

# Active Data Systems

A. L. Narasimha Reddy  
Department of Electrical and Computer Engineering  
Texas A & M University

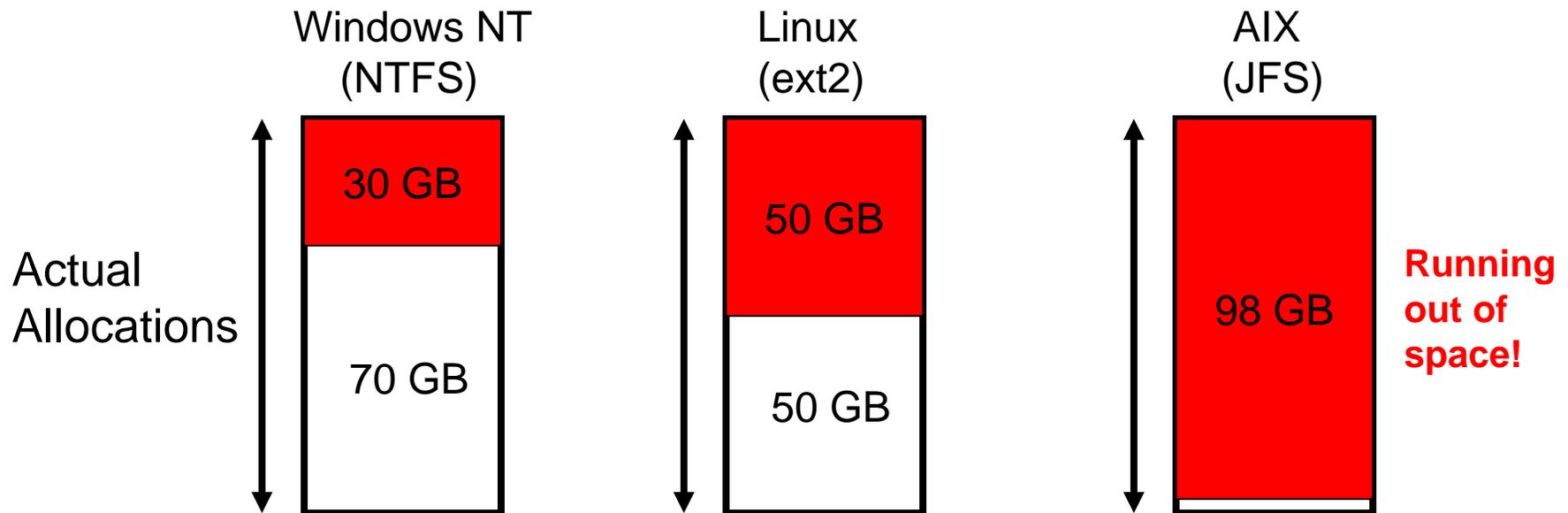
Students: Sukwoo Kang (now at IBM Almaden)  
John Garrison

# Outline

- **Flexible Storage Allocation**
  - Motivation
  - Design of Virtual Allocation
  - Evaluation
- **Conclusion**

# Storage Allocation

- Allocate entire storage space at the time of the file system creation
- Storage space owned by one operating system cannot be used by another

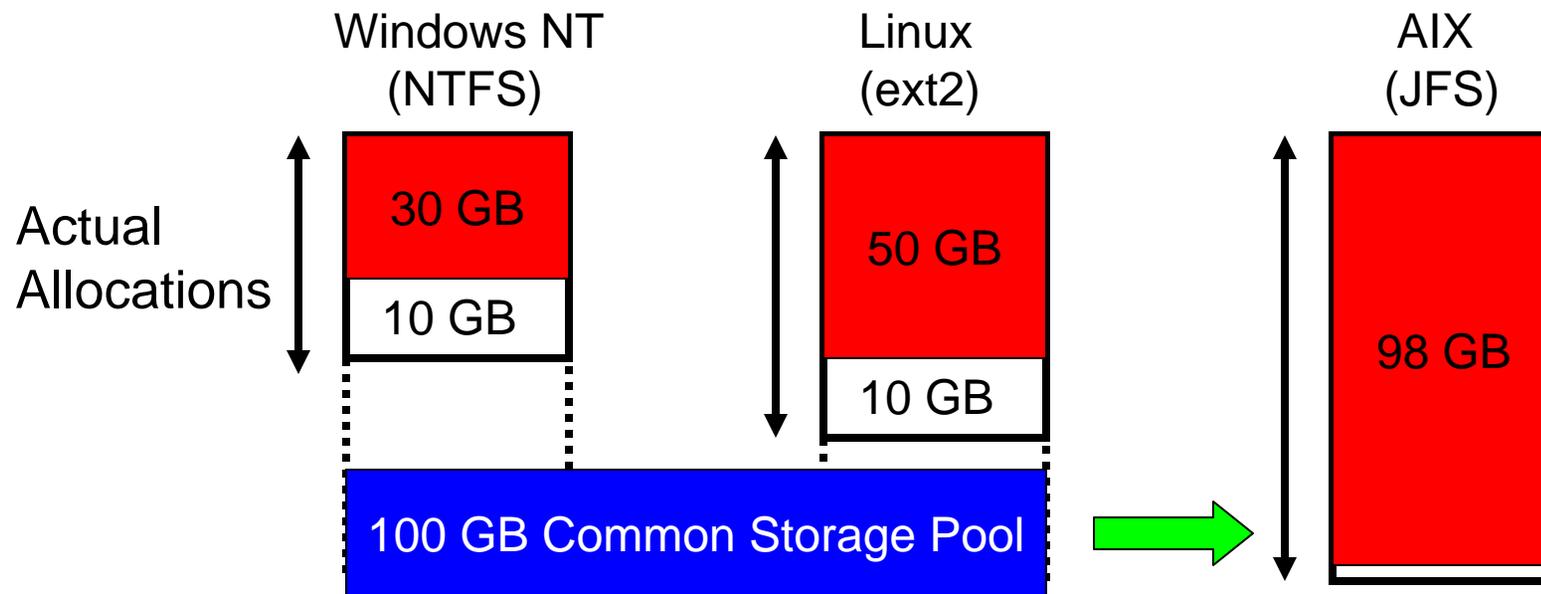


# Big Picture

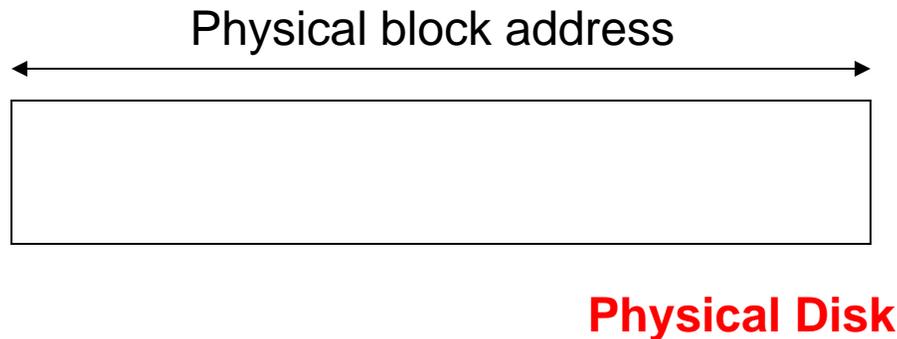
- **Memory systems employ virtual memory for several reasons**
- **Current storage systems lack such flexibility**
- **Current file systems allocate storage statically at the time of their creation**
  - Storage allocation: Space on the disk is not allocated well across multiple file systems

