NFS Tuning for High Performance

Tom Talpey

Usenix 2004 “Guru” session

tmt@netapp.com

Overview

- Informal session!
- General NFS performance concepts
- General NFS tuning
- Application-specific tuning
- NFS/RDMA futures
- Q&A

Who We Are

- **Network Appliance**
- **“Filer” storage server appliance family**
  - **NFS, CIFS, iSCSI, Fibre Channel, etc**
  - **Number 1 NAS Storage Vendor – NFS**

FAS900 Series
Unified Enterprise-class storage

NearStore®
Economical secondary storage

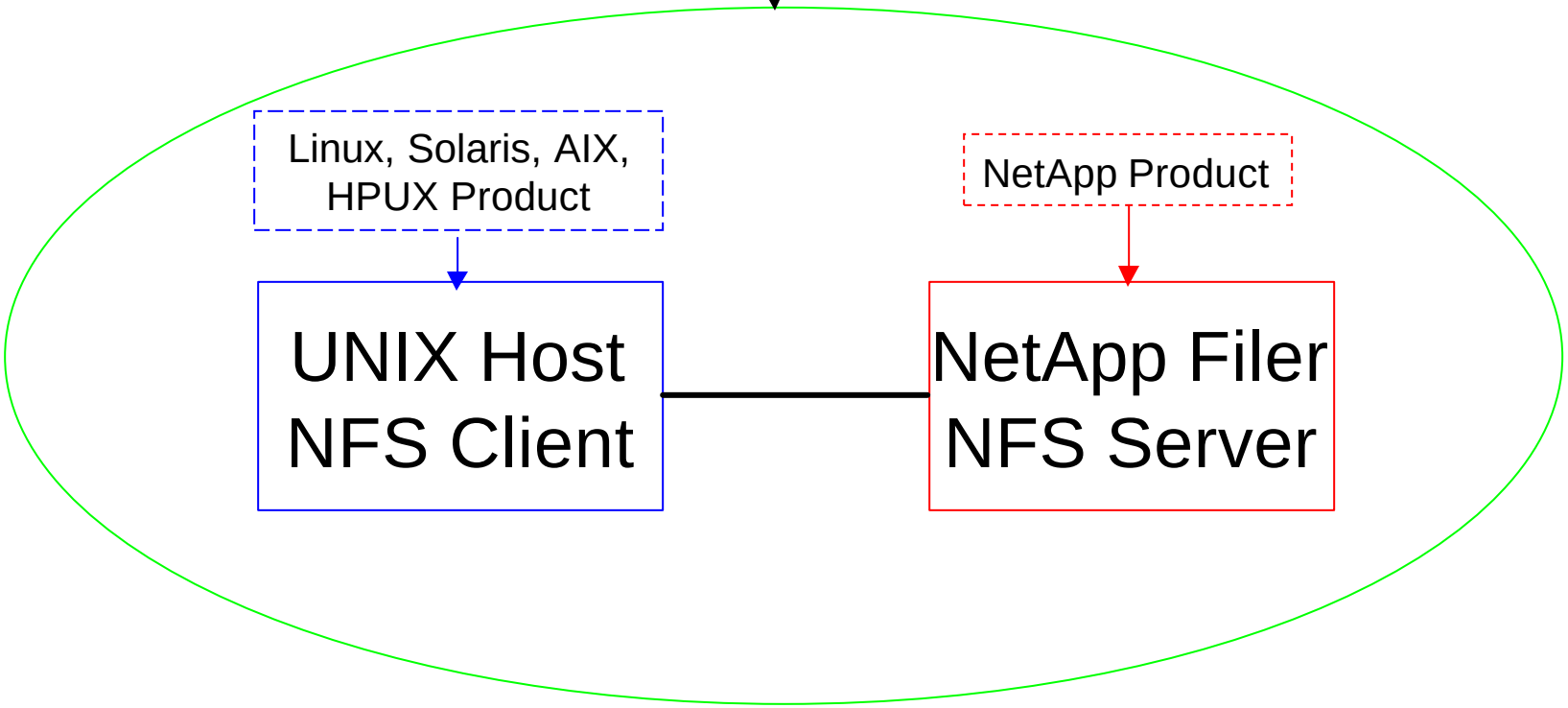
gFiler™
Intelligent gateway for existing storage

NetCache®
Accelerated and secure access to web content

FAS200 Series
Remote and small office storage

Why We Care

What the User Purchases and Deploys
An NFS Solution



Our Message

- NFS → Delivers real management/cost value
- NFS → Core Data Center
- NFS → Mission Critical Database Deployments
- NFS → Deliver performance of Local FS ???
- NFS → Compared directly to Local FS/SAN

Our Mission

- **Support NFS Clients/Vendors**
  - **We are here to help**
- **Ensure successful commercial deployments**
  - **Translate User problems to actionable plans**
- **Make NFS as good or better than Local FS**
  - **This is true under many circumstances already**
- **Disseminate NFS performance knowledge**
  - **Customers, Vendors, Partners, Field, Engineers**

NFS Client Performance

- **Traditional Wisdom**
  - **NFS is slow due to Host CPU consumption**
  - **Ethernets are slow compared to SANs**
- **Two Key Observations**
  - **Most Users have CPU cycles to spare**
  - **Ethernet is 1 Gbit = 100 MB/s. FC is on 2x**

NFS Client Performance

- **Reality – What really matters**
  - **Caching behavior**
  - **Wire efficiency (application I/O : wire I/O)**
  - **Single mount point parallelism**
  - **Multi-NIC scalability**
  - **Throughput IOPs and MB/s**
  - **Latency (response time)**
  - **Per-IO CPU cost (in relation to Local FS cost)**
  - **Wire speed and Network Performance**

Tunings

- The Interconnect
- The Client
- The Network buffers
- The Server

Don’t overlook the obvious!

- **Use the fastest wire possible**
  - Use a quality NIC (hw checksumming, LSO, etc)
  - 1GbE
  - Tune routing paths
- **Enable Ethernet Jumbo Frames**
  - 9KB size reduces read/write packet counts
  - Requires support at both ends
  - Requires support in switches

More basics

- Check mount options
  - Rsize/wsize
  - Attribute caching
    - Timeouts, noac, nocto, …
    - actimeo=0 != noac (noac disables write caching)
  - llock for certain non-shared environments
    - “local lock” avoids NLM and *re-enables caching of locked files*
    - can (greatly) improve non-shared environments, with care
  - forcedirectio for databases, etc

More basics

- NFS Readahead count
  - Server and Client both tunable
- Number of client “biods”
  - Increase the offered parallelism
  - Also see RPC slot table/Little’s Law discussion later

Network basics

- Check socket options
  - System default socket buffers
  - NFS-specific socket buffers
  - Send/receive highwaters
  - Send/receive buffer sizes
  - TCP Large Windows (LW)
- Check driver-specific tunings
  - Optimize for low latency
  - Jumbo frames

Server tricks

- Use an Appliance
- Use your chosen Appliance Vendor's support
- Volume/spindle tuning
  - Optimize for throughput
  - File and volume placement, distribution
- Server-specific options
  - "no access time" updates
  - Snapshots, backups, etc
  - etc

War Stories

- Real situations we've dealt with
- Clients remain Anonymous
  - NFS vendors are our friends
  - Legal issues, yadda, yadda
  - Except for Linux – Fair Game
- So, some examples…

