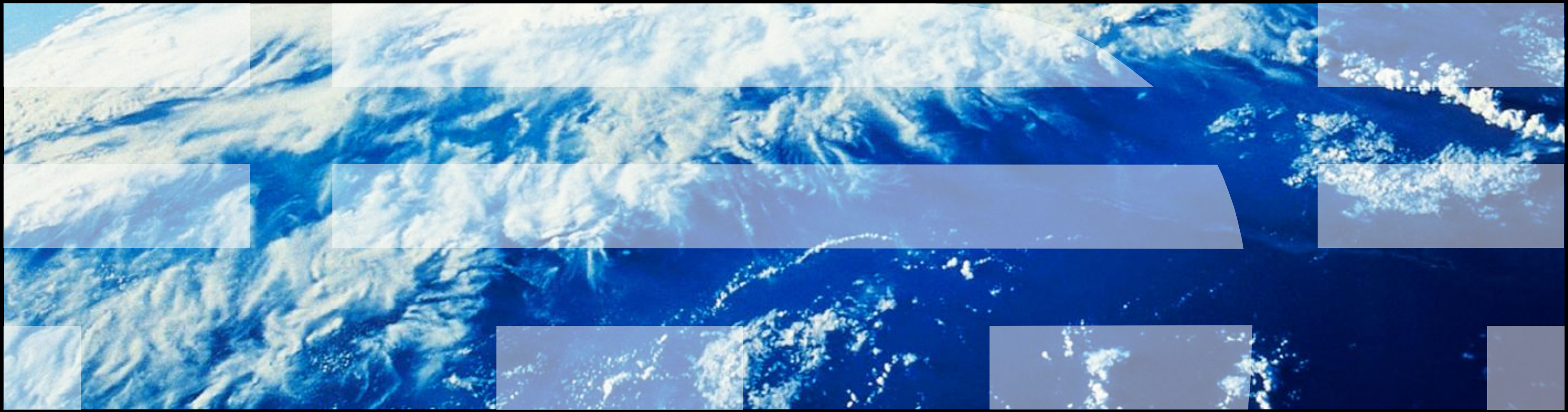
Revisiting the Storage Stack in Virtualized NAS Environments

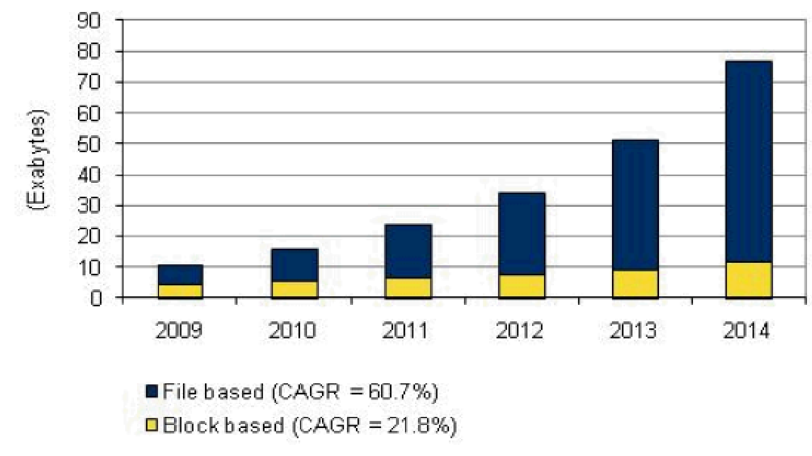
Dean Hildebrand, Anna Povzner, Renu Tewari – IBM Almaden
Vasily Tarasov – Stony Brook University



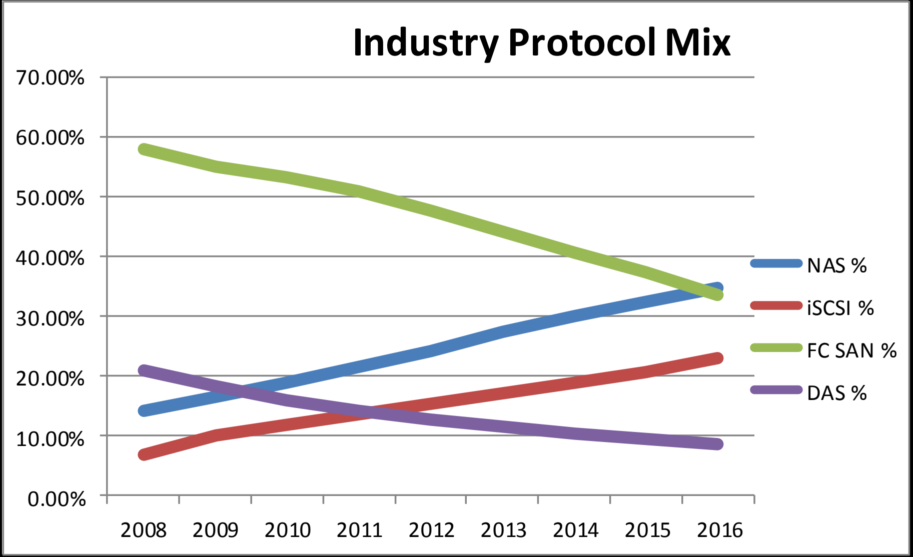
NAS On the Rise

- Increase in unstructured data
  - web, video, photograph, images, music
- Move to single storage network
  - Users migrating from SAN block storage to IPSAN
  - 10GigE becoming commonplace
- Virtual Mache Disk Images
  - Ease of movement
    - Migrate and run anywhere
  - Simplified and flexible storage management
  - Thin provisioning by default

