GANESHA NFS server

Administration guide
v0.3

Mon Sept 24 2007

Contents

Chapter 1

How to write the GANESHA's configuration file

1.1 Overall view of the configuration file

The configuration file for the GANESHA daemon may seemed huge at first glance. It is separated into several blocks. A block contain several items.

A block will look like this:

```
<BLOCK_TAG>
{
   <Config_Item_Tag> = <Value> ;

   # Comment
   <Config_Item_Tag> = <Value> ;
.
.
.
   <Config_Item_Tag> = <Value> ;
}
```

As we will see, there are many different blocks, each of them dedicated to the configuration of a specifc stuff in GANESHA. Not all the blocks are mandatory and some may not be explicitely specified. In this case, default values will be kept¿ Inside the blocks are items which are a way to set a internal constant in the daemon with direct effect to the program's behaviour. Like the blocks, not all the items are mandatory and some may be lacking. The following sections will described each blocks and items and explain all of them and their use.

Block's names and item's name matching algorithm is case insensitive. Naming a block EXPORT or Export or export work as well.

Every character below a '#' sign is a comment and will be ignored (except if it is inside a quoted string, or escaped with a backslash).

Block Name	Status	Used for...
EXPORT	Mandatory. Can be duplicated	Set an export entry
FSAL	Recommended. One instance.	FSAL generic configuration
FileSystem	Optionnal. One Instance	Filesystem's behaviour
HPSS	Recommended. One instance	FSAL_HPSS's specific configuration (only with hpss.ganesha.nfsd)
POSIX	Recommended. One instance	FSAL_POSIX's specific configuration (only with posix.ganesha.nfsd)
SNMP	Recommended. One instance	FSAL_SNMP's specific configuration (only with snmp.ganesha.nfsd)
NFS_Worker_Param	Recommended. One instance	Worker Thread configuration
NFS_Core_Param	Recommended. One instance	Daemon's core configuration
NFS_DupReq_Hash	Optionnal. One Instance	RPC Duplicate Request Hash configuration
NFS_IP_Name	Optionnal. One Instance	IPadress ↔ hostname resolution cache configuration
NFSv4_ClientId_Cache	Optionnal. One Instance	Configuration of the NFSv4 Clientid cache
NFSv4_StateId_Cache	Optionnal. One Instance	Configuration of the NFSv4 State id cache
NFS_KRB5	Recommended for using KRB5. One instance	KRB5 configuration. Needed only if sys=krb5, krb5i or krb5p is used
NFSv4	Recommended. One instance	NFSv4 Specific configuration items (e.g. lease lifetime)
CacheInode_Hash	Recommended. One instance	Configuration of the Cache Inode Hash Table
CacheInode_Client	Recommended. One instance	The configuration of the workers's clients to the Cache Inode
CacheInode_GC_Policy	Recommended. One instance	Garbagge Collection Policy for the Cache Inode entries
FileContent_Client	Recommended. One instance	The configuration of the workers's client to the File Content Cache
FileContent_GC_Policy	Recommended. One instance	Garbagge Collection Policy for the File Content cache entries
BUDDY_MALLOC	Recommended. One instance	Configuration of the internal memory manager

1.2 What to put in each block

1.2.1 The EXPORT Block

This tag is used to describe the way a FSAL tree is exported via NFS. It describes clients, who has root access, which versions of the protocols to be used and transport layers.

How to describe clients ?
Clients can be single machines, pack of machines, netgroups, and networks.

Single machine is identified by its name or ip address: e.g. localost or 127.0.0.1

Pack of machine: you may have "farms" of machines with similar name like "cluster0, cluster0, ..., cluster-gateway". With a cluster, you can have bunches of such names. Using "unix jokers" can help so far. GANESHA allows you to specify names like "cluster*" or "cluster1?", using the old style "shell joker syntax"

Networks: this should be a network name (you may have a /etc/networks describing them) or a reduced ip address describing addresses belonging to this networks. For example 12.13.0.0 will describe machines whose address is 12.13.x.y with netmask 0xffff0000

Netgroups: clients can be identified by the fact they belong to a given netgroup. This is shown by adding an arobas at the beginning of the netgroupname. To tell GANESHA to provide access to machine in netgroup "nfsclients" specify "@nfsclients"

Lists of clients is comma separated: "localhost,12.13.0.0,@nfsclient"

NB: Only 'Path' , 'Export_Id' and 'Pseudo' are mandatory keys.