# File Systems with Virtual Allocation

- **When a file system is created with  $x$  GB,**
  - Allows the file system to be created with only  $y$  GB, where  $y \ll x$
  - Remaining space used as one common available pool
  - As the file system grows, the storage space can be allocated on demand

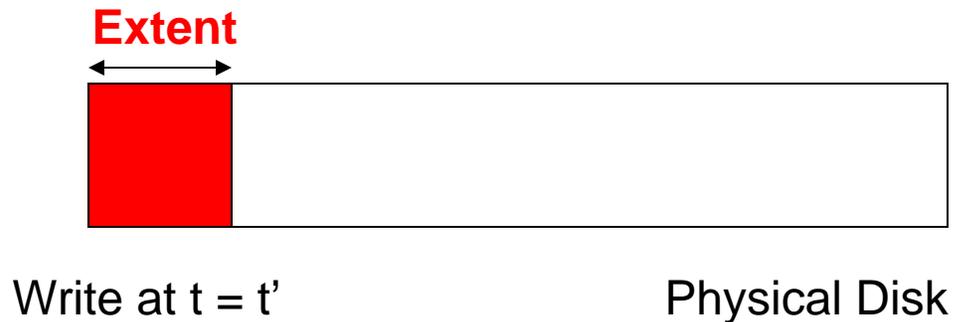


# Our Approach to Design



- **Employ *Allocate-on-write* policy**
  - Storage space is allocated when the data is written
  - Writes all data to disk sequentially based on the time at which data is written to the device
  - Once data is written, data can be accessed from the same location, i.e., data is *updated in-place*

# Allocate-on-write Policy



- **Storage space is allocated by the unit of the extent when the data is written**
- ***Extent* is a group of file system blocks**
  - Fixed size
  - Retain more spatial locality
  - Reduce information that must be maintained

# Allocate-on-write Policy

Write at  $t = t''$  (where  $t'' > t'$ )