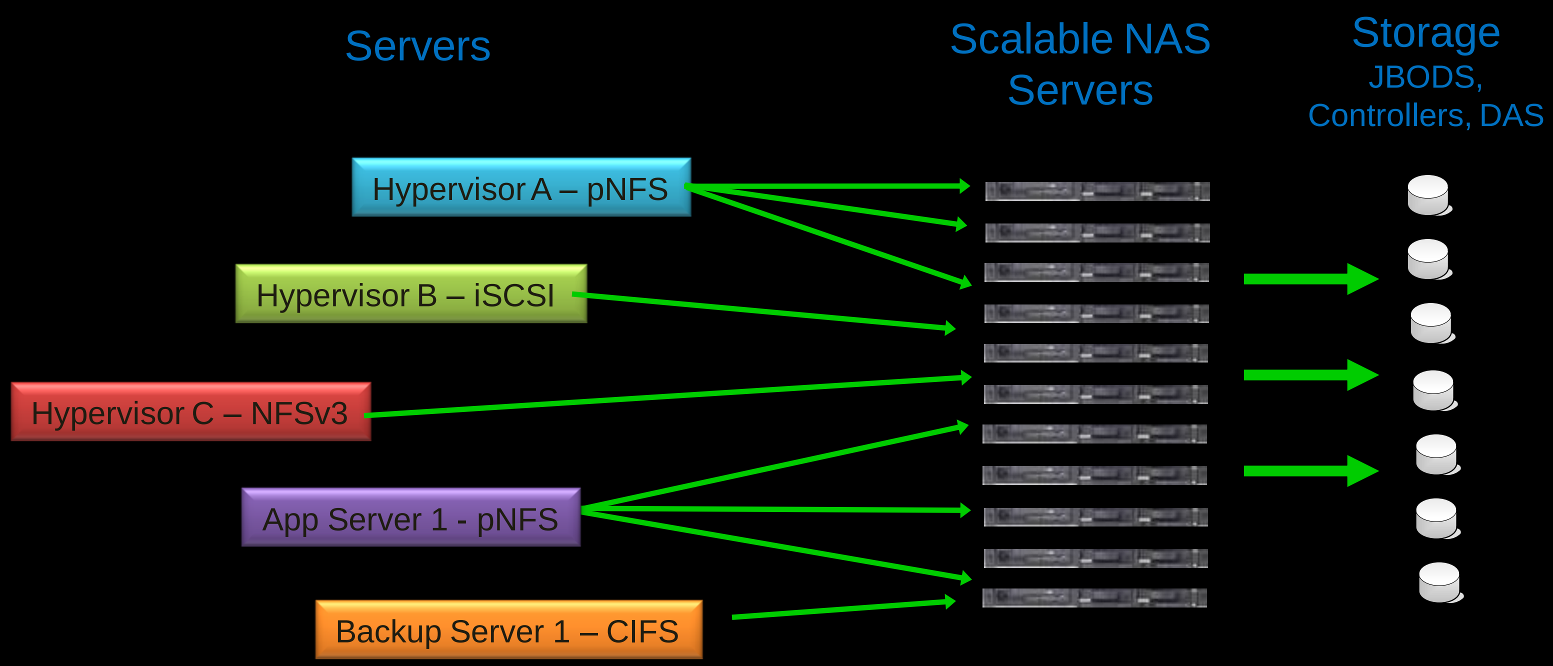
Worldwide File-Based Versus Block-Based Storage Capacity Shipments, 2009–2014