Caching – Weak Cache Consistency

- **Symptom**
  - **Application runs 50x slower on NFS vs Local**
- **Local FS Test**
  - **dd if=/dev/zero of=/local/file bs=1m count=5**
  - **See I/O writes sent to disk**
  - **dd if=/local/file of=/dev/null**
  - **See NO I/O reads sent to disk**
  - **Data was cached in host buffer cache**
- **NFS Test**
  - **dd if=/dev/zero of=/mnt/nfsfile bs=1m count=5**
  - **See I/O writes sent to NFS server**
  - **dd if=/local/file of=/dev/null**
  - **See ALL I/O reads send to disk ?!?**
  - **Data was NOT cached in host buffer cache**

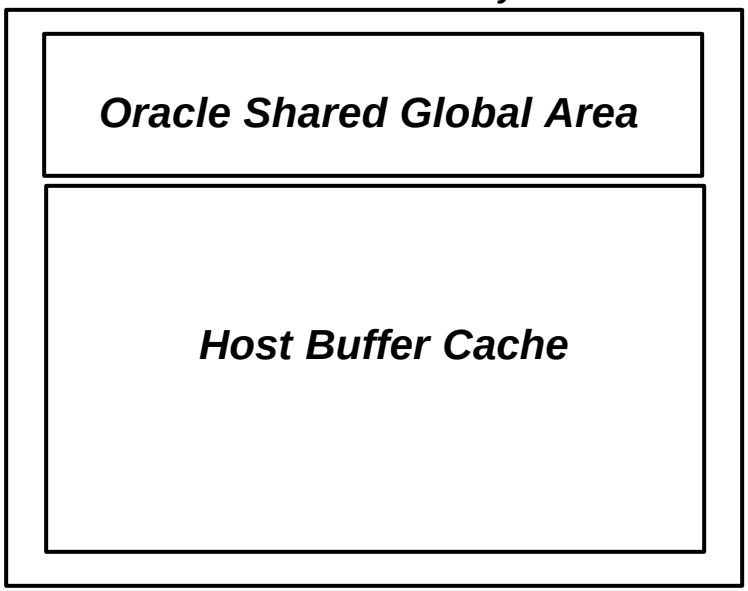
Caching – Weak Cache Consistency

- **Actual Problem**
  - Threads processing write completions
  - Sometimes completed writes out-of-order
  - NFS client spoofed by unexpected mtime in post-op attributes
  - NFS client cache invalidated because WCC processing believed another client had written the file
- **Protocol Problem ?**
  - Out-of-order completions makes WCC very hard
  - Requires complex matrix of outstanding requests
- **Resolution**
  - Revert to V2 caching semantics (never use mtime)
- **User View**
  - Application runs 50x faster (all data lived in cache)

Oracle SGA

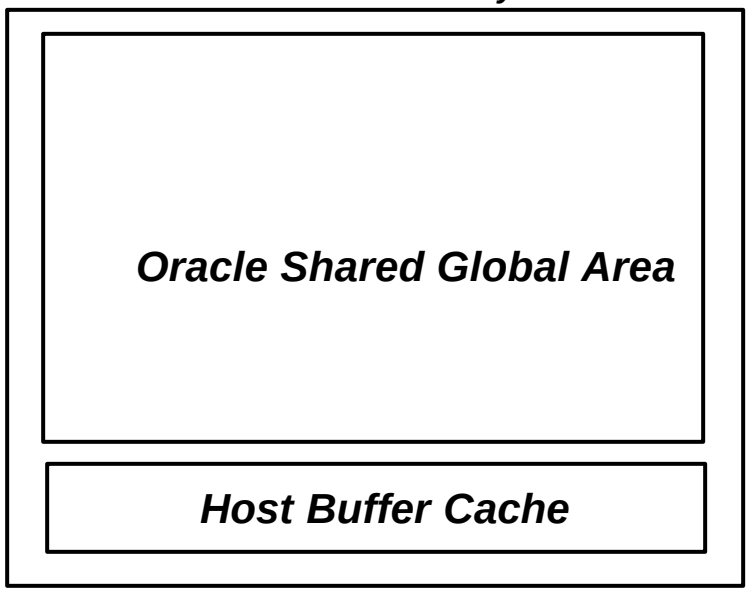
- **Consider the Oracle SGA paradigm**
  - **Basically an Application I/O Buffer Cache**

