NFS on the Fast track - fine tuning and futures

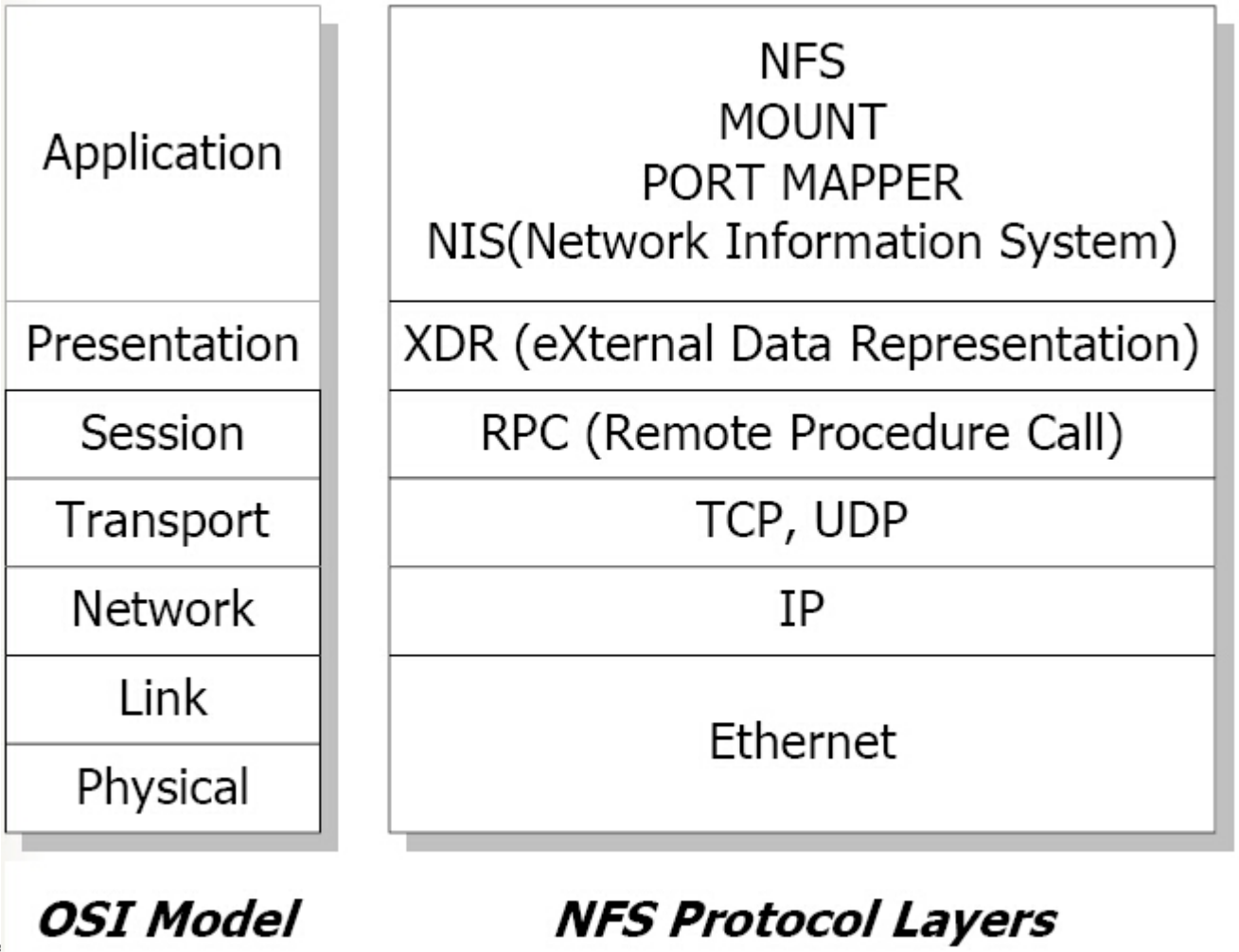
Bikash Roy Choudhury

Solutions Architect, NetApp

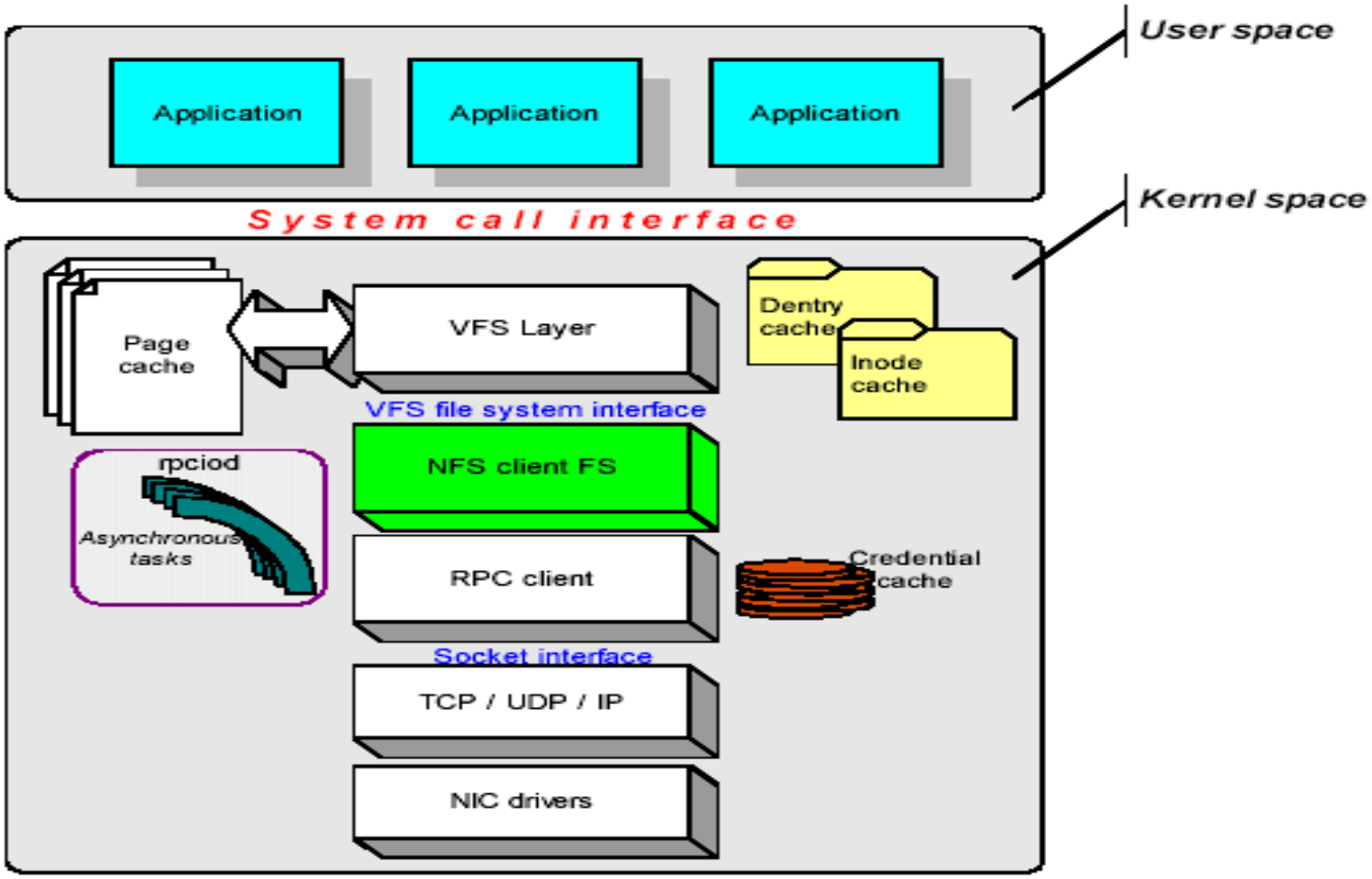
Agenda

- Overview of NFS layers – Linux Client
- Why is the NFS performance Slow?
- Understanding application behavior for NFS
- When is performance tuning needed
- TCP Tuning
- Network considerations for better NFS performance
- System Tuning guidelines
- Tools
- Status of the NFS development community
- QA

NFS and the OSI-ISO Network Model



Linux NFS Client Architecture



Slow NFS Performance ! Why?

- The slowness could mainly be for various **client** reasons
  - Old client OS version
  - Client OS not having recent patches
  - The client hardware platform is old
  - The NIC drivers on the client network interfaces are not up to date

Slow NFS Performance ! Why?

- The client does not have adequate memory for caching
- The mount options does not have recommended values
- Still on NFSv2?

