# Clotho: Transparent Data Versioning at the Block I/O Level

**Michail Flouris**
**Dept. of Computer Science**
**University of Toronto**
**flouris@cs.toronto.edu**

**Angelos Bilas**
**ICS- FORTH &**
**University of Crete**
**bilas@ics.forth.gr**

NASA/IEEE MSST 2004
12th NASA Goddard/21st IEEE Conference on
Mass Storage Systems & Technologies
The Inn and Conference Center
University of Maryland University College
Adelphi MD USA
April 13-16, 2004

# Advanced Storage Functionality

- Applications have increased requirements from storage systems
- Storage systems are required to provide advanced functionality in several areas
  - Reliability
  - Encryption
  - Compression
  - Quality Guarantees
  - Backup

# Online Storage Versioning

- Applications need to preserve many versions of data
- Advanced backup functionality motivated by current large & inexpensive disks
- Online versioning:
  - Perform continuous real-time versioning
  - Keep backup repositories on-line using disks
- Potential for lower administration overhead

# Previous Work

- Filesystem-level:
  - Elephant [SOSP '99], Plan 9 [Pike'90], CVFS [FAST '03], WAFL [Usenix'94], 3DFS [Usenix '92], Inversion FS [Usenix '93]
- Block-level: Petal [ASPLOS '96]
- Commercial Products:
  - EMC SnapView, Veritas FlashSnap, Sun Instant Image, etc.
- Not general-purpose and transparent
  - Require special filesystem or custom hardware

# Our goal

- Transparent & general-purpose versioning
  - Filesystem-agnostic versioning
  - Based on commodity hardware
- High performance (essential for online use)
- Provide versioning mechanisms not policies
  - Higher layers (e.g. daemons) implement policies:
    - Create new storage versions (when ?)
    - Delete storage versions (which ? when ?)
    - Compress storage versions (which ? when ?)

# Our approach: Examine versioning at block-level

- Satisfies our goal
- Timely & in-line with technology trends
  - Pushes functionality closer to disk
    - Can offload processing to storage back-end
  - Low-level
    - Less complexity
    - Facilitates firmware implementations (e.g. self-versioned disks)
  - Storage systems operate at block-level

# Clotho: Issue Overview

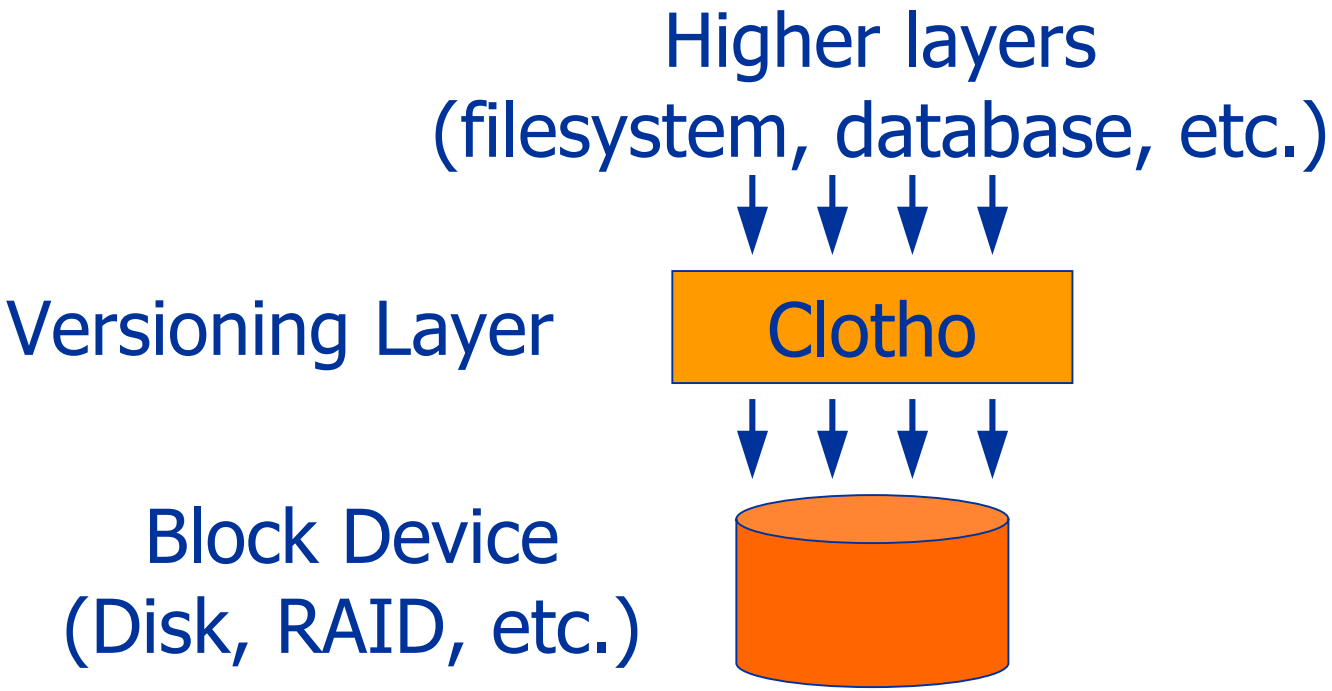
- ✓ Motivation
- System Design
  - Transparency & Flexibility
  - Performance
  - Overheads: Memory & Disk Space
  - Snapshot consistency
- Evaluation
- Conclusions

