An Empirical Study of File Systems on NVM

Priya Sehgal, Sourav Basu, Kiran Srinivasan, Kaladhar Voruganti

NetApp Inc.

Motivation

- Need a thin software stack to access data from fast NVM
  - Persistent memory abstractions
  - NVM optimized file systems

- What characteristics of traditional block based file systems are good for NVM?
  - Can traditional file systems be fine tuned using mount and format options?
  - Can it be optimized with minor changes?
  - How does the performance of traditional file systems compare with NVM-optimized one?
  - What file system features help improve performance on NVM?

Overview

- Related Work
- Experimental Methodology
- Experimental Results
- Recommendation and Conclusion

Overview

- Related Work
- Experimental Methodology
- Experimental Results
- Recommendation and Conclusion

Related Work

POSIX interface, re-designing the file system for NVM	POSIX library interposers or bypass file system	New Programming Models and Data-Structures	Adjust operating system, storage stack or file system configurations

Related Work

POSIX interface, re-designing the file system for NVM	POSIX library interposers or bypass file system	New Programming Models and Data-Structures	Adjust operating system, storage stack or file system configurations
• PMFS • SCMFS • BPFS			

Related Work

POSIX interface, re-designing the file system for NVM	POSIX library interposers or bypass file system	New Programming Models and Data-Structures	Adjust operating system, storage stack or file system configurations
• PMFS • SCMFS • BPFS	• Moneta-D • Bankshot • Aerie		

Related Work

POSIX interface, re-designing the file system for NVM	POSIX library interposers or bypass file system	New Programming Models and Data-Structures	Adjust operating system, storage stack or file system configurations
• PMFS • SCMFS • BPFS	• Moneta-D • Bankshot • Aerie	• Mnemosyne • PMem-Lib • NV-Heaps • NV-Tree • CDDS	

Related Work

POSIX interface, re-designing the file system for NVM	POSIX library interposers or bypass file system	New Programming Models and Data-Structures	Adjust operating system, storage stack or file system configurations
• PMFS	• Moneta-D	• Mnemosyne	• Lee et al
• SCMFS	• Bankshot	• PMem-Lib	• Our work.
• BPFS	• Aerie	• NV-Heaps	
		• NV-Tree	
		• CDDS	

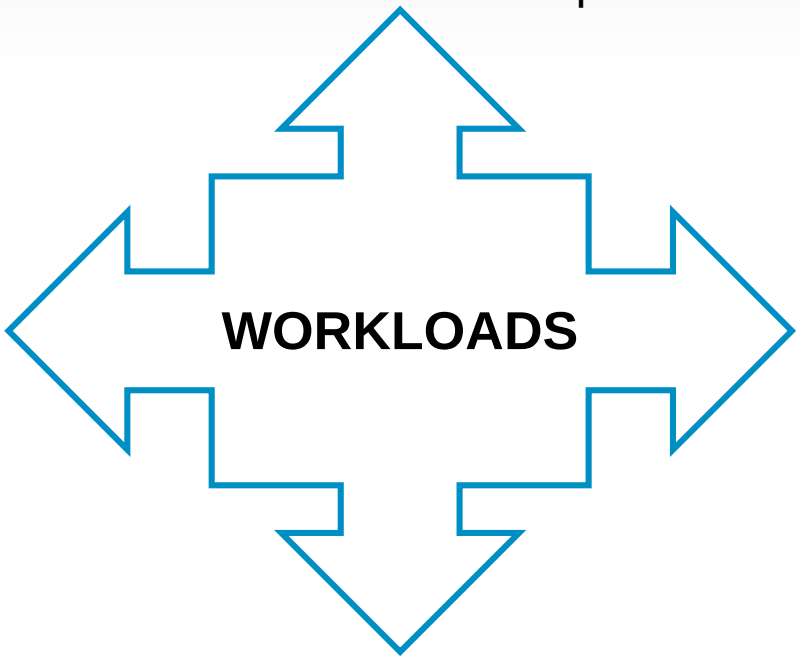
Overview

- Related Work
- Experimental Methodology
- Experimental Results
- Recommendation and Conclusion

Experimental Methodology

Workloads

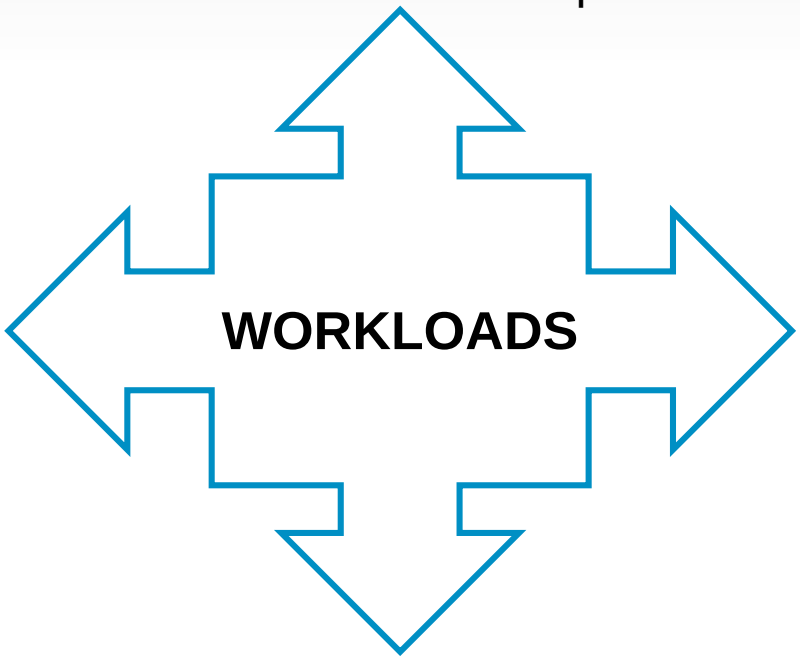
FileServer :Moderate directory depth, moderate no. of large files, data and metadata operations



Experimental Methodology

Workloads

FileServer :Moderate directory depth, moderate no. of large files, data and metadata operations



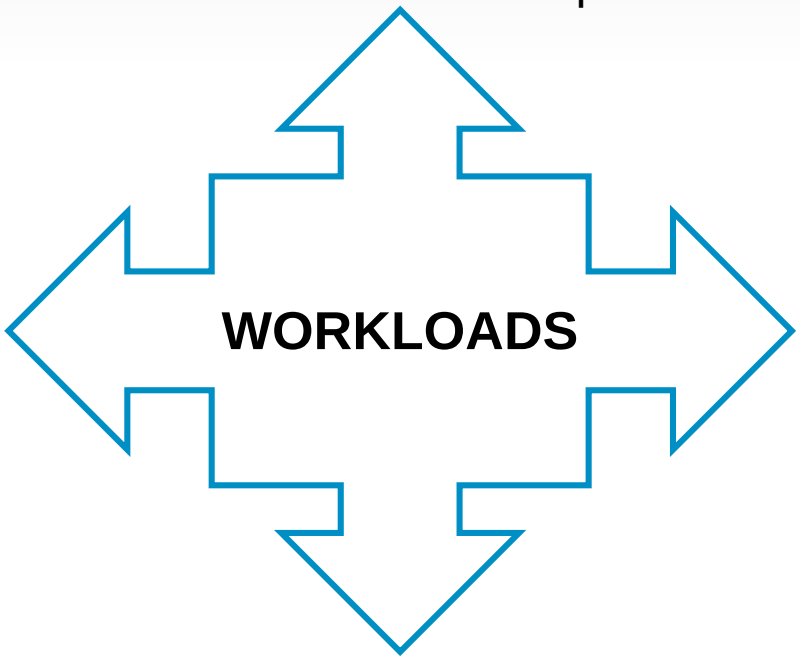
WebProxy:Flat namespace, large no. of small files, data and metadata operations

Experimental Methodology

Workloads

FileServer :Moderate directory depth, moderate no. of large files, data and metadata operations