Source: IDC's 2010 Enterprise Disk Storage Consumption Model

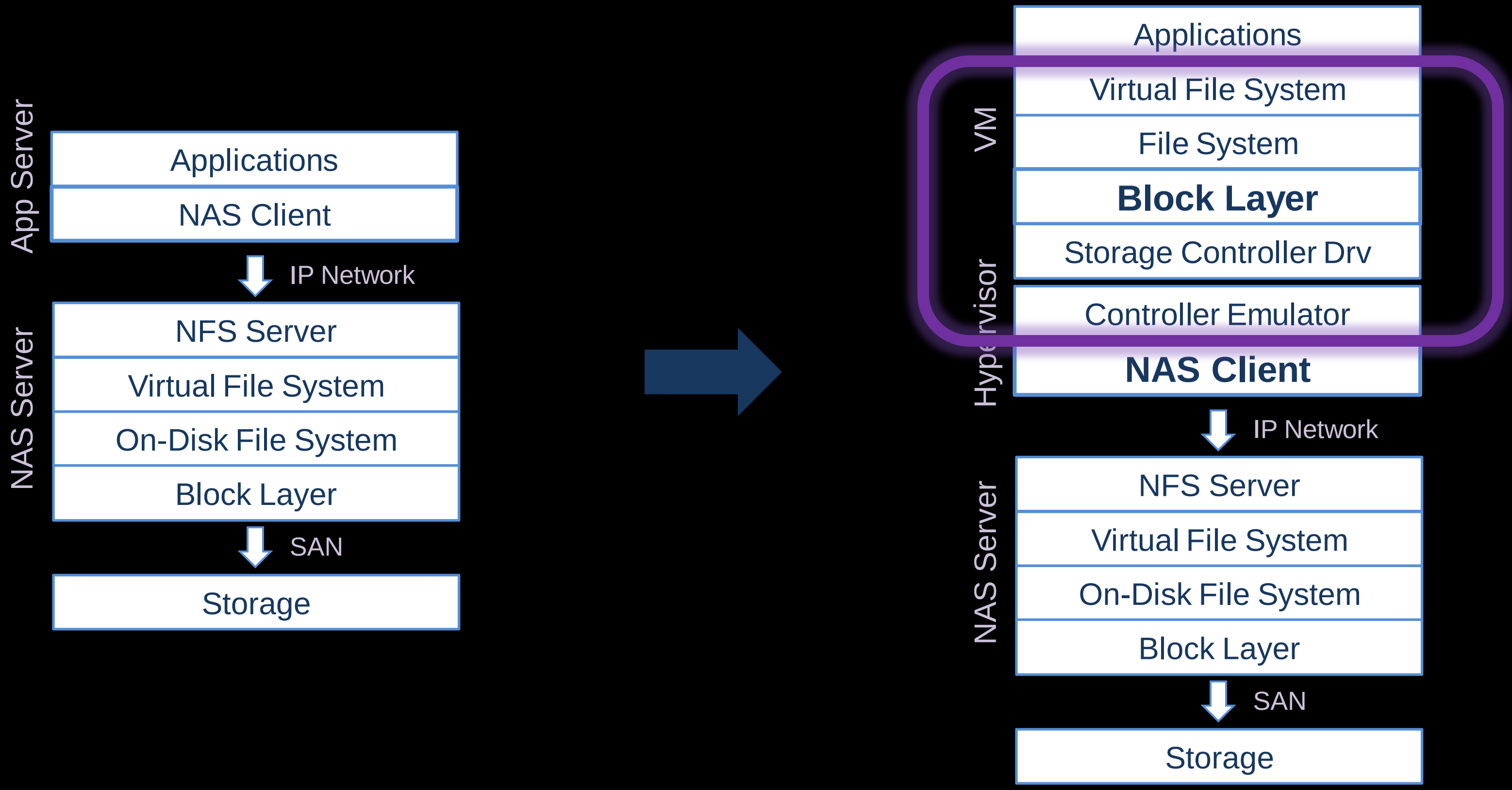


NAS in the Data Center



- Single scalable NAS storage system
  - Capacity and throughput limited only by budget
- Support all relevant NAS protocols
  - NFSv3/v4/v4.1/pNFS/v4.2, iSCSI, CIFS, SMB2

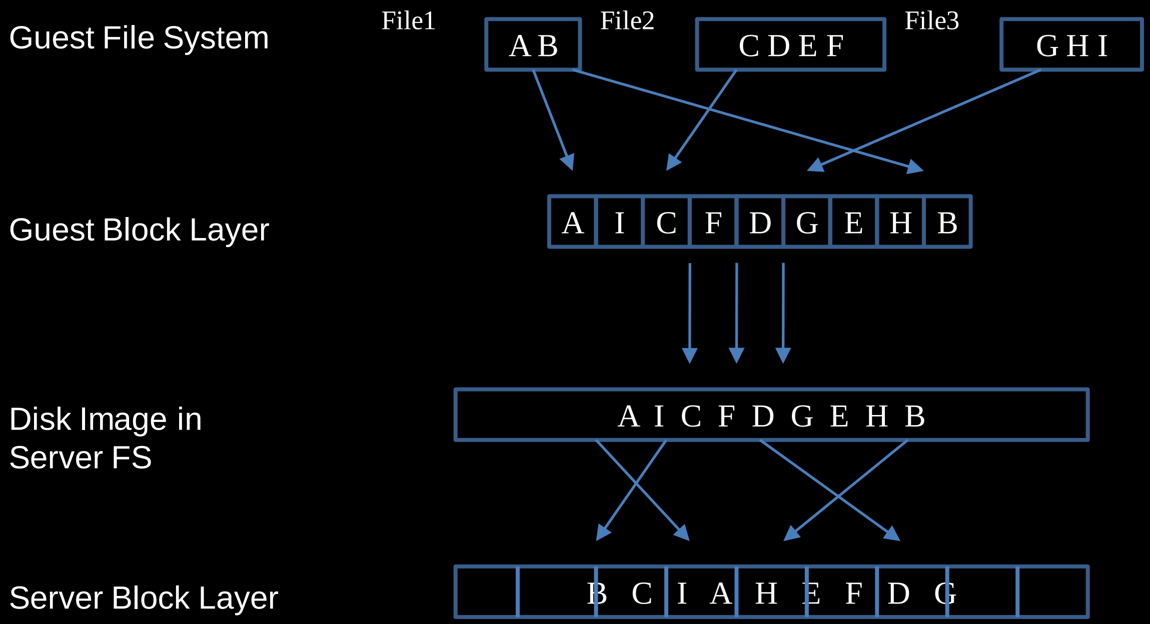
Applications migrating from traditional NAS environments



