



Education

# Running Database Applications On NAS: How and Why?

Stephen Daniel, NetApp

- The material contained in this tutorial is copyrighted by the SNIA.
  - Member companies and individuals may use this material in presentations and literature under the following conditions:
    - ◆ Any slide or slides used must be reproduced without modification
    - ◆ The SNIA must be acknowledged as source of any material used in the body of any document containing material from these presentations.
  - This presentation is a project of the SNIA Education Committee.
  - Neither the Author nor the Presenter is an attorney and nothing in this presentation is intended to be nor should be construed as legal advice or opinion. If you need legal advice or legal opinion please contact an attorney.
  - The information presented herein represents the Author's personal opinion and current understanding of the issues involved. The Author, the Presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.
- NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.**

- **Running Database Applications On NAS: How and Why?**
  - ◆ The use of NFS as a storage interconnect protocol for serious enterprise-class databases began around 1995. The number of businesses running production databases over NAS protocols has grown steadily since then. This talk will discuss the benefits and risks of using NFS as a database storage protocol, provide some historical examples of the subtle bugs that NFS clients have had that caused problems for databases, and show some performance measurements comparing NFS-based database performance to database performance using more traditional interconnects such as Fibre-Channel and iSCSI. The talk will conclude with some future directions for this technology.

# Database on NAS, in the beginning... SNIA

## ➤ Brief History

- ◆ 1995: First NAS appliance purchased for DB use
  - ◆ 2000: First major ERP-outsourcing shop built on NAS
  - ◆ Today: Approx 10% of fabric connected storage for databases uses NAS
- In the beginning there were many who said that databases on NAS was not supportable
- ◆ and, given the state of NAS at that time, they were basically right.

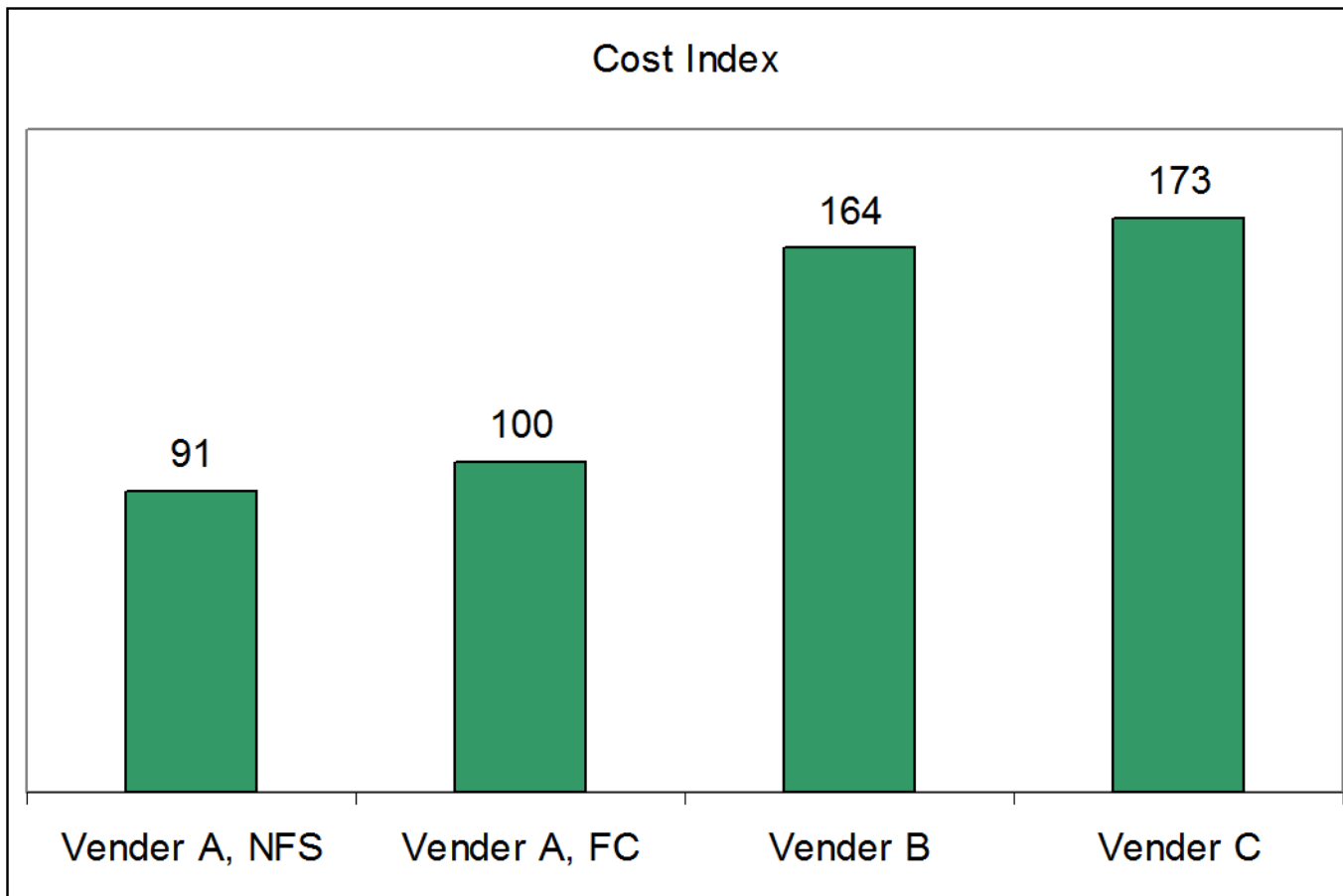
# Database on NAS, why?

