SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2015

High Resiliency Parallel NAS Cluster

Richard Levy
Peer Fusion

Topics

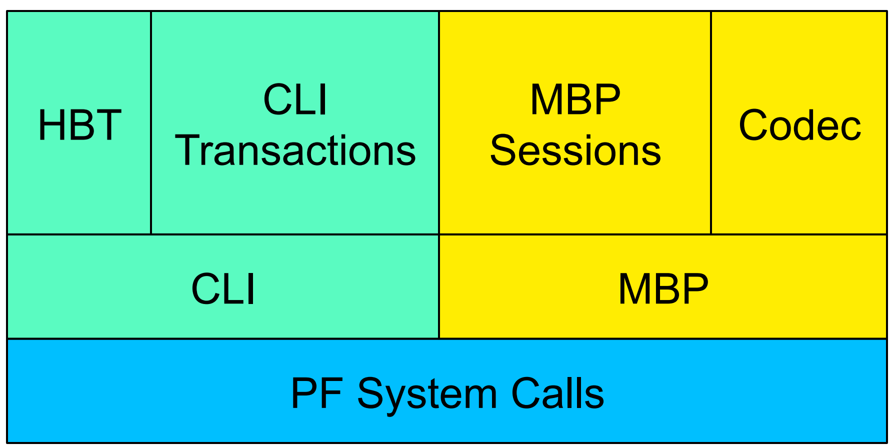
- What is the Peer Fusion File System?
- PFFS Motivating Goals
- PFFS Sub-systems
- Resiliency
- Performance
- Scalability
- Roadmap

What Is The Peer Fusion File System?

- A parallel file system
  - Peer-to-peer
  - Highly resilient
  - No data replication
  - Parallel everything: I/O, FEC
  - High performance
  - Simple to Administer

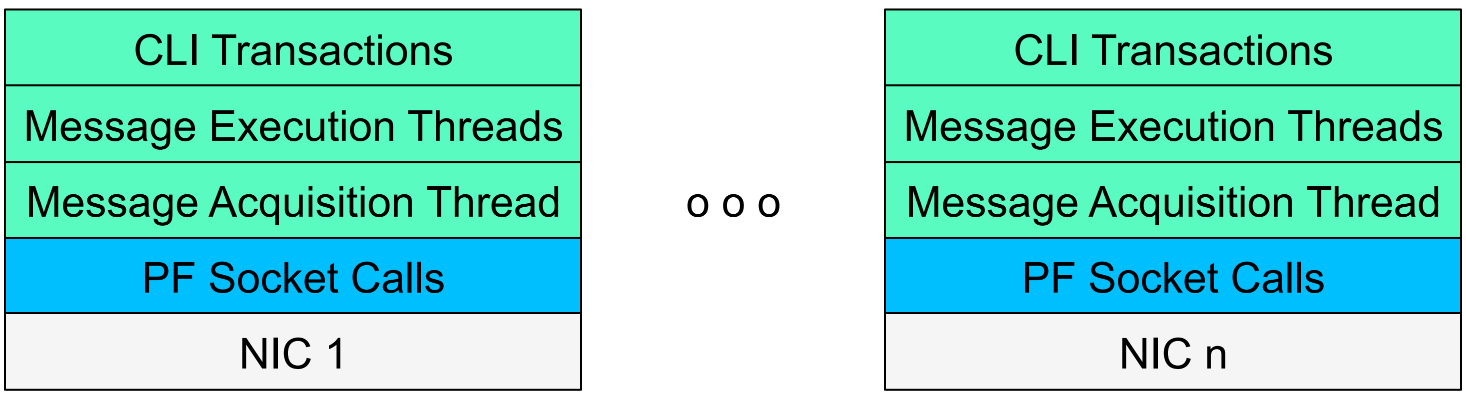
PFFS Modules

- CLI – the protocol used for unstructured messages
  - HBT – the heartbeat and cluster quorum module
  - CLI Transactions – the library of methods (open/link/mkdir/…)
- MBP – the protocol used for file I/O
  - MBP Sessions – the library of methods (read/write/trunc/…)
  - Codec – the Reed-Solomon high-performance codec