Configuration 1



- **Common w/32 bit Arch**
- **Or Multiple DB instances**

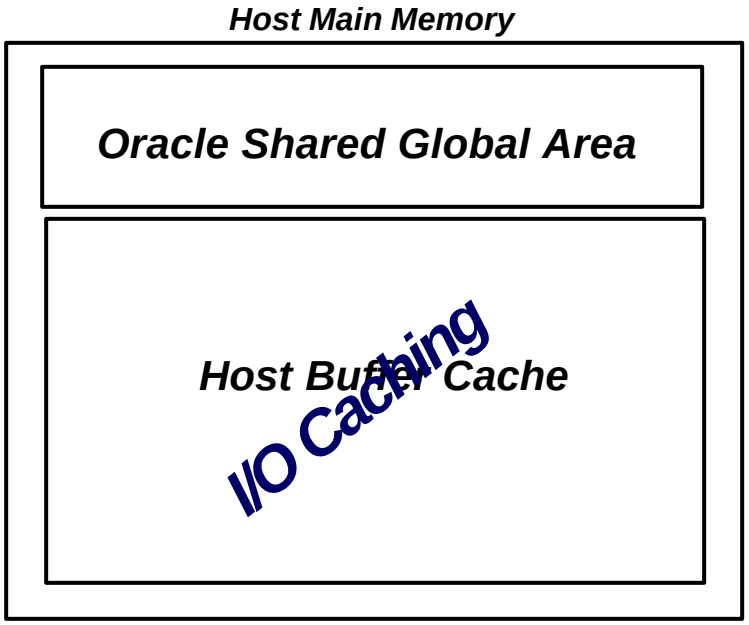
Configuration 2



- **Common w/64 bit Arch**
- **Or Small Memory Setups**

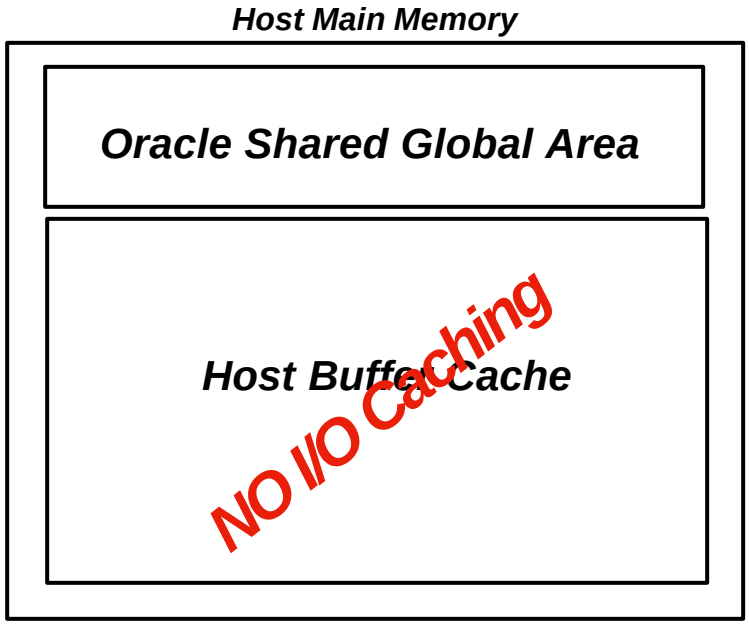
Oracle SGA – The "Cache" Escalation

With Local FS



- Very Little Physical I/O
- Application sees LOW latency

With NFS



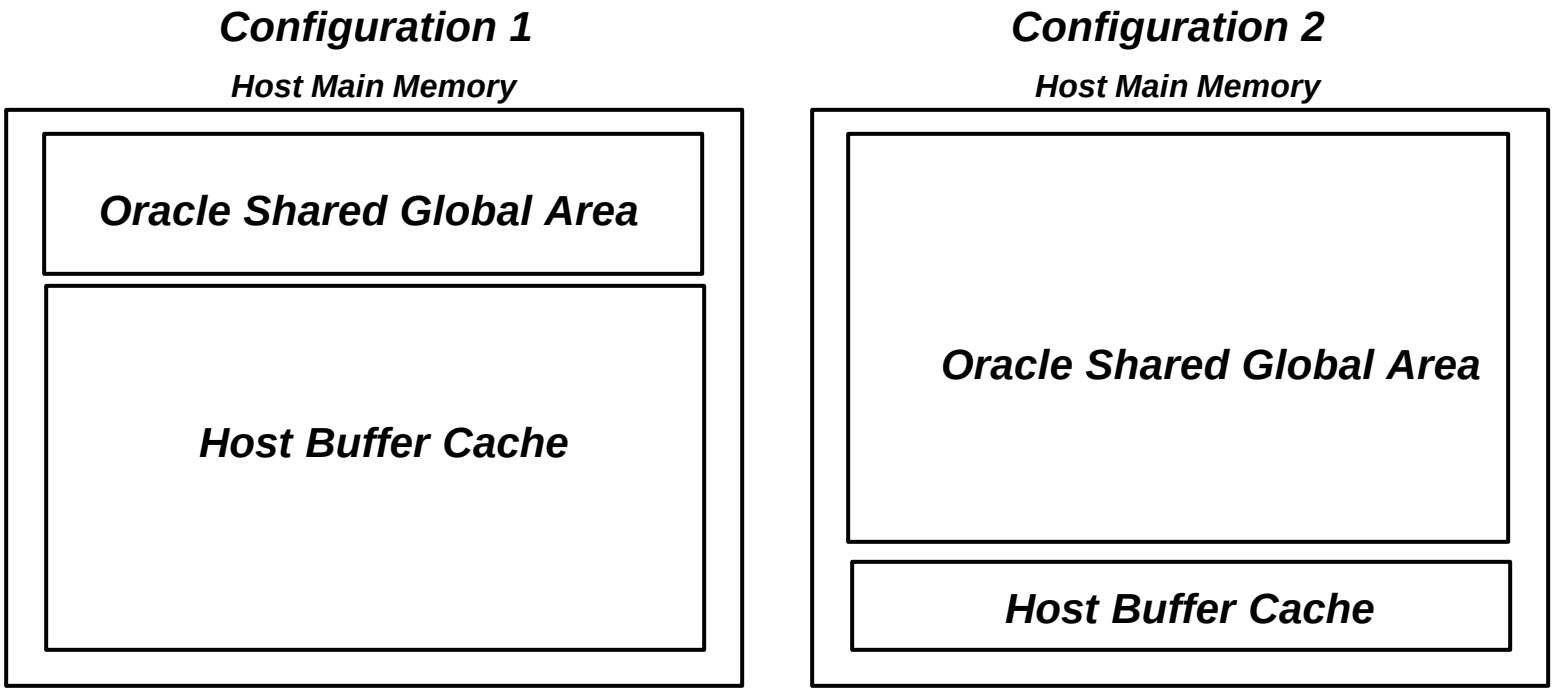
- Lots of Physical I/O
- Application sees HIGH latency

File Locks

- Commercial applications use different locking techniques
  - No Locking
  - Small internal byte range locking
  - Lock 0 to End of File
  - Lock 0 to Infinity (as large as file may grow)
- NFS Client behavior
  - Each client behaves differently with each type
  - Sometimes caching is disabled, sometimes not
  - Sometimes prefetch is triggered, sometimes not
  - Some clients have options to control behavior, some don’t
- DB Setups differ from Traditional Environment
  - Single host connected via 1 or more dedicated links
  - Multiple host locking is NOT a consideration

File Locks

- **Why does it matter so much?**
  - **Consider the Oracle SGA paradigm again**



- NOT caching here is deadly
- Locks are only relevant locally

- Caching here is a waste of resources
- Simply want to say “don’t bother”

Cache Control Features

- Most of the NFS clients have no “control”
  - Each client should have several “mount” options
    - (1) Turn caching off, period
    - (2) Don’t use locks as a cache invalidation clue
    - (3) Prefetch disabled
- Why are these needed
  - Application needs vary
  - Default NFS behavior usually wrong for DBs
  - System configurations vary

Over-Zealous Prefetch

- **Problem as viewed by User**
  - **Database on cheesy local disk**
    - **Performance is ok, but need NFS features**
  - **Setup bake-off, Local vs NFS, a DB batch job**
    - **Local results: Runtime X, disks busy**
  - **NFS Results**
    - **Runtime increases to 3X**
- **Why is this?**
    - **NFS server is larger/more expensive**
    - **AND, NFS server resources are SATURATED**
    - **?!? Phone rings…**

Over-Zealous Prefetch

- Debug by using a simple load generator to emulate DB workload
- Workload is 8K transfers, 100% read, random across large file
- Consider I/O issued by application vs I/O issued by NFS client

	Latency	App Ops	NFS 4K ops	NFS 32K ops	4Kops/App Op	32K ops/App op
8K 1 Thread	19.9	9254	21572	0	2.3	0.0
8K 2 Thread	7.9	9314	32388	9855	3.5	1.1
8K 16 Thread	510.6	9906	157690	80019	15.9	8.1

- NFS Client generating excessive, unneeded prefetch
- Resources being consumed needlessly
- Client vendor was surprised. Created a patch.
- Result: User workload faster on NFS than on Local FS

Poor Wire Efficiency – Some Examples

- **Some NFS clients artificially limit operation size**
  - **Limit of 8KB per write on some mount options**
- **Linux breaks all I/O into page-size chunks**
  - **If page size < rsize/wsize, I/O requests may be split on the wire**
  - **If page size > rsize/wsize, operations will be split and serialized**
- **The User View**
  - **No idea about wire level transfers**
  - **Only sees that NFS is SLOW compared to Local**

RPC Slot Limitation

- **Consider a Linux Setup**
  - **Beefy server, large I/O subsystem, DB workload**
  - **Under heavy I/O load**
    - **Idle Host CPU, Idle NFS server CPU**
    - **Throughput significantly below Wire/NIC capacity**
    - **User complains workload takes too long to run**
- **Clues**
  - **Using simple I/O load generator**
  - **Study I/O throughput as concurrency increases**
  - **Result: No increase in throughput past 16 threads**

RPC Slot Limitation

- **Little's Law**
  - I/O limitation explained by Little's Law
  - Throughput is proportional to latency and concurrency
  - To increase throughput, increase concurrency
- **Linux NFS Client**
  - RPC slot table has only 16 slots
  - At most 16 outstanding I/O's per mount point, even when there are hundreds of disks behind that mount point
  - Artificial Limitation
- **User View**
  - Linux NFS performance inferior to Local FS
  - Must Recompile kernel or wait for fix in future release

Writers Block Readers

- **Symptom**
  - Throughput on single mount point is poor
  - User workload extremely slow compared to Local
  - No identifiable resource bottleneck
- **Debug**
  - Emulate User workload, study results
  - Throughput with only Reads is very high
  - Adding a single writer kills throughput
  - Discover writers block readers needlessly
- **Fix**
  - Vendor simply removed R/W lock when performing direct I/O

Applications Also Have Issues

- Some commercial apps are “two-brained”
  - Use “raw” interface for local storage
  - Use filesystem interface for NFS storage
  - Different code paths have major differences
    - Async I/O
    - Concurrency settings
    - Level of code optimization
- Not an NFS problem, but is a solution inhibitor

Why is this Happening?