- Export_Id : This tag is used to set the the id for this export. This is mostly used internally, but this is a mandatory value. The value is to be a non-zero integer value
  e.g: Export_Id = 1 ;
- Access: The list if clients that can access the export entry
  e.g: Access = "Machine3,Machine10*,NetworkB,@netgroupY";
  NB: Using Access = "*" ; will allow everybody to access the mount point.
- Root_Access: The list of clients (see above) that have root access rights (root remains root when crossing the mount point).
  e.g: Root_Access = "localhost,12.13.0.0,@nfsclient" ;
- Access_Type: Describes the kind of access you can have of the mount point. Acceptable values are:
  - RO: Read-Only mount point
  - RW: Read-Write mount points
  - MDONLY: The mount point is RW for metadata, but data accesses are forbidden (both read and write). This means you can do everything on md, but nothing on data.
  - MDONLY_RO: Like RO, but reading data is forbidden. You can read only metadata.

  e.g: Access_Type = "RW" ;
- Anonymous_root_uid : The uid to be used for root when no root access was specified for this clients. This is the definition of the 'nobody' user. "Traditional" value is -2
  e.g: Anonymous_root_uid = -2 ;
- NOSUID: a flag (with value TRUE or FALSE) showing if setuid bit is kept.
  e.g : NOSUID= TRUE;
- NOSGID: a flag (with value TRUE or FALSE) showing if setgid bit is kept.
  e.g : NOSGID= TRUE;
- NFS_Protocols: The version(s) of the NFS protocol you can use for mounting this export entry
  e.g: NFS_Protocols = "2,3,4" ;

- Transport_Protocols: The transport layer to use for mount this entry This should be UDP or TCP or a list.
  e.g : Transport_Protocols = "UDP,TCP" ;

- Sec_type: List of supported RPC_SEC_GSS authentication flavors for this export. It can be a coma-separated list of the following values: sys, krb5i, krb5p.
  e.g : SecType = "sys,krb5";

- MaxOffsetRead: Maximum offset allowed for a read operation (no limit if not specified). This could be useful for prevently from "evil" use of NFS read (like access a TB long file via NFS)
  e.g : MaxRead = 409600;

- MaxOffsetWrite: Like MaxOffsetRead, but for Write operation
  e.g : MaxWrite = 409600;

- MaxRead, MaxWrite, PrefRead, PrefWrite, PrefReaddir: The value to be returned to client when NFS3_FSINFO is called. Better solution is not to use this keys and so keep the default values, optimized for NFSv3.

- Filesystem_id: The filesystem id to provide to the client for this export entry. NFS Client will use this value to address their internal metadata cache. In NFSv4, both major and minor values are used, in NFSv2 and NFSv3, only the major is used.
  e.g. : Filesystem_id = 100.1 ;

- PrivilegedPort: A flag (TRUE or FALSE) to specify is a client using an ephemere port should be rejecting or not.
  e.g. PrivilegedPort = FALSE ;

- Cache_Data: A flag (TRUE or FALSE) To specify if files content should be cached by the GANESHA server or not
  e.g : Cache_Data = TRUE ;

- MaxCacheSize: if export entry is datacached, this value defines the maximum size of the files stored in this cache.

- FS_Specific: a comman separated list of information used by the FSAL to perform initialization. See FSAL documentation for detail.
  e.g. (for HPSS/FSAL): FS_Specific = "cos=1" ;

- Path: The path to export via NFS. Should have a leading slash.
  e.g: Path = "/nfs/my_exported_directory" ;

- Tag: A way of providing a shorter path for mounting an entry. For example, you could mount an entry like this:

  ```
  mount -o vers=3 nfsserver:/nfs/my_exported_directory /mnt
  ```

  But if you specified "Tag = ganesha;", you can simply do

  ```
  mount -o vers=3 nfsserver:ganesha /mnt
  ```

- Pseudo: a NFSv4 specific key that shows the path, in NFSv4 pseudo file system were the 'actual' mount point resides.
  e.g : Pseudo = "/nfsv4/pseudofs/nfs_mount_entry_#1" ;

1.2.2 The FSAL Block

This block provides with general information about the FSAL. A given FSAL could use other information, more specific, but not from this block which remains generic.