- 1980's wisdom: run databases on raw disks
- 1990's wisdom: run databases on files
  - ◆ Performance of raw not worth the hassle
- Early implementations of DB on NAS were seeking file system features only found in NAS servers
  - ◆ Snapshots (point-in-time, consistent logical copies)
  - ◆ Inexpensive asynchronous replication

# Database on NAS, why?

- Top 5 Reasons to run Database on NAS today
  - ◆ Easy to manage Ethernet infrastructure
  - ◆ Ease of provisioning
  - ◆ Very flexible, manageable, binding of object names to storage locations
  - ◆ Some NAS protocols are very fault tolerant
  - ◆ NAS servers continue to offer advanced features that are missing or harder to use in SAN configurations
- Summary: Lower cost

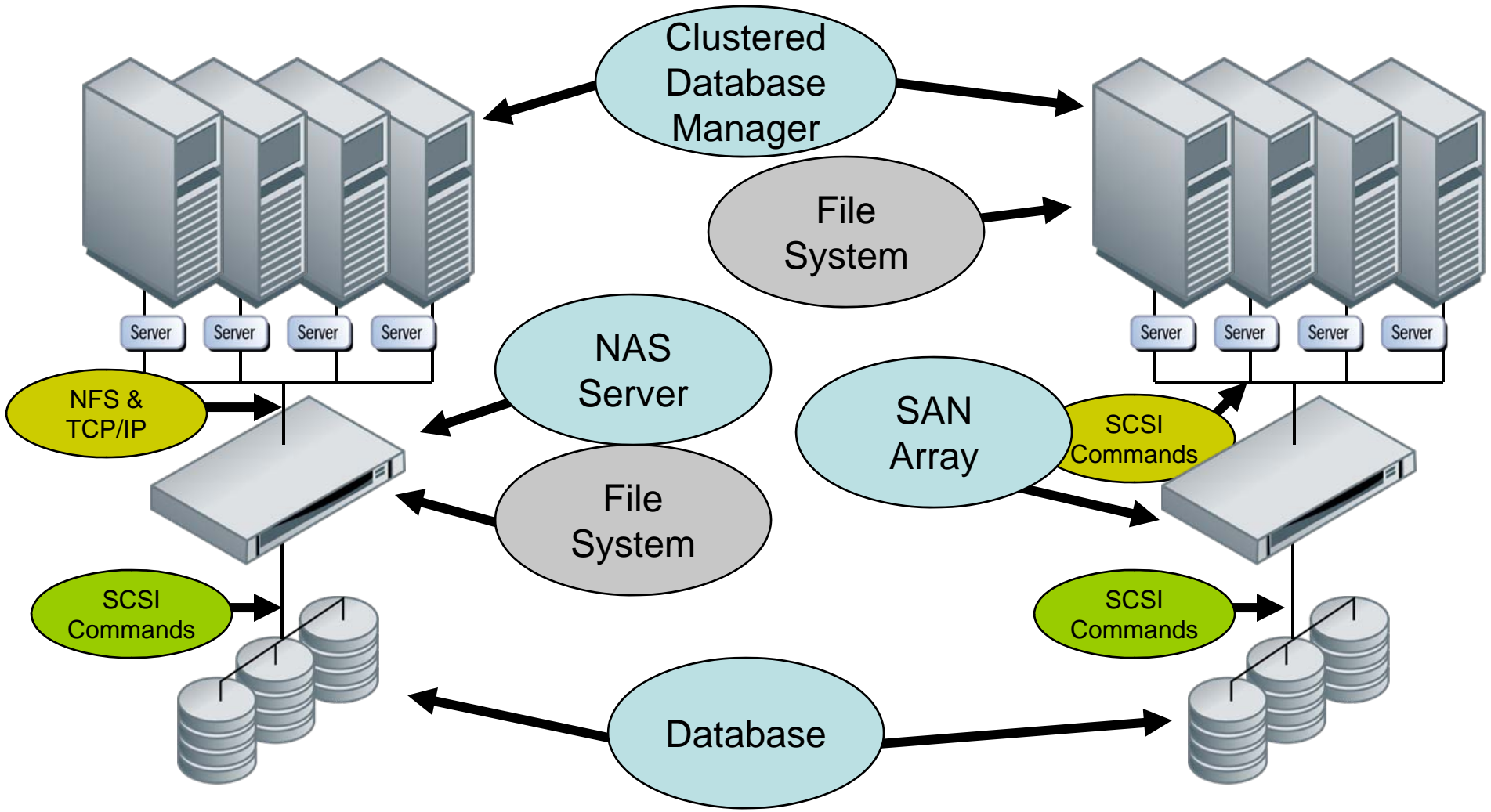
# Cost Advantages



**Note:** Cost index is the relative total-cost-of-ownership, as measured by an independent market research organization, looking at 4 different, comparable solutions.

See <http://media.netapp.com/documents/ar1038.pdf> for more details

# Block Diagrams: SAN and NAS



# Functional Issues

- Write-through caches (you'd think `O_SYNC` would suffice)
  - ◆ VM subsystem locking issues will cause modern Linux kernels to occasionally fail to write through even in the presence of `O_SYNC`
  - ◆ Old UNIX and Linux kernels had issues with confused handling of post-operation attributes that sometimes prevented write-through
- With current kernels
  - ◆ Current UNIX systems: No known write-through bugs
  - ◆ Linux: must use non-caching mode

# Functional Issues

## ➤ Non-atomic writes

- ◆ Writes must be atomic to the size of a database block
- ◆ Failure to maintain atomicity causes database recovery problems, possibly including loss of data
- ◆ Be wary of any implementation where the kernel's page size is smaller than the database block size

## ➤ With current kernels

- ◆ Current UNIX systems: No known atomicity bugs