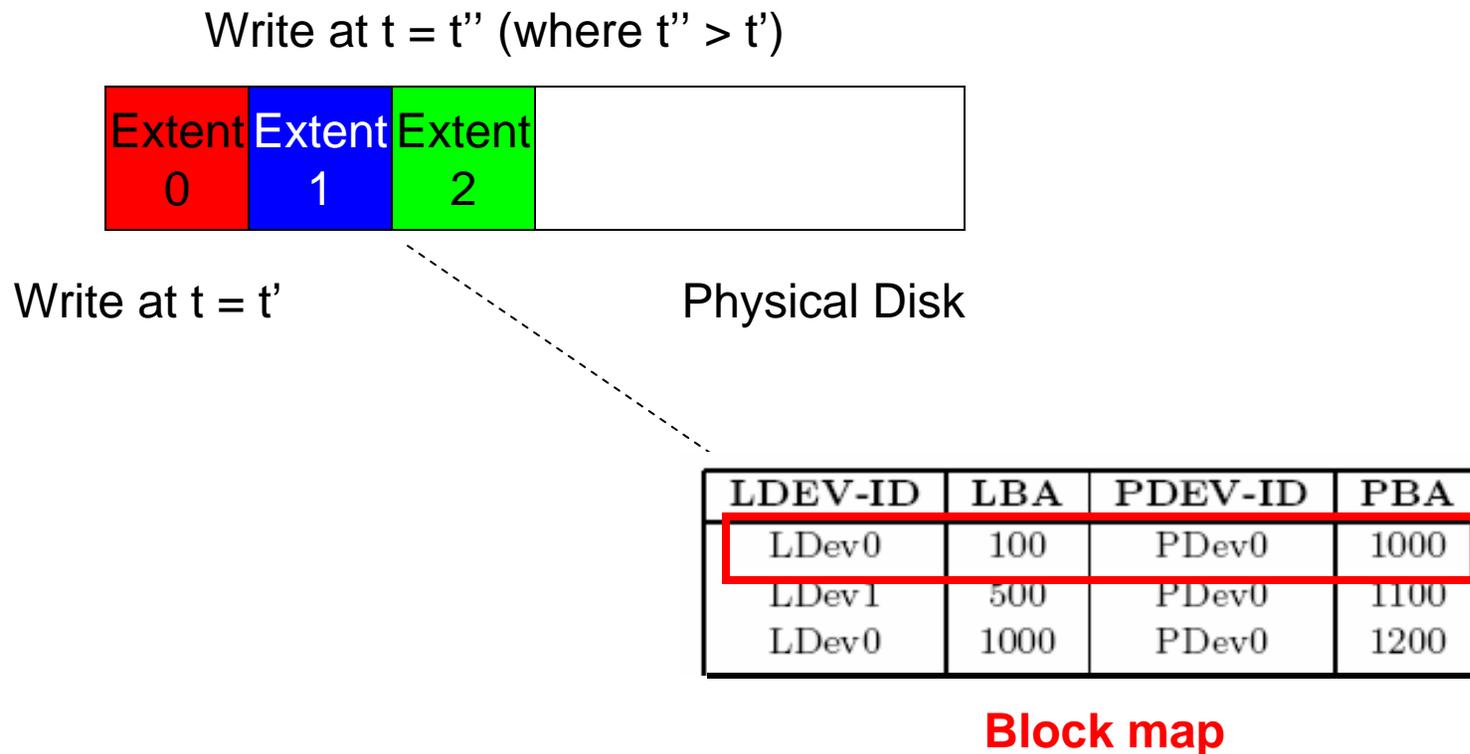


Write at  $t = t'$

Physical Disk

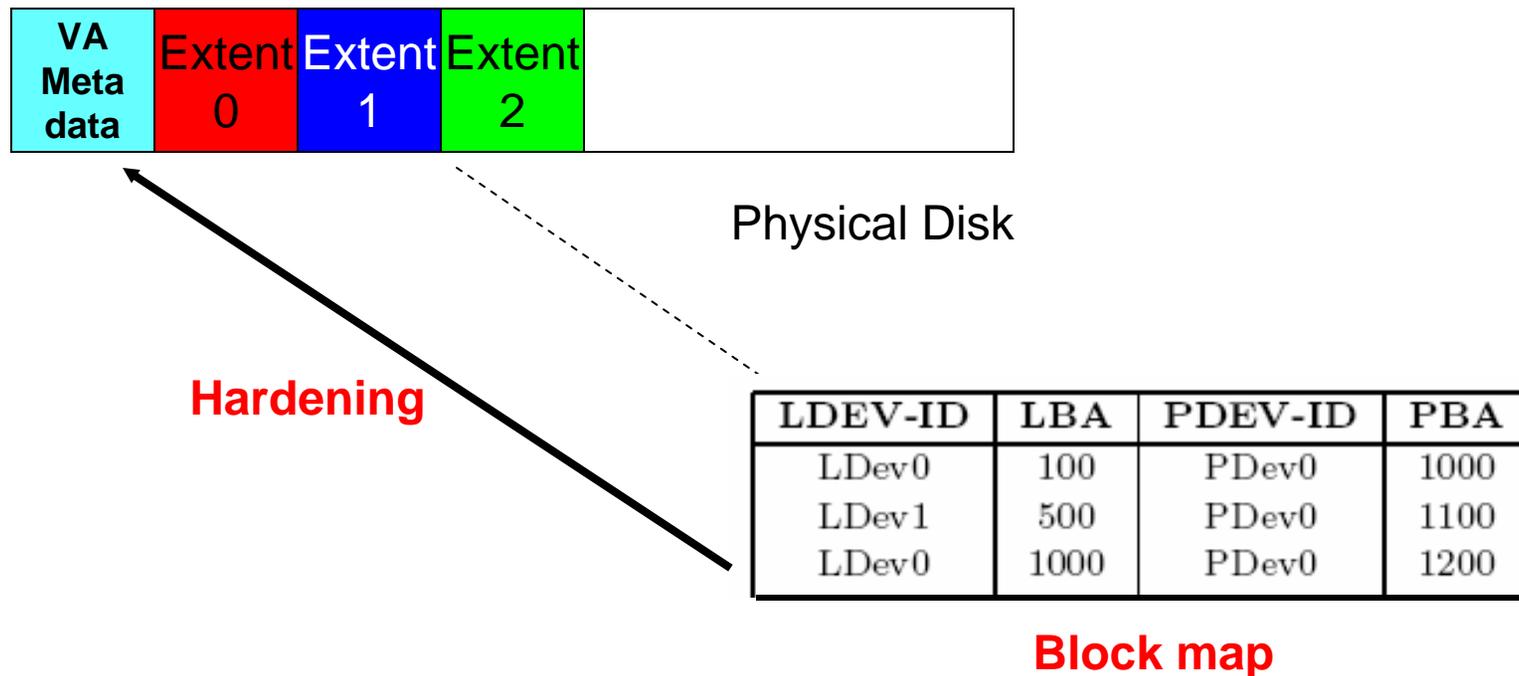
- **Data is written to disk sequentially based on write-time**
  - Further writes to the same data updated in-place
  - *VA (Virtual Allocation)* requires additional data structure

# Block Map



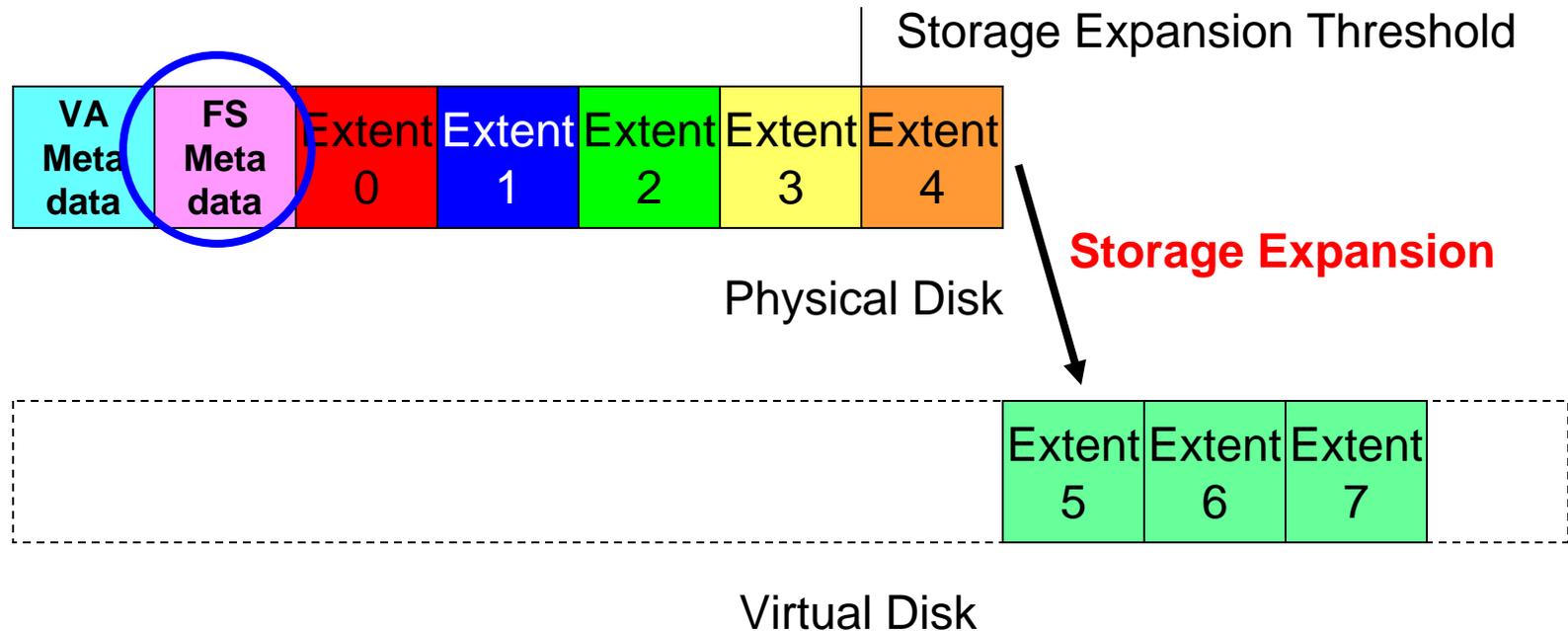
- ***Block map*** keeps a mapping of logical storage locations and real (physical) storage locations