- DebugLevel: the level of verbosity for FSAL logs. Acceptable values are:
  - NIV_NULL (no logging)
  - NIV_MAJ (log only major events)
  - NIV_CRIT (log only erroneous events)
  - NIV_EVENT (log only important events)
  - NIV_DEBUG (log all events)
  - NIV_FULL_DEBUG (log all internal processing)

  e.g: DebugLevel = "NIV_EVENT" ;
- LogFile: The path to use for the logfile.
  e.g: LogFile = "/var/log/ganesha.fsal.log" ;
- max_FS_calls: FSAL provides a way (with a semaphore) to limit the number of requests performed by the worker threads. This is useful to prevent from 'FS flood via NFS'. If none is specify, no limits exists and the semaphore will remain ususued.
  e.g : max_FS_calls = 20 ;

1.2.3 The FileSystem Block

This block describes the beahavior of the file system on some points.

- MaxRead: Maximum read buffer size for this filesystem.
  e.g. : MaxRead = 1048576 ;
- MaxWrite: Maximum write buffer size for this filesystem
  e.g. : MaxWrite = 1048576 ;
- Umask: If set, this mask is applied on the mode of created objects. This should be an octal value, do not forget the leading 0.
  e.g.: Umask = 0002 ;
- CanSetTime, Symlink_support, Link_support: Three flags (TRUE or FALSE) to specify if hard links, symbolic links are allowed or if time is settable.
- auth_xdev_export: A flag (TRUE of FALSE) to specify if crossing junction is allowed or not.

1.2.4 The HPSS Block (Only for use with HPSS/FSAL)

This is the items needed for configuring the HPSS CLAPI that the HPSS/FSAL uses for its calls.

- AuthMech: CLAPI Authentication mechanism. Should be "krb5" or "unix"
  e.g. : AuthMech = "krb5" ;
- PrincipalName: The Auth principal to be used by the HPSS/FSAL to run CLAPI calls. Principal hpssfs is strongly suggested.
  e.g. : PrincipalName = "hpssfs";
- KeytabPath: The Keytab file to be use for acquiring principal identify
  e.g. : KeytabPath = "/var/hpss/etc/hpss.keytabs" ;

- CredentialLifetime : The duration in seconds after which a CLAPI credential is to be renewed.
  e.g. : CredentialLifetime = 3600 ;
- NumRetries: Number of retries in CLAPI for connection failure with HPSS. A value of 0 means no retry, a value of -1 means "retry forever". This last value may cause troubles because worker threads may hang. A value of 100 is a good compromise.
  e.g : NumRetries = 100 ;
- BusyDelay: retry delay (in seconds) if core server is busy
  e.g.: BusyDelay = 1 ;
- BusyRetries: Number of retries when core is busy (0: no retry, -1: retry forever)
  e.g. : BusyRetries = -1;
- MaxConnections : Maximum number of connection to the Core Server (check HPSS configuration for a coherent value)
  e.g. : MaxConnections = 100;

1.2.5 The POSIX Block (only for use with a posix filesystem)

This block is related to POSIX/FSAL, it contains the necessary items to properly init the FSAL built on top of POSIX calls.

- DB_Host : The hostname for the machine running the DB engine
  e.g. : DB_Host = "dbserver.localdomain" ;
- DB_Port : The port to use for contacting the DB engine
  e.g. : DB_Port = 5432;
- DB_Name : The name for the database to use for the POSIX/FSAL
  e.g. : DB_Name = FSAL_POSIX_DB ;
- DB_Login : The username to use to connect to the database
  e.g.: DB_Login = DB_USER ;
- DB_keytab : The keytab to use to acquire the 'DB_Login' identity.
  e.g : DB_keytab = "/var/etc/posix.db.keytab" ;

1.2.6 The SNMP Block (Only for use with SNMP/FSAL)

This block is related to SNMP/FSAL, it contains necessary items to properly init the FSAL built on top of a SNMP tree.

- snmp_version: this indicates the SNMP protocol version that GANESHA will use for communicating with SNMP agent. Expected values are `1`, `2c` or `3`. Default is `"2c"`.
- snmp_server: this is the address of the SNMP master agent. A port number can also be specified by adding ":¡port¿" after the address. Default is `"localhost:161"`.
- community: this is the SNMP community used for authentication. Default is `"public"`.
- nb_retries: number of retries for SNMP requests. Default value is `SNMP_DEFAULT_RETRIES`, defined in net-snmp library.
- microsec_timeout: number of microseconds until first timeout, then an exponential backoff algorithm is used for next timeouts. Default value is `SNMP_DEFAULT_TIMEOUT`, defined in net-snmp library.