- ◆ Linux: must use non-caching mode

# Functional Issues

## ➤ Delayed duplicates

- ◆ At least one major database repeatedly writes the same block with updated information
- ◆ Failure to detect delayed duplicates causes corruption
- ◆ Failures were found when client-reconnect happened on a new port and the server used source-port as part of the tuple that detects duplicates

# Functional Issues

## ➤ Non-issue: locking

- ◆ **Concern:**
  - NFS file locking implementation are not suitable for high-performance distributed lock management
  - Neither high performance nor robust
- ◆ **However,**
  - No current enterprise-class database uses file-locking system calls to lock pages or rows in the database
  - All such databases use the same locking code on all databases (SAN, NAS, DAS)
- ◆ **Historical note: Cobol ISAM implementations are a different story...**

# Functional Issues

- Protocols that don't work for DB on NAS:
  - ◆ CIFS – Windows' file sharing protocol
    - › Stateful sessions are not handled well
    - › Other issues are plentiful, and never fully debugged
    - › Beginning to see some DB on NFS on Windows
    - › However, Microsoft continues to improve the protocols and implementations, so this may change in the future
  - ◆ NFS/UDP (as opposed to NFS/TCP):
    - › Insufficient sequence number space in UDP
    - › Corruption due to delayed duplicate fragments has been observed in production

# Functional Issues

## ➤ Security

- ◆ Some customers have lots of concerns about security of NFS for production data
- ◆ **Methods:**
  - › Physical or logical (VLAN) isolation of the network
  - › Tight control of the various protocols and daemons that control routing and authentication
  - › Result can be as theoretically secure (or insecure) as Fibre Channel SAN
- ◆ **Not yet seen:**
  - › Kerberized NFS in a database deployment
- ◆ **Note: DB on NFS used in secure government sites**

