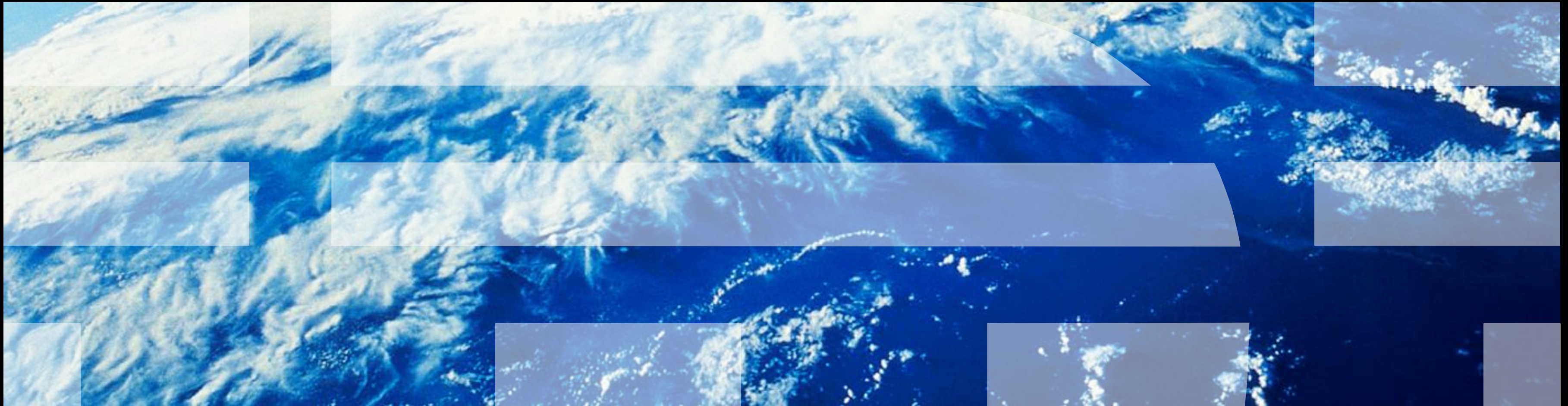
Dean Hildebrand

dhildeb@us.ibm.com

IBM

Sparse Support for NFSv4.2

Dean Hildebrand, Marc Eshel– IBM Almaden



© 2010 IBM Corporation

Sparse File Support – Problem Statement

- Sparse file are common way to represent huge files
  - Database files
  - HPC applications
  - Virtual machine images
- Problem
  - Application not aware of file organization
    - Many cases where NFS blindly read/copies files
  - Read and prefetch holes
    - 'cp', 'rsync' can only omit zeroes after transferred to client
    - Once hole is a allocated, the zeroes remain forever
    - 'Thin Provisioning' advantages lost
  - No write problem…

"Why haven't we done this already?"

Source: If applicable, describe source origin

© 2009 IBM Corporation

IBM

Sparse File Support – Advantages and Goals

- Advantages
  1. Maintain file sparseness and support Thin Provisioning
  2. Reduce data on network
  3. Improve read performance

- Goals
  - Read performance only improves, never degrades
    - Ensure no degenerative cases
      - E.g., File hole and data alternate every 4K
      - The word ‘hurts performance’ should never be uttered
  - Support any number of file holes
    - Millions, billions, …
  - Performance benefits with data sharing
  - Ensure solution works for pNFS
  - Handle ‘holes’ AND all zero regions
  - Simple addition to NFS protocol
  - Never break close-to-open
  - Compatibility with Server-Side-Copy
  - Required feature

Source: If applicable, describe source origin

© 2009 IBM Corporation

IBM

Sparse File Support – Possible Approaches

- READ modifications
  - Prohibited by NFSv4.1 minor versioning rules
- Compression
  - Require all implementations to agree on a single compression algorithm
  - Computational overhead
- Sparse Map / Deduplication
  - Can be very large (> few MBs)
    - Degenerative cases can 'hurt performance' (my ears!)
    - E.g., Disk images can be over 100 GBs and have a map well over several MBs
  - Data sharing reduces effectiveness
    - Map can change frequently
    - Invalidated on write to the file from other clients
  - Difficult to handle 'all zero' regions
    - How does the server file system track zero regions?
  - Hard to support pNFS
    - How does Map on MDS reflect zero regions on data server
    - MDS knows allocation maps, not data contents

Source: If applicable, describe source origin

© 2009 IBM Corporation

OPERATION: READPLUS – Read from file with extensible results

ARGUMENTS

```
struct READPLUS4args {
        /* CURRENT_FH: file */
        stateid4        stateid;
        offset4         offset;
        count4          count;
};
```

No Changes

RESULTS

```
  union nfs_readplusreshole switch  (holeres4 resop) {
      CASE HOLE_NOINFO:
          void;
      CASE HOLE_INFO:
          offset4         hole_offset;
          length4         hole_length;
   };
   union nfs_readplusresok4 switch  (readplusrestype4 resop) {
      CASE READ_OK:
           opaque          data<>;
      CASE READ_HOLE:
           nfs_readplusreshole   reshole4;
   };
   union READPLUS4res switch (nfsstat4 status) {
    case NFS4_OK:
            bool            eof;
            nfs_readresok4  resok4;
    default:
            void;
   };
```

Per-READ improvement

Per-HOLE improvement
(Size of Hole)

New result type
(more possible)

© 2009 IBM Corporation

READPLUS - Details

- Based on NFS4 READ operation (a superset)
  - Extra ‘flag’ to indicate result structure
  - New READ_HOLE type
- If read ‘entirely’ within hole, return READ_HOLE (instead of zeroes)
  - If hole info available, also return offset+length of current hole
    - Client satisfy READ/READPLUS requests into hole without contacting server
    - Hole info valid until ‘change’ attribute changed
  - If hole info not available, simply return HOLE_NOINFO
- Reads that ‘extend’ into holes
  - Server can return ‘short’ read
  - Client send another request for remaining data

Source: If applicable, describe source origin

© 2009 IBM Corporation

READPLUS - pNFS

- Data servers MAY return a READ_HOLE
- Use hole information together with layout
  1. If DS cannot determine hole info, data server SHOULD return HOLE_NOINFO.
  2. If DS can only obtain hole information that data server
     - DS SHOULD return HOLE_INFO and byte range of hole stored on that DS
  3. If DS can obtain hole information for the entire file (w/o severe performance impact)
     - DS MAY return HOLE_INFO and the byte range of the entire file hole
- Basically, DS should try its best…

Source: If applicable, describe source origin

© 2009 IBM Corporation