NFS Software Stack

VM-NAS Software Stack

VM-NAS Write Example



Guest File System

File1 A B

File2 C D E F

File3 G H I

Guest Block Layer

A I C F D G E H B

Disk Image in Server FS

A I C F D G E H B

Server Block Layer

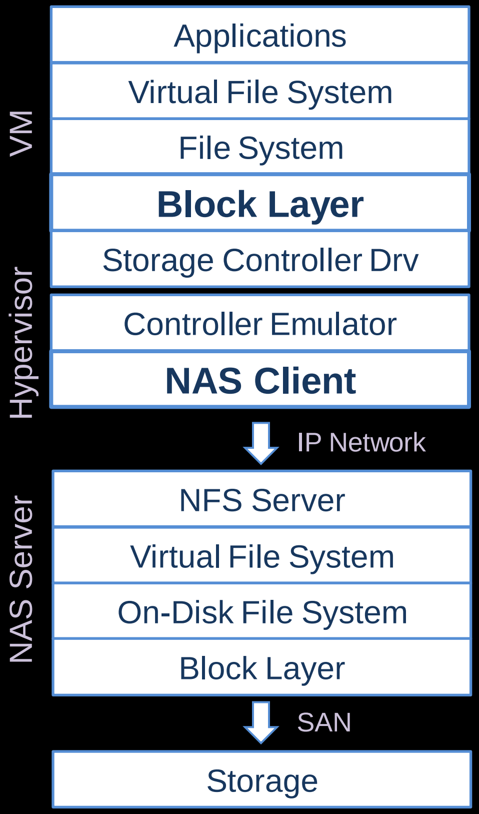
B C I A H E F D G

Lots of opportunities for inefficiencies

Virtual Machines and NAS – Plug and Play?

Potential Hurdles

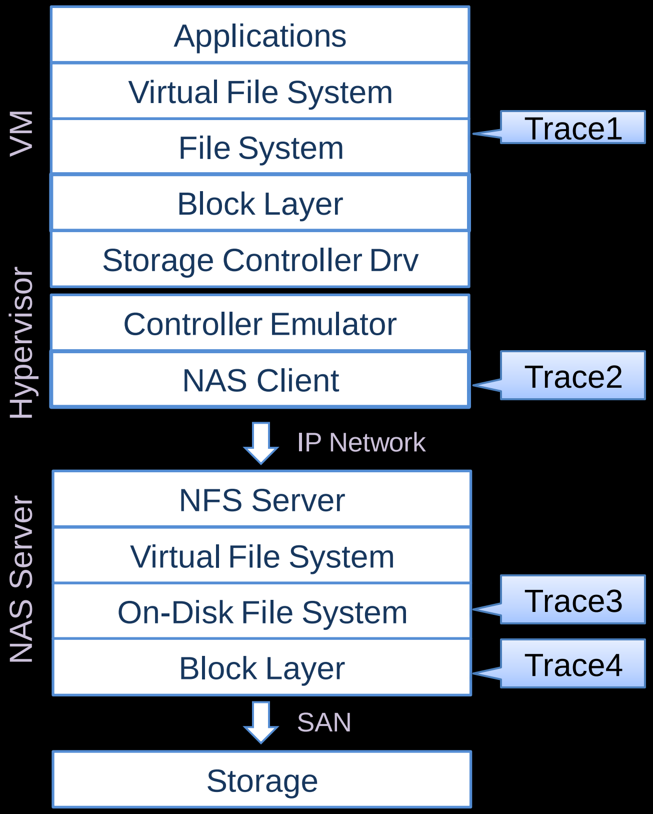
- I/O workloads revamped
  - Typical DB, Web Server, etc workloads may no longer applicable
    - Workloads changes as I/O requests flow through virtualization and NAS software stack
  - Server file system may not handle these new workloads
  - For example, workload may change from many small files to a small number of large files.
- Block on File
  - NFS must support block requests
  - VM block driver in layer above NFS client
  - Basic file system optimizations now handled by VM
    - NFS client can no longer leverage techniques such as readahead, write-back cache, and write gathering
- I/O Optimization layering
  - Does VM or NAS client implement performance optimizations such as caching, readahead, write gathering, etc.
- Out-of-band storage management operations
  - Server-side copy, clones, snapshots, space reservations, etc