# VA Metadata



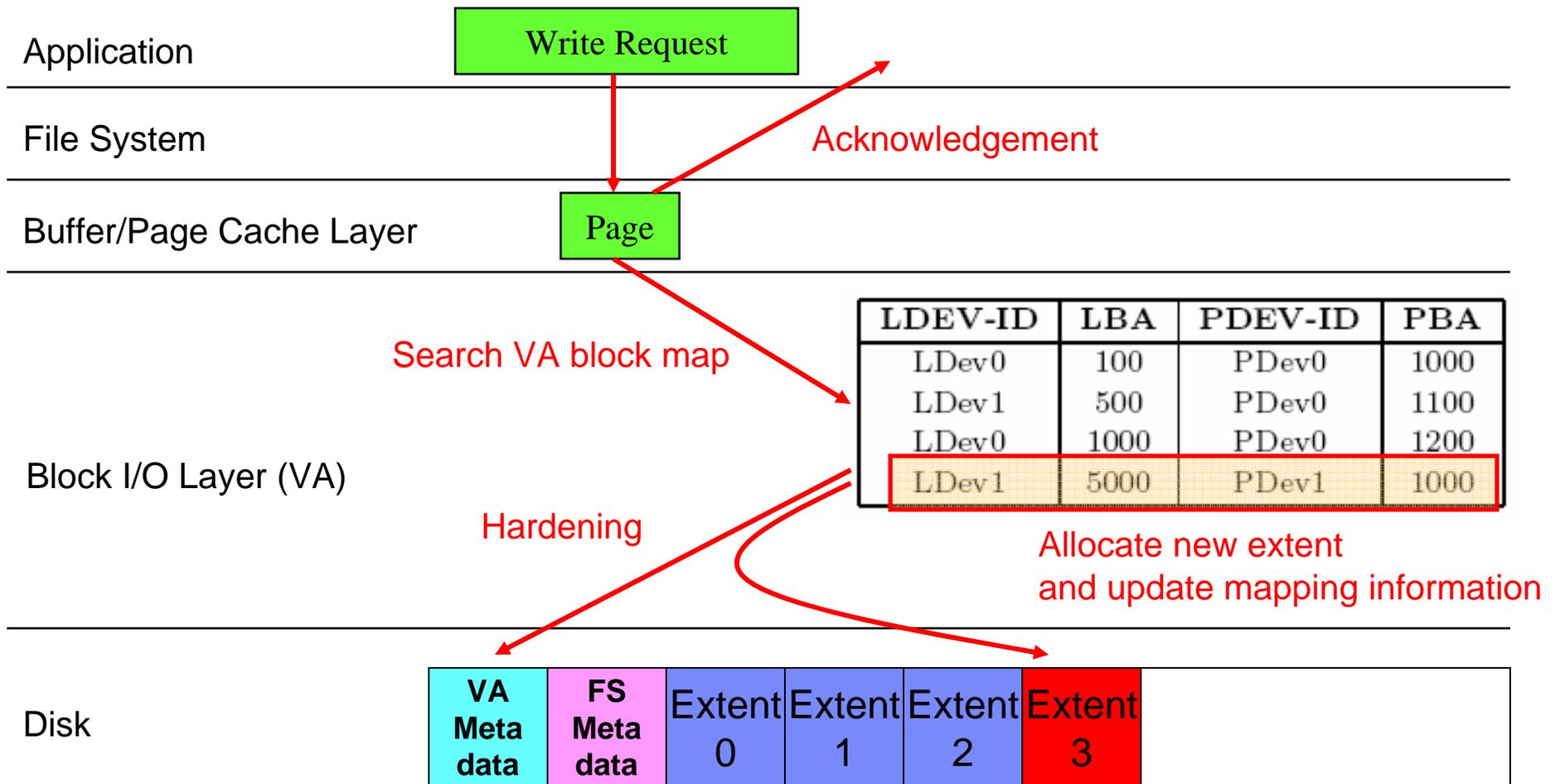
- This block map is maintained in memory and regularly written to disk for hardening against system failures
- *VA Metadata* represents the on-disk block map

# On-disk Layout & Storage Expansion

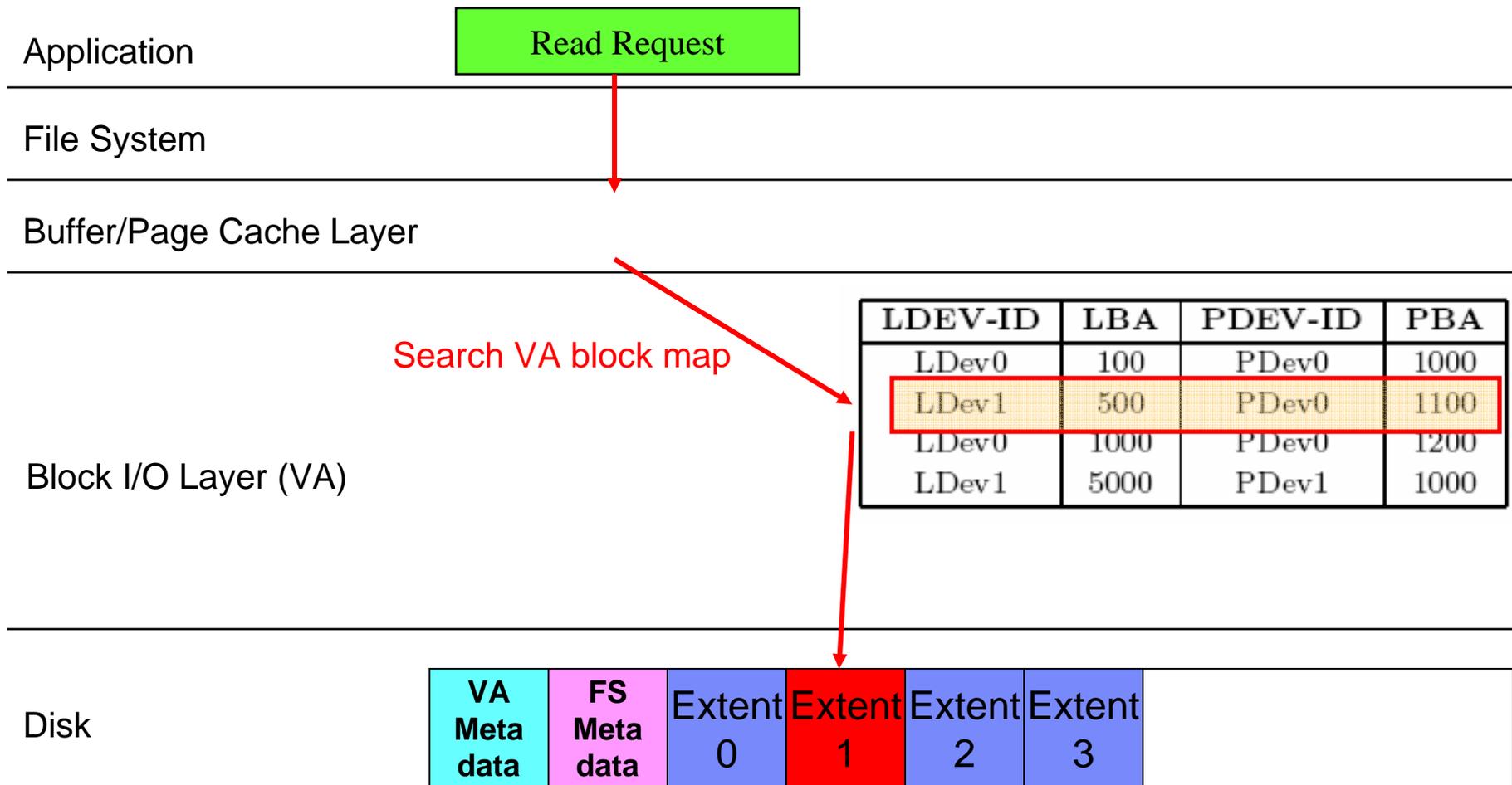


- **When the capacity is exhausted or reaches *storage expansion threshold*, a physical disk can be expanded to other available storage resources**
  - File system unaware of the actual space allocation and expansion

# Write Operation



# Read Operation



## Allocate-on-write vs. Other Work

- **Key difference from log-structured file systems (LFS)**
  - Only allocation is done at the end of log
  - Updates are done *in-place* after allocation

	Allocation	Update
LFS	dynamic	not in-place
FS	static	in-place
FS w/ VA	dynamic	in-place