- client_name: this is the client name that could be used internally by net-snmp for its traces. Default value is `"GANESHA"`.
- snmp_getbulk_count: this indicates the number of responses wanted for each SNMP GET-BULK request. Default value is `64`.

1.2.7 The BUDDY_MALLOC Block

GANESHA manages its own memory on its own way. This is pretty useful for memory use optimization. The algorithm is the Budd Block algorithm. Each thread in GANESHA manages the memory it uses via this method.

- Page_Size: The size of a page. This MUST be a power of 2.
  e.g. : Page_Size = 8388608;
- Enable_OnDemand_Alloc: a flag (TRUE or FALSE) to specify if a thread can extend its number of page when it lacks memory. Value of TRUE is strongly recommended.
  e.g. : Enable_OnDemand_Alloc = TRUE ;
- Enable_Extra_Alloc : a flag (TRUE or FALSE) to specify if buddy memory manager allow threads to alloc memory areas that are larger than Page_sSize value.
  e.g. : Enable_Extra_Alloc = TRUE;
- Enable_GC : a flag (TRUE or FALSE) to specify if buddy memory manager can release unused pages, according to GC_Keep_Factor and GC_Keep_Min parameters. TRUE is strongly recommended.
  e.g. : Enable_GC = TRUE;
- GC_Keep_Factor : Buddy's GC must keep at least GC_Keep_Factor times the current number of used pages.
  e.g. : GC_Keep_Factor = 2;
- GC_Keep_Min: GC must keep at least this number of pages.
  e.g. : GC_Keep_Min = 2;
- LogFile: The path of the log file for BUDDY_MALLOC messages (not very verbose, shows only critical behaviours).
  e.g. : LogFile: /var/log/ganesha.buddy_malloc.log ;

1.2.8 The 'CacheInode_Hash' block

This block defines the behavior of hash table used for the internal metadata cache.
It consists of the following key/value peers:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the underlying filesystem's handle, i.e. 256. However, if you notice a bad balancing in your hash tables, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the forecasted number of metadata entries, divided by the number of worker threads.
  e.g.: for 300k entries, and 30 worker threads, we'll have about 10k entries by thread. Thus, a value between 2500 and 10000 should be set (it will result in 1 to 4 memory allocations during the whole life of the thread)

1.2.9 The 'CacheInode_Client' block

This block defines the behavior of the metadata cache. It consists of the following key/value peers:

- LogFile: The file where the metadata cache events are logged.
- DebugLevel: The verbosity level for the metadata cache log. The values can be:
  - NIV_NULL (no logging)
  - NIV_MAJ (log only major events)
  - NIV_CRIT (log only erroneous events)
  - NIV_EVENT (log only important events)
  - NIV_DEBUG (log all events)
  - NIV_FULL_DEBUG (log all internal processing)
- LRU_Nb_Call_Gc_invalid: Each worker maintains a LRU list of the last entries it handled. When a worker handles an entry, it sets it invalid in other threads' LRU's, in order to make them garbaging it. This parameter so defines the periodicity for garbaging invalid entries in this list. Thus, a worker will garbage its own list after processing this amount of NFS requests.
- LRU_Prealloc_PoolSize: This parameter sets the number of LRU entries that are preallocated for each worker thread. Given that the total amount of theses entries equals the number of cache entries + a certain working set (number of working threads * LRU_Nb_Call_Gc_invalid), this parameter must be close to the forecasted number of metadata entries divided by the number of worker threads, + the value of LRU_Nb_Call_Gc_invalid.
- Entry_Prealloc_PoolSize: This parameter is the number of preallocated cache entries for each worker thread. It must be close to the forecasted number of metadata entries divided by the number of worker threads.
- DirData_Prealloc_PoolSize: This parameter is the number of preallocated directory cache entries for each worker thread. It must be close to the number of directories in the exported filesystem divided by the number of worker threads.
- ParentData_Prealloc_PoolSize: This parameter is the number (for each thread) of preallocated entries' references to their parents (for example, a hard linked object can have several parents).
  It must be close to the forecasted number of metadata entries divided by the number of worker threads, except if the filesystem contains a huge amount of hard links (you should then multiply this value by the average number of hard links on each object).
- State_v4_Prealloc_PoolSize: This parameter is the number (for each thread) of preallocated state to be used by each worker for NFSv4 State management.
  It is used only if NFSv4 is used.
- Attr_Expiration_Time: The expiration delay (in seconds) for cached attributes. A value of "0" disables attributes cache expiration.
- Symlink_Expiration_Time: The expiration delay (in seconds) for symbolic link content. A value of "0" disables symlink cache expiration.
- Directory_Expiration_Time: The expiration delay (in seconds) for directory content. A value of "0" disables directory cache expiration.