# Direct I/O Methods

- Over time most functional (and many performance) problems have been solved through the use of non-cached, direct I/O methods
  - ◆ Most UNIX Systems have an NFS mount option to disable all caching (“forcedirectio”, “cio”, ...)
  - ◆ Linux has an open flag (O\_DIRECT), but no mount option – many enterprise-class databases support O\_DIRECT
- Remaining issue: shared disk clusters
  - ◆ Require actime=0 or similar to disable attribute cache

# High Availability Issues

- Most NAS servers achieve H/A through fail-over techniques
- There are a number of subtle issues with this
  - ◆ Similar issues after any server reboot
  - ◆ Sequencing of NFS protocols
  - ◆ Sequencing of services on the NAS server

# Summary of functional requirements

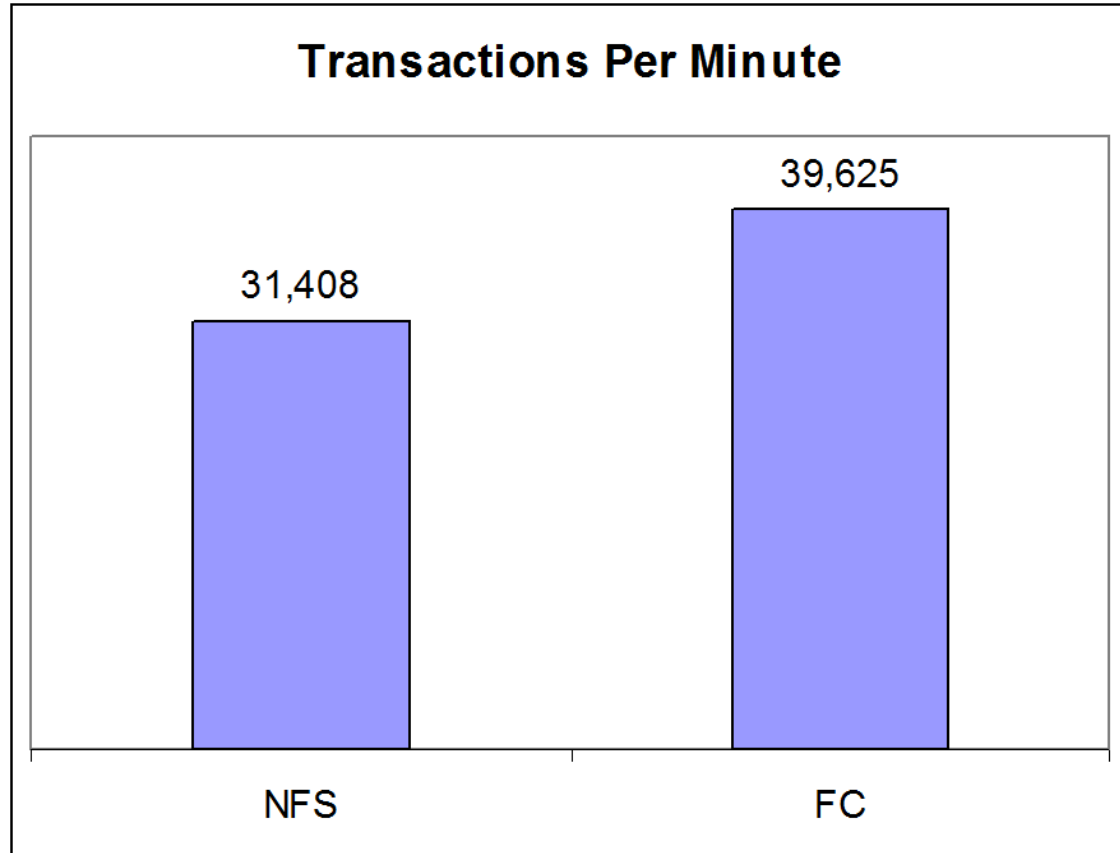
- Caches must be write-through
- Shared-disk databases require no read cache and no attribute cache
- Writes must be atomic
- Robust delayed-duplicate detection

# Performance – Non Issues

- “TCP/IP will add too much latency”
  - ◆ No measurable difference in response time as long as the systems are not at 100% utilization
- “CPU Overhead of NFS is too high”
  - ◆ Per-operation overheads are low
  - ◆ Cost of using NFS for OLTP-Type workloads runs from 0% to 20% in CPU-limited workloads