- **LVM still ties up storage at the time of file system creation**

	Block map
LVM	static
FS w/ VA	dynamic

# Design Issues

- **VA Metadata Hardening (File System Integrity)**
  - Must keep certain update ordering of VA metadata and FS (meta)data

- **Extent-based Policy Example (with Ext2)**

- $I$  (inode),  $B$  (data block),  $V$  (VA block map)
- $A \rightarrow B$  ( $B$  is allocated to  $A$ )

Operation	In-memory	On-disk
1. $I$ alloc data block	$I \rightarrow B$	
2. $B$ write to disk		
2-1. $V$ alloc extent	$V \rightarrow B$	
2-2. $V$ write to disk		$V$ written
2-3.		$B$ written

- **File system-based Policy Example (with Ext3 *ordered* mode)**

Operation	In-memory	On-disk
1. $I$ alloc data block	$I \rightarrow B_1$	
2. $I$ alloc data block	$I \rightarrow B_2$	
3. $I$ write to journal		
3-1. $B_1$ write to disk		
3-2. $V$ alloc extent	$V \rightarrow B_1$	
3-3. $B_2$ write to disk		
3-4. $V$ alloc extent	$V \rightarrow B_2$	
3-5.		$B_1$ written
3-6.		$B_2$ written
3-7. $V$ write to disk		$V$ written
3-8.		$I$ written
4. $I$ write to disk		$I$ written

## Design Issues (cont.)

- **Extent Size**

- Larger extent size: Reduce block map size, retain more spatial locality, cause data fragmentation

- **Reclaiming allocated storage space of deleted files**

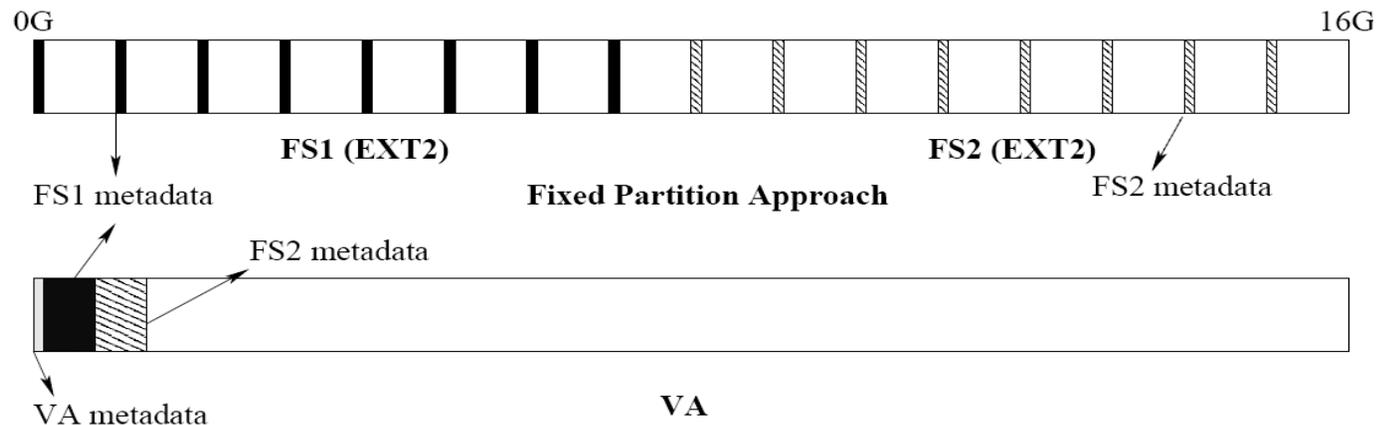
- Needed to continue to provide the benefits of virtual allocation
- Without reclamation, possible to turn virtual allocation into static allocation

- **Interaction with RAID**

- RAID remaps blocks to physical devices to provide device characteristics
- VA remaps blocks for flexibility
- Need to resolve performance impact of VA's extent size and RAID's chunk size

# Spatial Locality Observations & Issues

- **Metadata and data separation**
- **Data clustering: Reduce seek distance**
- **Multiple file systems**



- **Data placement policy**
  - Allocate hot data in a high data region of disk
  - Allocate hot data in the middle of the partition

# Implementation & Experimental Setup

- **Virtual allocation prototype**

- Kernel module for Linux 2.4.22
- Employ a hash table in memory for speeding up VA lookups

- **Setup**

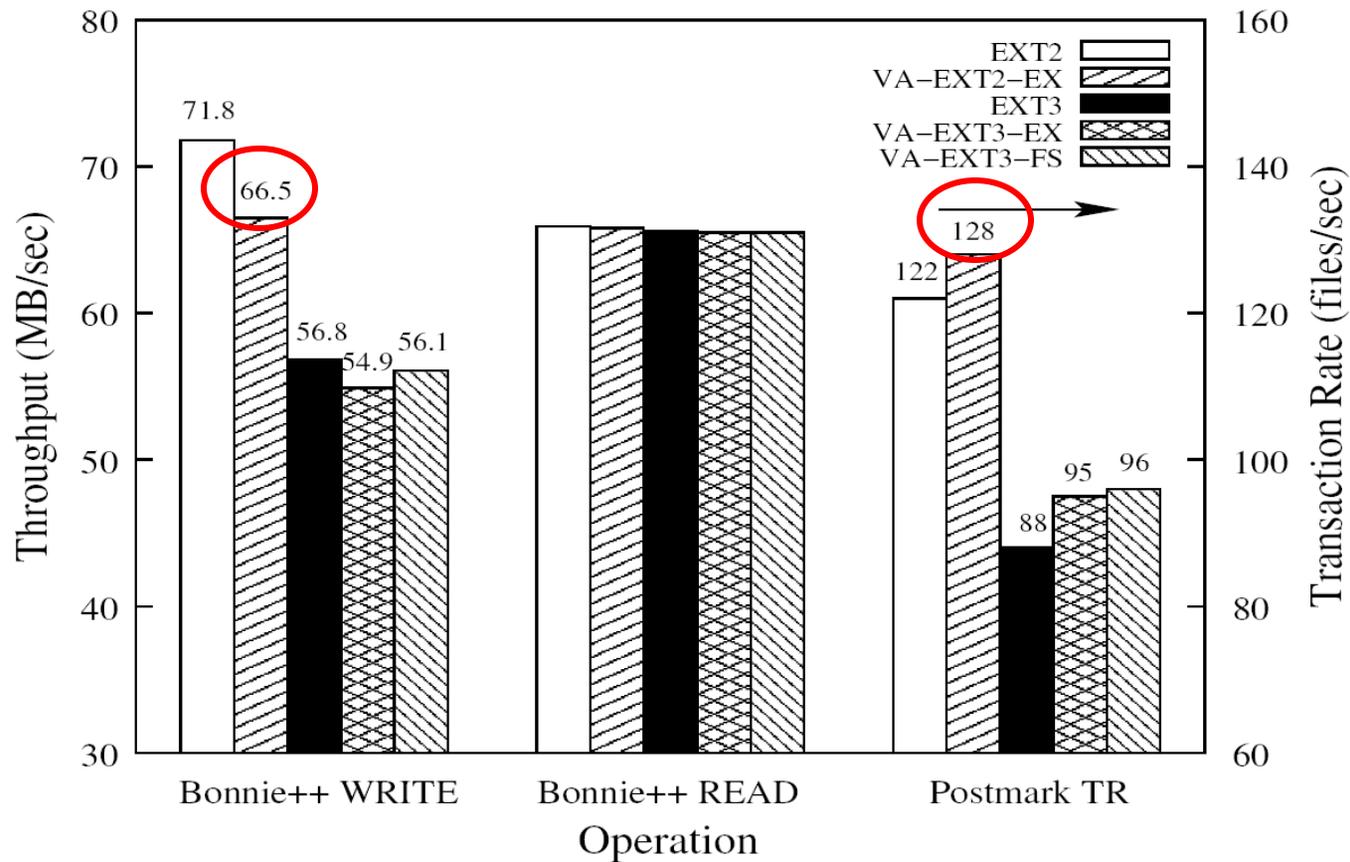
- A 3GHz Pentium 4 processor, 1GB main memory
- Red Hat Linux 9 with a 2.4.22 kernel
- Ext2 file system and Ext3 file system