VM-NAS Software Stack

Test Harness and Multi-Level Tracing

- Setup
  - Virtual Environment (VM-NFS)
    - Hypervisor: ESX 4.1 with NFSv3
    - Guest: Fedora 14
    - Disk image: Ext2
  - NFS Environment (NFS)
    - Linux 2.6.34
  - NFS
    - rsize = 64KB, wsize = 512KB (ESX maximums)
    - 32 nfsd threads
  - Server File System: GPFS
- Tracing at four levels
  - Guest VFS – What is the app doing?
  - vscsistats – What is coming out of the VM?
  - Server file system – What is NFS sending to the server?
  - Server block layer – What is the FS sending to the disks?



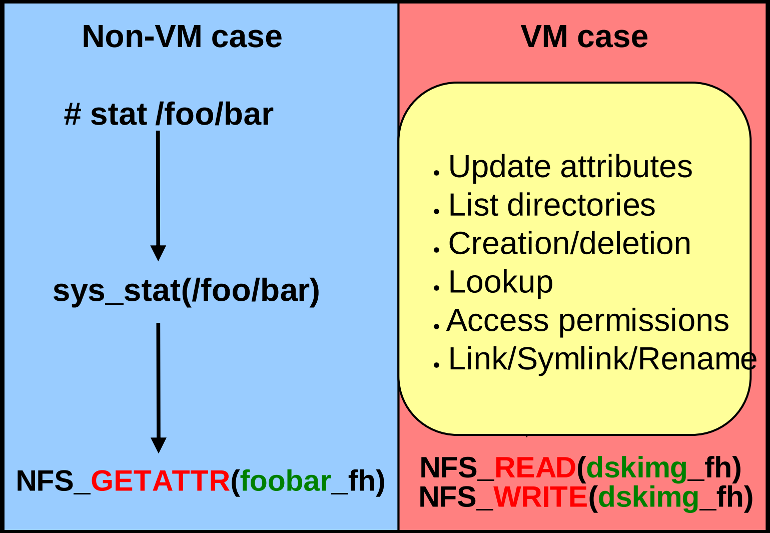
VM-NAS Software Stack

Point 1: Block on File

- All metadata operations converted to read/write
  - create, remove, etc, converted to writes
  - stat, readdir, etc, converted to reads

- Virtual machine's block controller dictates I/O requests to NFS client
  - NFS client must satisfy block requests immediately without buffering
    - Philosophy is to leverage VM OS cache
    - For example, VMWare's proprietary NFSv3 client has the following properties
      - Synchronous
      - All writes direct to disk (stable flag turned on)
      - No readahead
      - No write behind

Example



Non-VM case

# stat /foo/bar

sys_stat(/foo/bar)

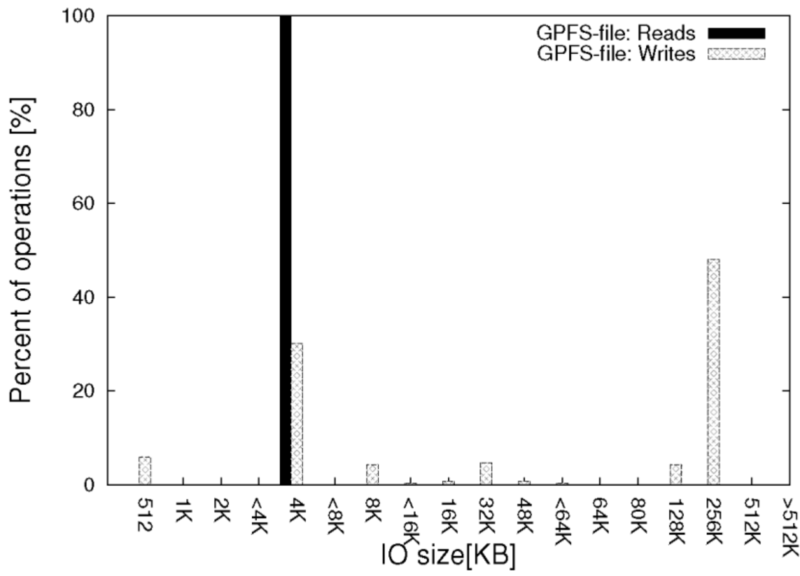
NFS_GETATTR(foobar_fh)