# Transparency & Flexibility (I)

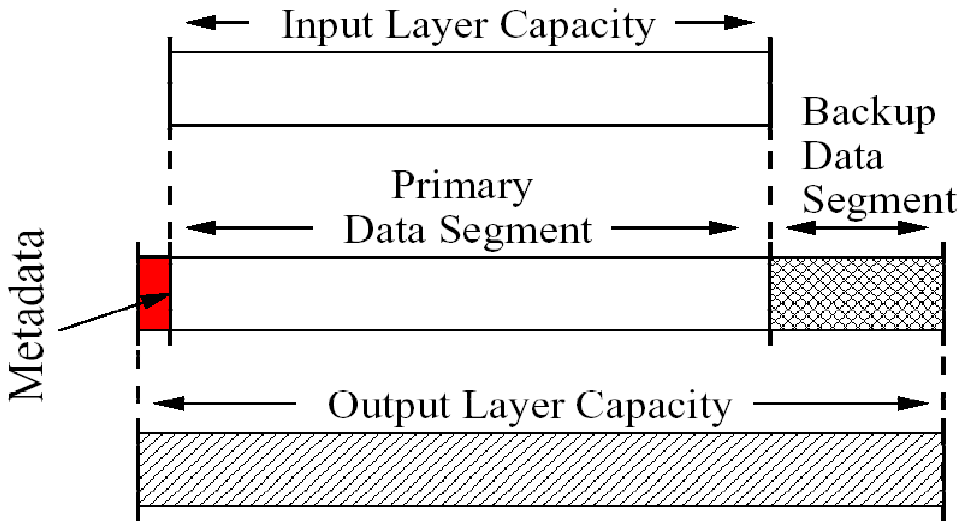
- Clotho is a versioning layer in the OS I/O hierarchy
- Appears as a virtual block device
- Versions whole block volumes (snapshots)
- Read-only old volume versions
  - Appear as new virtual block devices
  - Accessible in parallel with the latest version
  - Practically unlimited versions

# Transparency & Flexibility (II)

- Clotho can be inserted arbitrarily in an I/O hierarchy over any other block device