- Is NFS a bad solution? Absolutely not!
- NFS began with a specific mission
  - Semi-wide area sharing
  - Home directories and shared data
- Note: problems are NOT with NFS protocol
  - Mostly client implementation issues
- Are the implementations bad? …

Why is this Happening?

- **The implementations are NOT bad.**
- **The Mission has changed!**
  - **Narrow sharing environment**
  - **Typically dedicated (often p2p) networks**
  - **Data sharing → High-speed I/O Interconnect**
  - **Mission evolved to Mission Critical Workloads**
- **Actually, NFS has done ok**
  - **Credit a strong protocol design**
  - **Credit decent engineering on the implementations**

Why are things Harder for NFS?

- **What makes Database + NFS different than Local FS?**
  - **For Local Filesystem Caching is simple**
    - **Just do it**
    - **No multi-host coherency issues**
  - **NFS is different**
    - **By default must be concerned about sharing**
    - **Decisions about when to cache/not, prefetch/not**

Why are things Harder for NFS?

- **Database + Filesystem Caching is complex**
  - **Most database deployments are single host (modulo RAC)**
    - **So, cross host coherency not an issue**
    - **However, Users get nervous about relaxing locks**
  - **Databases lock files (many apps don't)**
    - **Causes consternation for caching algorithms**
  - **Databases sometimes manage their own cache (ala Oracle SGA)**
    - **May or may not act in concert with host buffer cache**

Whitepaper on Solaris, NFS, and Database

- **Joint Sun / NetApp White Paper**
  - **NFS and Oracle and Solaris and NetApp**
  - **High level and Gory Detail both**
- **Title**
  - **Database Performance with NAS: Optimizing Oracle on NFS**
- **Where**
  - **http://www.sun.com/bigadmin/content/nas/sun_netapps_rdbms_wp.pdf**
  - **(or http://www.netapp.com/tech_library/ftp/3322.pdf)**

NFS Performance Considerations

NFS Implementation

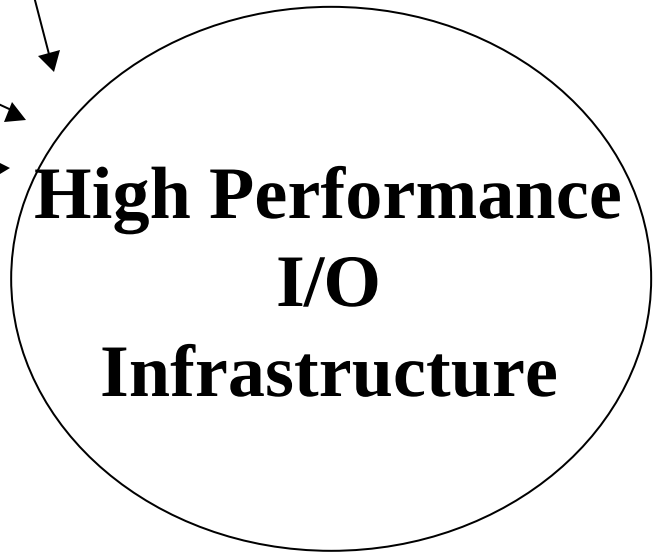
- **Up-to-date Patch levels**
- **NFS Clients – Not all Equal**
  - **Strengths/Weaknesses/ Maturity**
- **NFS Servers**
  - **NetApp filers – most advanced**

Network Configuration

- **Topology – Gigabit, VLAN**
- **Protocol Configuration**
  - **UDP vs TCP**
  - **Flow Control**
  - **Jumbo Ethernet Frames**

NFS Configuration

- **Concurrency and Prefetching**
- **Data sharing and file locking**
- **Client caching behavior**



NFS Scorecard – What and Why

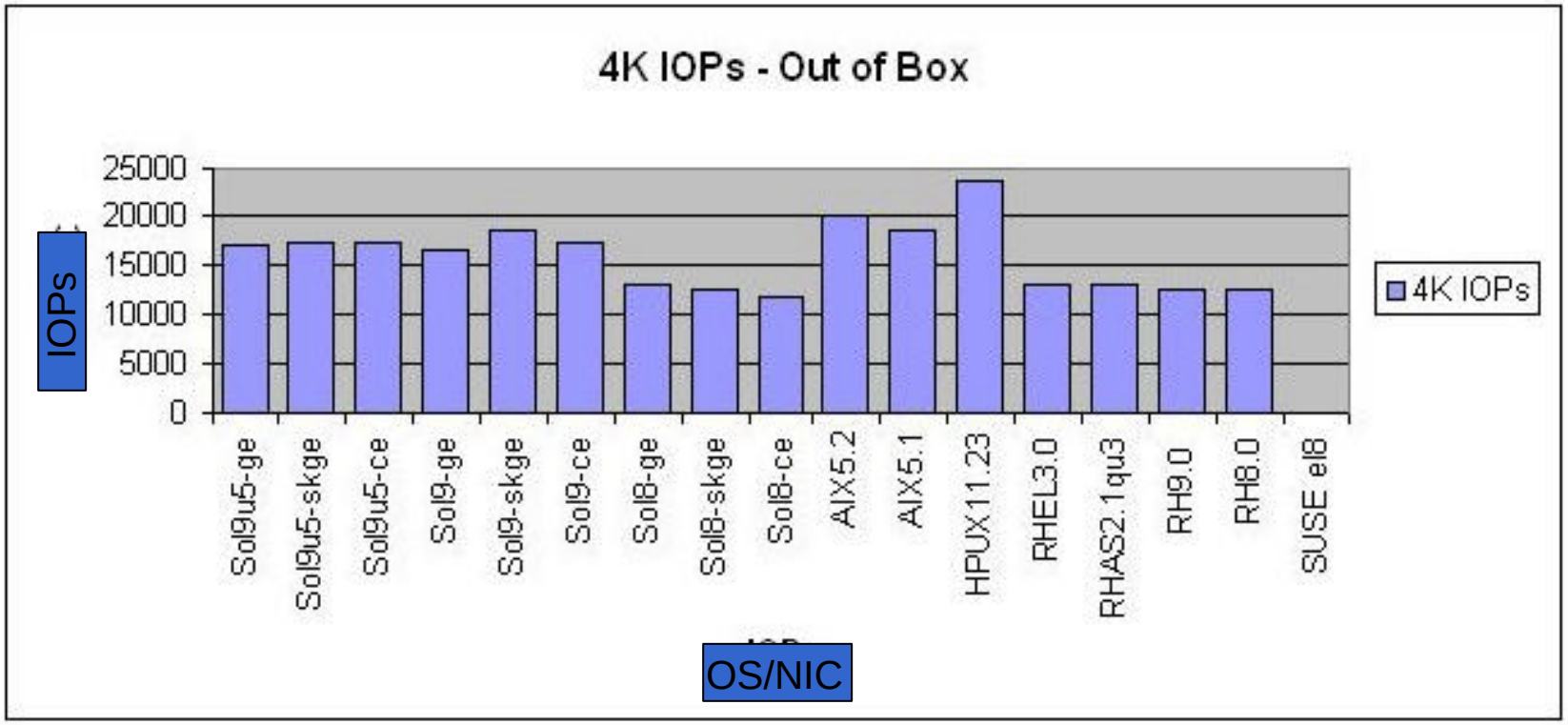
- **Comparison of all NFS clients**
  - **On all OS platforms, releases, NICs**
- **Several major result categories**
  - **Out of box basic performance**
    - **Maximum IOPs, MB/s, and CPU Cost of NFS vs Local**
  - **Others**
    - **Well-Tuned Basic Performance**
    - **Mount Features**
    - **Filesystem Performance and Semantics**
    - **Wire Efficiency**
    - **Scaling / Concurrency**
    - **Database Suitability**