VM case

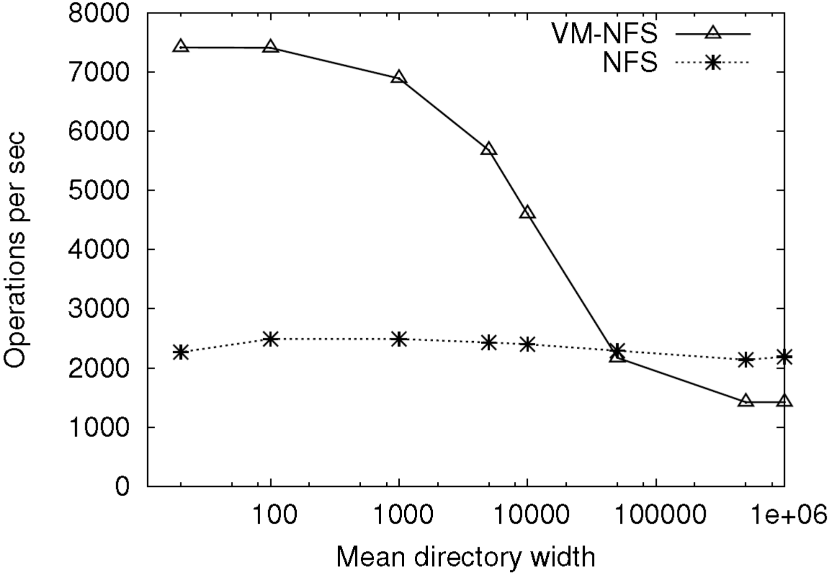
- Update attributes
- List directories
- Creation/deletion
- Lookup
- Access permissions
- Link/Symlink/Rename

NFS_READ(dskimg_fh)
NFS_WRITE(dskimg_fh)

File Create (100K files)



Read and Write Sizes at GPFS with a single directory

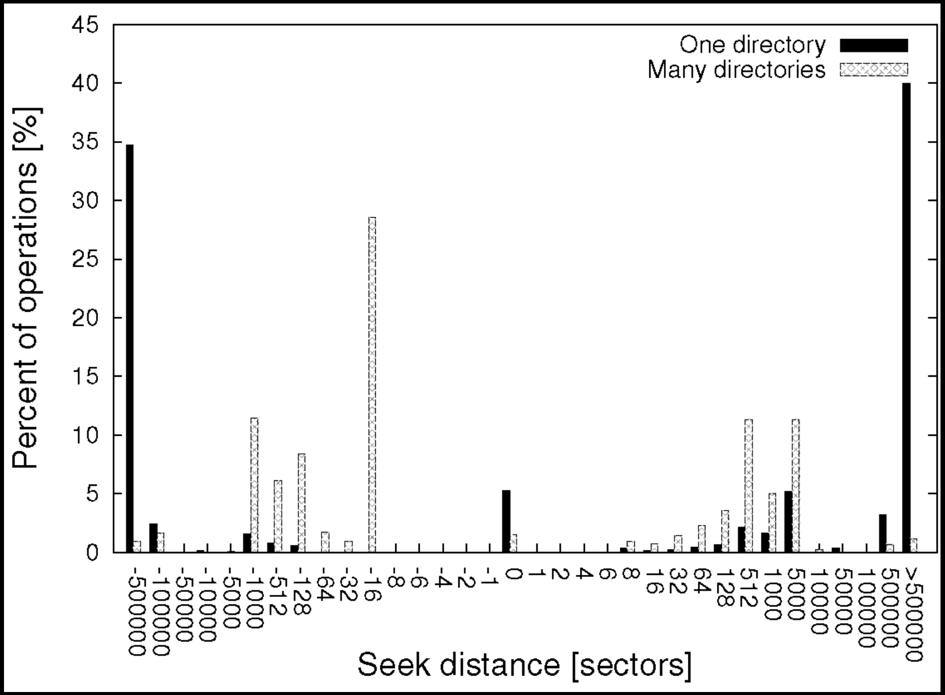


File Create Performance

Dir width - # files in a directory

- With single directory
  - VM-NFS
    - reads 21.5MB and writes 21MB (209 bytes per dir)
    - 98% of reads and 52% of writes are sequential
  - NFS cause GPFS to receive ~500K getattr calls (in addition to the 100K creates)