- **Workloads**

- Bonnie++ (Large-file workload)
- Postmark (Small-file workload)
- TPC-C (Database workload)

# VA Metadata Hardening

- Compare *EXT2* and *VA-EXT2-EX*
- Compare *EXT3* and *VA-EXT3-EX*, *VA-EXT3-FS*



# Reclaiming Allocated Storage Space

- **Reclaim operation for deleted large files**

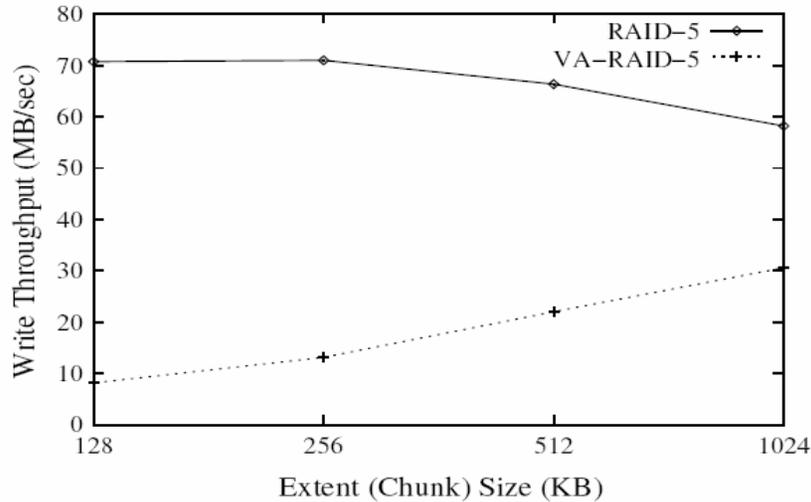
Operation	EXT2 Live Blocks	VA Live Extents (Number of Blocks)
1. Initial	0	0
2. FS create	27,509	<u>217 (27,776)</u>
3. File write to disk	552,311	4316 (552,448)
4. File delete	27,509	4316 (552,448)
5. Reclaim	27,509	<u>217 (27,776)</u>

- **How to keep track of deleted files?**

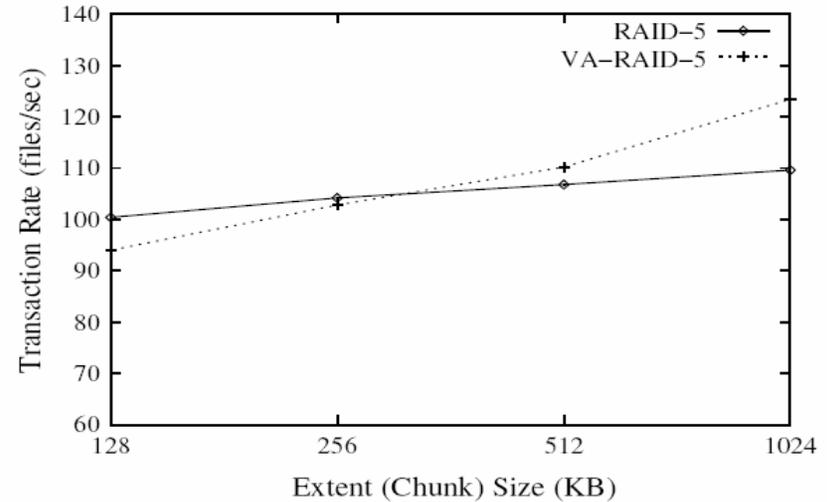
- Employed stackable file system: Maintain duplicated block bitmap
- Alternatively, could employ “Life or Death at Block-Level” (OSDI’04) work