NFS Scorecard - caveat

- This is a metric, not a benchmark or measure of goodness
- “Goodness” is VERY workload-dependent
- For example
  - High 4KB IOPS is key metric for databases
  - But possibly not for user home directories
  - Low overhead is also key, and may not correlate
- But this is a start…

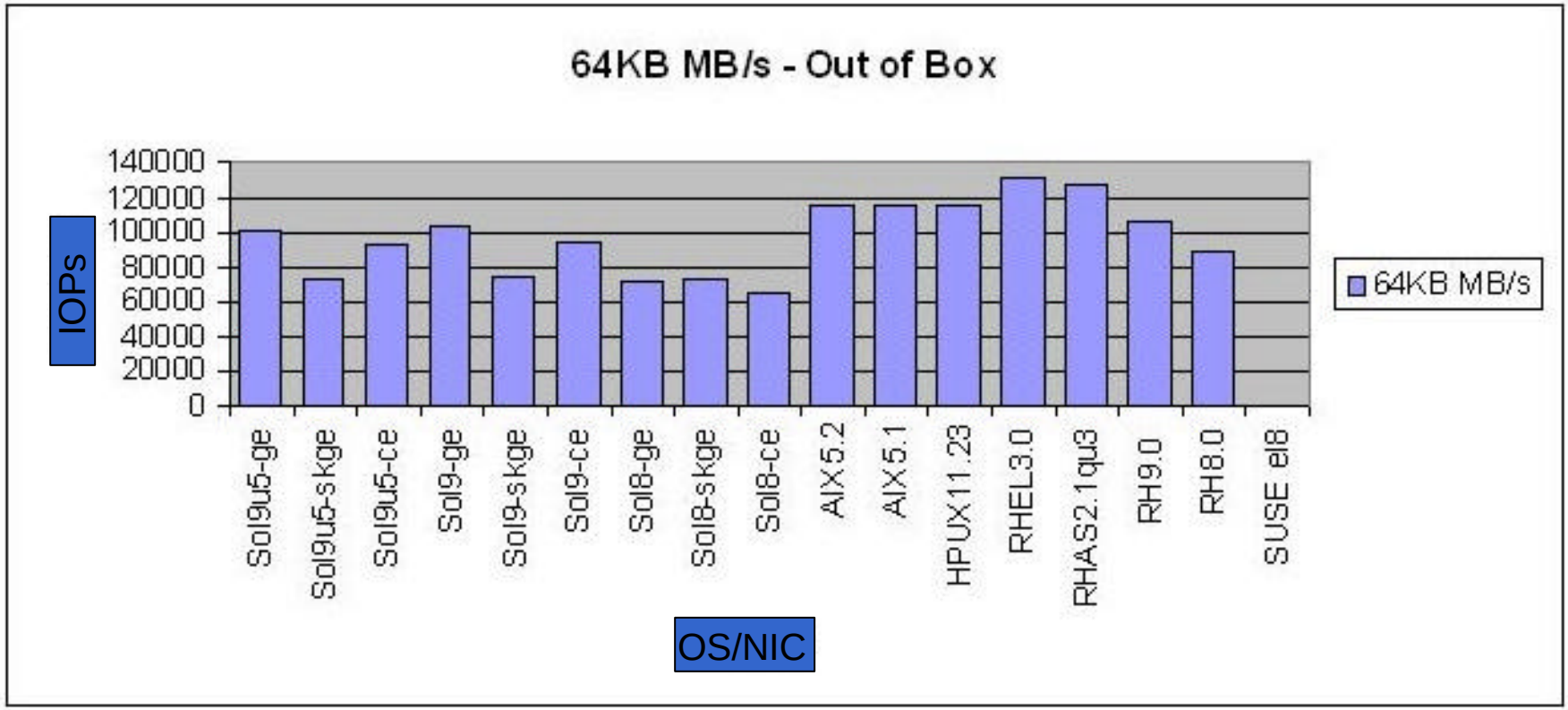
NFS Scorecard – IOPs and MB/s

- **4K IOPs Out-of-box**



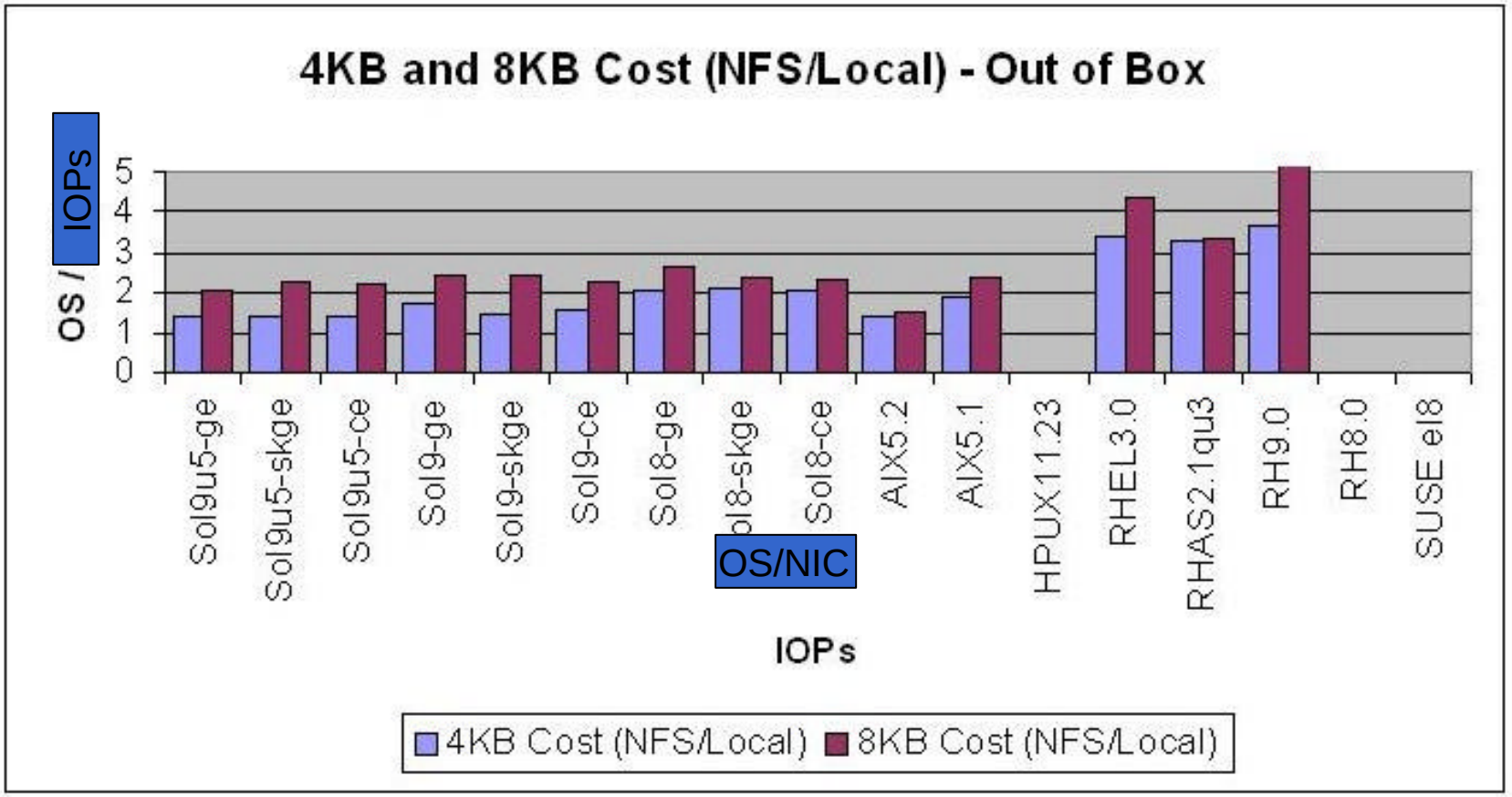
NFS Scorecard – IOPs and MB/s

- **64K MB/s Out-of-box**



NFS Scorecard – Costs

- 4K and 8K Cost per I/O – NFS / Local
- Bigger is Worse!