# Performance – OLTP

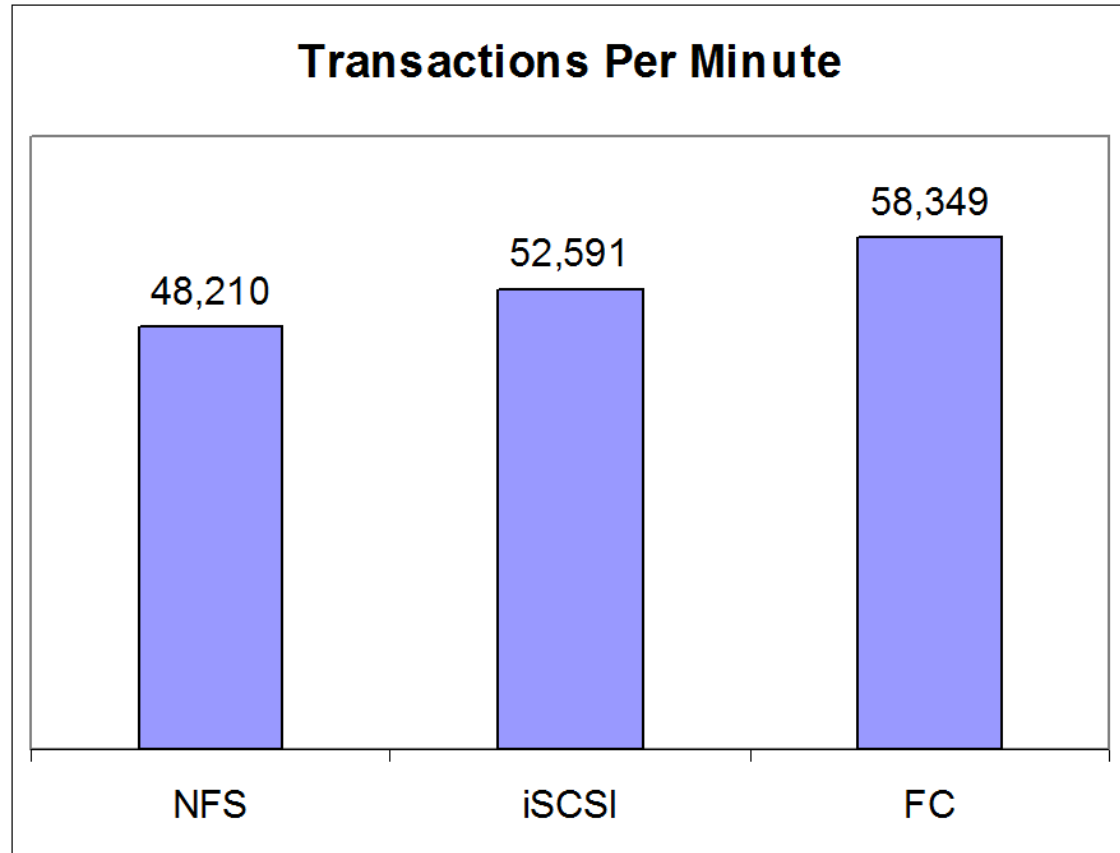
- Comparing host-limited OLTP performance, NFS vs Fibre-Channel
- Data from a RISC UNIX System



- See <http://media.netapp.com/documents/tr-3496.pdf> for more details.

# Performance – OLTP

- Comparing host-limited OLTP performance, NFS vs iSCSI and Fibre-Channel
- FC & iSCSI are raw, no Linux FS
- Data from a modern Linux kernel



- Unpublished data. Contact author for more details.

# Performance Issues

- NFS Client reader/writer locks
  - ◆ Older NFS clients allow only one concurrent write or many concurrent reads
  - ◆ This is a cheap way of satisfying POSIX serialization requirements
  - ◆ Most clients solve this issue by abandoning POSIX serialization on direct-I/O calls

# Performance Issues

## ➤ Block size mismatches

- ◆ Some NFS clients will fetch a minimum amount of data (typically 32 KB) on each random read
- ◆ Direct I/O methods avoid this issue