# VA with RAID-5

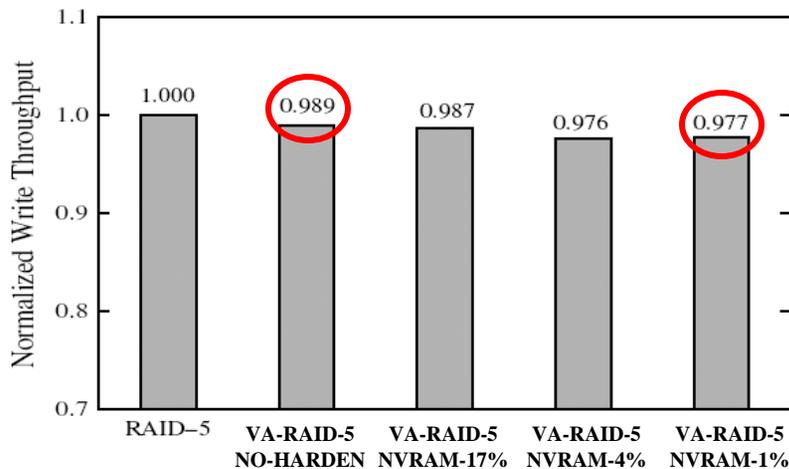
## Large-file workload



## Small-file workload



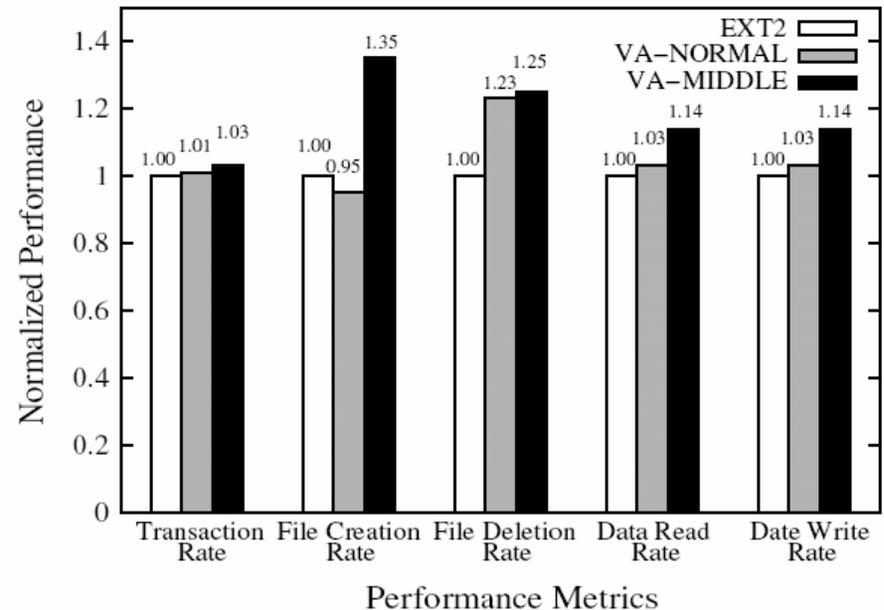
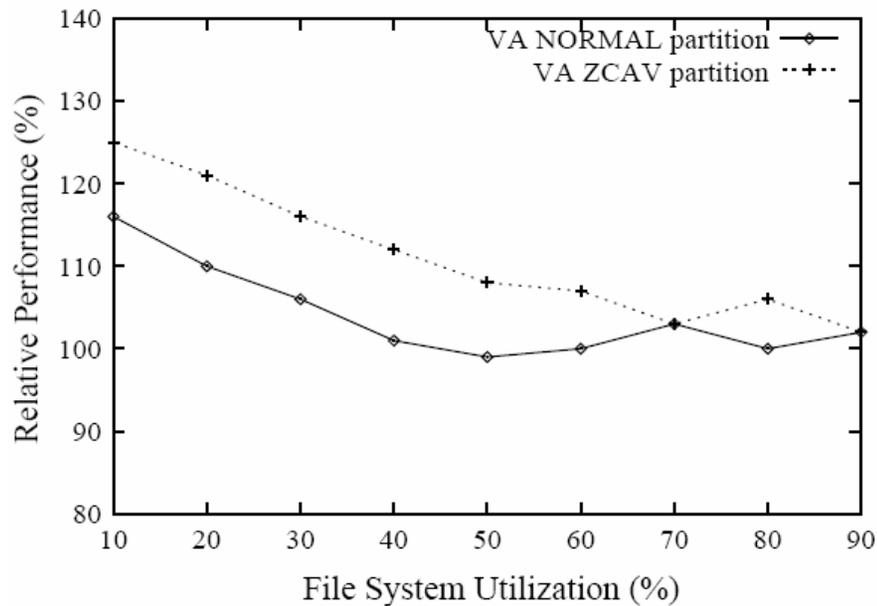
## Large-file workload with NVRAM



- Used Ext2 with software RAID-5 + VA
- NVRAM- $X\%$ :  $X\%$  of total VA metadata size

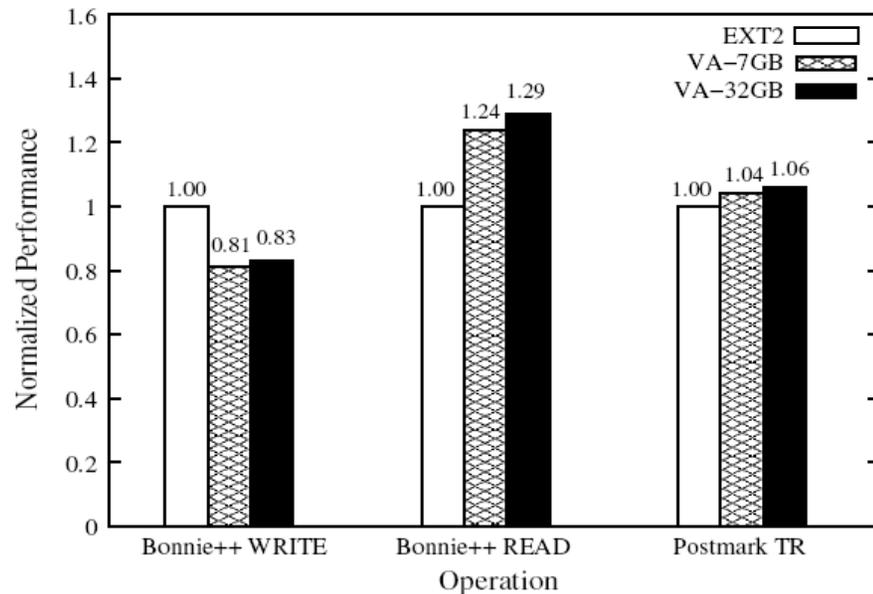
## Data Placement Policy (Postmark)

- **VA NORMAL partition:** Same data rate across a partition
- **VA ZCAV partition:** Hot data is placed in high data region of a partition
- **VA-NORMAL:** start allocation from the outer cylinders
- **VA-MIDDLE:** start allocation from the middle of a partition