OLTP: data intensive

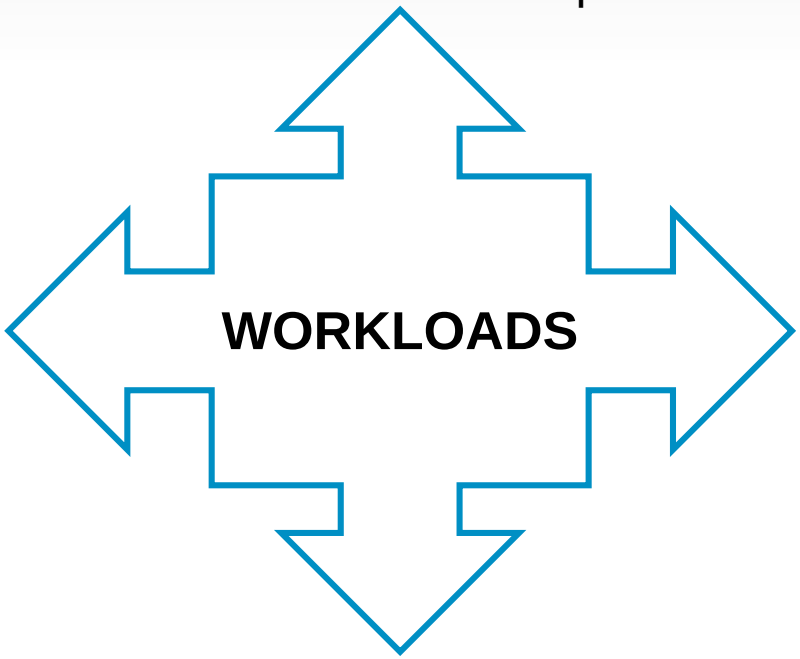


WebProxy:Flat namespace, large no. of small files, data and metadata operations

Experimental Methodology

Workloads

FileServer :Moderate directory depth, moderate no. of large files, data and metadata operations



OLTP: data intensive

WebProxy:Flat namespace, large no. of small files, data and metadata operations

Key-Value Store: Uses Memory-Mapped Semantics (load/stores)

Experimental Methodology

File System Characteristics (varied using mount and format options)

- **Inode Structure**: Linear vs B+ Tree
- **Block Size**: Fixed vs Variable sized extent
- **Layout/Update**: In-place vs Log-structured vs Hybrid
- **Allocation Strategy**: Immediate vs Delayed
- **Parallel Allocation** (Concepts like Allocation/Block group)
- **Journal**: Ordered vs Write-Back vs Data
- **Execute-in-place(XIP)**

Experimental Methodology

File System Characteristics (varied using mount and format options)

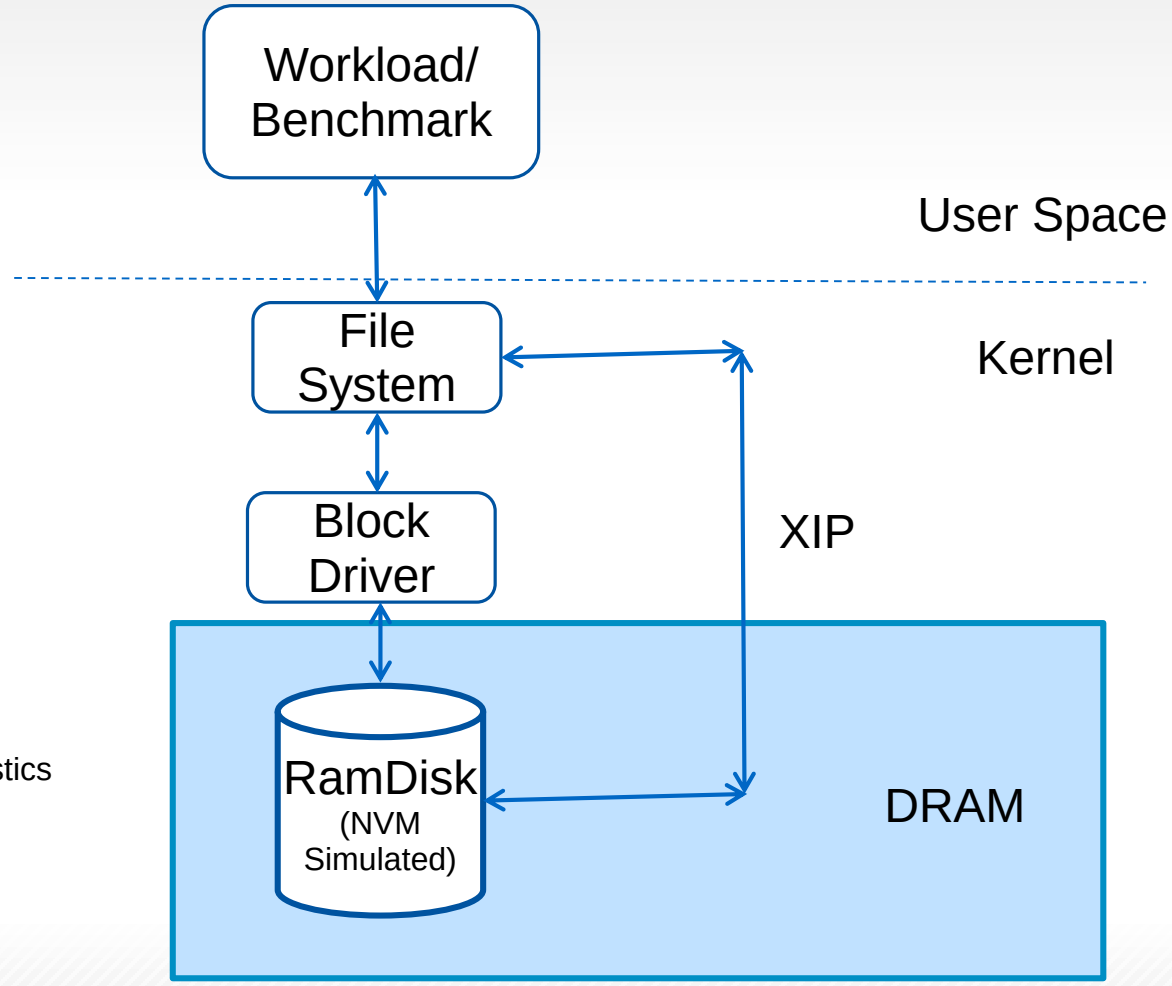
- **Inode Structure**: Linear vs B+ Tree
- **Block Size**: Fixed vs Variable sized extent
- **Layout/Update**: In-place vs Log-structured vs Hybrid
- **Allocation Strategy**: Immediate vs Delayed
- **Parallel Allocation** (Concepts like Allocation/Block group)
- **Journal**: Ordered vs Write-Back vs Data
- **Execute-in-place(XIP)**

File Systems Evaluated

Ext2, Ext3, Ext4, XFS, F2FS, NILFS2, PMFS

Experimental Methodology

Experimental Setup



Limitations:

