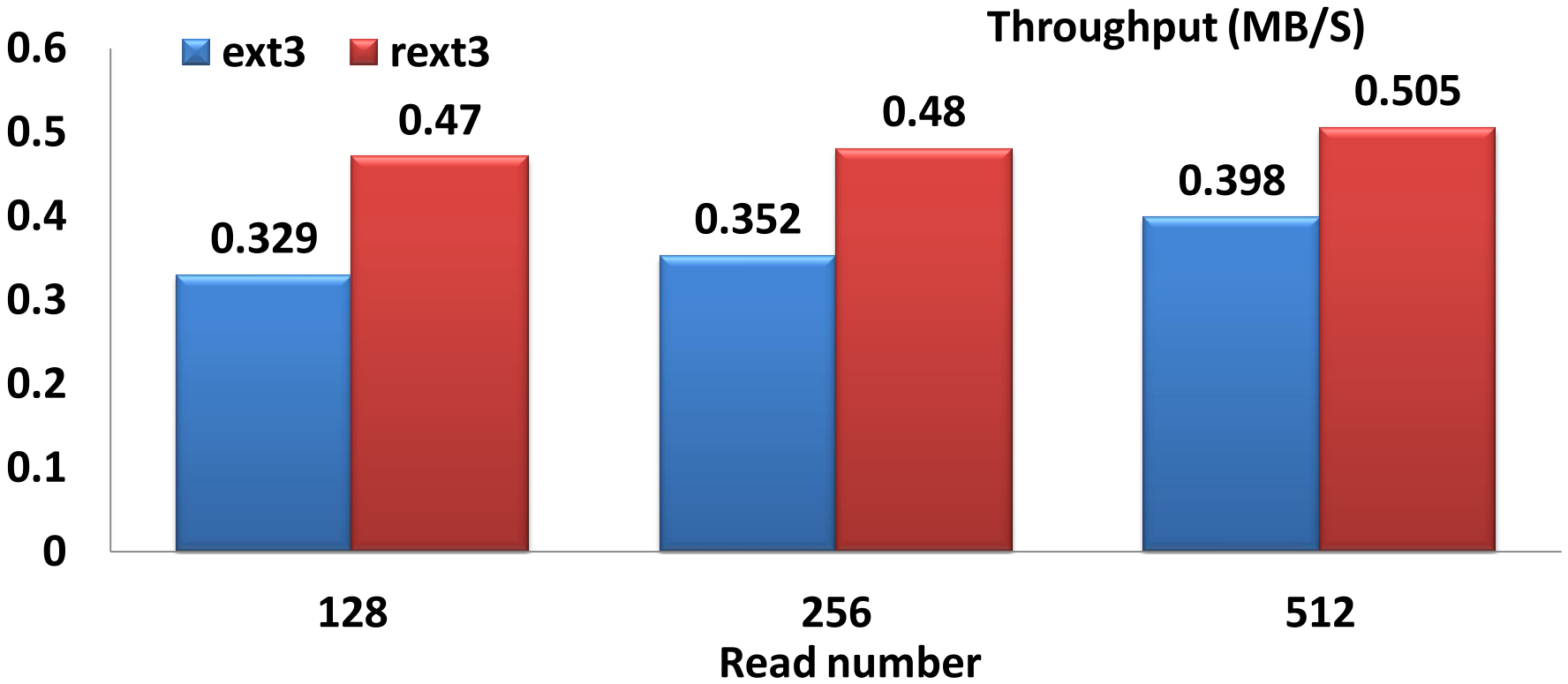
Indirect region aids ext3's ordered journaling mechanism

© Copyright 2013 EMC Corporation. All rights reserved.