# Performance Issues

- Inappropriate NFS client cache in the presence of file locks
  - ◆ Some databases will lock the files to prevent simultaneous access by multiple instances
  - ◆ Some NFS clients assume shared access by multiple clients to any file that has file locks present
  - ◆ Such an NFS client cache will invalidate the entire cache on every write
  - ◆ Net result: lots of CPU time spent managing the cache for no benefit

# Performance Issues

## ➤ I/O Concurrency

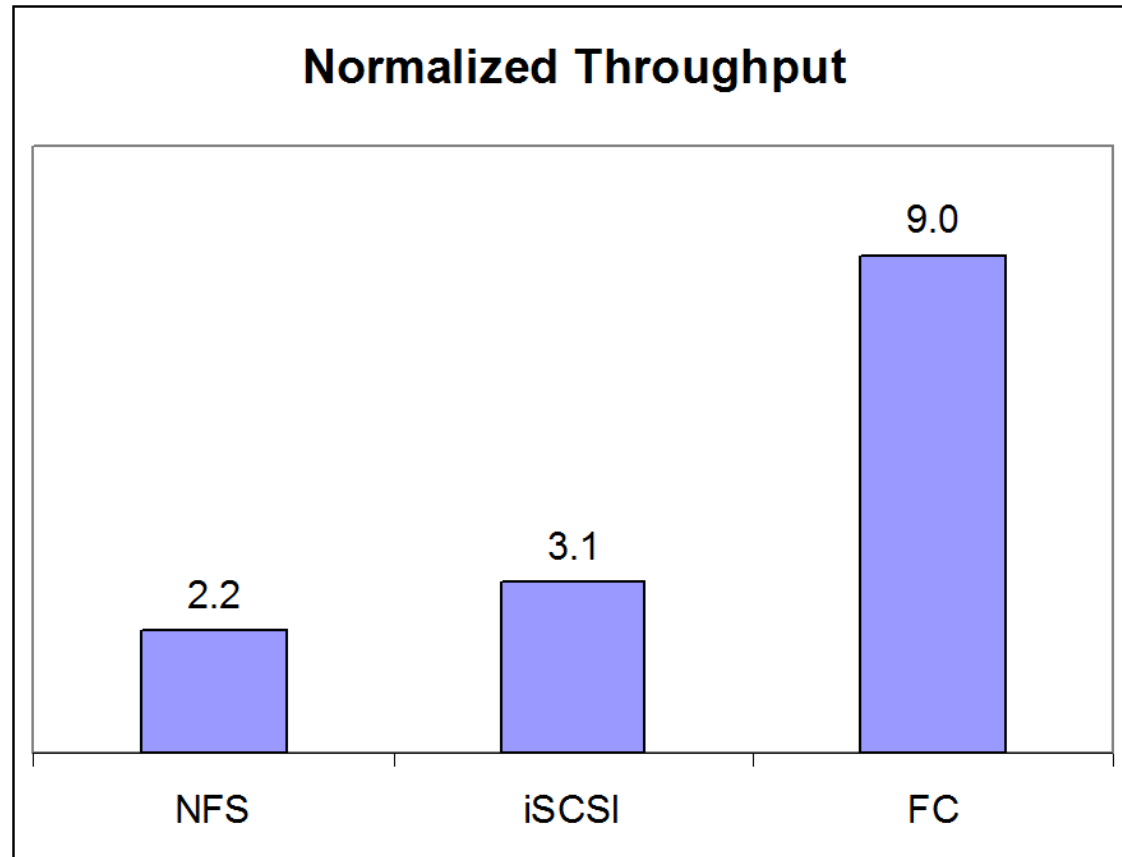
- ◆ Large storage systems require multiple concurrent I/O requests to achieve maximum throughput
- ◆ Some systems (i.e. Linux) limit the number of concurrent I/O requests over a single NFS mount to 16
  - › Fixed limit replaced with `/etc/sysctl` option in 2.6 kernels
  - › For large systems the new limit of 256 is still an issue

# Performance Issues

- Load-balancing across multiple-wires
- High CPU overhead per-byte
  - ◆ At least one NFS client used sufficient CPU time to copy the data 7 times

# Performance – Table Scans

- Comparing host-limited host CPU usage for large table scans
- Units are GB/sec per CPU of system time
- FC & iSCSI are raw, no Linux FS
- Data from a modern Linux kernel



- Unpublished data. Contact author for more details.