SIO – What and Why

- **What is SIO?**
  - **A NetApp authored tool**
    - **Available through support channel**
  - **Not magic. Similar tools exist. Just useful.**
  - **Simulated I/O generator**
    - **Generate I/O load with specifics:**
      - **read/write mix, concurrency, data set size**
      - **I/O size, random/sequential**
    - **Works on all devices and protocols: files, blocks, iscsi**
    - **Reports some basic results**
      - **IOPs, MB/s (others also)**

SIO – What and Why (cont)

- **Why use SIO?**
  - Controlled workload is imperative
  - Same tool on all platforms
  - Emulate multiple scenarios
  - Easy to deploy and run
  - Better than
    - dd – single threaded (most cases)
    - cp – who knows what is really happening
    - real world setup – often hard to reproduce
  - Demonstrate performance for
    - Users, validation, bounding maximum
  - Find performance bottlenecks

NFS Futures – RDMA

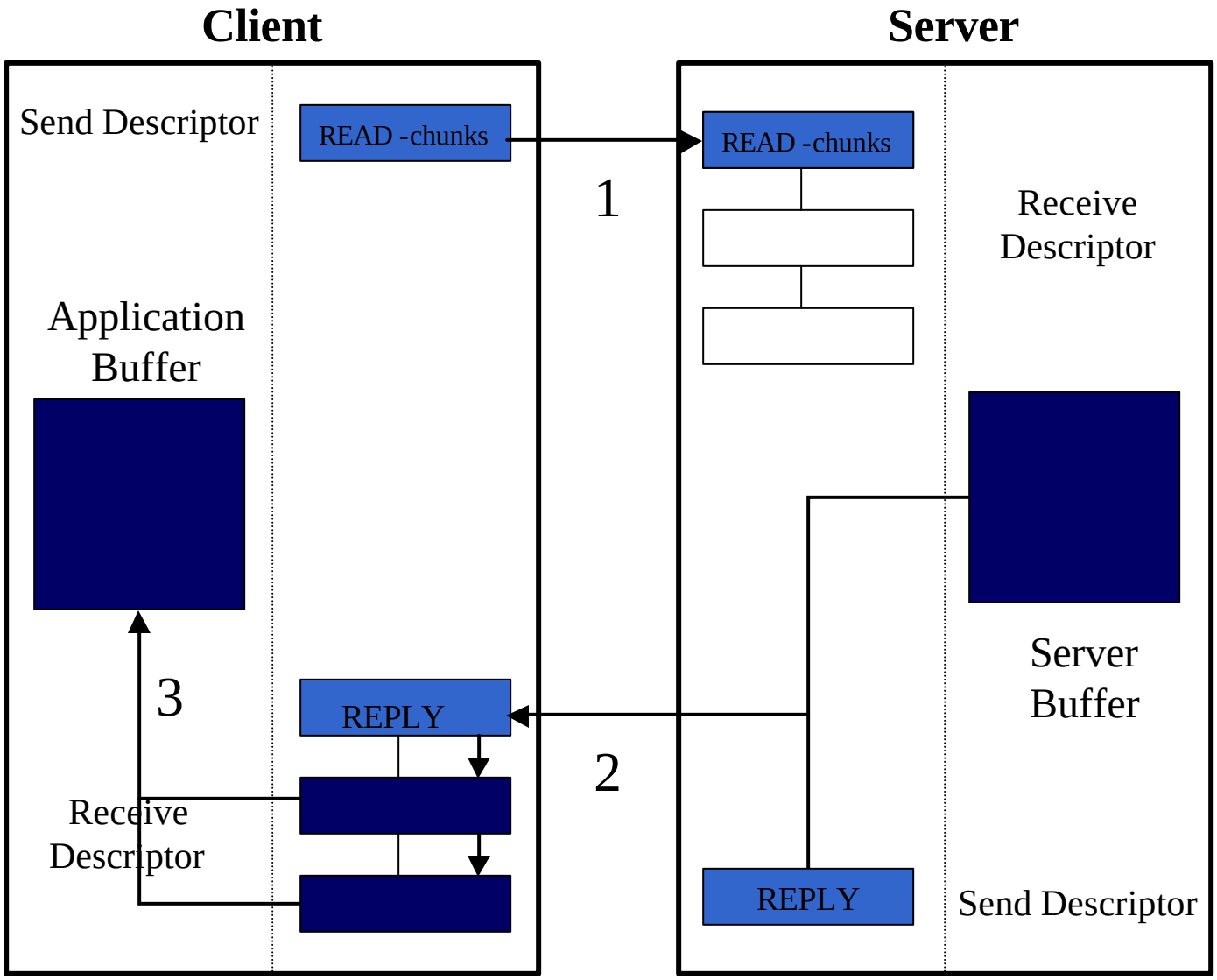
What is NFS/RDMA

- A binding of NFS v2, v3, v4 atop RDMA transport such as Infiniband, iWARP
- A significant performance optimization
- An enabler for NAS in the high-end
  - Databases, cluster computing, etc
  - Scalable cluster/distributed filesystem