# Disk Space Management



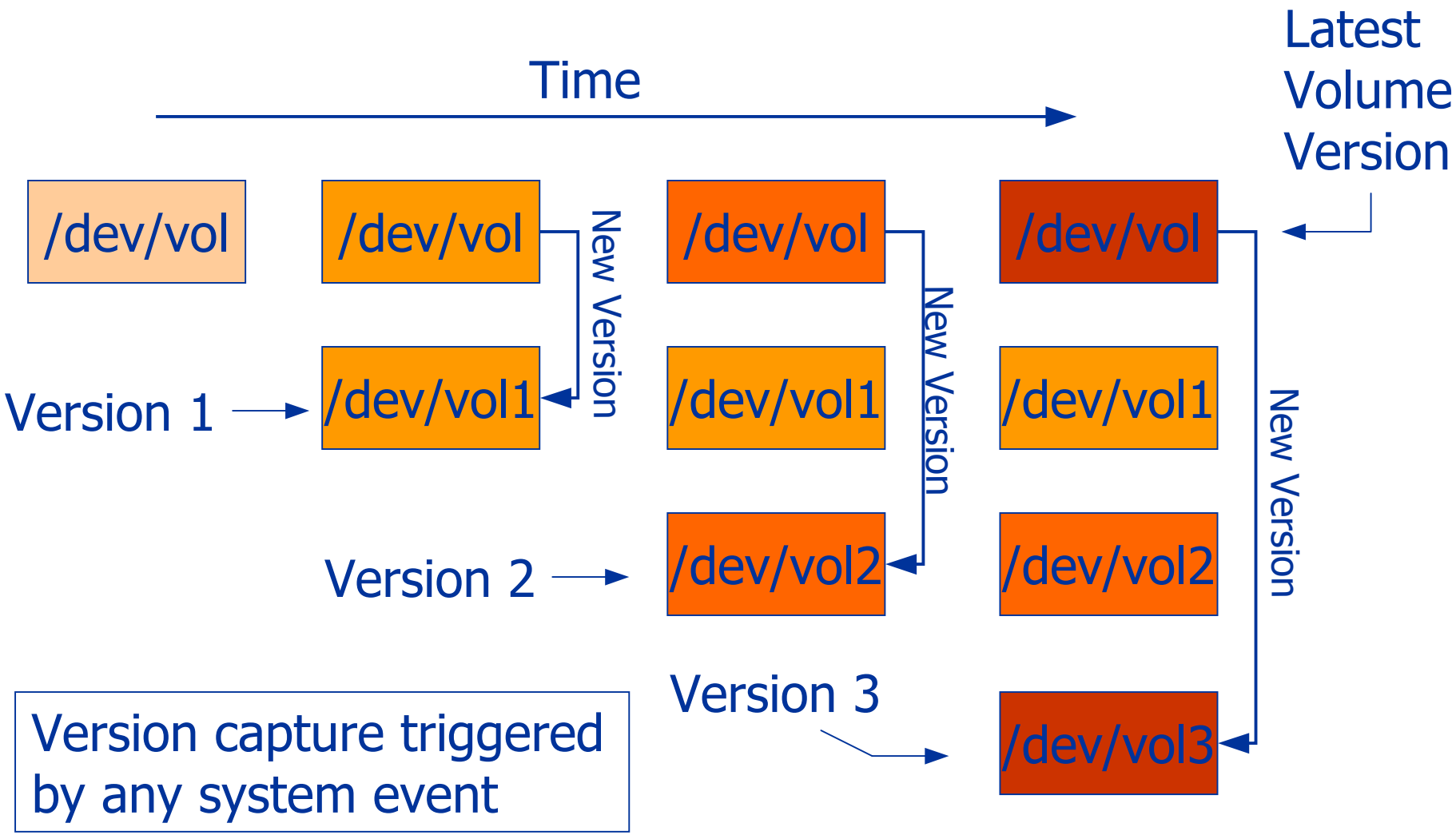

Input Capacity

Logical Segments

Output Device Capac.

- Reserving backup capacity from higher layer
- Data Capacity guaranteed
- Need metadata to index backup blocks

# Volume Evolution



# Clotho: Issue Overview

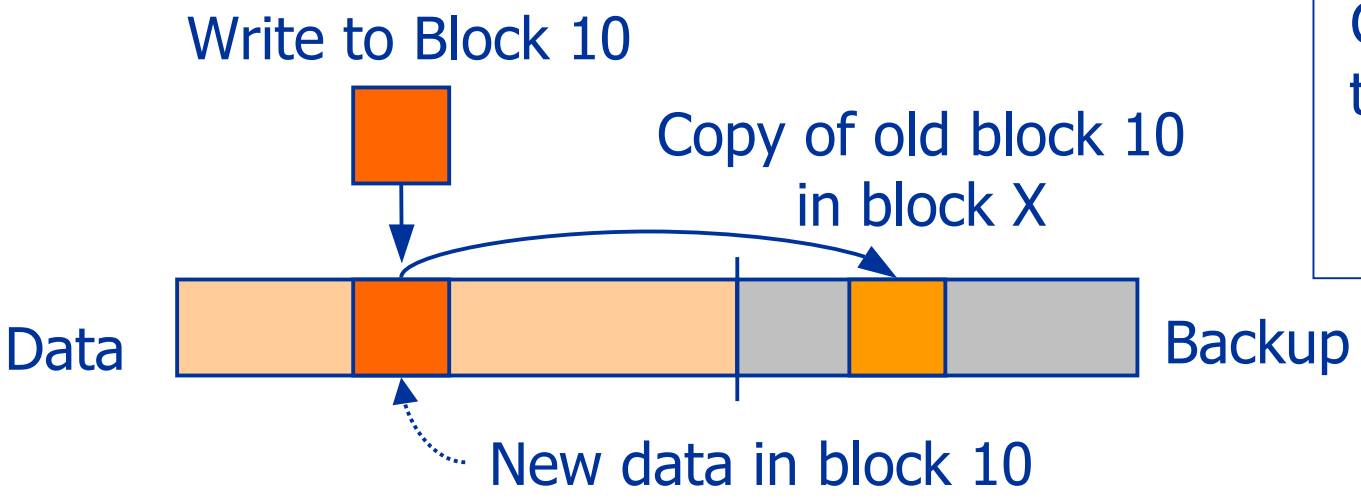
- ✓Motivation
- ✓System Design
  - ✓Transparency & Flexibility
  - – Performance
  - – Overheads: Memory & Disk Space
  - – Snapshot consistency
- Evaluation
- Conclusions

# Low Overhead Snapshots

- Issue: Creating volume snapshots fast !
- Clotho services a stream of I/O requests
  - Need to make a versioning decision per write request (read requests do not modify data)
- Clotho uses a “latest version” counter and a version number per block
  - On writes check the two counters to decide
  - Create new snapshot: simply increase the “latest version” counter
  - No overhead : simple pointer manipulation

# Handling Writes (I)

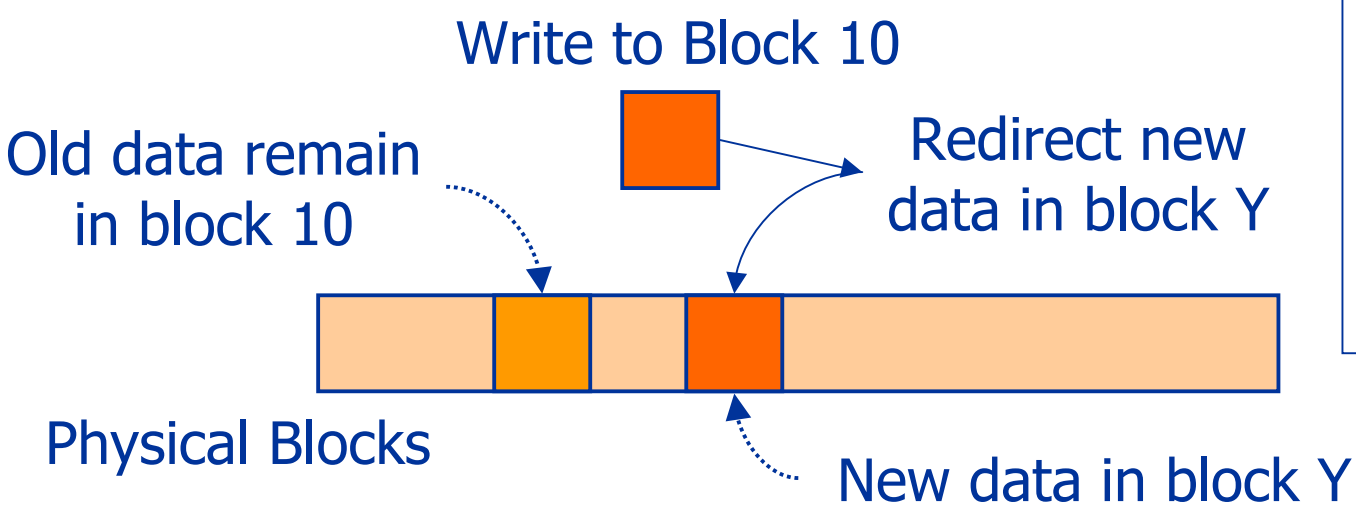
- Issue: Creating new block versions fast !
- First method: Copy-on-write
  - No layout change for user volume
  - Slow (introduces a copy)