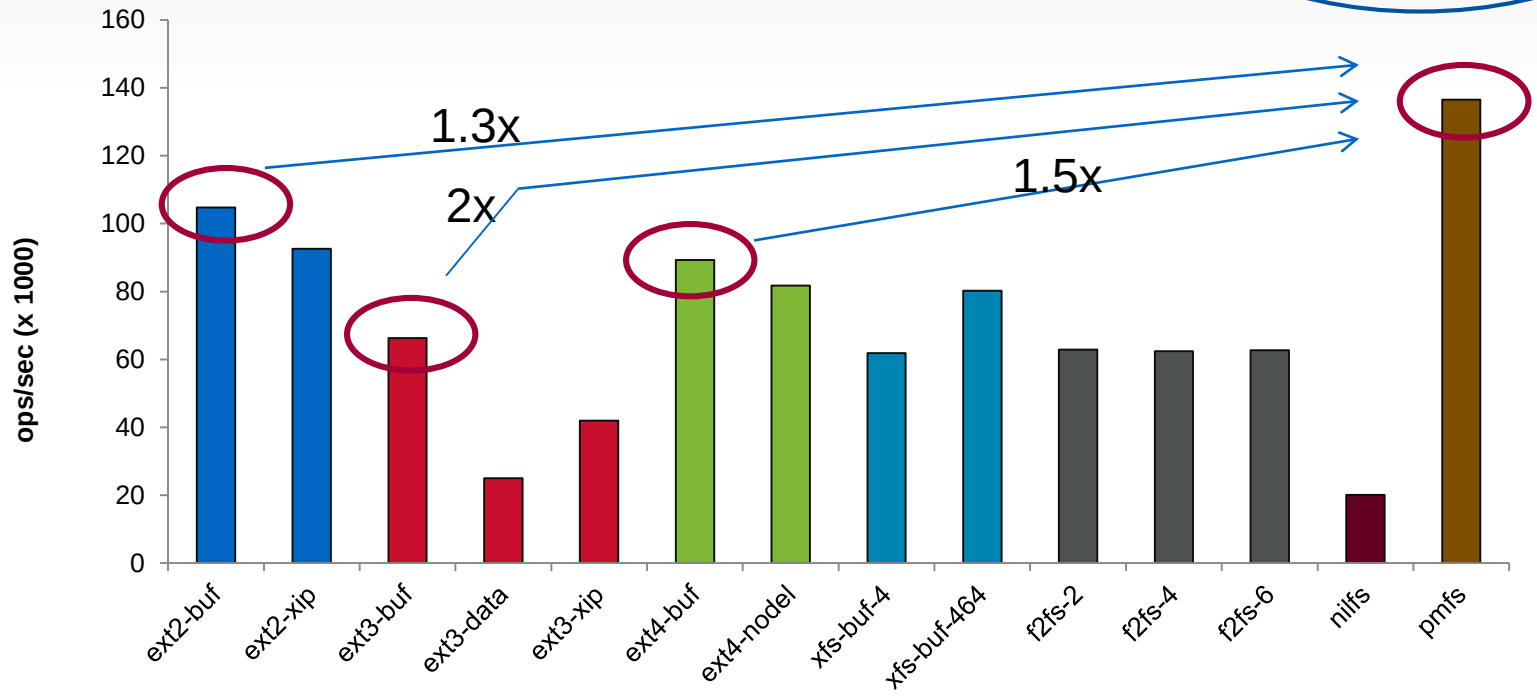
- NVM characteristics not simulated
- Not considered recoverabiliy

Overview

- Related Work
- Experimental Methodology
- Experimental Results
- Recommendation and Conclusion

FileServer (Throughput)