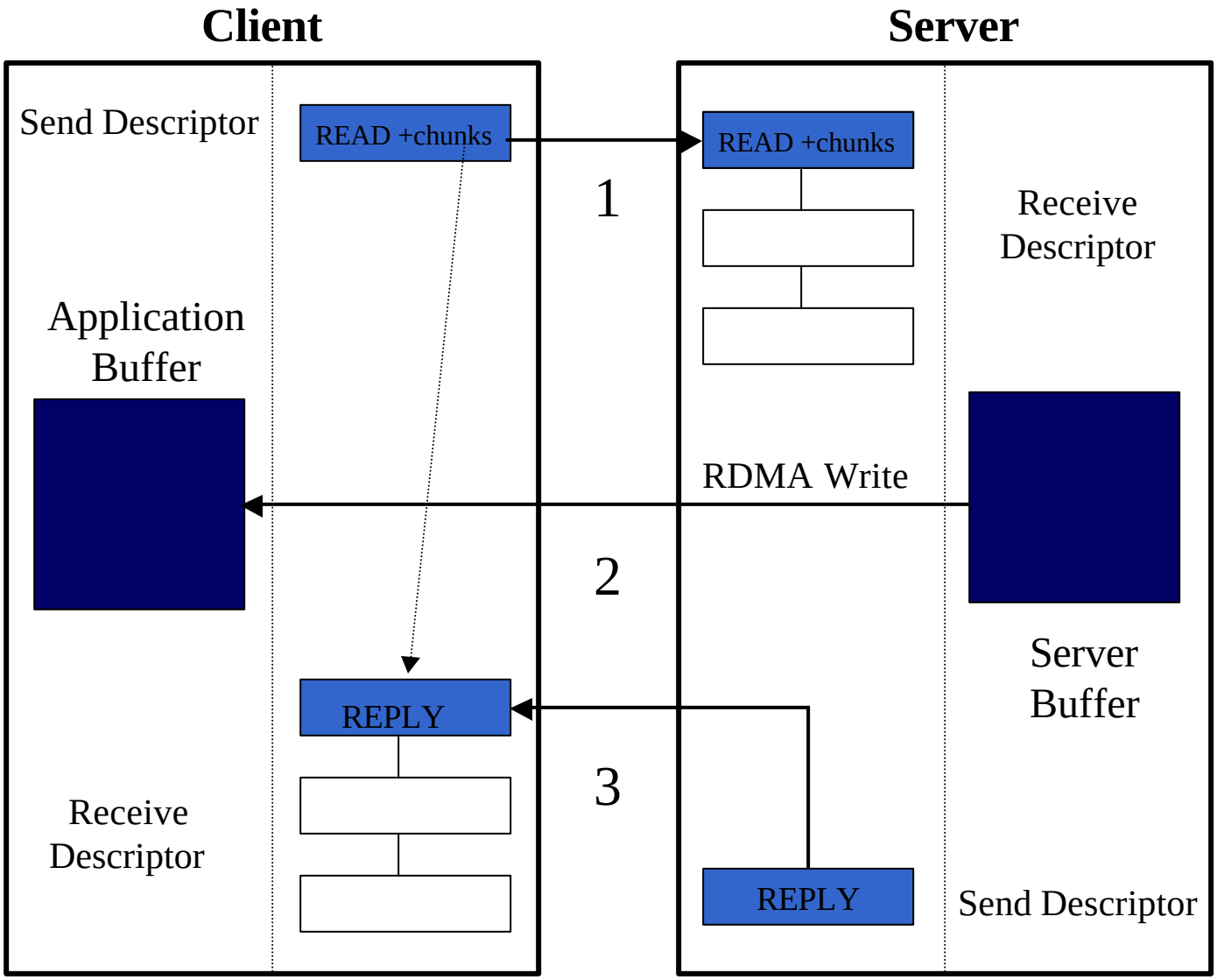
Benefits of RDMA

- Reduced Client Overhead
- Data copy avoidance (zero-copy)
- Userspace I/O (OS Bypass)
- Reduced latency
- Increased throughput, ops/sec

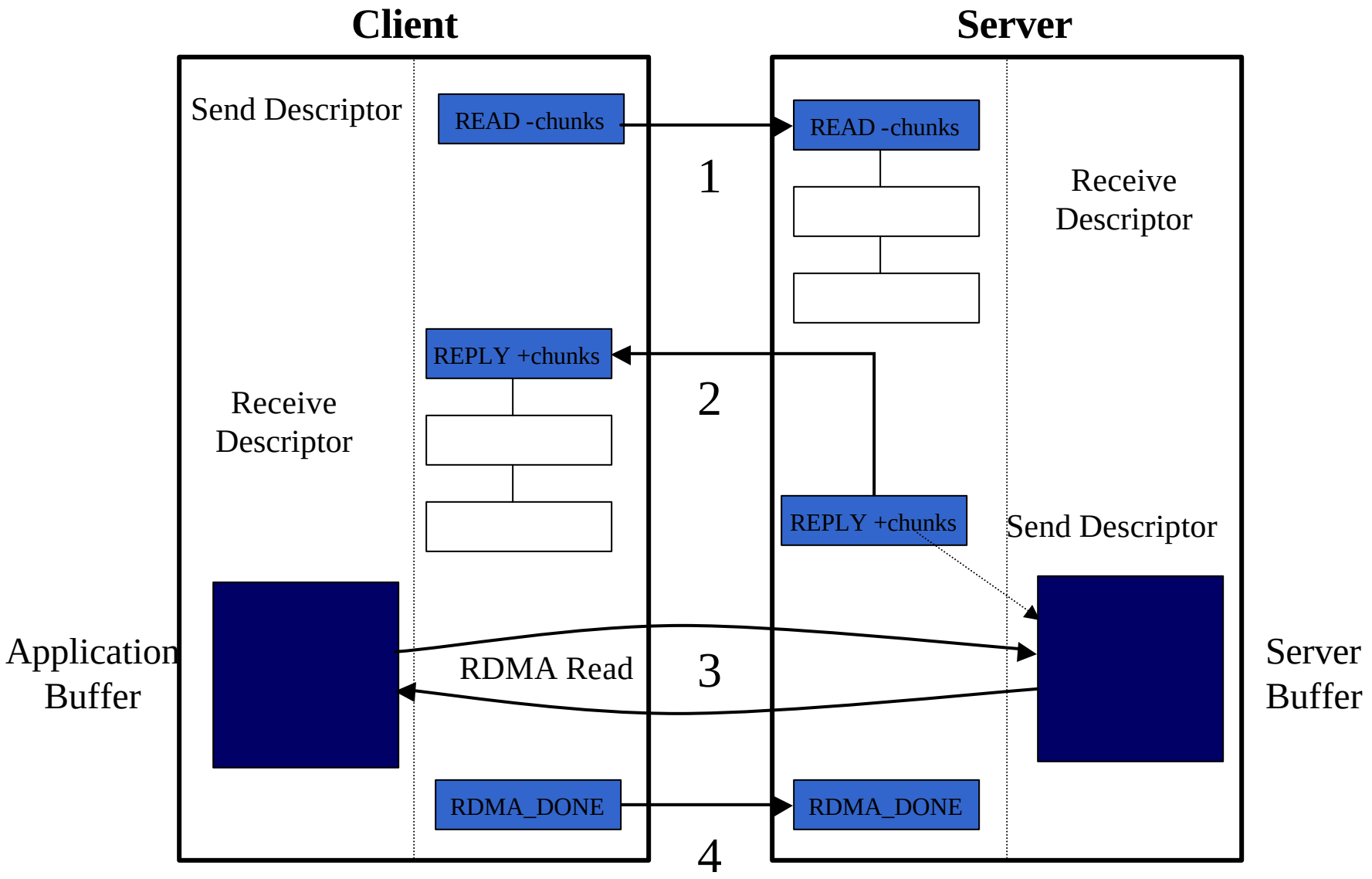
Inline Read


Client
Server
Send Descriptor
READ -chunks
READ -chunks
1
Receive Descriptor
Application Buffer
Server Buffer
3
REPLY
2
Receive Descriptor
REPLY
Send Descriptor