Algorithm:
Copy old data,
then overwrite
the old block
in place

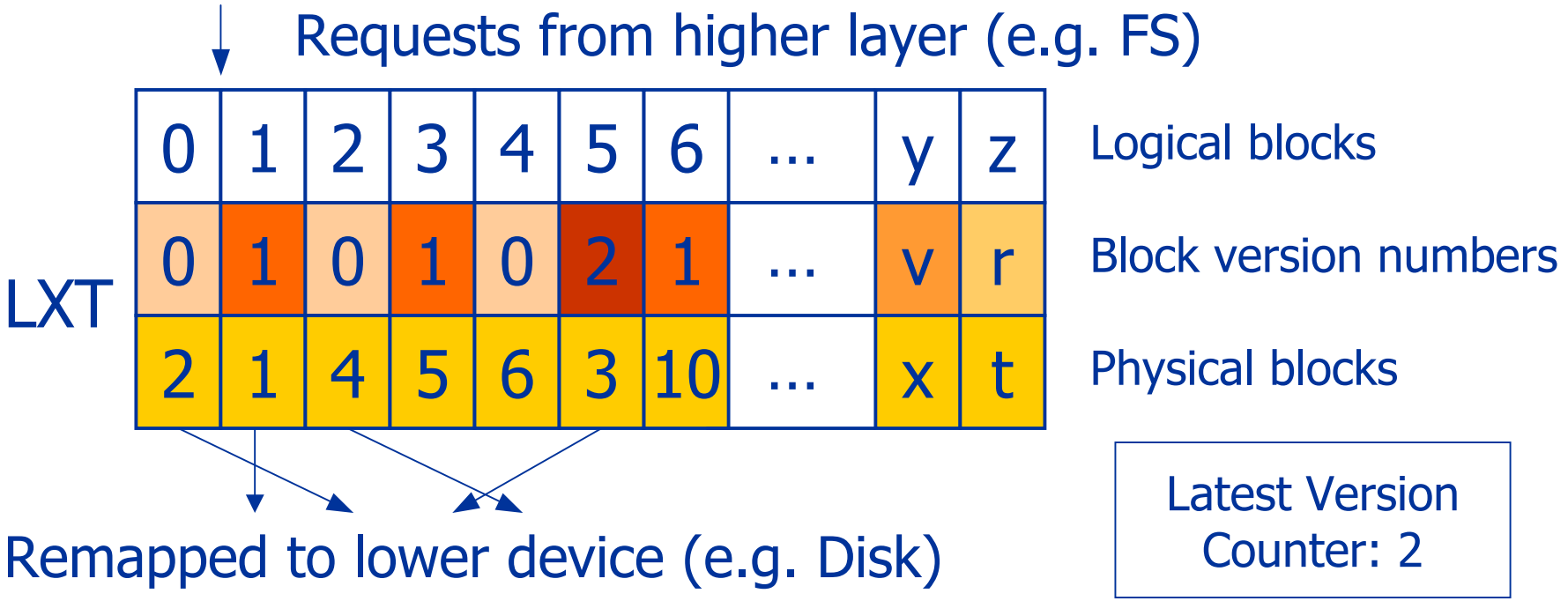
# Handling Writes (II)

- Second method: Remapping-on-write (Clotho)
  - Low overhead (no copy)
  - Changes block layout (use own layout algorithm)
- Augmenting existing logical-to-physical block translation table




Algorithm: Redirect new data to free block, keep old block in place

# Block Translation




- Extending existing block translation table
- Versioning granularity: One block
- Clotho uses scan-based layout algorithm

