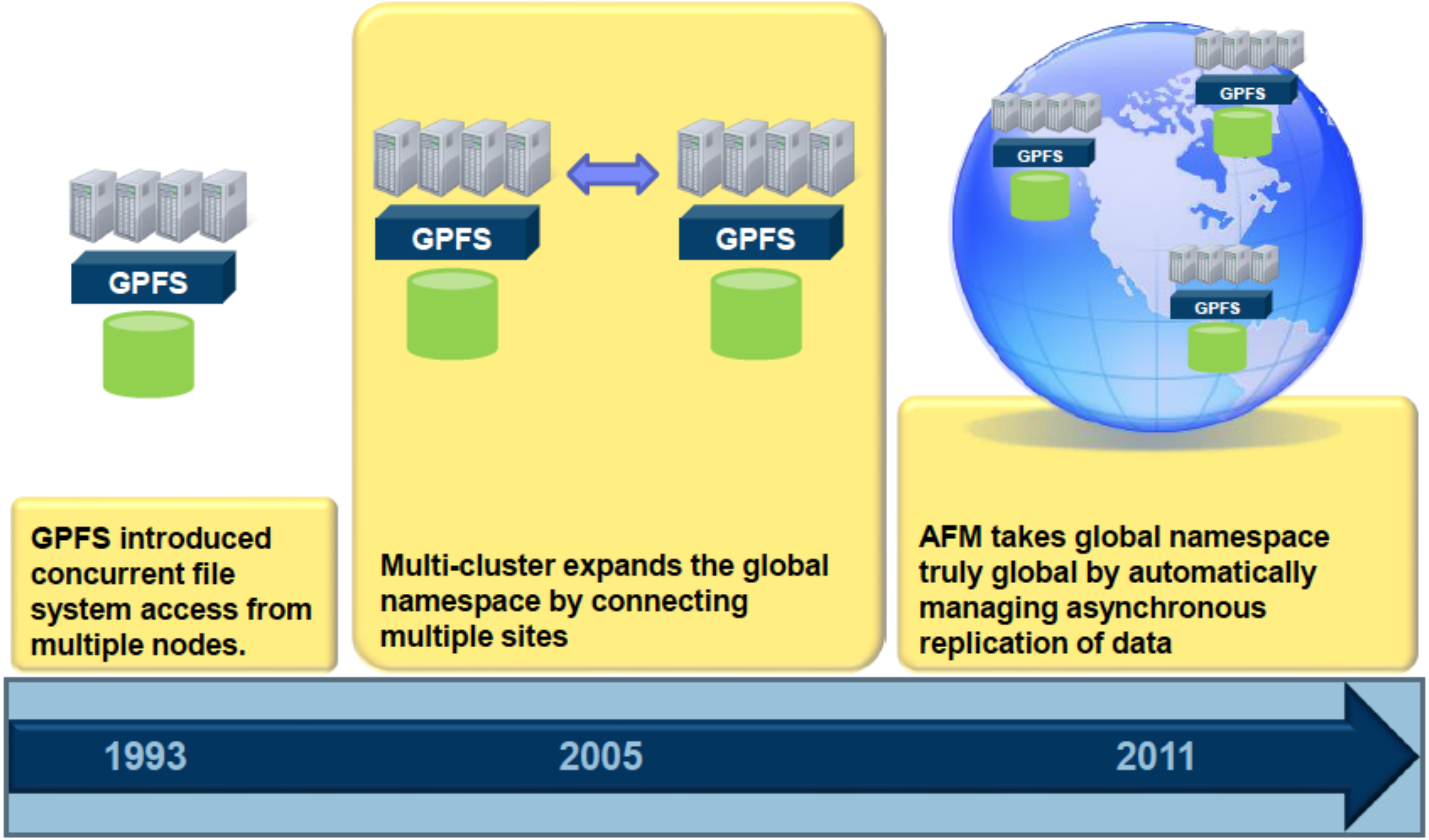
Corso sul file system parallelo distribuito GPFS

Part 4. New features: AFM and Native RAID

New features in GPFS 3.5

- Active File Management
- High Performance Extended Attributes
- Independent Filesets
- IPv6 support
- GPFS Native RAID

Evolution of the global namespace: GPFS Active File Management (afm)