- Use_Getattr_Directory_Invalidation: This boolean indicates if a cached directory content is invalidated when its mtime has changed on the underlying filesystem. Setting this parameter to TRUE will result in an extra "getattr" operation on the filesystem for each NFS "readdir" call, so it could strongly impact the readdir performances.
  However, it must be set if your filesystem tree is continuously modified by an external actor (another NFS server, ...) and if you need a good re-synchronisation of GANESHA's NFS server cache.

- Use_Test_Access: If set to TRUE (strongly recommended), NFS "access" calls will be performed according to the cached attributes (mode, group, owner,...).
  Else, each NFS "access" operation will result in an "access" call to the underlying filesystem.

- Use_OpenClose_cache: If this boolean is set to TRUE, files will not be opened and closed at each read/write NFS call: GANESHA will cache a certain amount of opened file descriptors for better I/O performances (recommended).

- Max_Fd: When datacaching is disabled and 'Use_OpenClose_cache' is enabled, this parameter indicates the maximum number of files that can be kept openned for each thread.
  NB: when datacaching is enabled, use the FileContent_Client::Max_fd parameter instead

- OpenFile_Retention: When datacaching is disabled and 'Use_OpenClose_cache' is enabled, this parameter indicates the minimum time (in seconds) a file must be kept opened.
  If the 'max_fd' limit is reached and all files have been opened more recently than the 'OpenFile_Retention' time, no more file descriptors are cached until the previous ones are older than this value.

1.2.10 The 'CacheInode_GC_Policy' block

This section defines the garbage collection policy for the metadata cache.

In GANESHA, each worker thread ensures the garbage collection of the cache entries it was the last to deal with.

The following parameters describe the conditions that must be meet for a worker, in order for it to launch a garbage collection of its entries.

- Nb_Call_Before_GC: This parameter indicates, for each worker thread, the number of NFS calls it has to process between checking for garbage collection conditions.

- Runtime_Interval: The period (in seconds) for checking cache high-watermark. Thus, a worker thread does not check GC conditions until the interval since the last garbage collection has not been enlapsed.

- NbEntries_HighWater: Garbage collections are launched only if the number of entries in the cache is over this value.

- NbEntries_LowWater: A garbage collection stops when the number of entries in the metadata cache falls to this value.

- File_Lifetime: the minimum delay (in seconds) a file has not been accessed, for beeing a candidate to metadata cache garbage collection.
  A value of "-1" disables files garbage collection.

- Directory_Lifetime: the minimum delay (in seconds) a directory has not been accessed, for beeing a candidate to metadata cache garbage collection. Note that a directory is not garbaged until all its childs have not been garbaged before.
  A value of "-1" disables directory garbage collection.

1.2.11 The 'FileContent_Client' config block

This block sets the behavior for the datacache. It consists of the following key/value peers:

- LogFile: The file where the datacache events are logged.
- DebugLevel: The verbosity level for the datacache log. The values can be:
  - NIV_NULL (no logging)
  - NIV_MAJ (log only major events)
  - NIV_CRIT (log only erroneous events)
  - NIV_EVENT (log only important events)
  - NIV_DEBUG (log all events)
  - NIV_FULL_DEBUG (log all internal processing)
- LRU_Nb_Call_Gc_invalid: Each worker maintains a LRU list of the last entries it handled for read/write operations through the datacache. When a worker accesses an entry, it sets it invalid in other threads' LRU's, in order to make them garbaging it. This parameter so defines the periodicity for garbaging invalid entries in this list. Thus, a worker will garbage its own list after processing this amount of read/write operations in the cache.
- LRU_Prealloc_PoolSize: This parameter sets the number of LRU entries that are preallocated for each worker thread. Given that the total amount of theses entries equals the number of datacached entries + a certain working set (number of working threads * LRU_Nb_Call_Gc_invalid), this parameter must be close to the number of datacached files divided by the number of worker threads, + the value of LRU_Nb_Call_Gc_invalid.
- Entry_Prealloc_PoolSize: This parameter is the number of preallocated datacache entries for each worker thread. It must be close to the forecasted number of entries that are to be datacached, divided by the number of worker threads.
- Cache_Directory: The local directory where the GANESHA's datacache is to be stored.
- Refresh_FSAL_Force: Force to refresh a datacached file when it has been manually modified in the cache ???
- Use_OpenClose_cache: If this boolean is set to TRUE, cached files will not be opened and closed at each read/write NFS call: GANESHA will cache a certain amount of opened file descriptors for better I/O performances (recommended).
- Max_Fd: When datacaching and 'Use_OpenClose_cache' are enabled, this parameter indicates the maximum number of files that can be kept openned for each thread.
  NB: when datacaching is disabled, use the CacheContent_Client::Max_fd parameter instead
- OpenFile_Retention: When datacaching and 'Use_OpenClose_cache' are enabled, this parameter indicates the minimum time (in seconds) a file must be kept opened.
  If the 'max_fd' limit is reached and all files have been opened more recently than the 'OpenFile_Retention' time, no more file descriptors are cached until the previous ones are older than this value.

1.2.12 The 'FileContent_GC_Policy' block

This section defines the garbage collection policy for the data cache.

- Emergency_Grace_Delay: When doing a global datacache flush (option -F of the ganesha), files who are younger than this delay will neither be flushed nor removed from the cache.

- Lifetime: the minimum delay (in seconds) a file has not been accessed, for beeing a candidate to flush and removal from the datacache.
  A value of "-1" disables data flushing.
- (not implemented yet) Inactivity_Before_Flush: Will be used for automatic flushing (without removal).
- Df_HighWater: When the local datacache filesystem usage is over this value (in percent), a garbage collection of the datacache is launched.
- Df_LowWater: A datacache garbage collection stops when the filesystem usage falls to this value (in percent).
- Runtime_Interval: The interval between checking datacache filesystem usage.
- Nb_Call_Before_GC: This parameter indicates, for each worker thread, the number of read/write calls through the datacache it has to process between checking for filesystem usage.

1.2.13 The 'NFS_Worker_Param' block

This section consists of the parameters for worker threads.

- Pending_Job_Prealloc: Each worker has a queue of pending jobs (requests that have been received from NFS clients, and that have to be processed yet).
  This parameter indicates the number of preallocated pending jobs for each worker thread. This must be close to the 'Nb_Before_GC' parameter + the potential size of requests floods.
- LRU_Pending_Job_Prealloc_PoolSize: The pending jobs are sorted into a LRU list. Thus, this parameter should be equal to the 'Pending_Job_Prealloc' parameter.
- Nb_Before_GC: When a worker is processing a flood of requests, it does not clean its pending jobs immediatly. The 'Nb_Before_GC' parameter indicates how many requests a worker must process before cleaning its queue.
- Nb_DupReq_Prealloc: This indicates the number of preallocated entries (for each worker) for the duplicate request cache. This must be close to the total number of requests stored in it, divided by the number of workers. This greatly depends on the number of NFS requests/sec and the lifetime of this cache (see the NFS_Core_Param::DupReq_Expiration parameter)
- LRU_DupReq_Prealloc_PoolSize: The pending requests are sorted into a LRU list. Thus, this parameter should be equal to the 'Nb_DupReq_Prealloc' parameter.
- Nb_DupReq_Before_GC: The 'Nb_DupReq_Before_GC' parameter indicates how many requests a worker must process before garbaging the duplicate request cache.
- Nb_IP_Stats_Prealloc: The number of preallocated entries for the IP to statistics cache. This must be close to the number of client nodes.
- Nb_Client_Id_Prealloc: The number of preallocated entries for client information cache in NFSv4. This must be close to the number of client nodes.

1.2.14 The 'NFS_Core_Param' block

This section gives general parameters for the NFS deamon.