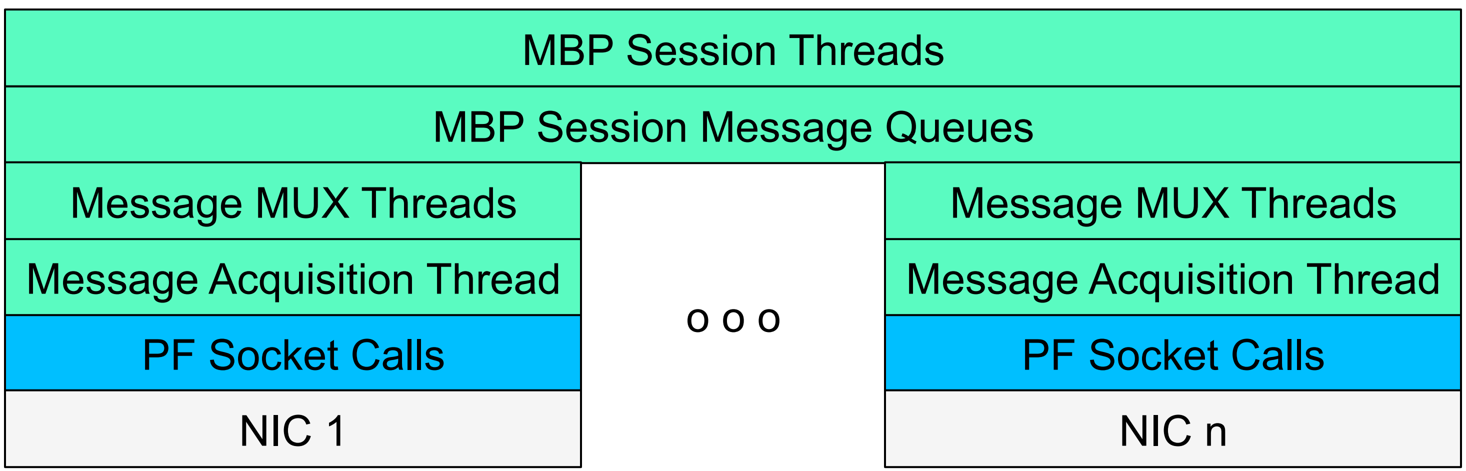
PFFS CLI Module

- A dedicated thread per NIC receives and queues messages
- A configurable number of threads are available to process queued messages
  - Parse the message header to determine its type
  - Invoke the corresponding method



PFFS MBP Module

- A dedicated thread per NIC receives and queues messages
- A configurable number of threads are available to process queued messages
  - Lookup the MBP session by its identification in the message
  - Insert the message into the corresponding MBP session queue and wake up the session thread



PFFS Motivating Goals

- Resiliency
  - No file data replication
  - Configurable to multiple peer failures
  - Peer failures must not disrupt applications
  - Gradual degradation of performance with each peer failure
  - Maintain high I/O throughput

PFFS Motivating Goals

- Performance
  - Balanced and distributed cluster workload
    - All the work is performed by the peers
      - Files: read/write/encode/decode/repair/…
      - Namespace: create/open/mkdir/rmdir/link/unlink/…
    - All the peers share the work equally
  - High throughput
  - Highly multi-threaded

PFFS Motivating Goals

- Scalability - Additional Peers Provide
  - Greater cluster resiliency
  - Greater cluster storage capacity
  - Greater Cluster performance:
    - Network bandwidth, CPU, RAM
    - More parallel I/O
    - Less I/O per peer

PFFS Motivating Goals

- Low Cost of Ownership
  - Commodity hardware
  - No data replication
  - No idle peers
- Ease of Administration
  - Easy to install
  - Simple to configure
  - Plug and Play

Design Considerations - Gateways

- Configuration file (describes the cluster)
- Perform network discovery (plug-and-play)
- Exchange heartbeats that describe the node's LANs, status and various metrics
- No persistent user data on gateways
  - Namespace is replicated across the peers
  - User data is striped across the peers
- POSIX compliant file system:
  - Kernel VFS (FreeBSD)
  - FUSE (Linux)

Design Considerations - Peers

- Configuration file (describes local resources)
- Perform network discovery (plug-and-play)
- Exchange heartbeats that describe the node's LANs, status and various metrics
- Peers receive configuration information from the gateway upon network discovery and any changes in heartbeats
- All work is to be done by the cluster (peers):
  - Namespace
  - I/O to storage
  - Encoding, decoding and on-the-fly repairs
  - Healing/re-striping