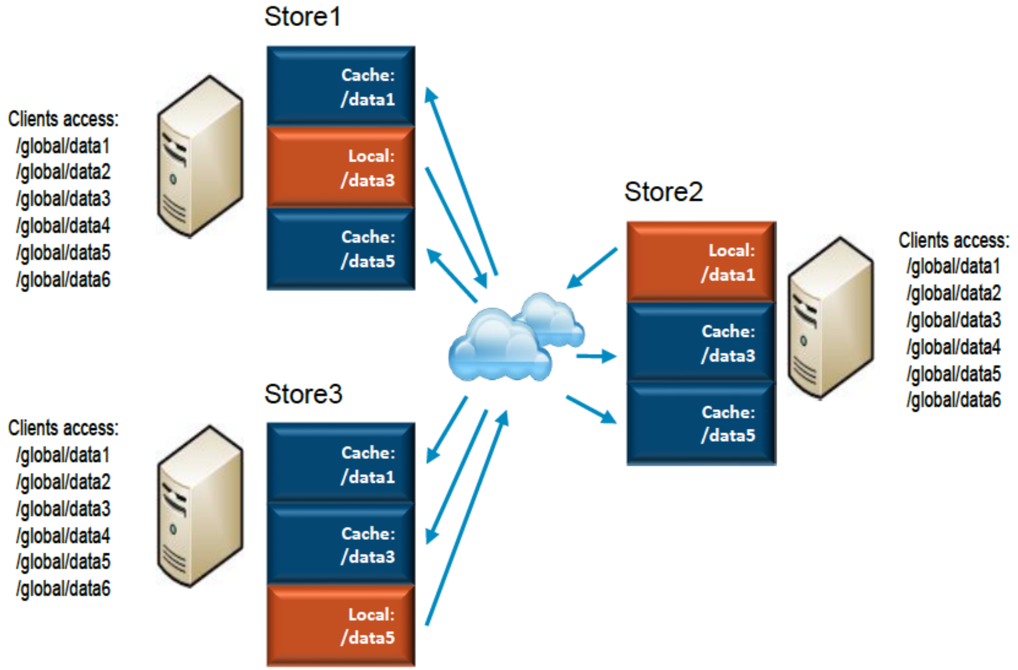
GPFS introduced concurrent file system access from multiple nodes.

Multi-cluster expands the global namespace by connecting multiple sites

AFM takes global namespace truly global by automatically managing asynchronous replication of data

Active File Management

- Enables sharing data across unreliable or high latency networks
- location and flow of file data between GPFS clusters can be automated.
- Relationships between GPFS clusters using AFM are defined at the fileset level.
  - A fileset in a file system can be created as a “cache” that provides a view to a file system in another GPFS cluster called the “home.” File data is moved into a cache f



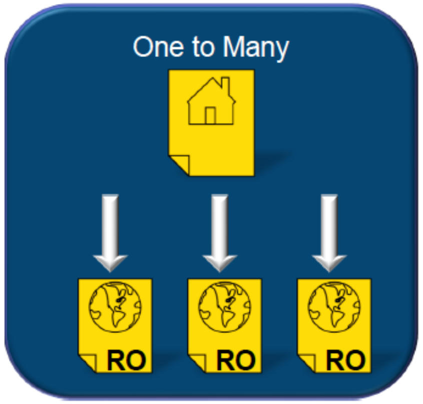
can be used to create a global namespace within a data center, across a campus or between data centers located around the world. AFM is designed to enable efficient data transfers over wide area network (WAN) connections. Transfer home -> cache can happen in parallel within a node called a *gateway or across multiple gateway nodes.*

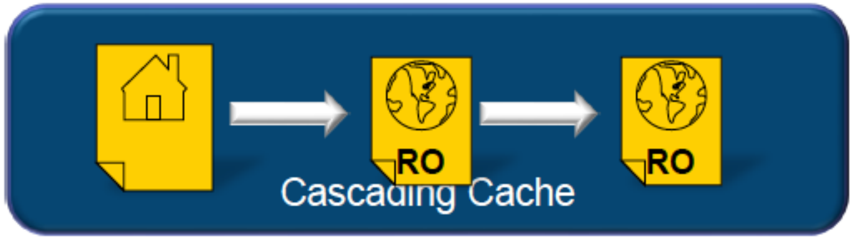
Active File Management Caching Basics

- Cache basics
  - Data update are asynchronous
  - Writes can continue when the WAN is unavailable
  - Communication between sites uses TCP/IP
- Two sides to a cache relationship
  - Home
    - Where the information lives
  - Cache
    - Data written to the cache is copied back to home as quickly as possible
    - Data is copied to the cache when requested
- Multiple cache relationships per file system
  - Cache relationships are at a fileset level
  - A file system can contain multiple homes, caches and non-cached data
- Multiple caching modes to meet your needs
  - Read-Only
  - Single Writer
  - Cache-Wins
  - High Availability