XIP support, atomic updates and fine grained logging

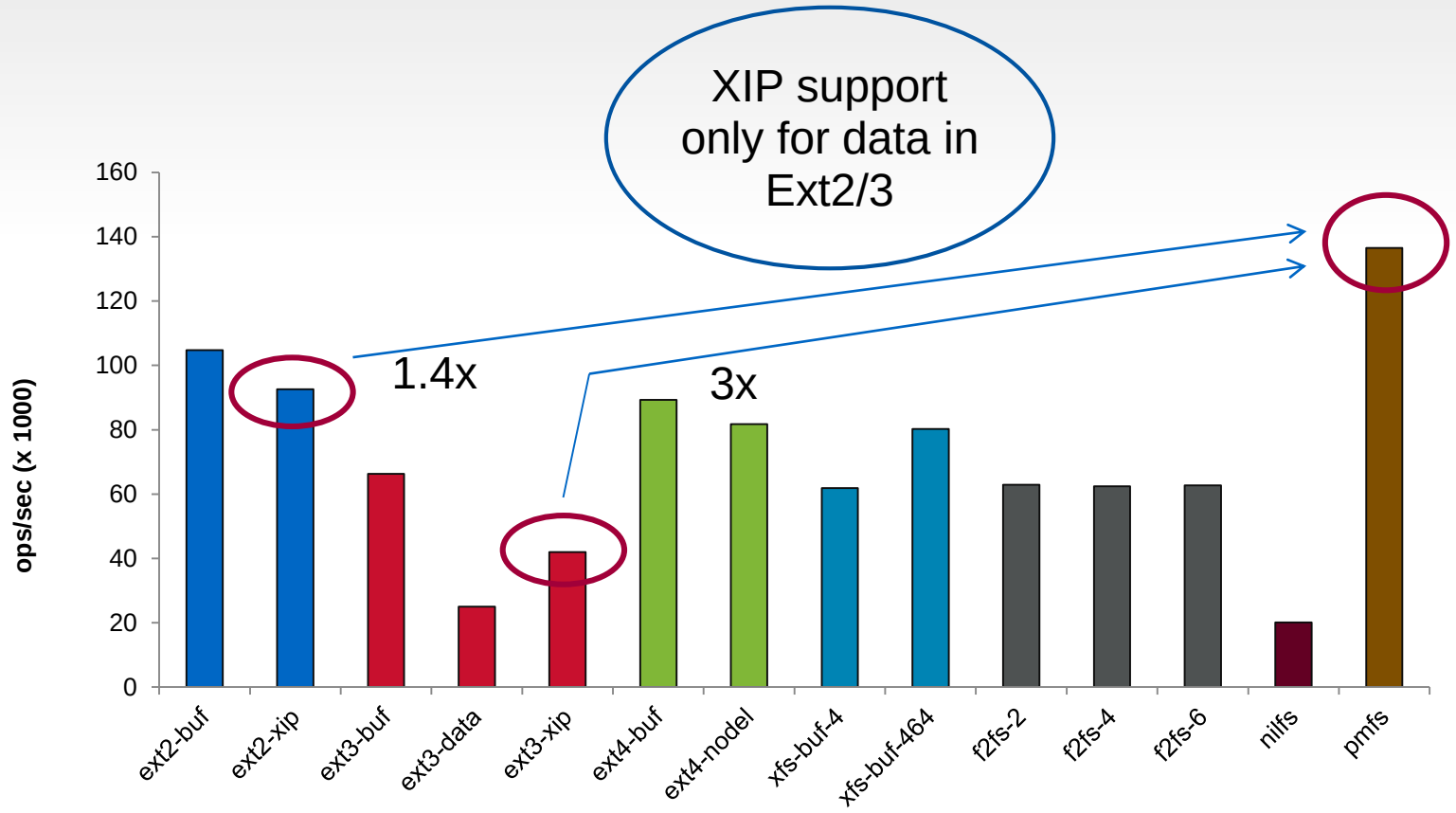


100K files of size 128KB

Higher is better

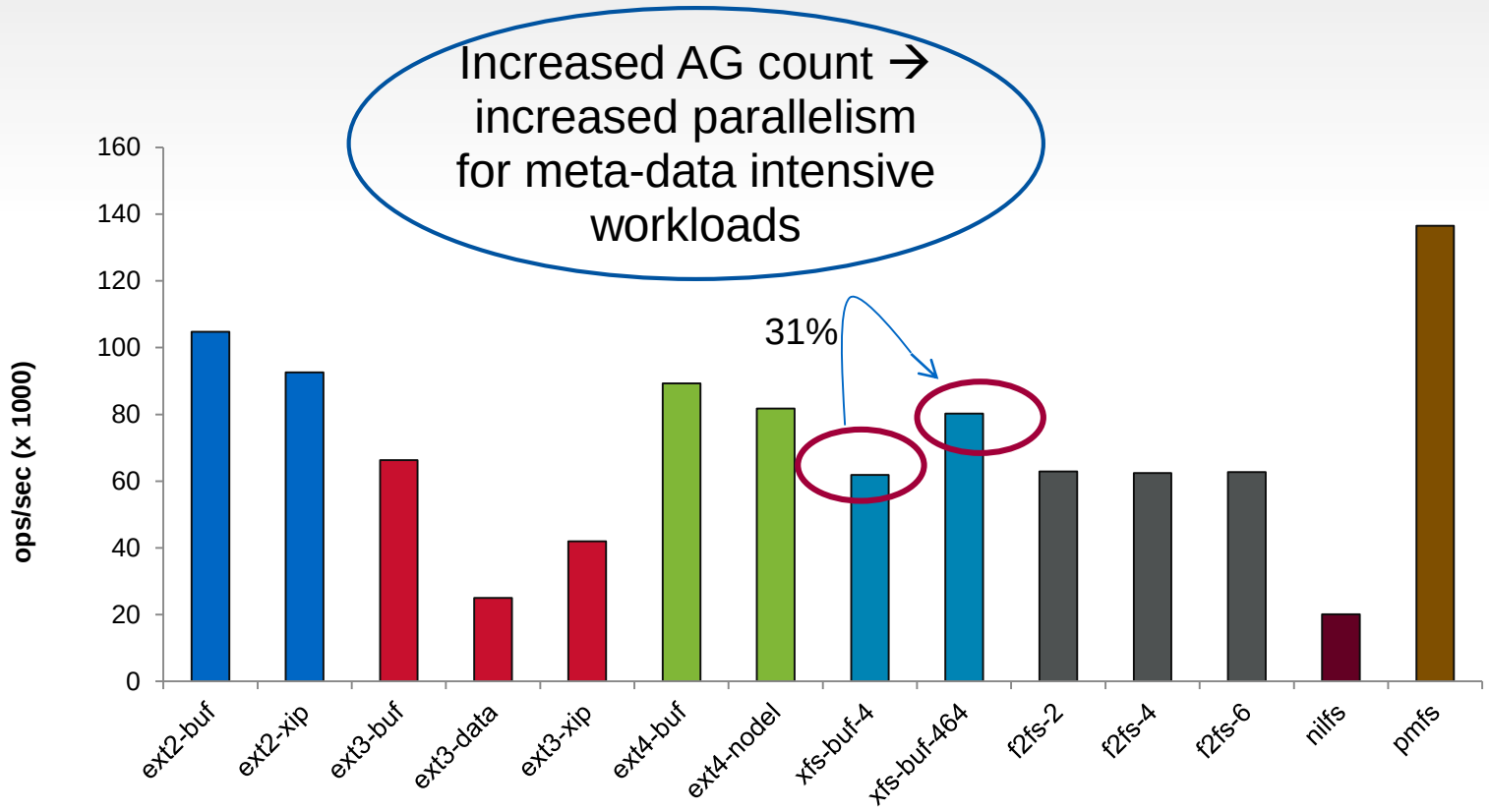
NetApp

FileServer (Throughput)



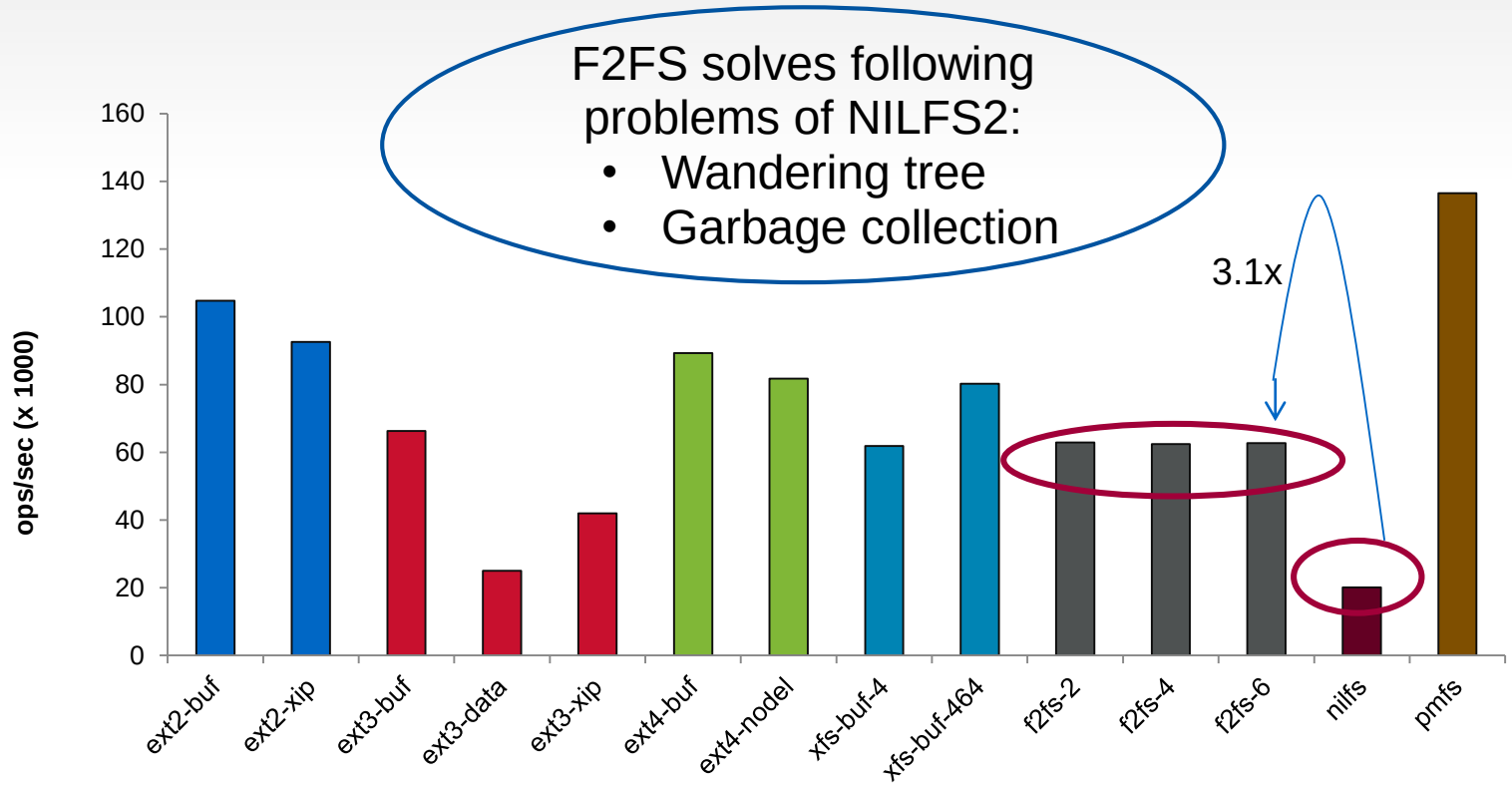
Higher is better

FileServer (Throughput)



Higher is better

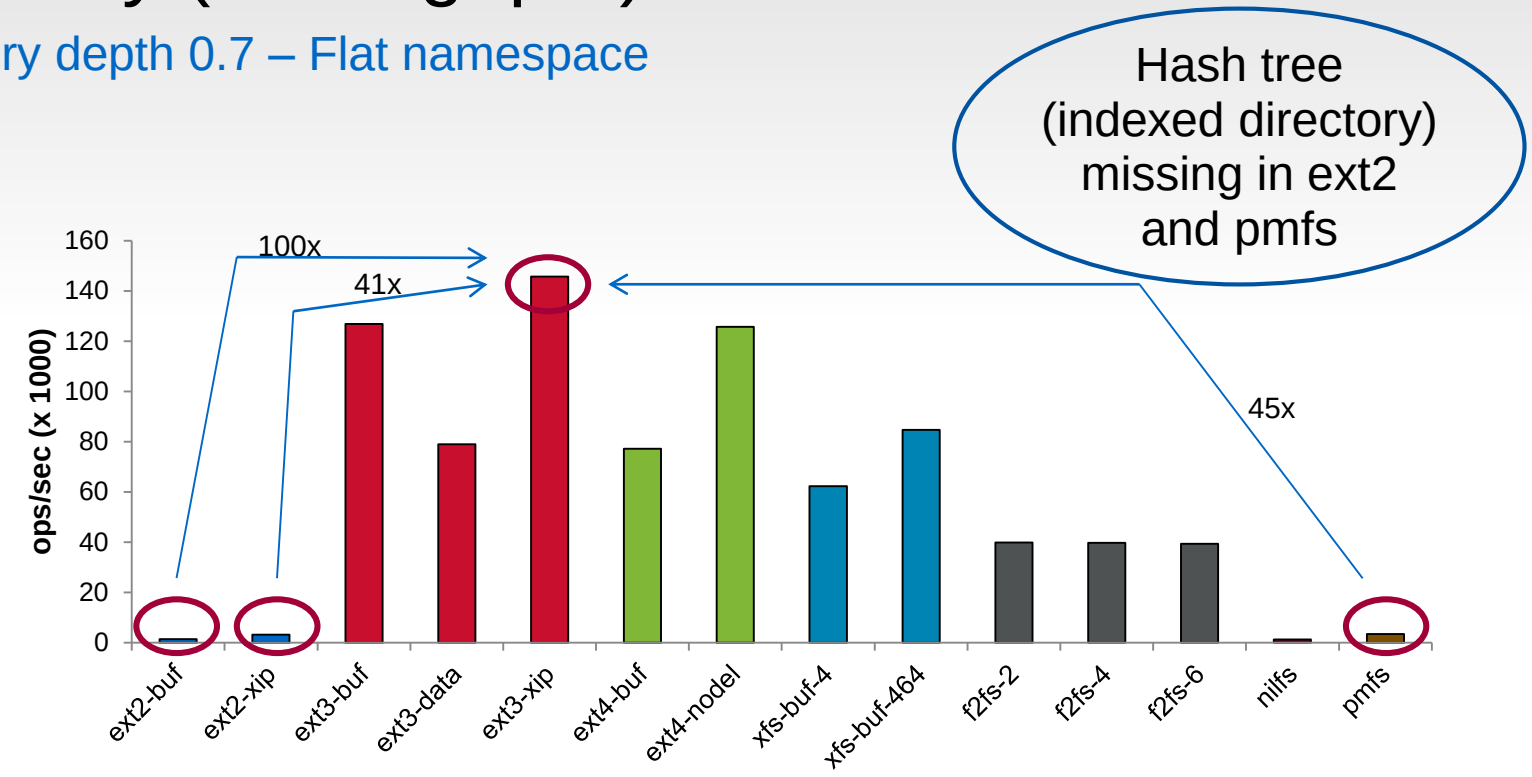
FileServer (Throughput)