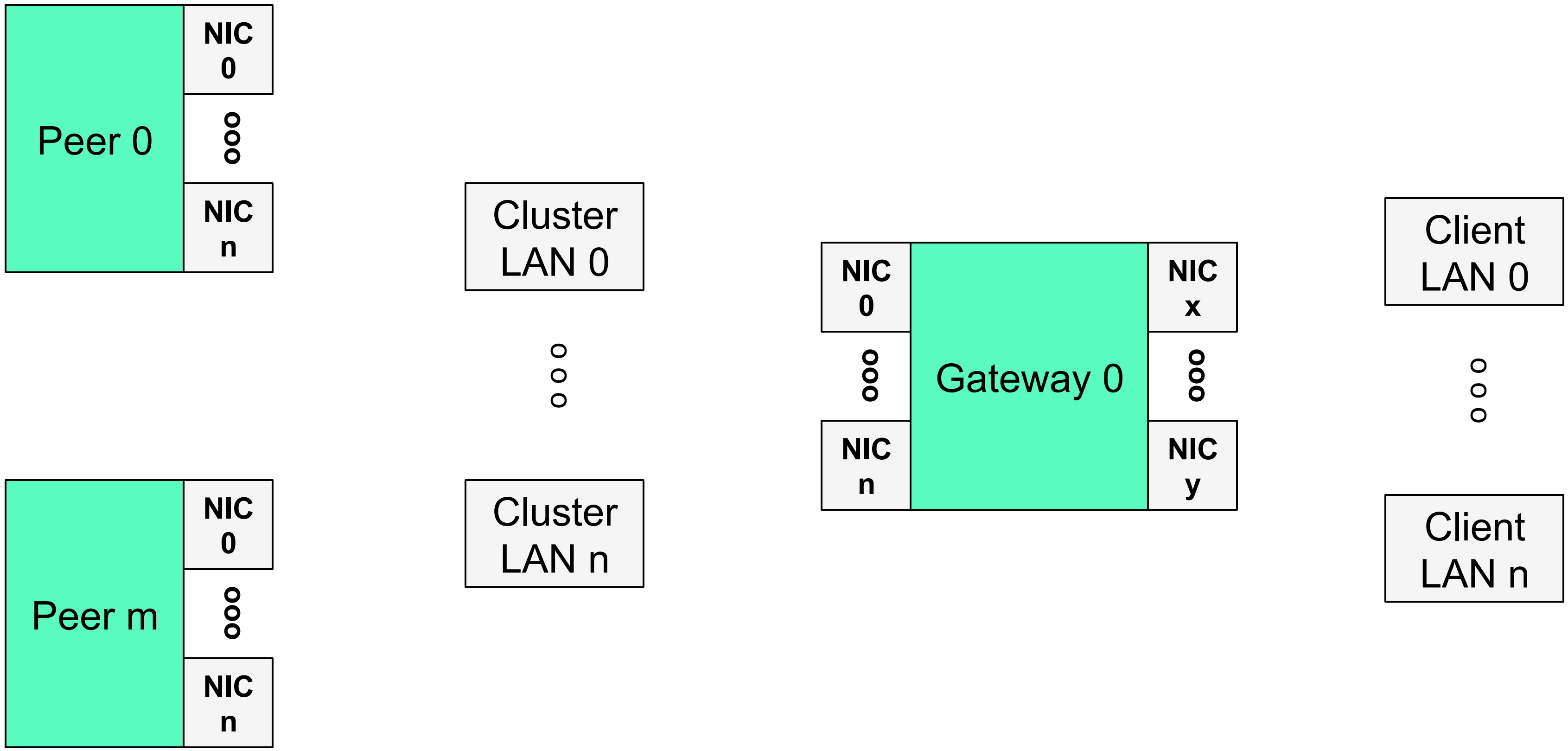
Design Considerations - Data Storage

- All stripe files have a Host Map File header specifying:
  - The resiliency/encoding configuration including the count of failed peers (e.g. 20:5:2)
  - The peers storing a stripe of this file and the size of each stripe
  - Whether the file is compressed
  - Whether the file is encrypted
- Opening an existing file is the process of gathering its HMF data from all peers and achieving a quorum
- Some stripe files may be missing (peer failures)

Design Considerations - Multicast

- Multicast over UDP/IP is the most efficient way to transfer data from the gateway to the peers
  - Send a packet and all the peers receive it (ideally)
  - Very high throughput via multiple LANs (almost wire speed)
  - The cluster LANs have very little packet loss (isolated)
- Multicast is not perfect
  - A protocol is needed to smooth out UDP/IP artifacts
  - Use burst windows to manage packet loss
  - Use inference to avoid time-outs
  - Limit multicast to the cluster LAN

Design Considerations - Topology



Resiliency – Fault Tolerance

- The protection level is configurable
  - Minimum two active peers per cluster
  - Multiple peer failures allowable
- No fail-over of peers
  - No idle peers
- On-the-fly repairs
  - No disruptions – only cluster is aware of faults