- Nb_Worker: The number of worker threads (threads that process NFS requests)
- NFS_Port: The port number for incoming NFS requests (default is 2049)
- MNT_Port: The port number for mount protocol (default is any available port)

- NFS_Program: The RPC program number for NFS (default is 100003)
- MNT_Program: The RPC program number for mount protocol (default is 100005)
- Drop_IO_Errors: This parameter defines the behavior of the server when an EIO error is returned by the filesystem.
  TRUE indicates that the client request will be dropped, so the client will retry it later.
  FALSE indicates that an IO error is return to the client (and as a result, to the client application).
- Drop_Inval_Errors: This parameter defines the behavior of the server when an EINVAL error is returned by the filesystem.
  TRUE indicates that the client request will be dropped, so the client will retry it later.
  FALSE indicates that an IO error is return to the client (and as a result, to the client application).
- DupReq_Expiration: This defines the lifetime for the duplicate NFS requests cache.
- Core_Dump_Size: This indicates the core size in case of a server crash (default is 0)
- Stats_File_Path: This indicates the file for dumping server's statistics.
- Stats_Update_Delay: This indicates the delay for dumping server's statistics.
- Dump_Stats_Per_Client: Should be set to TRUE or FALSE. If set TRUE, the statistics per client will be dumped . (default is FALSE)
- Stats_Per_Client_Directory: If Dump_Stats_Per_Client is TRUE, this directory will contain one file per client. (default is /tmp)

1.2.15 The 'NFS_DupReq_Hash' block

Duplicate requests are found from their 'rpcxid' using a hash table. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the rpcxid, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the forecasted number of duplicate requests cache entries, divided by the number of worker threads. Thus, it must be equal to 'NFS_Worker_Param::Nb_DupReq_Prealloc'.

1.2.16 The 'NFS_IP_Name' block

Hostnames are found from the associed IP address using a hash table. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)

- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the ip address, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the number of client hosts, divided by the number of worker threads.
- Map: The hash table content can be preloaded when the NFS server is starting so that it won't have to issue any DNS requests at runtime. This parameter specifies the file that contains the DNS items to be preloaded.

1.2.17 The 'UidMapper_Cache' block

User names are used for the NFSv4 protocol. They are found from the associated uid using a hash table. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the uid, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the total number of users, divided by the number of worker threads.
- Map: The hash table content can be preloaded when the NFS server is starting so that it won't have to issue any ldap/nis request about users at runtime. This parameter specifies the file that contains the passwd items to be preloaded.

1.2.18 The 'GidMapper_Cache' block

Group names are used for the NFSv4 protocol. They are found from the associated gid using a hash table. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the gid, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the total number of groups, divided by the number of worker threads.
- Map: The hash table content can be preloaded when the NFS server is starting so that it won't have to issue any ldap/nis request about groups at runtime. This parameter specifies the file that contains the group items to be preloaded.

1.2.19 The 'NFSv4_ClientId_Cache' block

Client ids are used in NFSv4 protocol, in order to keep informations about clients. Those informations are stored into a hashtable. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the client_id, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the number of client hosts, divided by the number of worker threads.

1.2.20 The 'NFSv4_StateId_Cache' block

State ids are used in NFSv4 protocol, in order to keep informations about clients. Those informations are stored into a hashtable. This section specifies the parameters for this hastable:

- Index_Size: The size of the hash table. This MUST be a prime number, greater enough compared to the number of worker threads (specified in the 'NFS_Core_Param' block)
- Alphabet_Length: A parameter for the hashing algorithm. This must be set to the number of possible values for each byte of the client_id, i.e. 256.
  However, if you notice a bad balancing in your hash table, you can try decreasing this value (but it should not exceed 256).
- Prealloc_Node_Pool_Size: For better performances, each slot of the hash table consists of a Red-Black Tree. Thus, this parameter is the number of preallocated RBT entries for each worker thread. As a result, this must be set to a value that is close to the number of client hosts, divided by the number of worker threads.

1.2.21 The 'NFS_KRB5' block

This section specifies the parameters for RPCSEC_GSS authentication.

- PrincipalName: The principal name the NFS server (default is nfs@localhost.localdomain)
- KeytabPath: The Kerberos5 keytab for this principal

1.2.22 The 'NFSv4' block

This block is for NFSv4 specific parameters.

- Lease_Lifetime: This specifies the NFSv4 lease time (see RFC 3530 for more details)
- FH_expire: This specifies if NFSv4 FH will expire (see RFC 3530 for more details)
- Returns_ERR_FH_EXPIRED: Specifies if the serveur should return NFS4ERR_FHEXPIRED. This will be used only if FH_expire is TRUE.