AFM Mode: Read-Only caching

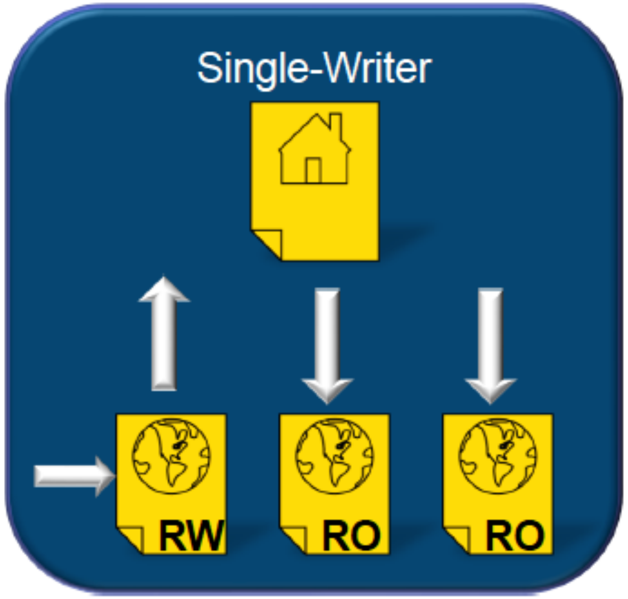
- Read caching mode
  - Data exists on the home fileset and one or more cache sites
- Data is moved to the cache on-demand.
  - File Metadata caching: Listing the contents of a directory moves the file metadata information into the cache
  - Data – Opening a file copies the data in the cache
  - Getting data to the cache
    - On-demand when opened
    - Pre-fetch using a GPFS policy
    - Pre-fetch using a list of files
- Caching behavior
  - Many to one
  - Optional LRU cleaning of cache
  - Cascading caches





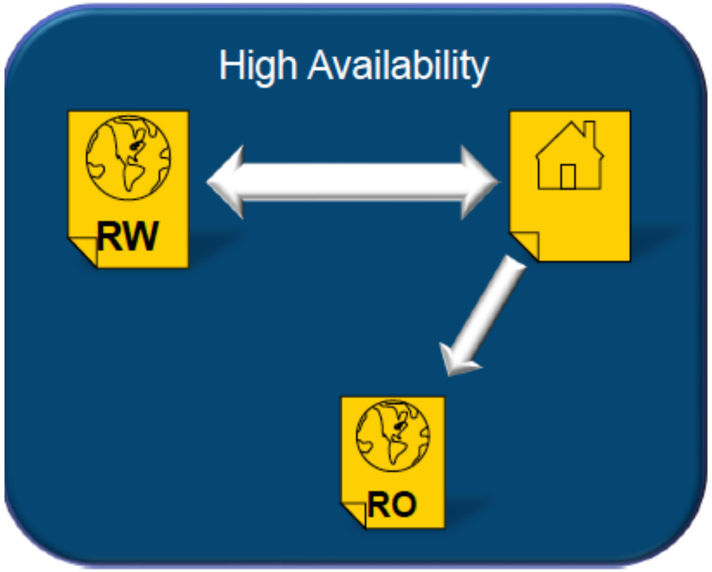
AFM Mode: Single-Writer

- Data written to a cache
- Asynchronous replication back to home
- Can have multiple read-only caches



AFM Mode: Cache-Wins

- Multiple cache nodes
- All nodes can write data
- Conflict resolution
  - Default: The last writer wins



Communication between AFM clusters

- Communication is done using NFSv3
  - Already tested with NFSv4
  - Architecture is designed to support future protocols
- GPFS has it's own NFSv3 client
  - Automatic recovery in case of a communication failure
  - Parallel data transfers (even for a single file)
  - Transfers extended attributes and ACL's
- Additional Benefits
  - Standard protocol can leverage standard WAN accelerators
  - Any NFSv3 server can be a "Home"