# Clotho: Issue Overview

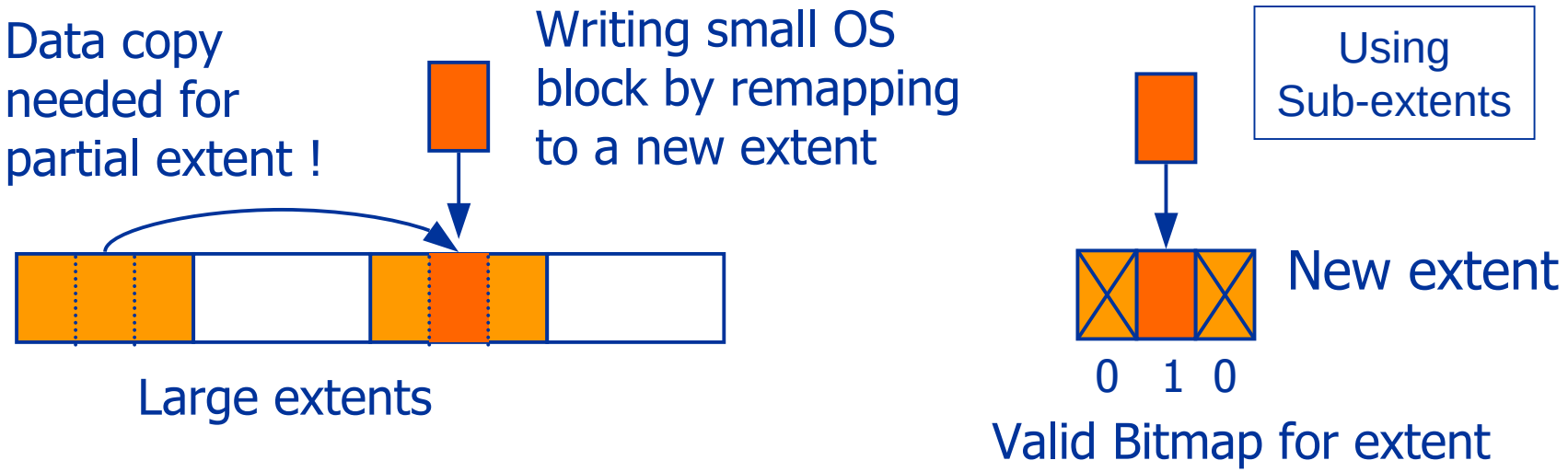
- ✓Motivation
- ✓System Design
  - ✓Transparency & Flexibility
  - ✓Performance
  - – Overheads: Memory & Disk Space
  - – Snapshot consistency
- Evaluation
- Conclusions

# Memory Overhead

- Clotho services I/O requests with the OS block size
- Issue: Small OS block size increases metadata size
  - Metadata are cached in memory
  - Metadata size depends on capacity & block size
  - 4KByte blocks require 4MB RAM per GByte of disk
- Solution: Clotho uses internal blocks (“extents”) to transparently minimize metadata memory usage
- Large “extents” reduce the metadata size
  - 32KB extents require 500KB per GByte of disk space (for keeping all the metadata in memory)

# Partial Update Issue

- Issue: external (OS) block size smaller than the "extent" size causes data copy
- Solution: use "sub-extents" equal to the external block size and valid bitmap



# Disk Space Overhead

- Storing many or all data versions requires much disk space (naturally !)
- Clotho minimizes disk usage by applying differential compression
  - Deltas/diffs between consecutive block versions
- Compact versions are accessible on-line
  - Full blocks reconstructed dynamically on reads
- Issue: Need new block mapping
- Solution: Storing many diffs in a normal block

# Snapshot consistency (I)

- Snapshots are raw volume images at a point in time
- Issue: OS & higher layer buffering may lead to snapshot inconsistencies
  - E.g. Unsynchronized system or filesystem buffers at snapshot capture time
- Solution: Clotho flushes all system and filesystem buffers before snapshot capture

# Snapshot Consistency (II)

- Filesystem specific issue
  - Some files are open at snapshot capture time
  - Clotho stores open file list on snapshot
- Application buffer issue
  - When block-level applications cache I/O buffers
  - When applications use specialized consistency rules (e.g. atomic set of I/O operations)
  - Versioning must be triggered by applications or recovery mechanisms applied