Higher is better

WebProxy (Throughput)

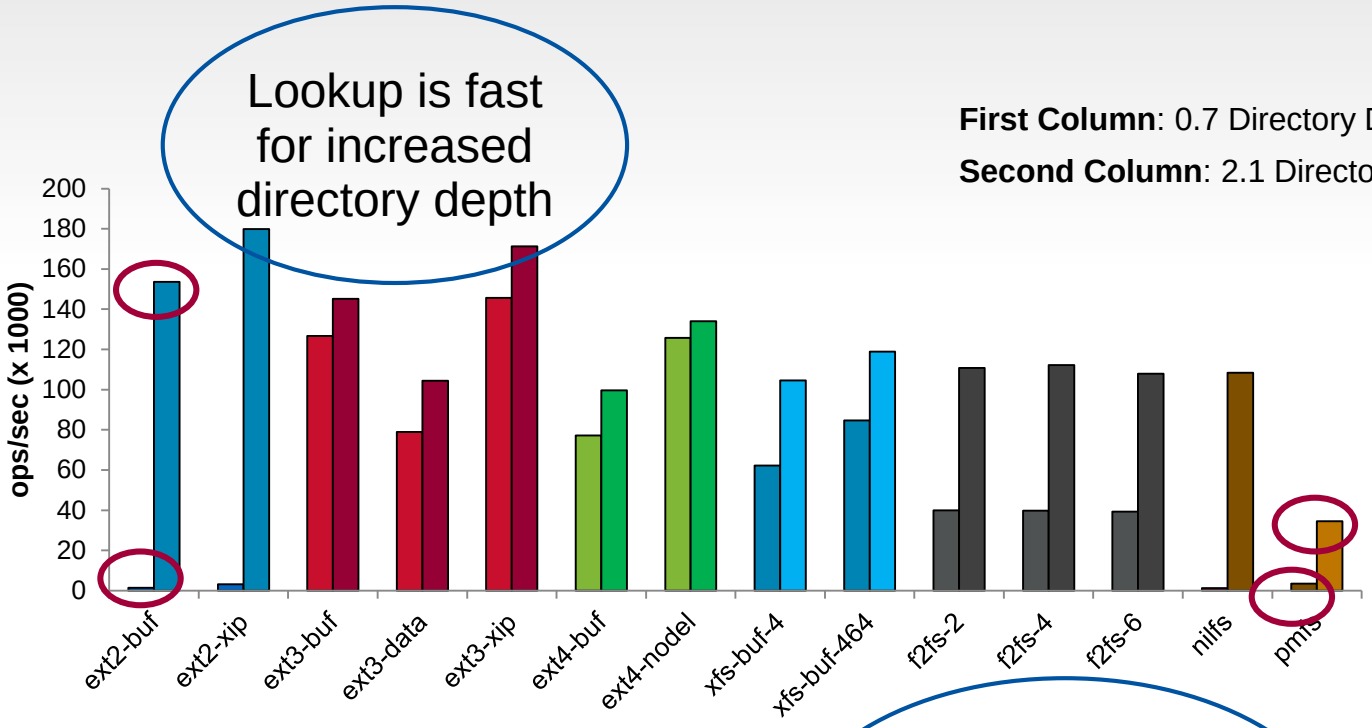
At directory depth 0.7 – Flat namespace



500K files of size 32KB

Higher is better

WebProxy (Throughput)

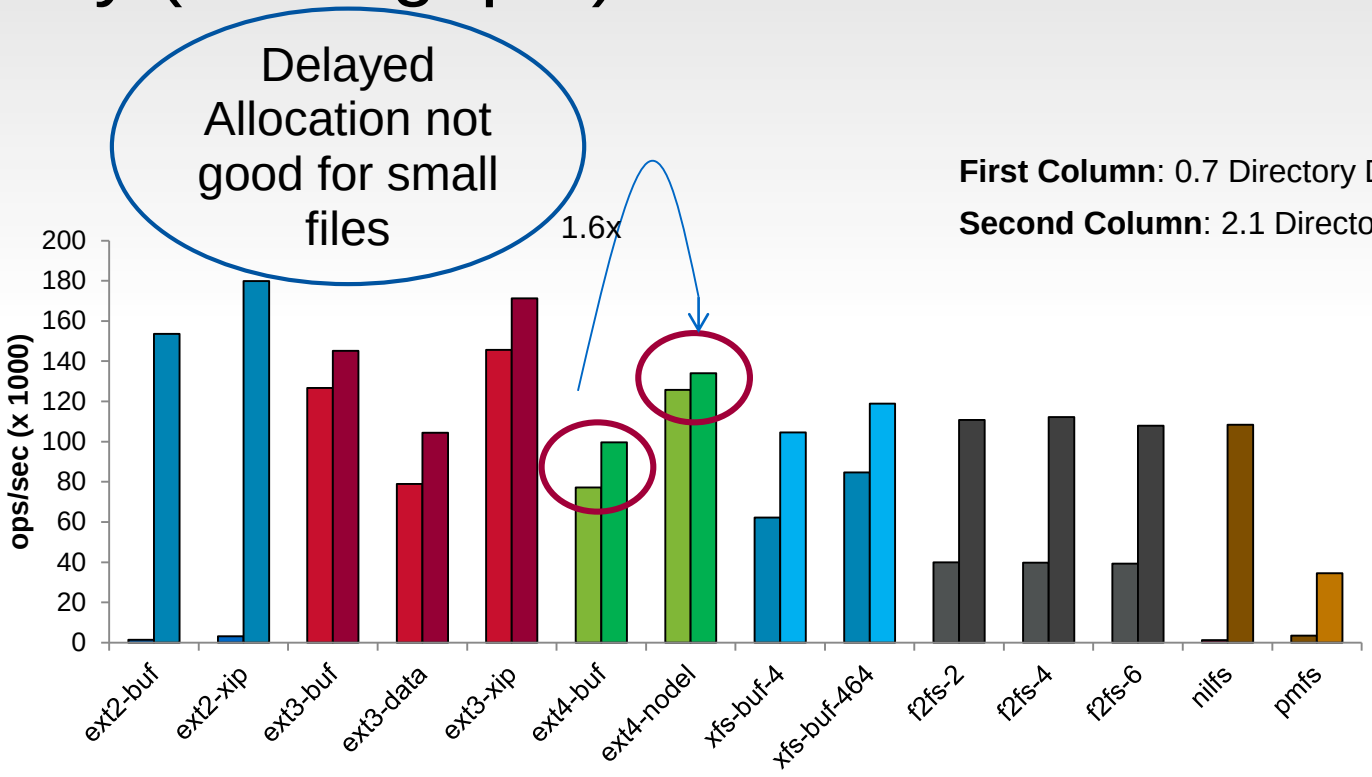


First Column: 0.7 Directory Depth

Second Column: 2.1 Directory Depth

Higher is better

WebProxy (Throughput)