Random Read



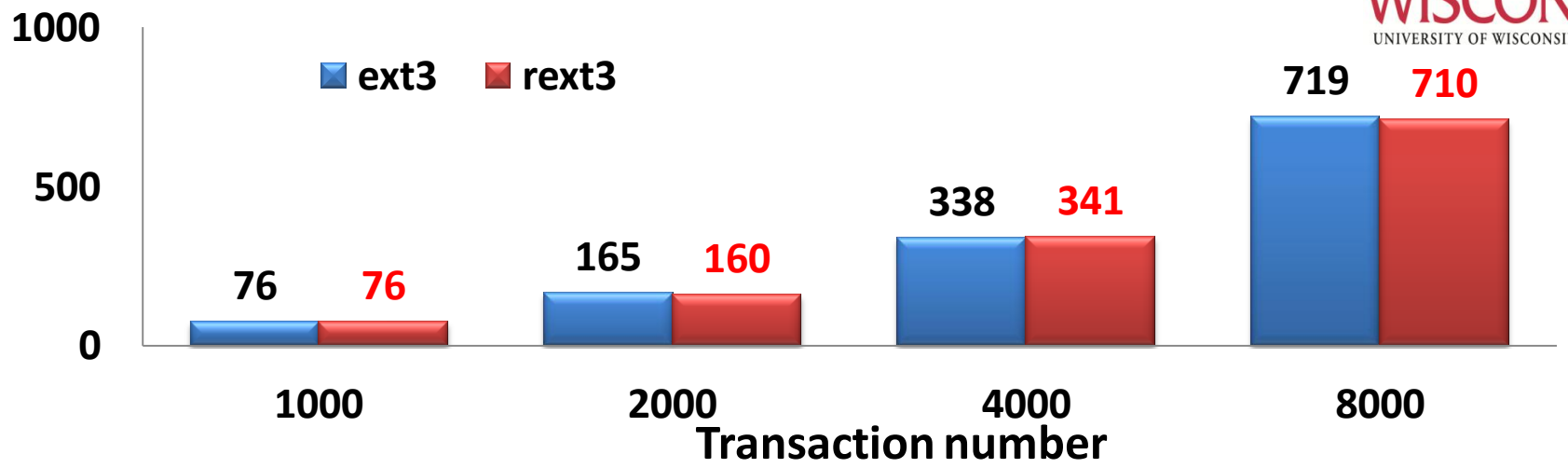
Randomly read 4KB blocks from a 2GB file



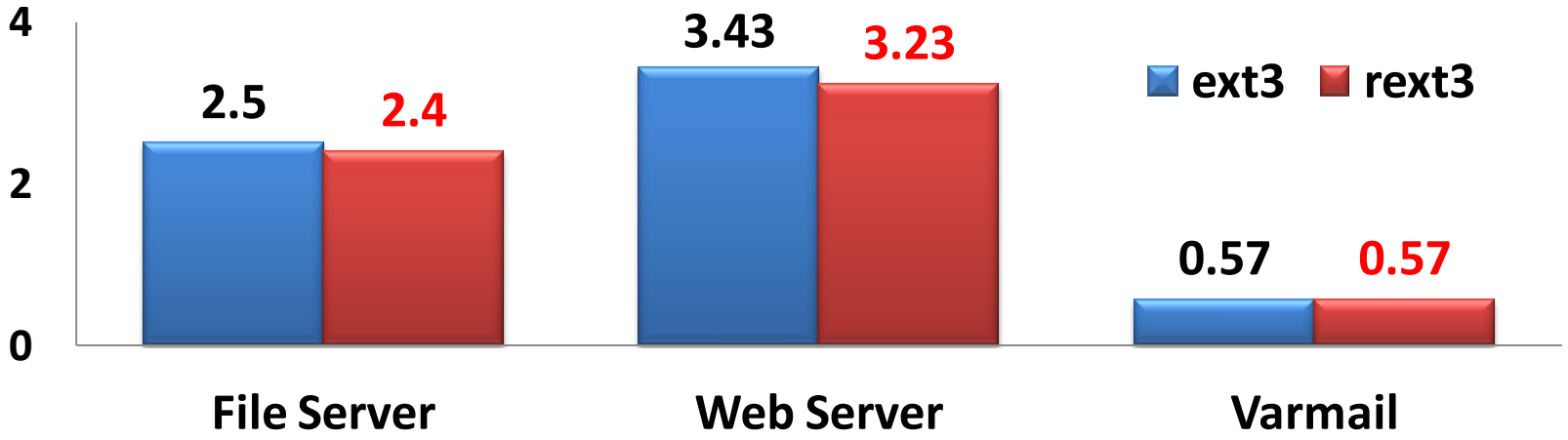
27% - 43% improvement

Indirect region benefits from disk buffer

Postmark



Filebench



Competitive performance

© Copyright 2013 EMC Corporation. All rights reserved.

Summary



Make fast repair a primary concern of FS design

FS provides direct support for the fast checker

Benefits:

- 10 times checking speed
- Big improvement for large writes and random reads
- Small penalty for small reads

© Copyright 2013 EMC Corporation. All rights reserved.

Conclusion

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

- How to protect against corruptions is well-known
- FS repairing is important but receives little attention
- Build the checker as an integral component rather than a peripheral addition
- ffsck is not a universal solution, other FSes may require other methods

© Copyright 2013 EMC Corporation. All rights reserved.





WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Thanks!

Questions?

Wisconsin Institute on Software-defined Datacenters in Madison

http://wisdom.cs.wisc.edu/

© Copyright 2013 EMC Corporation. All rights reserved.