AFM Configuration example

- Setting up the Home cluster
  - NFS v3 server
    - recommended to use the GPFS cNFS
    - Should have "Cluster IP"
  - Define gateway nodes
    - Cache data is transferred between the GPFS clusters through gateway nodes

```
mmchnode --gateway –N node1
```

  - Setting up a cache relationship
    - best practice to define the NFS mount points at fileset junction points
    - On the home:

```
mmcrfileset master1 master_t1
mmlinkfileset master1 master_t1 –J /gpfs/master1/master_t1
#vi /etc/exports /gpfs/master1/master_t1 *(rw,no_root_squash,sync,fsid=92496)
```

AFM Configuration example (2)

- On the cache:
  - create an independent fileset using -p parameter:

```
mmcrfileset cache2 master_t1 -p afmtarget=node1:/gpfs/master1
   /master_t1 -p afmmode=ro --inode-space=new
mmlinkfileset cache2 master_t1 –J /gpfs/cache2/master_t1
```

  - Once the fileset is linked you are ready to start caching data

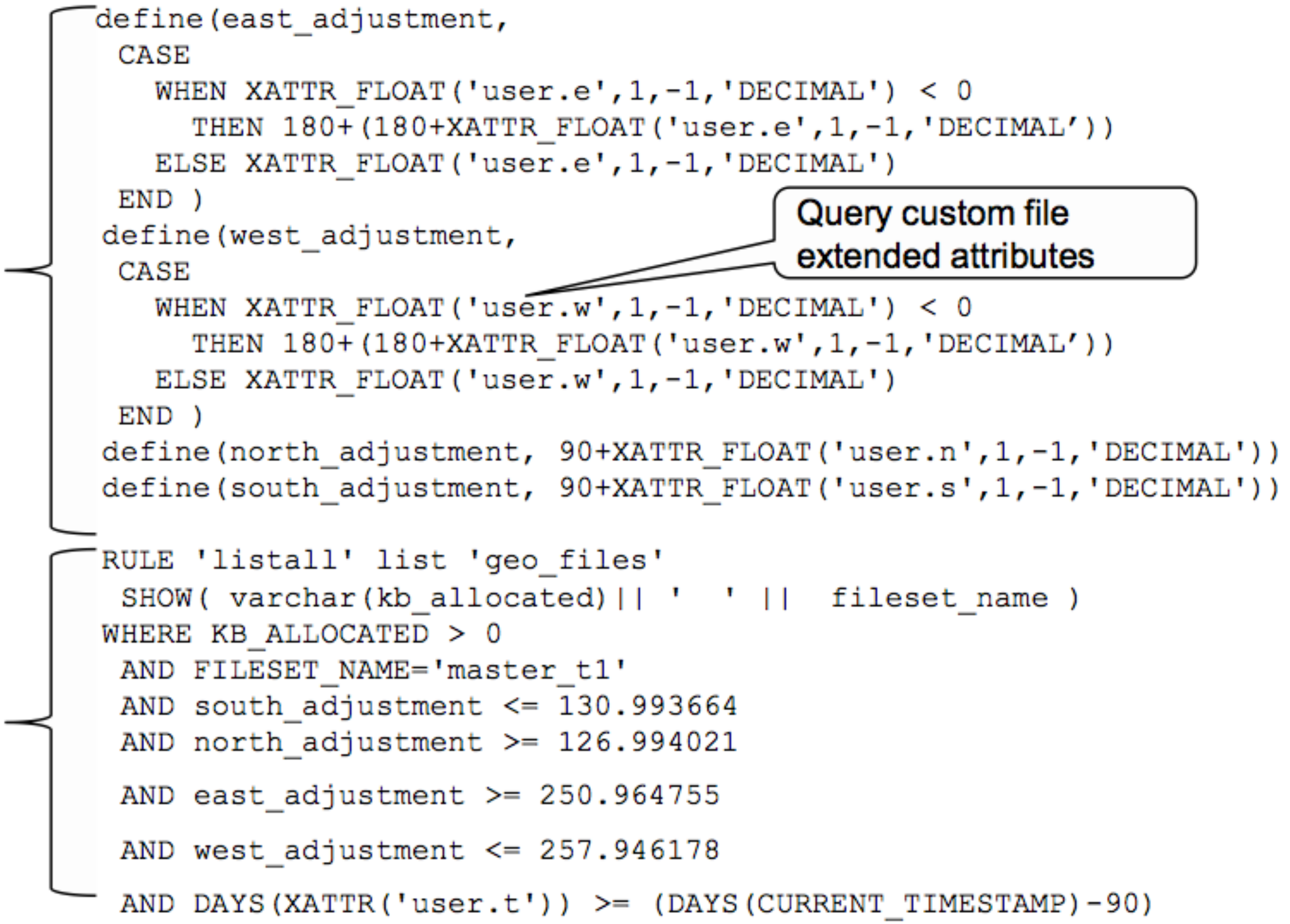
High Performance Extended Attributes

- Extended attributes in GPFS since 3.2
    - not commonly used, in part because of performance concerns.
    - GPFS 3.4: redesign of the extended attributes support infrastructure was implemented, → significant performance improvements.
    - GPFS 3.5: extended attributes are accessible by the GPFS policy engine →**policy rules** can use your custom file attributes.
- Now an application can use standard POSIX interfaces t to manage extended attributes and the GPFS policy engine can utilize these attributes.

Policy example extended attributes

Macros manipulate data

Policy calls macros



```
define(east_adjustment,
 CASE
   WHEN XATTR_FLOAT('user.e',1,-1,'DECIMAL') < 0
     THEN 180+(180+XATTR_FLOAT('user.e',1,-1,'DECIMAL’))
   ELSE XATTR_FLOAT('user.e',1,-1,'DECIMAL')
 END )
define(west_adjustment,
 CASE
   WHEN XATTR_FLOAT('user.w',1,-1,'DECIMAL') < 0
     THEN 180+(180+XATTR_FLOAT('user.w',1,-1,'DECIMAL’))
   ELSE XATTR_FLOAT('user.w',1,-1,'DECIMAL')
 END )
define(north_adjustment, 90+XATTR_FLOAT('user.n',1,-1,'DECIMAL'))
define(south_adjustment, 90+XATTR_FLOAT('user.s',1,-1,'DECIMAL'))
```

```
RULE 'listall' list 'geo_files'
 SHOW( varchar(kb_allocated)|| '  ' ||  fileset_name )
WHERE KB_ALLOCATED > 0
 AND FILESET_NAME='master_t1'
 AND south_adjustment <= 130.993664