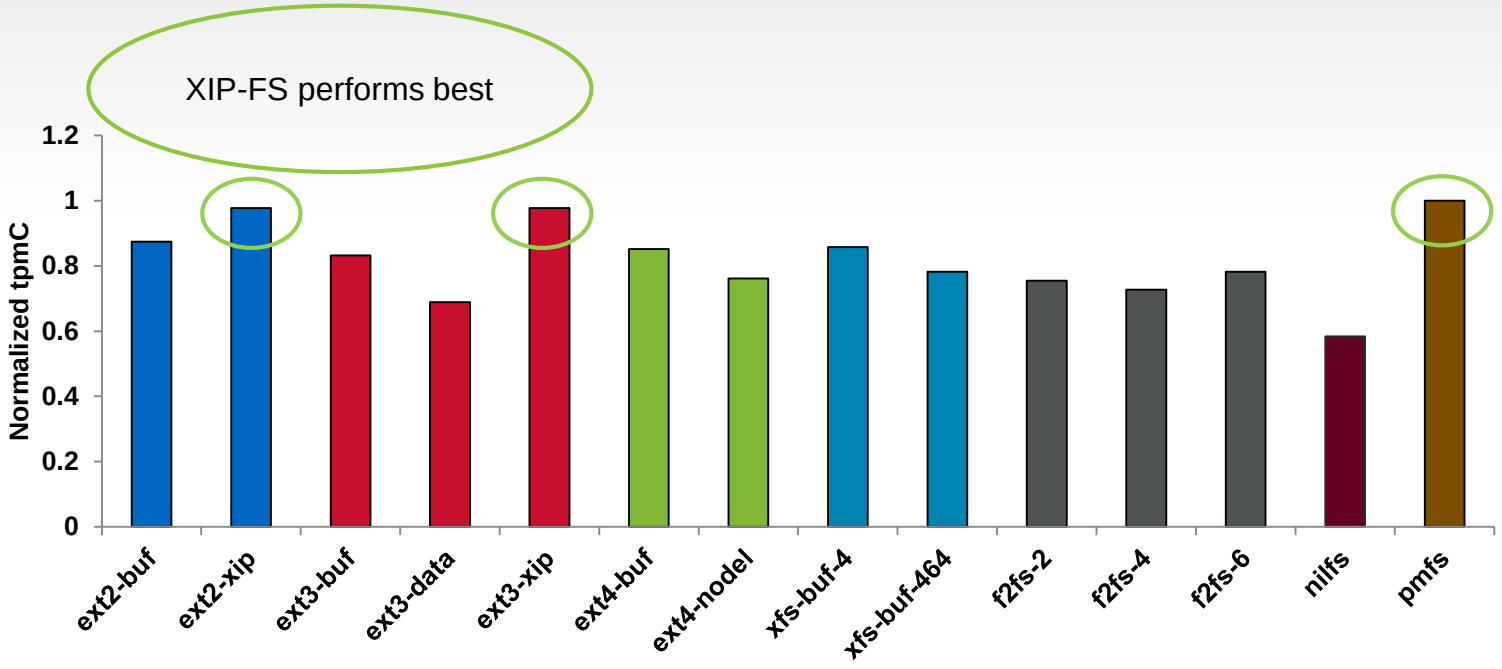


First Column: 0.7 Directory Depth

Second Column: 2.1 Directory Depth

Higher is better

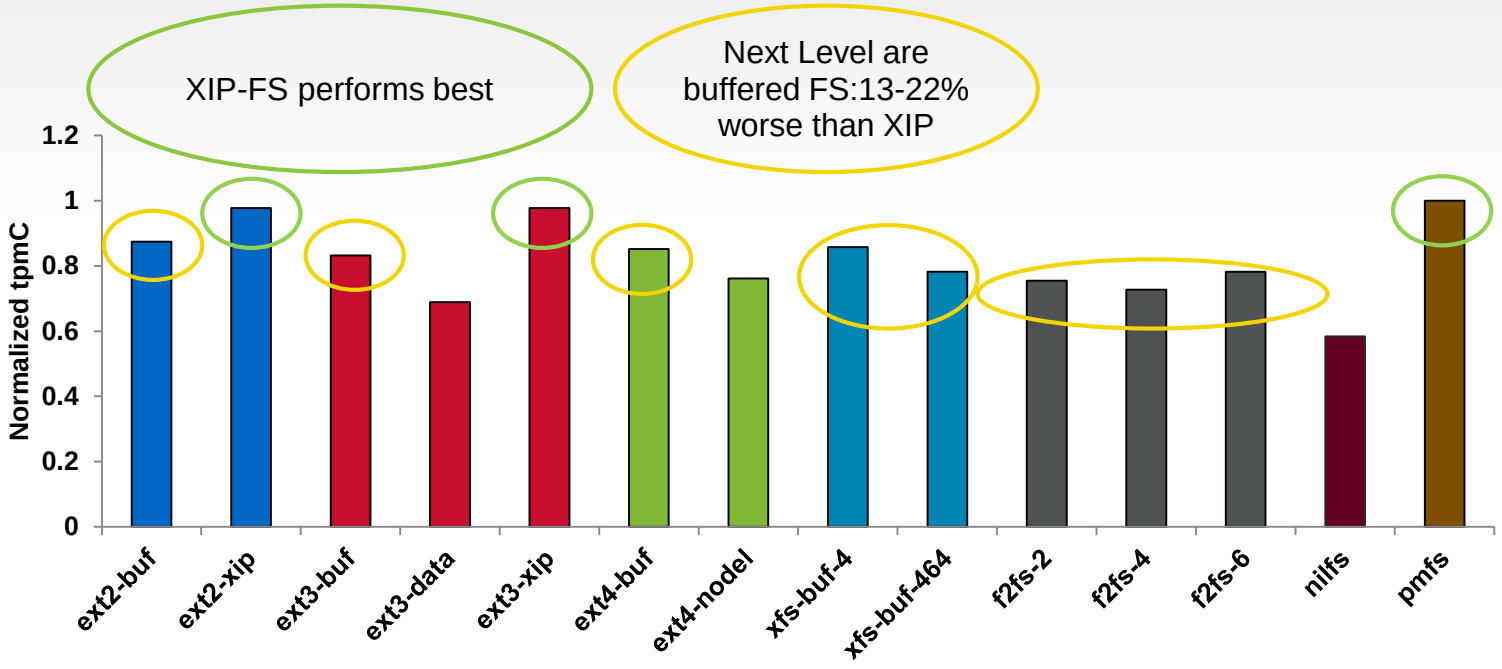
OLTP Database(TPC-C on MySQL)