Things to Consider for Applications over NFS

- Apps. should avoid unnecessarily opening and closing files – Too many ***GETATTR***s
- App-based read-ahead can benefit from NFS preset large read and write transaction size – ***rsize/wsize*** set to 64k or more
  - rw,vers=3,rsize=65536,wsize=65536,hard, proto=tcp,timeo=600,retrans=5

Things to Consider for Applications over NFS

- Apps. should be able to handle the ***throughput*** and the ***latency*** of NFS

- Apps. should be responsible to set appropriate ***locks*** on files
  - App. should choose to ***lock*** and ***unlock*** the file before and after every individual write operation

High-performance NFS infrastructures

- **Sharing**
  - share data coherently across multiple host platforms
- **Caching**
  - Data read once can be cached in the host buffer cache
  - Data written to the host buffer cache is first written to the NFS server
  - Data set size plays a role in host buffer caching

High-performance NFS infrastructures

- **Locking**
  - lock manager places locks on the storage system
    - Locks are scoped only to the local host (***nolock***)
  - Locks *must* be managed at the file level so that all the nodes are aware of file locks

When to tune NFS Clients?

- Tuning is not always required if recommended values of NFS parameters are followed

- All tuning parameters are not standard for all kind of workloads

When to tune NFS Clients?

- Use the latest client OS version with the recent patches/packages and drivers
- Understand the nature of the application and its workload before any tuning

TCP Tuning

- **Use the TCP transport**
  - More reliable and low risk of data corruption and better congestion control compared to UDP
- **Enlarge TCP window size for fast response**
  - TCP Read buffer
    - **net.ipv4.tcp_rmem = 4096 524288 16777216**
  - TCP Write buffer
    - **net.ipv4.tcp_wmem = 4096 524288 16777216**
  - TCP Buffer Space
    - **net.ipv4.tcp_mem = 16384 16384 16384**
- **Miscellaneous TCP settings**
  - net.ipv4.tcp_timestamps - toggles TCP timestamp support
  - net.ipv4.tcp_sack - toggles SACK (Selective ACK) support

Network Topology Considerations

- Use **10**Gige Ethernet
- Use jumbo frames – **9000** or **8192** MTU size wherever needed
  - Make appropriate changes in the **/etc/config/network-scripts**
- Set **net.core.netdev_max_backlog = 30000**
  - how many unprocessed rx packets before kernel starts to drop them
- Enable TCP window scaling
  - **sysctl –w net.ipv4.tcp_window_scaling=1**

Some general system Tuning guidelines

- To maximize throughput
  - Disable irqbalance
    - service irqbalance stop
    - chkconfig irqbalance off
  - Disable CPU Speed
    - default gov=ondemand, set governor to performance
  - Set the "**sunrpc.tcp_slot_table_entries**" to 128
    - Removes a throttle and improves IO between the Linux® node and the backend storage system
  - Set the IO Scheduler to NOOP
    - Handles the IO much better with RAID storage

Tools

- mpstat - reveals per cpu stats, Hard/Soft Interrupt usage
- vmstat – vm page info, context switch, total ints/s, cpu
- netstat – per nic status, errors, statistics at driver level
- ethtool – View and change Ethernet card settings
- sysctl – View and set */proc/sys settings*
- ifconfig – View and set ethX variables
- netperf – Can run a bunch of different network tests
- oprofile – system level profiling, kernel/driver code

Status of the NFS development community

- Oracle is largely responsible for the IPv6 project.
- CITI, Panasas and NetApp have collaborated on NFSv4.1 and pNFS
- Red Hat contributes a lot of stability and performance related patches
  - Also focussed on toolchain (nfs-utils, libtirpc and rpcbind)
- This all adds up to a healthy development model!
  - A lot more differing interests are being served.
  - There are no monopolies.

General NFS client changes

- Finer grained spin locks to reduce SMP contention.
  - The Big Kernel Lock is finally gone!
  - Where possible, some NFSv4 locks were converted to lockless schemes (i.e. Read Copy Update)
- Cachefs has finally been merged into the upstream kernel
  - So far, it provides read-only file caching
- Support for IPv6 and RDMA transport mechanisms
  - RPC RDMA runs on Infiniband and iWARP hardware in native RDMA mode..

Experimental support for NFSv4.1

- Development is being driven by a community consisting of teams from the University of Michigan/CITI, IBM, NetApp and Panasas.
- A basic NFSv4.1 client has been merged into Linux 2.6.31.
  - Builds on top of the existing NFSv4 code, and shares code where possible.
  - Contains the minimal functionality required in order to correctly interoperate with a NFSv4.1 server
    - Mainly NFSv4.1 session and backchannel support.
  - pNFS clients are under development, but have not yet been merged into mainline.

Thank You

bikash@netapp.com