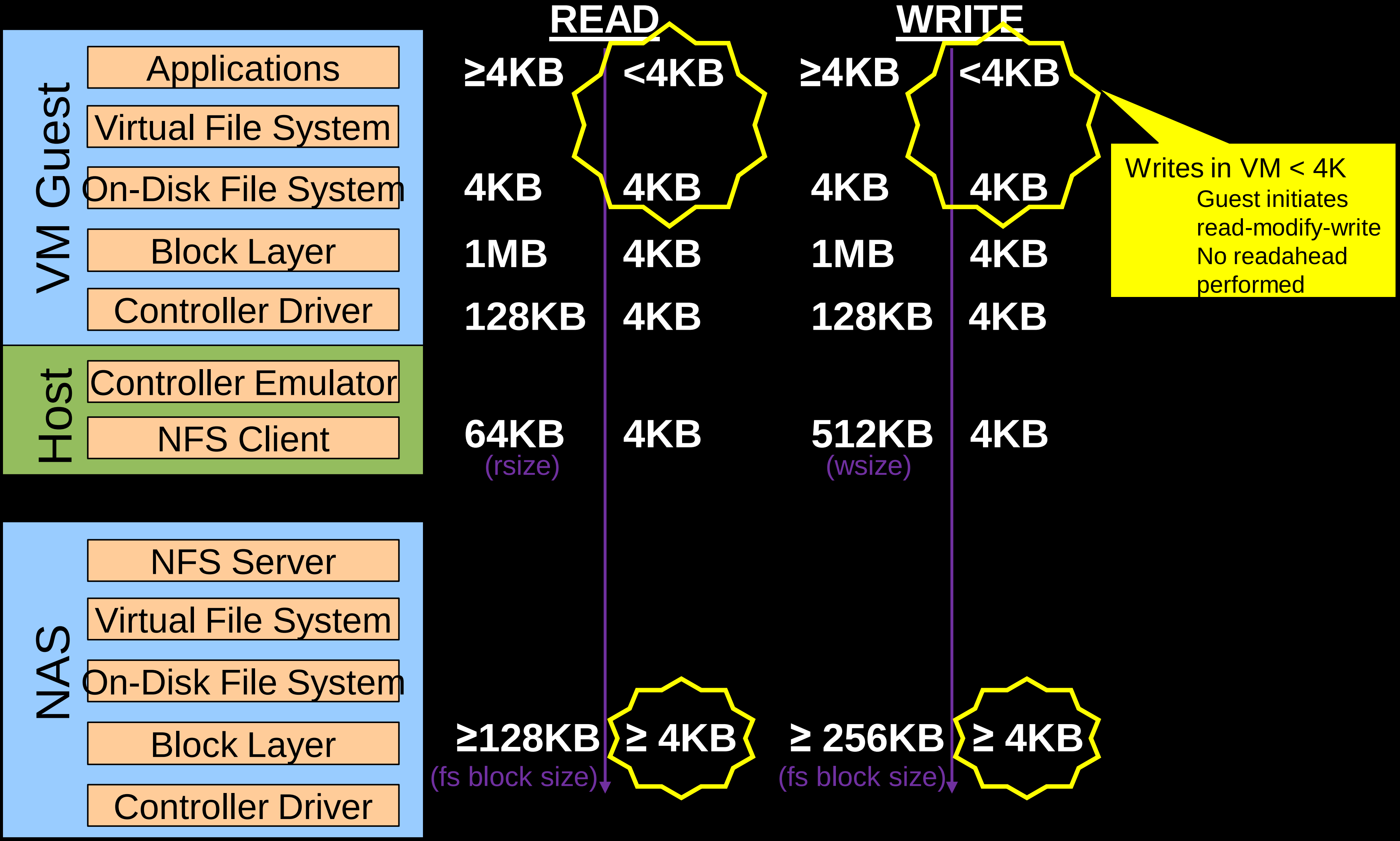
File Stat (100K files)



GPFS Seek Distance

- Collocation of inodes in disk image reduces seek distance
  - Randomness of read requests decreases with number of directories
- With single directory
  - VM-NFS reads 26.3MB (8622 ops/sec) (276 bytes/op)
  - NFS caused GPFS to receive ~610K lookup/getattr calls (2656 ops/sec)