400 warehouse, total size 38GB

Higher is better

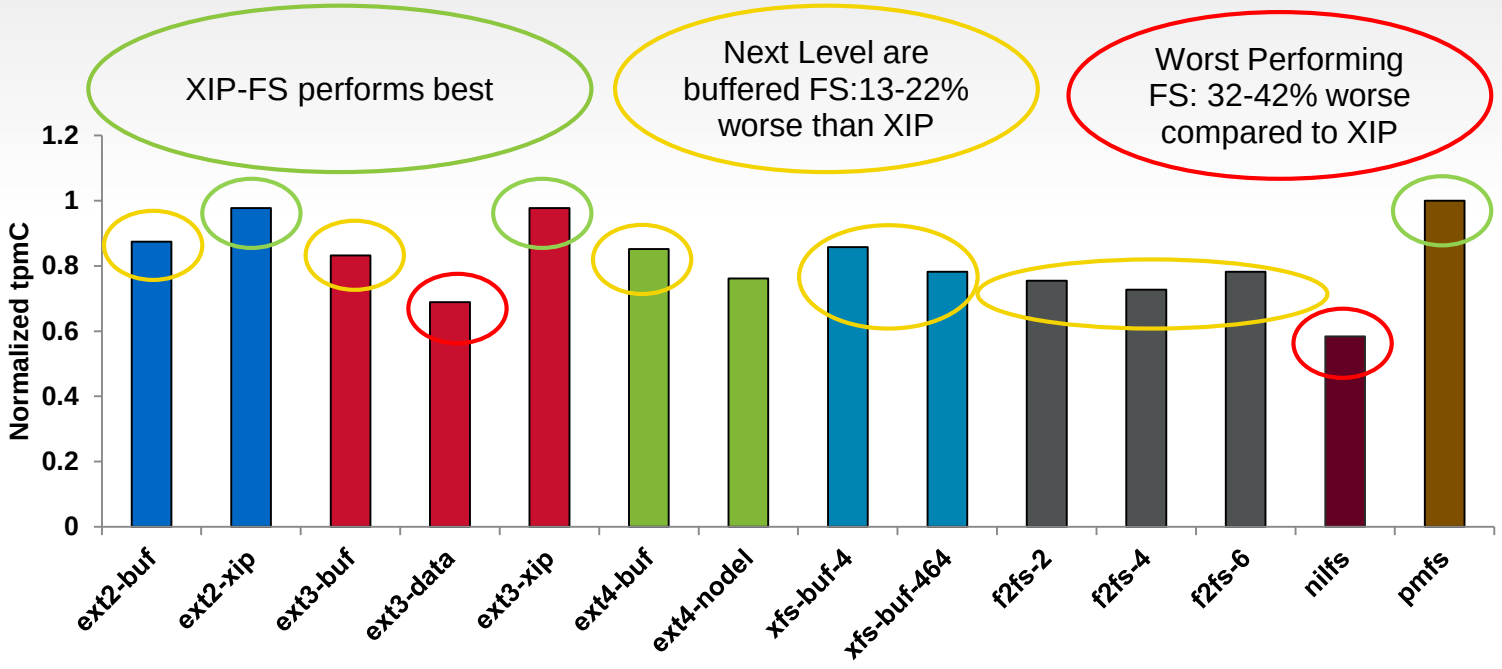
OLTP Database(TPC-C on MySQL)



400 warehouse, total size 38GB

Higher is better

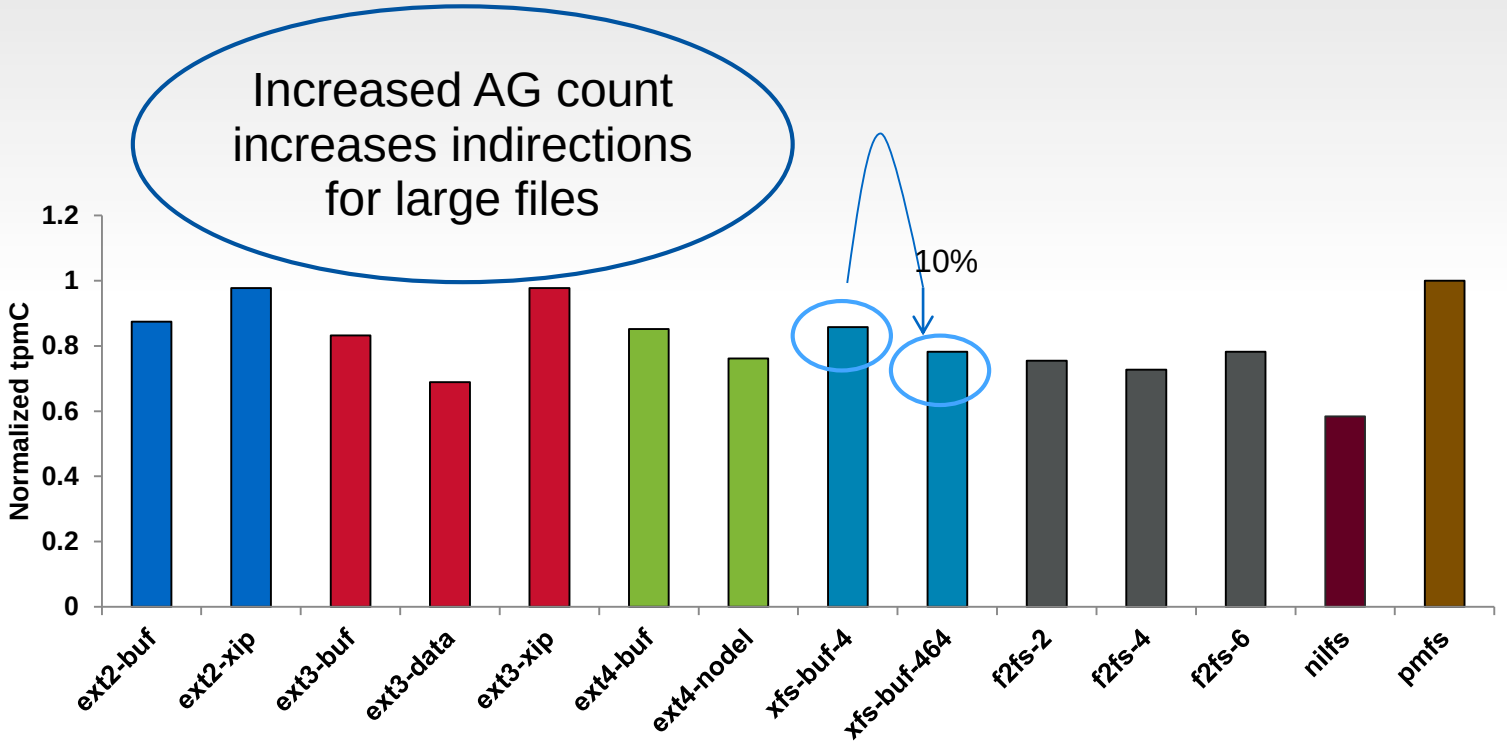
OLTP Database(TPC-C on MySQL)



400 warehouse, total size 38GB

Higher is better

OLTP Database(TPC-C on MySQL)