 AND north_adjustment >= 126.994021
 AND east_adjustment >= 250.964755
 AND west_adjustment <= 257.946178
 AND DAYS(XATTR('user.t')) >= (DAYS(CURRENT_TIMESTAMP)-90)
```

Query custom file extended attributes

Independent Filesets

- To effectively manage a file system with billions of files requires advanced file management technologies.
- independent fileset has its own inode space.
  - independent fileset can be managed similar to a separate file system but still allow you to realize the benefits of storage consolidation.
- An example of an efficiency introduced with independent filesets is improved policy execution performance.
- GPFS only needs to scan the inode space represented by that fileset, so if you have 1 billion files in your file system and a fileset has an inode space of 1 million files, the scan only has to look at 1 million inodes. This instantly makes the policy scan much more efficient.
- Independent filesets enable other new fileset features in GPFS 3.5.
- **Fileset Level Snapshots**
  - Snapshot granularity is now at the fileset level in addition to file system level snapshots.
- **Fileset Level Quotas**
  - User and group quotas can be set per fileset

Other features

- **File Cloning**
  - File clones are space efficient copies of a file where two instances of a file share data they have in common and only changed blocks require additional storage. File cloning is an efficient way to create a copy of a file, without the overhead of copying all of the data blocks.
- **IPv6 Support**
  - IPv6 support in GPFS means that nodes can be defined using multiple addresses, both IPv4 and IPv6.
- **Independent metadata block size**
  - Up to 1/32 of data block size

GPFS Native RAID

- GPFS brings storage RAID management into the GPFS NSD server.
- With GNR GPFS directly manages JBOD based storage.
  - This feature provides greater availability, flexibility and performance for a variety of application workloads.
- GNR implements a Reed-Solomon based de-clustered RAID technology
  - provide high availability and keep drive failures from impacting performance by spreading the recovery tasks over all of the disks.
  - Unlike standard Network Shared Disk (NSD) data access GNR is tightly integrated with the storage hardware.
  - For GPFS 3.5 GNR is available on the IBM Power 775 Supercomputer platform.