# Clotho: Issue Overview

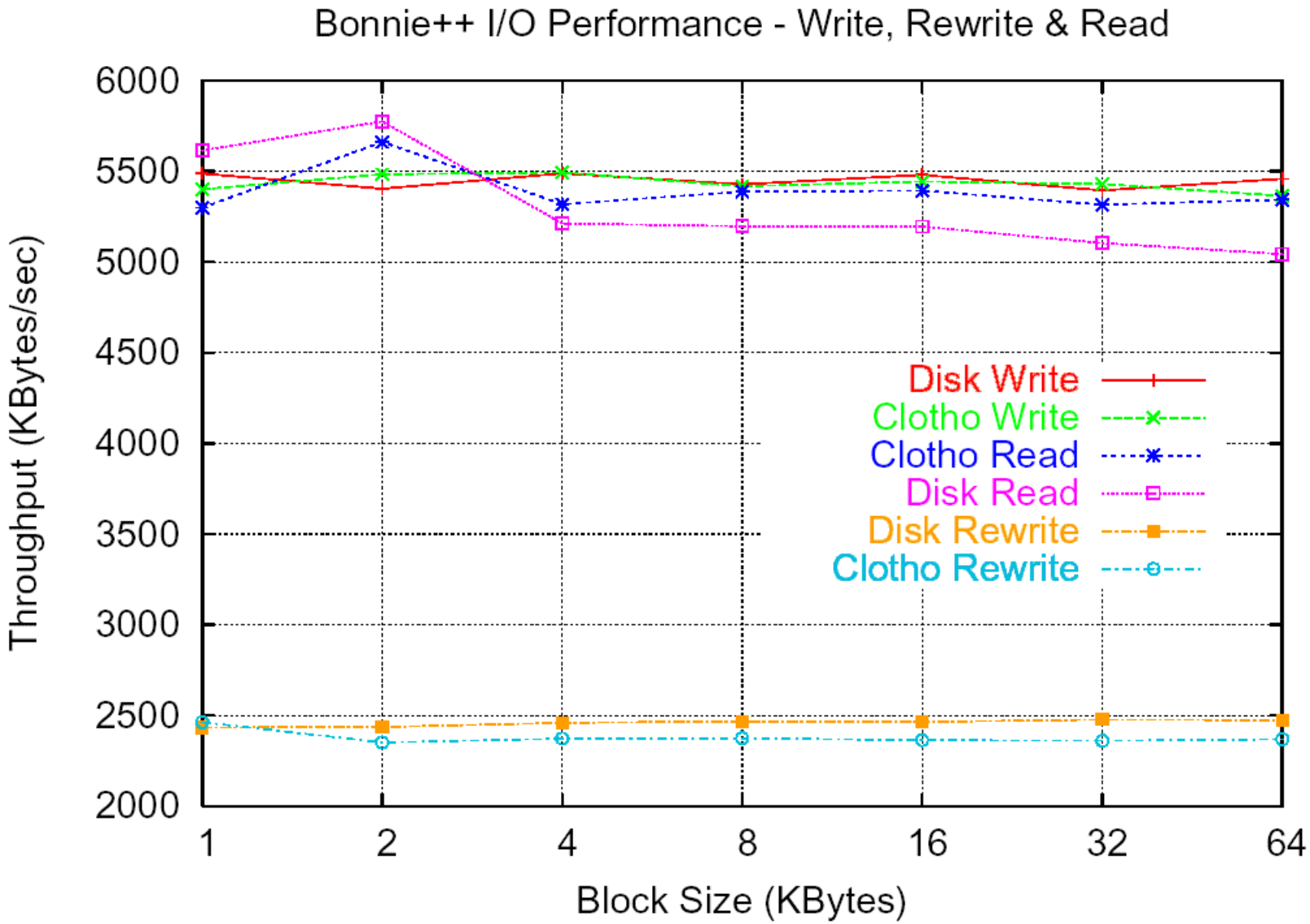
- ✓Motivation
- ✓System Design
  - ✓Transparency & Flexibility
  - ✓Performance
  - ✓ Overheads: Memory & Disk Space
  - ✓Snapshot consistency
- Evaluation
- Conclusions

# Implementation

- Linux 2.4 block device driver (~6,000 lines)
- Loadable kernel module
- Implementation uses kernel I/O request remapping techniques
  - As in LVM and Linux RAID
- Ioctl() interface for custom commands
- Exposes state through the /proc interface