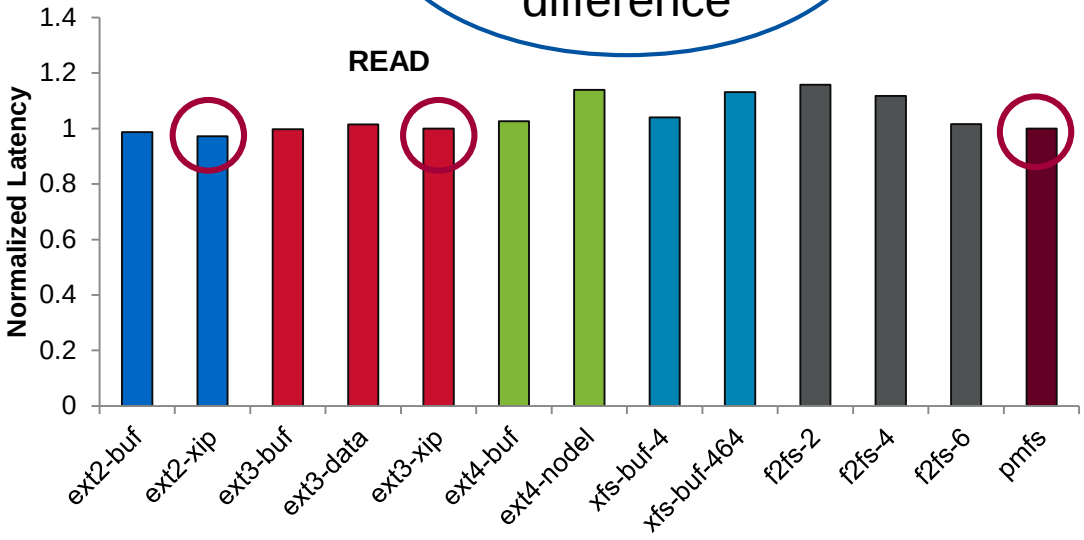


Higher is better

Key-Value Stores (YCSB on MongoDB) - Latency

WORKLOAD C

Reduced page faults blurs performance difference

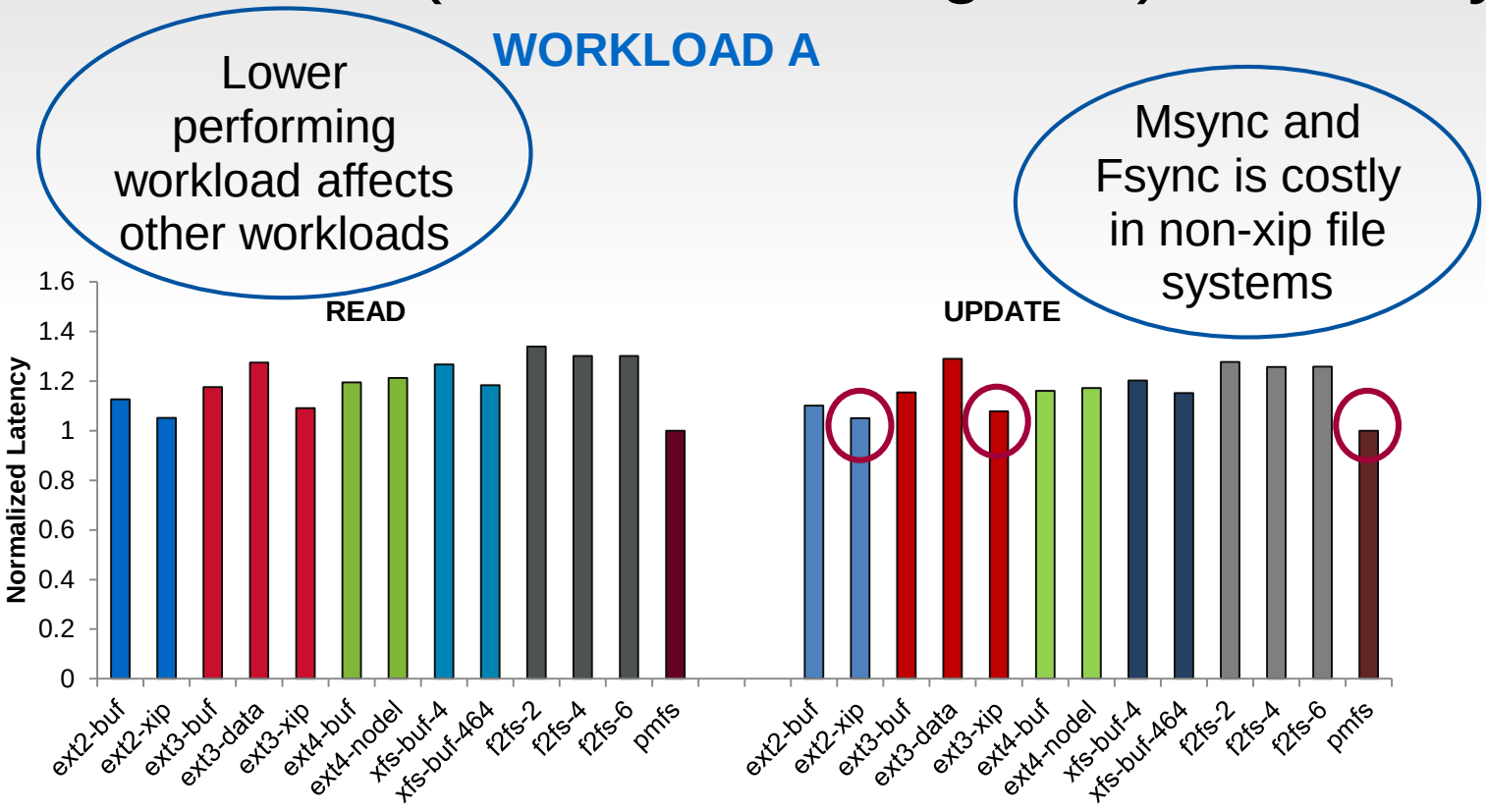


READ=100%

10 million records, total size 36GB

Lower is better

Key-Value Stores (YCSB on MongoDB) - Latency

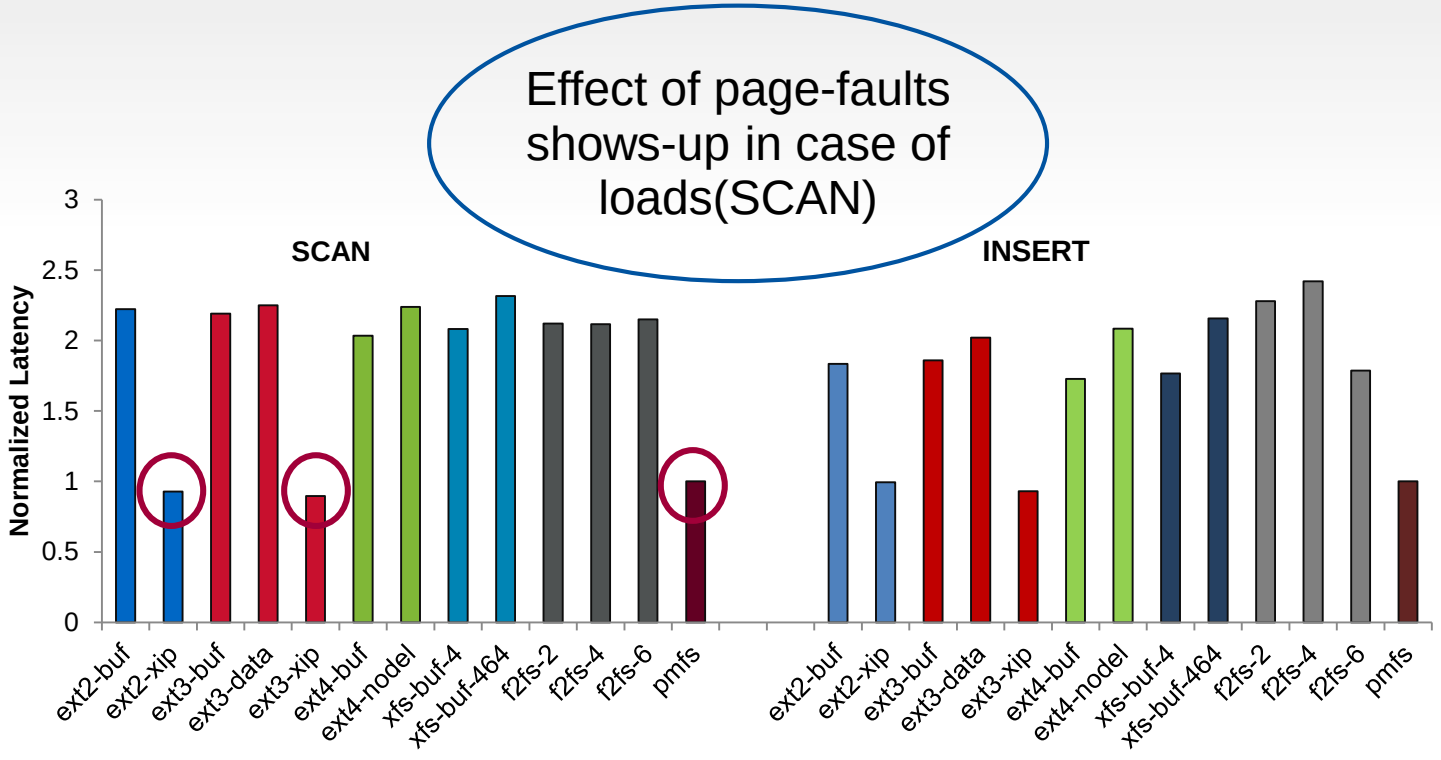


READ=50% UPDATE=50%

Lower is better

Key-Value Stores (YCSB on MongoDB) - Latency

WORKLOAD E



SCAN=95% INSERT=5%

Lower is better

Overview

- Related Work
- Experimental Methodology
- Experimental Results
- Recommendation and Conclusion

Recommendation and Conclusion

- Recommendation for traditional and new file systems
  - In-place update layout
  - Execute In Place
  - Simple and parallel allocation strategy
  - Fixed sized data blocks

Thank you