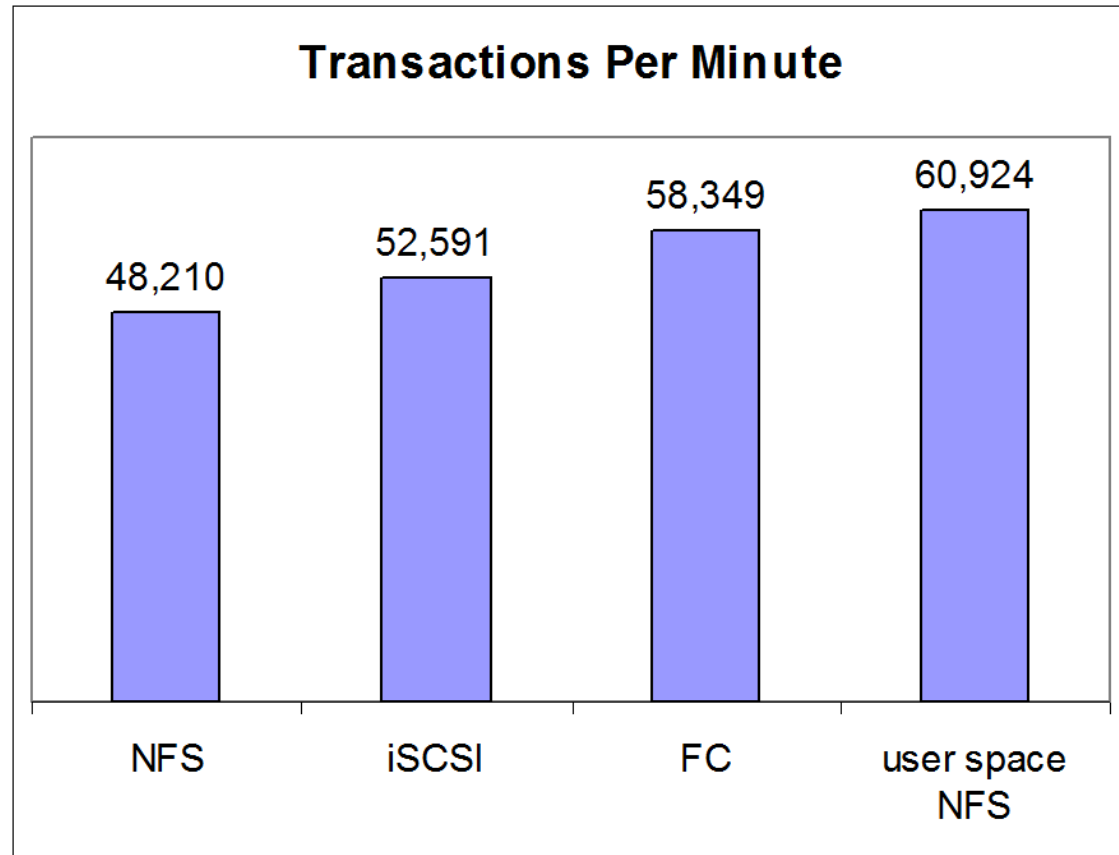
# Future Directions

## ➤ User-space NFS clients

- ◆ One major database has implemented their own NFS client in user-space
- ◆ ... prompting other database vendors to consider the same

# Performance – OLTP

- Comparing host-limited OLTP performance, user-space NFS vs other protocols
- FC & iSCSI are raw, no Linux FS
- Data from a modern Linux kernel



- Note: User space NFS is faster because the implementation is optimized for scalable, non-caching use.
- Pre-publication data. Contact author for more details.

# Future Directions

## ➤ NFS v4

- ◆ Caching
- ◆ pNFS
- ◆ Other extensions

## ➤ Security

# SAN vs NAS

- Common developer question:
  - ◆ Can I run a reliable database over NAS?
  - ◆ Yes, provided your NAS is NFS/TCP
- Common customer question:
  - ◆ Should I run my database on NAS or SAN?
  - ◆ Selection Criteria
    - > Goals
      - Minimum TCO?
      - Maximum performance?
      - Flexibility?
    - > Existing infrastructure / knowledge-base

- Please send any questions or comments on this presentation to SNIA: [trackapplications@snia.org](mailto:trackapplications@snia.org)

**Many thanks to the following individuals  
for their contributions to this tutorial.**

**- SNIA Education Committee**

**John Elliott  
Dan Morgan  
Joshua Tseng**

**Saad Jafri  
Philippe Nicolas**