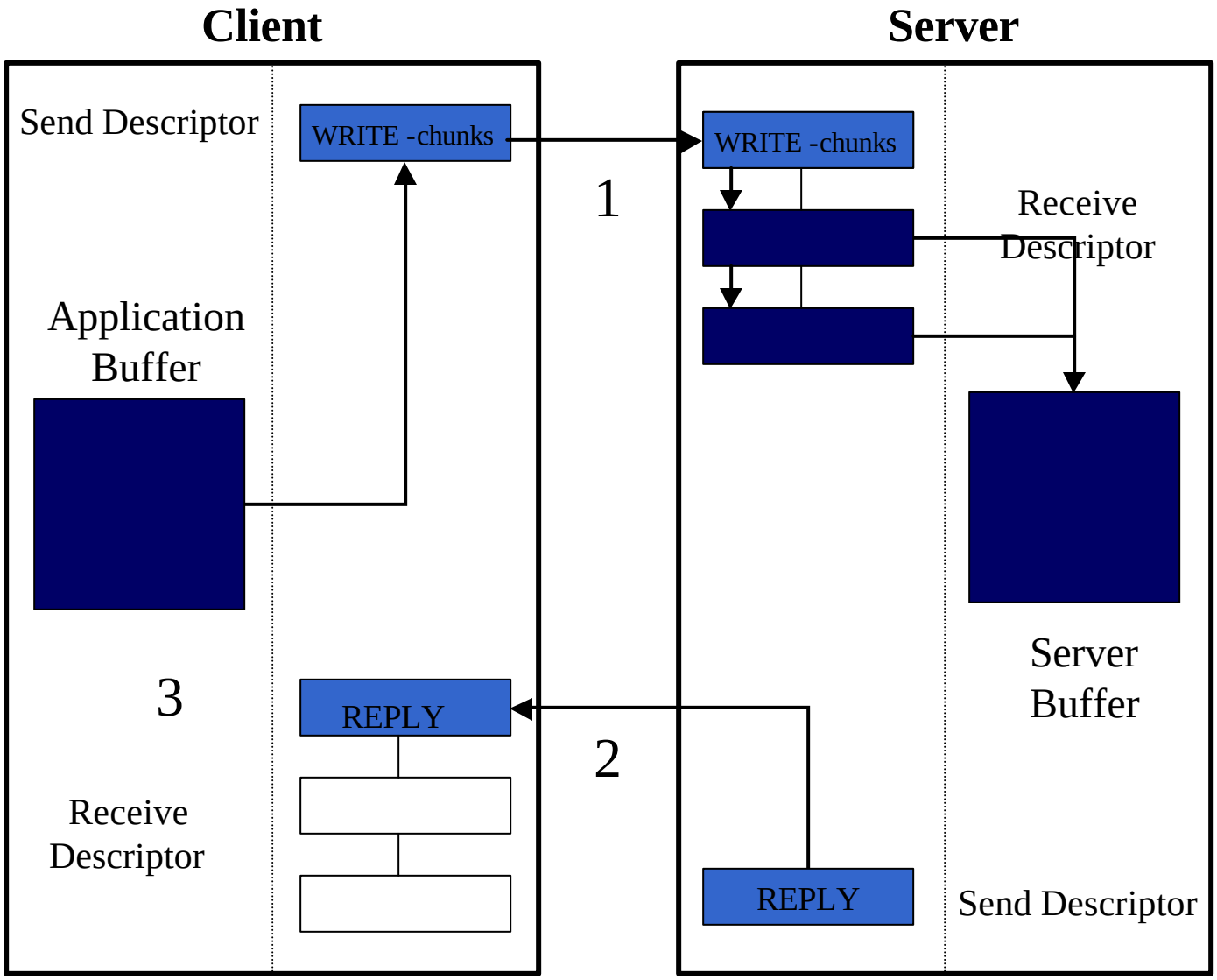
Direct Read (write chunks)



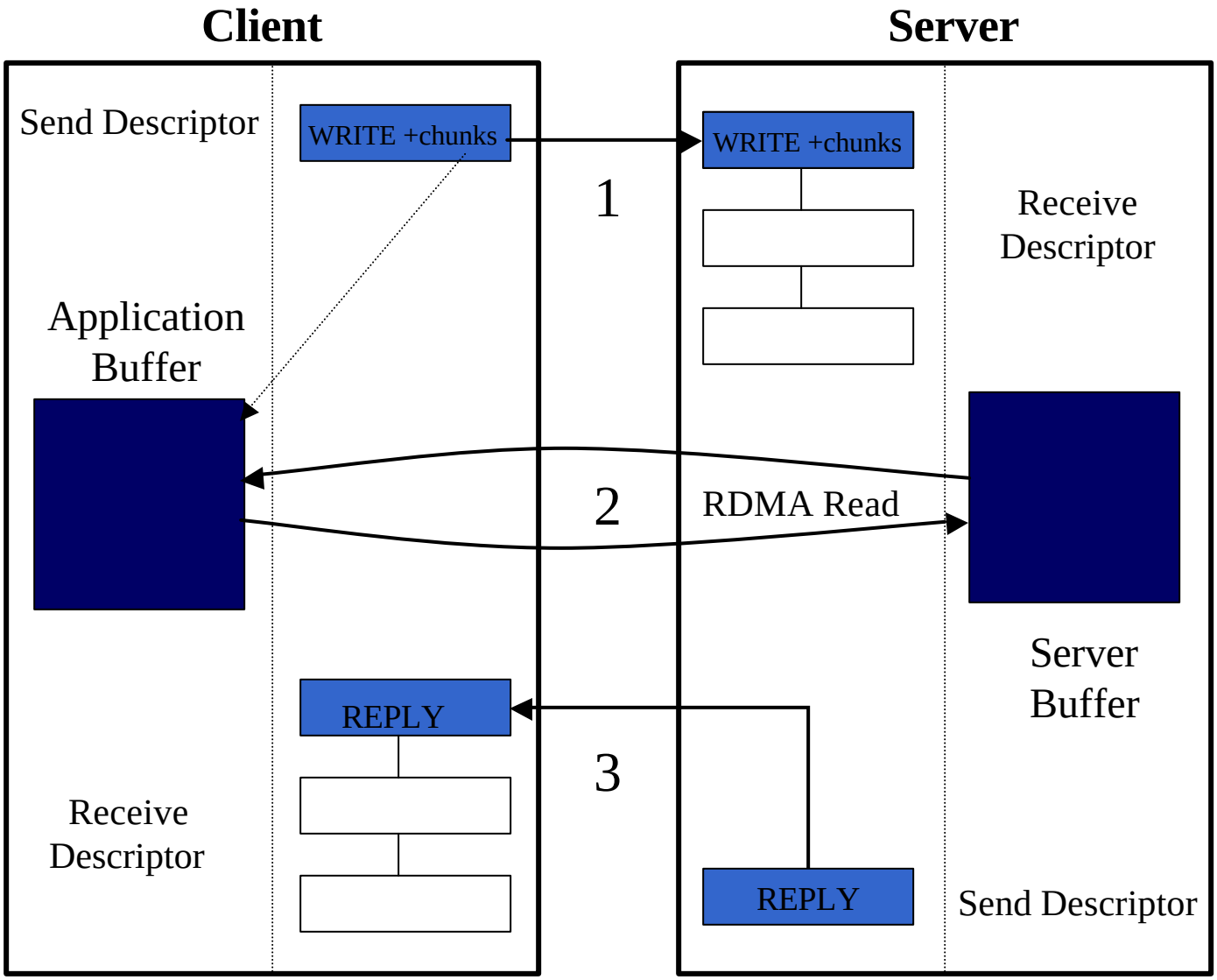
Direct Read (read chunks) – Rarely used



Inline Write



Direct Write (read chunks)



NFS/RDMA Internet-Drafts

- IETF NFSv4 Working Group
- RDMA Transport for ONC RPC
  - Basic ONC RPC transport definition for RDMA
  - Transparent, or nearly so, for all ONC ULPs
- NFS Direct Data Placement
  - Maps NFS v2, v3 and v4 to RDMA
- NFSv4 RDMA and Session extensions
  - Transport-independent Session model
  - Enables exactly-once semantics
  - Sharpens v4 over RDMA

ONC RPC over RDMA

- **Internet Draft**
  - **draft-ietf-nfsv4-rpcrdma-00**
  - **Brent Callaghan and Tom Talpey**
- **Defines new RDMA RPC transport type**
- **Goal: Performance**
  - **Achieved through use of RDMA for copy avoidance**
  - **No semantic extensions**

NFS Direct Data Placement

- Internet Draft
  - draft-ietf-nfsv4-nfsdirect-00
  - Brent Callaghan and Tom Talpey
- Defines NFSv2 and v3 operations mapped to RDMA
  - READ and READLINK
- Also defines NFSv4 COMPOUND
  - READ and READLINK

NFSv4 Session Extensions

- **Internet Draft**
  - **draft-ietf-nfsv4-session-00**
  - **Tom Talpey, Spencer Shepler and Jon Bauman**
- **Defines NFSv4 extension to support:**
  - **Persistent Session association**
  - **Reliable server reply caching (idempotency)**
  - **Trunking/multipathing**
  - **Transport flexibility**
    - **E.g. callback channel sharing w/operations**
    - **Firewall-friendly**

Others

- NFS/RDMA Problem Statement
  - Published February 2004
  - draft-ietf-nfsv4-nfs-rdma-problem-statement-00
- NFS/RDMA Requirements
  - Published December 2003

Q&A

- **Questions/comments/discussion?**