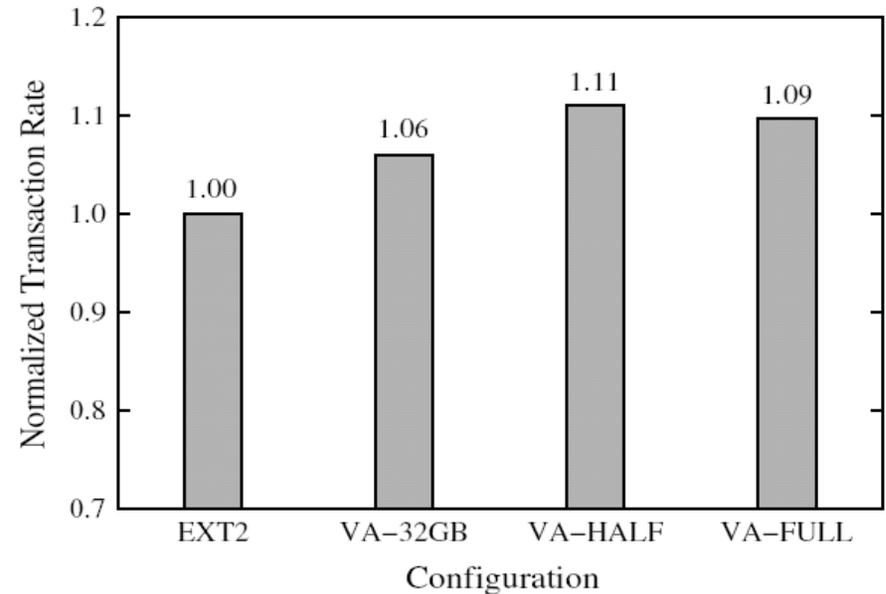


# Multiple File Systems

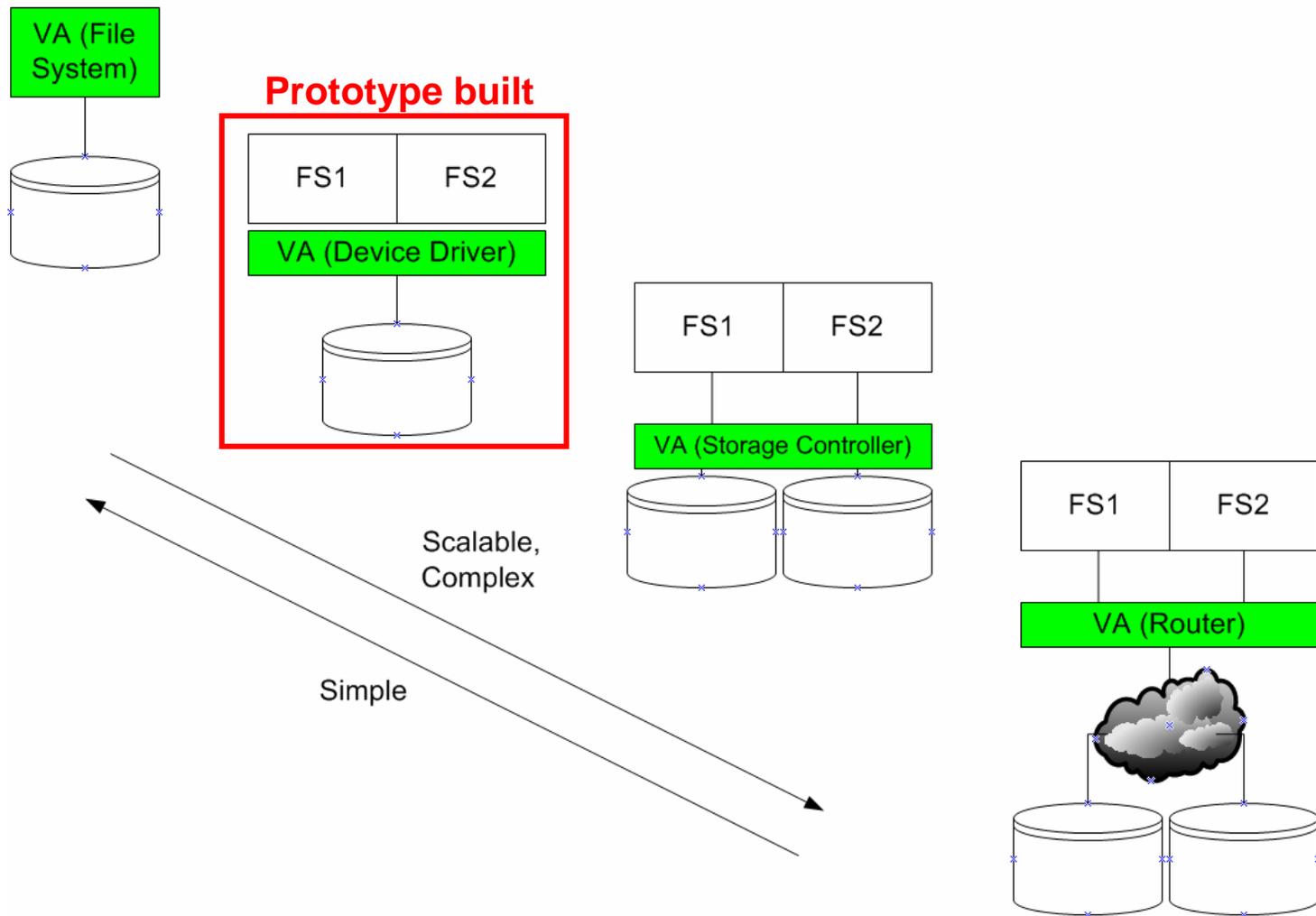
- **VA-7GB:** 2 x 3.5GB partition, 30% utilization
- **VA-32GB:** 2 x 16GB partition, 80% utilization



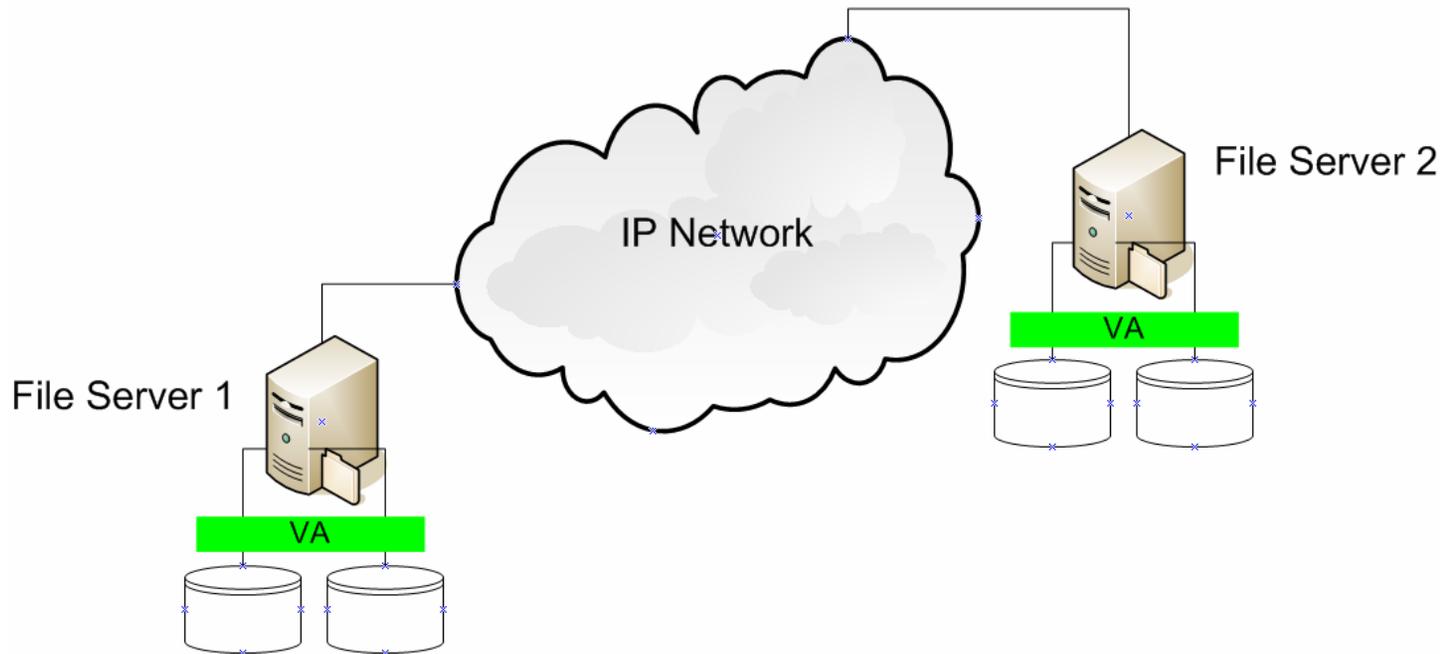
- Used Postmark
- **VA-HALF:** The 2<sup>nd</sup> file system is created after 40% of the 1<sup>st</sup> file system is written
- **VA-FULL:** 80%



# Real-World Deployment of Virtual Allocation



# VA in Networked Storage Environment



- **Flexible allocation provided by VA leads to**
  - Balancing locality vs. load balance issues

# Conclusion

- **Flexible use of storage space**
  - Virtual Allocation
  - Little overhead, sometimes improves performance
  - Allows other opportunities