Resiliency – Fault Tolerance Use Case #1

- Consider a 20:15:0 cluster *(max 20 peers, min 15 peers, 0 failures)*
  - Files consist of 20-block codewords (15 data blocks and 5 checksum blocks) striped across the 20 peer cluster
  - Upon opening a file each peer knows its assigned stripe and computes its expected contributions for I/O requests without further chatter
- Upon the failure of 5 peers:
  - The remaining 15 peers and the gateway vote (20:15:5)
  - Opened file sessions are updated with the erasure signature
  - In subsequent I/O requests, all nodes will compute their expected contributions, including as needed for on-the-fly repairs, without further chatter

Resiliency – Fault Tolerance Use Case #2

- Consider a 20:15:3 cluster *(max 20 peers, min 15 peers, 3 failures)*
  - Opening an existing file consists of aggregating the HMF data from each active peer to achieve a quorum
    - Some peers may not have the file
    - Some peers may have a stale version of the file
  - Peers that (re) join the cluster will join opened file sessions in which the file was not modified. This immediately increases both the resiliency and performance of the session
  - Files that were modified require healing to restore their resiliency

Resiliency - Healing

- High performance cluster healing
  - Namespace healing
  - Walk a directory or mount point (e.g. /pf0)
    - Stat and repair directories, symbolic links, files
    - Only repair stale files – *not every block in every disk*
    - File healing heals all peers simultaneously via multicast
    - Healing performance is not impacted by the count of peers being healed (healing 100 peers is as fast as healing 1 peer)
    - File healing throughput is hundreds of MB/s (*limited by the available network bandwidth*)

The Multicast Burst Protocol (MBP)

- A highly efficient many-to-many protocol based upon UDP/IP multicast
- Used for moving data between peers and gateways
- Each MBP message has a header that fully describes the context (session id, command, offset, count, etc.)
- Peers that missed a MBP command derive the context fully from multicast cluster replies thus avoid time-outs
- Upon receiving a command peers compute their tasks and the tasks of all other peers thus eliminating the need for any communication beyond the fulfillment of the command

Benchmark Data

- Benchmarks do not always reflect real world usage but they're fun
- Tests measure data to/from the cluster as gateways do no caching
- Hardware used: Dell R720, Intel Xeon E5-2603 1.80GHz, 4GB, 3x 10k 400GB, 2xGigE)
- Performance improves as peers are added for a given FEC setting (percentage of checksum vs. input symbols)
  - A cluster configured at 20% FEC will be almost twice as fast as a cluster configured at 75% FEC
- Both a 20:16:0 and a 5:4:0 cluster can lose up to 4 peers, but:
  - The 20:16:0 is at 20% FEC – very fast and economical
  - The 5:4:0 is at 80% FEC – very inefficient

Sample Benchmark Data

- Linux/FUSE (3:2:0 cluster)

```
dd bs=4k count=16384 if=/dev/zero of=/pf0/dd
16384+0 records in
16384+0 records out
67108864 bytes (67 MB) copied, 0.310384 s, 216 MB/s

dd bs=4k count=16384 if=/pf0/dd of=/dev/zero
16384+0 records in
16384+0 records out
67108864 bytes (67 MB) copied, 0.30576 s, 219 MB/s

dd bs=1M count=1000 if=/dev/zero of=/pf0/dd
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 4.78512 s, 214 MB/s

dd bs=1M count=1000 if=/pf0/dd of=/dev/zero
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB) copied, 4.57891 s, 223 MB/s
```

Available Platforms

- The PFFS was originally developed under FreeBSD as a kernel-mode VFS
  - The PFFS in user-mode (90% single source with kernel-mode) is used for initial debugging and especially for access to valgrind
  - Some performance metrics come from this build
- The PFFS was recently ported to Linux/FUSE
  - With some very minor tweaking of FUSE we have found the performance of PFFS to be comparable to FreeBSD kernel mode

Testing the PFFS

- Testing the PFFS comprises two parts:
  - Functional test suites both custom and standard (e.g. iometer, bonnie++, etc.) are used to ensure nominal functionality and performance
  - Fault injection is necessary to demonstrate that the system operates correctly during peer failures:
    - The software can be built for fault injection in which case every method (CLI and MBP) asserts whether a fault should be injected at run-time
    - A test program generator produces every iteration of peer failures for a specified cluster configuration. The template specifies for each test which faults should be inserted and when (e.g. inject a MBP fault on stripes 11, 34, 35 on the 200th read)

Roadmap

- Performance:
  - Real-time, block-level, intra-peer tiering
- Scalability:
  - Clusters of more than 64K peers
  - More than 256 simultaneous peer failures
- Resiliency:
  - Seamless Gateway fail-over
- Disaster Recovery:
  - Geographically split clusters
  - Inter-cluster replication

Thank You!

Richard Levy
Peer Fusion, Inc.
Founder/CEO
richard@peerfusion.com