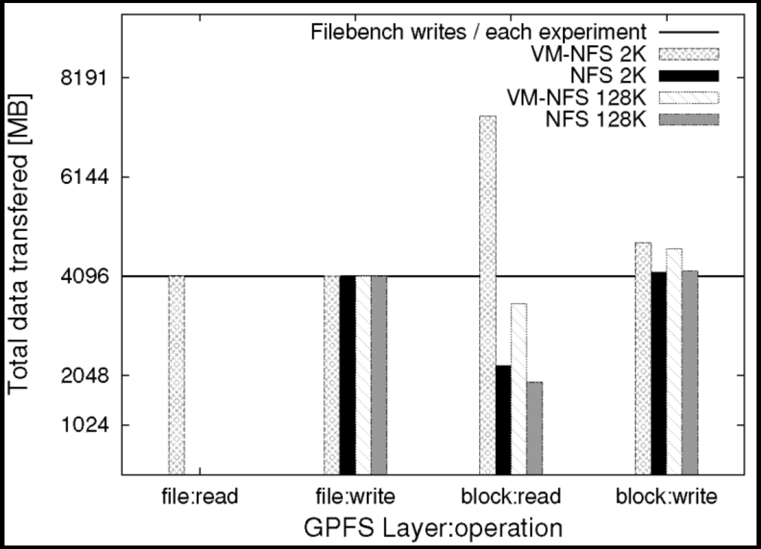
Point 2: I/O Size Transformation



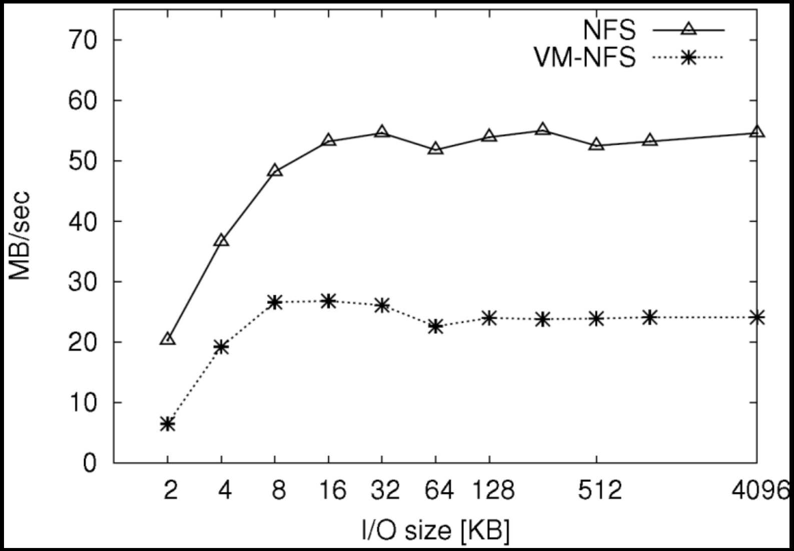
		READ		WRITE	
VM Guest	Applications	≥4KB	<4KB	≥4KB	<4KB
	Virtual File System				
	On-Disk File System	4KB	4KB	4KB	4KB
	Block Layer	1MB	4KB	1MB	4KB
	Controller Driver	128KB	4KB	128KB	4KB
Host	Controller Emulator				
	NFS Client	64KB (rsize)	4KB	512KB (wsize)	4KB
NAS	NFS Server				
	Virtual File System				
	On-Disk File System				
	Block Layer	≥128KB (fs block size)	≥ 4KB	≥ 256KB (fs block size)	≥ 4KB
	Controller Driver				

Writes in VM < 4K
Guest initiates read-modify-write
No readahead performed

Sequential Write



Sequential Write Data Transfer



Sequential Write Performance

- Data Transfer
  - VM-NFS 2KB
    - Read-modify-write from NFS client AND server file system
  - VM-NFS and NFS experience read-modify-write on server
- Performance
  - VM-NFS suffers from stable writes, client-side read-modify-write
    - NV-RAM or SSDs would help…

Random 2KB Reads (2GB from a 4GB file)

	2KB reads	
	VM-NFS	NFS
MB/s	0.6MB/s	2.5MB/s
app reads	2048M	2048M
file reads	4710M	1140M
block reads	6758M	5853M

- Machine has 500MB RAM
- GPFS uses 256KB block size, 50MB cache

Future Work: NFS Has Potential!

- Current performance degradation artifact of current implementations
  - Each software layer acting independently
    - Every write need not necessarily be stable
    - Need alignment across the layers
  - Single client Linux NFS supports POSIX semantics
    - In some cases, NFS is more strict
      - E.g., POSIX supports unstable file creates

- Think of different ways of accessing disk images

	Performance (MB/s)
VM-NFS	36.3
Linux-NFS	98.3
Guest-NFS	66.2

Comparing performance of 3 different ways of using NFS

Other Ongoing Work

- NFSv4.2 turning into the “VM” protocol
  - Cloning
  - Server-side copy
  - Hole punch
  - Space reservations
  - Sparse reads
    - http://www.ietf.org/id/draft-hildebrand-nfsv4-read-sparse-02.txt
  - I/O hints
    - http://www.ietf.org/id/draft-hildebrand-nfsv4-fadvise-01.txt
- pNFS
- Study of workload transformations
  - Real workloads (DB, webserver)
  - Effect of fragmentation, etc
- Improved benchmarks and workload models

Summary

- Block on file makes NFS do unnatural things
  - Stable writes
  - Client-side read-modify-write
  - *Small* reads in the guest can double the amount of data read at the NAS level.

- Server file systems need to adapt to new workloads
  - SpecSFS and creates/second are a thing of the past
    - Sequential I/O in guest highly likely to be transformed to random I/O
    - NAS workload changes from many smaller files to a small number of considerably larger files

- Need to rethink how we use NFS to access disk images
  - Depending on your server file system, using Guest NFS client may be preferable
    - Existing NAS-based applications may scrap disk images altogether

Thank You

Questions?