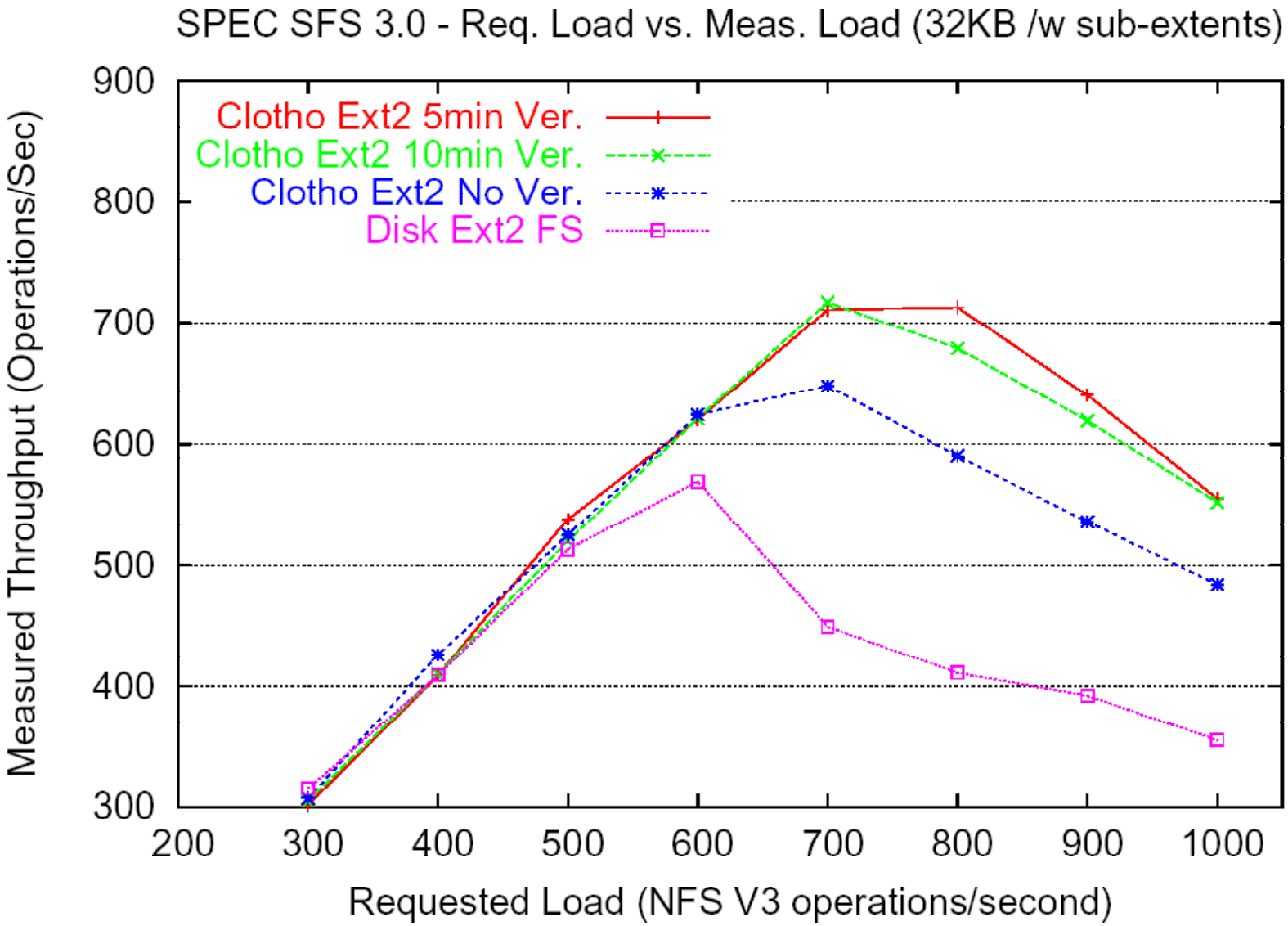
# Evaluation

- Interested in:
  - Common path performance (I/O + versioning)
  - Read performance on compact versions
- Platform: 2x P3 866MHz, 768MB RAM, 100MBps NIC, IBM 45GB (7200rpm) disk
- OS: Linux 2.4.18
- Filesystems measured on top of Clotho:
  - Linux Ext2 FS
  - Reiser FS
- FS Benchmarks: Bonnie, SPEC SFS 3.0

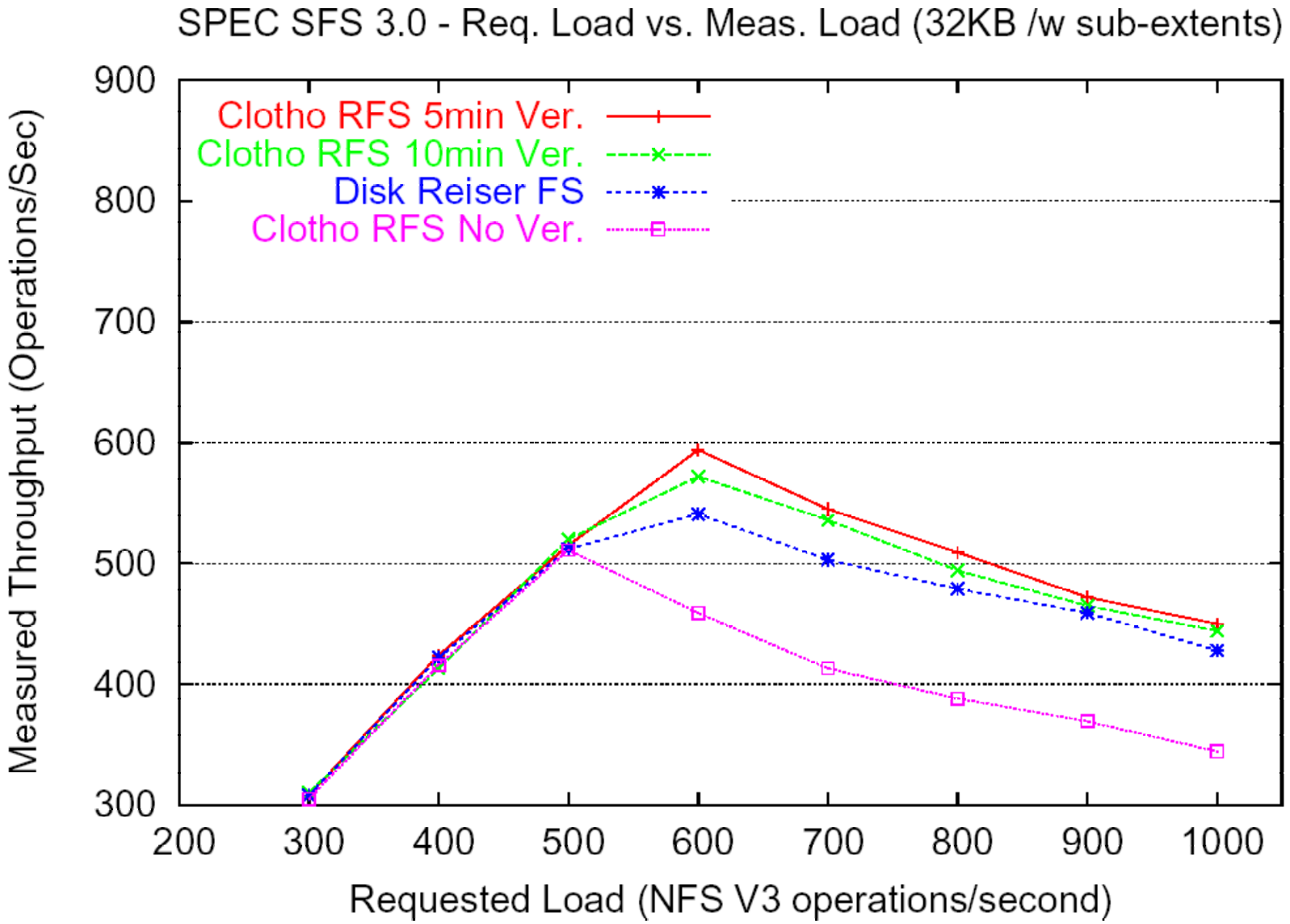


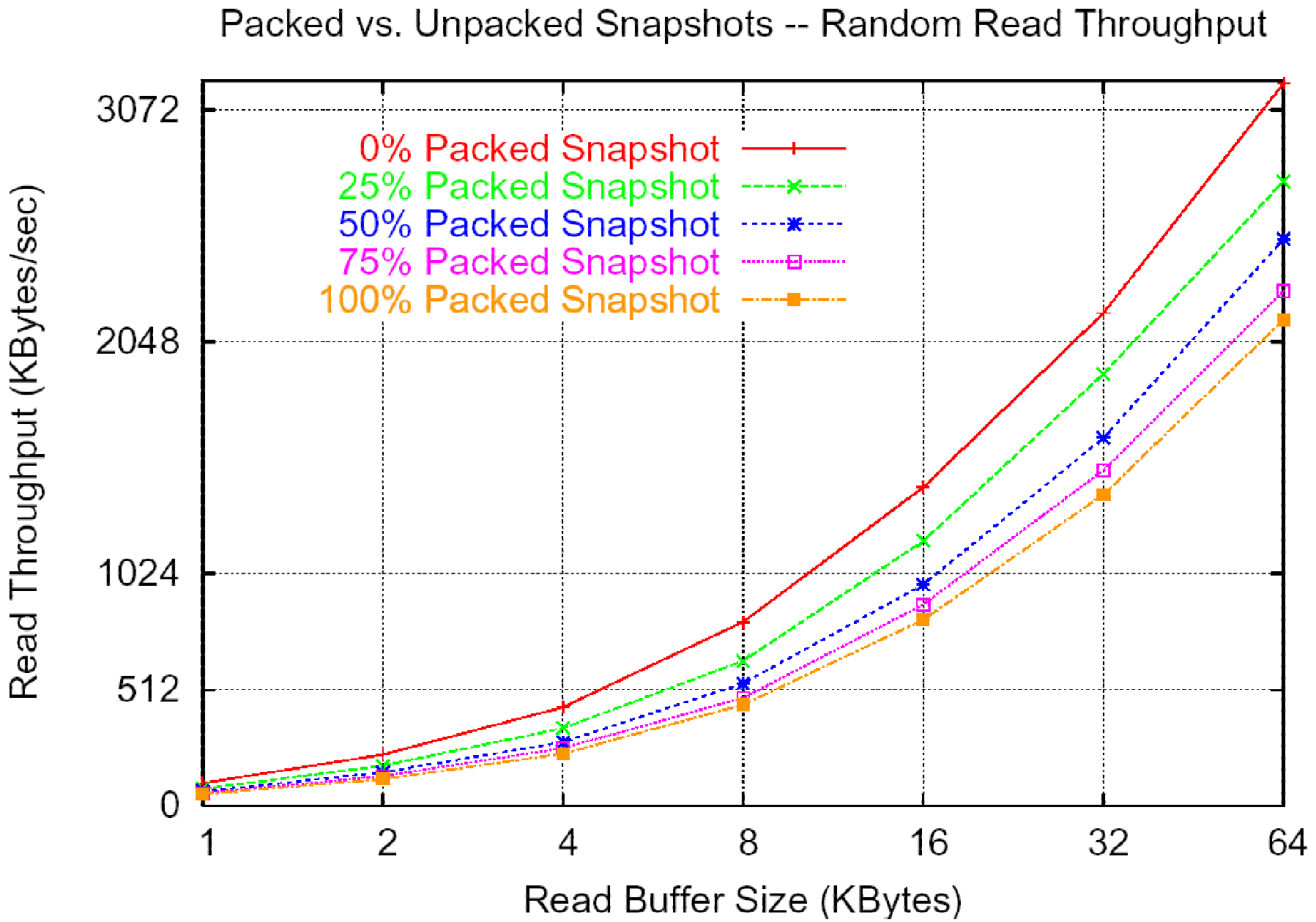

# 2GB data, No versions, 4KB extents, Ext2 FS

# Linux Ext2 FS Results



# Linux Reiser FS Results




Packed vs. Unpacked Snapshots -- Random Read Throughput
Read Throughput (KBytes/sec)
3072
2048
1024
512
0
1
2
4
8
16
32
64
Read Buffer Size (KBytes)
0% Packed Snapshot
25% Packed Snapshot
50% Packed Snapshot
75% Packed Snapshot
100% Packed Snapshot

# Limitations

- Issue if higher layer (e.g. FS) assumes certain disk layout
- No support for multiple devices (version capture must be synchronized across devices)

# Conclusions

- On-line storage versioning at the block level
  - Feasible and efficient
- Main advantage
  - Transparent and general-purpose
- Challenges
  - Flexibility & Transparency
  - Performance
  - Main memory & Disk overheads
  - Snapshot consistency

# Questions ?

Paper title:

"Clotho: Transparent Data Versioning at the Block I/O Level"

Authors:

- Michail Flouris, flouris@cs.toronto.edu
  Web: www.eecg.toronto.edu/~flouris

- Angelos Bilas, bilas@ics.forth.gr
  Web: www.ics.forth.gr/~bilas