

[2/3] POHMELFS: Documentation.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-07/msg02843.html>

- *From:* Evgeniy Polyakov <johnpol@xxxxxxxxxxx>
 - *Date:* Mon, 7 Jul 2008 22:10:19 +0400
-

Design notes, usage cases and protocol description.

Signed-off-by: Evgeniy Polyakov <johnpol@xxxxxxxxxxx>

```
diff --git a/Documentation/filesystems/pohmelfs/design_notes.txt
b/Documentation/filesystems/pohmelfs/design_notes.txt
new file mode 100644
index 0000000..0b20a47
--- /dev/null
+++ b/Documentation/filesystems/pohmelfs/design_notes.txt
@@ -0,0 +1,70 @@
+POHMELFS: Parallel Optimized Host Message Exchange Layered File System.
+
+Evgeniy Polyakov <johnpol@xxxxxxxxxxx>
+
+Homepage: http://tservice.net.ru/~s0mbre/old/?section=projects&item=pohmelfs
+
+It was first started as network filesystem with coherent local data and metadata caches,
+but it is being evolved into parallel distibuted filesystem now.
+
+Main features of this FS include:
+ * Local coherent (notes cache for data and metadata:
+ http://tservice.net.ru/~s0mbre/blog/devel/fs/2008\_05\_17.html)
+ * Completely async processing of all events (hard, symlinks and rename are the
+ only exceptions) including object creation and data reading and writing.
+ * Flexible object architecture optimized for network processing.
+ Ability to create long pathes to object and remove arbitrary huge
+ directoris in single network command.
+ (like removing the whole kernel tree via single network command).
+ * Very high performance.
+ * Fast and scalable multithreaded userspace server. Being in userspace it works
+ with any underlying filesystem and still is much faster than async in-kernel NFS one.
+ * Client is able to switch between different servers (if one goes down, client
+ automatically reconnects to second and so on).
+ * Transactions support. Full failover for all operations.
+ Resending transactions to different servers on timeout or error.
+ * Read requests (data read, directory listing, lookup requests) balancing between multiple servers.
+ * Write requests are sent to multiple servers and completed only when all of them sent an ack.
+ * Ability to add and/or remove servers from working set at run-time from userspace (via netlink,
```

[2/3] POHMELFS: Documentation.

- + so the same command can be processed from real network though, but since server does not support it yet, I dropped network part).
- +
- +
- +POHMELFS is based on transactions, which are potentially long-standing objects, which live in clients memory. Each transaction contains all information needed to process given command (or set of command, which is frequently used during data writing: single transaction can contain creation and data writing commands). Transaction is committed by all servers, where it was sent, and in case of failure of one or another, it will be eventually resent or dropped with error, so, for example, reading will return error, if no servers are available.
- +
- +POHMELFS uses novel asynchronous approach of data processing. Courtesy to transactions, it is possible to detouch reply from request, and if command requires data to be received, caller just sleeps waiting for it. Thus it is possible to issue multiple read commands to different servers and async threads will pick replies in parallel, find appropriate transactions in the system and put data where it belongs (like page or inode cache).
- +
- +Main feature of the POHMELFS is writeback data and metadata cache.
- +Only few non-performance critical operations use write-through cache and are synchronous: hard and symbolic link creation and object rename. Creation and removal of objects, as long as writing, are asynchronous and are sent to the server during system writeback.
- +When server receives some request for given object in the system (like data reading, or file creation or whatever else), it stores appropriate client information in own cache, so when subsequent request comes from different client, all previous could be notified (for example when several clients read data from file, and then new client writes there, appropriate pages on clients will be invalidated, so subsequent write will force them to read page from the server). Because of this feature POHMELFS is extremely fast in metadata intensive workloads, and can fully utilize bandwidth to servers when doing bulk data transfers.
- +
- +POHMELFS client operates with working set of servers, and is capable of balancing reading, i.e. no modification, like lookup or directory listing, workload between all servers it was conneted to.
- +Administrator can add or remove servers from the set in run-time via special command (described in Documentation/pohmelfs/info.txt file). Writing is performed to all servers.
- +
- +POHEMLFS is capable of full data channel encryption and/or strong crypto hashing.
- +One can select kernel supported cipher, encryption mode, hash type and operation mode (hmac or digest). It is also possible to use both or neither (default). Crypto configuration is checked during mount time, and if server does not support it, appropriate capabilities will be disabled or mount will fail (if 'crypto_fail_unsupported' mount option is specified).
- +Crypto performance heavily depends on number of crypto threads, which asynchronously perform crypto operations and send data to server or submit it higher to the stack. Number of crypto threads is controlled via mount option.

```
diff --git a/Documentation/filesystems/pohmelfs/info.txt b/Documentation/filesystems/pohmelfs/info.txt
new file mode 100644
index 0000000..f96f09d
--- /dev/null
+++ b/Documentation/filesystems/pohmelfs/info.txt
@@ -0,0 +1,81 @@
```

- +POHMELFS usage information.
- +
- +Mount options:

+idx=%u
+ Each mountpoint is associated with special index via this option.
+ Administrator can add or remove servers from given index, so all mounts,
+ which were attached to it, were updated.
+ Default it is 0.
+
+trans_scan_timeout=%u
+ This timeout, expressed in milliseconds, specifies time to scan transaction
+ trees looking for stale requests, which have to be resent, or if number of
+ retries exceed specified limit, dropped with error.
+ Default is 5 seconds.
+
+drop_scan_timeout=%u
+ Internal timeout, expressed in milliseconds, which specifies how frequently
+ inodes marked to be dropped are freed. It also specifies how frequently
+ system checks, that servers has to be added or removed from current working set.
+ Default is 1 second.
+
+wait_on_page_timeout=%u
+ Number of milliseconds to wait for reply from remote server for data reading command.
+ If this timeout is exceeded, reading returns error.
+ Default is 5 seconds.
+
+trans_retries=%u
+ Number of times, transaction will be resent to the server, which did not answer for the
+ last @trans_scan_timeout milliseconds. When number of resends exceeds this limit,
+ transaction is completed with error.
+ Default is 5 resends.
+
+crypto_thread_num=%u
+ Number of crypto processing threads. Threads are used both for RX and TX traffic.
+ Default is 2, or no threads if crypto operations are not supported.
+
+trans_max_pages=%u
+ Maximum number of pages in single transaction. This parameter also control number of pages,
+ allocated for crypto processing (each crypto thread has pool of pages, number of which is
+ equal to 'trans_max_pages').
+ Default is 100 pages.
+
+crypto_fail_unsupported
+ If specified, mount will fail if server does not support requested crypto operations.
+ By default mount will disable non-matching crypto operations.
+
+Usage examples.
+
+Add (or remove if it already exists) server server1.net:1025 into working set with index \$idx
+with appropriate hash algorithm and key file and cipher algorithm, mode and key file:
+\$cfg -a server1.net -p 1025 -i \$idx -K \$hash_key -H "hmac(sha1)" -C "cbc(aes)" -k \$cipher_key
+
+Mount filesystem with given index \$idx to /mnt mountpoint.
+Client will connect to all servers specified in working set via previous command:

[2/3] POHMELFS: Documentation.

```
+mount -t pohmel -o idx=$idx q /mnt
+
+One can add or remove servers from working set after mounting too.
+
+
+Server installation.
+
+Creating a server, which listens at port 1025 and 0.0.0.0 address.
+Working root directory (note, that server chroots there, so you have to have appropriate permissions)
+is set to /mnt, server supports SHA1 hasing (with 'hash_key' key file) and AES-CBC encryption
+(with 'cipher_key' key file, its size specifies cipher blocksize)
+Number of working threads is set to 10.
+
+# ./fserver -a 0.0.0.0 -p 1025 -r /mnt -w 10 -K hash_key -H "hmac(sha1)" -C "cbc(aes)" -k cipher_key
+
+-r root - path to root directory. Default: /tmp.
+-a addr - listen address. Default: 0.0.0.0.
+-p port - listen port. Default: 1025.
+-w workers - number of workers per connected client. Default: 1.
+-K file - hash key size. Default: none.
+-H hash - hash type. Default: none.
+-k file - cipher key size. Default: none.
+-C hash - cipher type. Crypto mode should be specified as 'mode(cipher)'. Default: none.
+-h - this help.
+
+Number of worker threads specifies how many workers will be created for each client.
+Bulk single-client transfers usually are better handled with smaller number (like 1-3).
diff --git a/Documentation/filesystems/pohmelfs/network_protocol.txt
b/Documentation/filesystems/pohmelfs/network_protocol.txt
new file mode 100644
index 0000000..bc13058
--- /dev/null
+++ b/Documentation/filesystems/pohmelfs/network_protocol.txt
@@ -0,0 +1,220 @@
+POHMELFS network protocol.
+
+Basic structure used in network communication is following command:
+
+struct netfs_cmd
+{
+  __u16 cmd; /* Command number */
+  __u16 csize; /* Attached crypto information size */
+  __u16 cpad; /* Attached padding size */
+  __u16 ext; /* External flags */
+  __u32 size; /* Size of the attached data */
+  __u32 trans; /* Transaction id */
+  __u64 id; /* Object ID to operate on. Used for feedback.*/
+  __u64 start; /* Start of the object. */
+  __u64 iv; /* IV sequence */
+  __u8 data[0];
+};
```

+
+Commands can be embedded into transaction command (which in turn has own command),
+so one can extend protocol as needed without breaking backward compatibility as long
+as old commands are supported. All string lengths include tail 0 byte.
+
+@cmd – command number, which specifies command to be processed. Following
+ commands are used currently:
+
+ NETFS_READDIR = 1, /* Read directory for given inode number */
+ NETFS_READ_PAGE, /* Read data page from the server */
+ NETFS_WRITE_PAGE, /* Write data page to the server */
+ NETFS_CREATE, /* Create directory entry */
+ NETFS_REMOVE, /* Remove directory entry */
+ NETFS_LOOKUP, /* Lookup single object */
+ NETFS_LINK, /* Create a link */
+ NETFS_TRANS, /* Transaction */
+ NETFS_OPEN, /* Open intent */
+ NETFS_INODE_INFO, /* Metadata cache coherency synchronization message */
+ NETFS_JOIN_GROUP, /* Joing metadata update group */
+ NETFS_LEAVE_GROUP, /* Leave metadata update group */
+ NETFS_PAGE_CACHE, /* Page cache invalidation message */
+ NETFS_READ_PAGES, /* Read multiple contiguous pages in one go */
+ NETFS_RENAME, /* Rename object */
+ NETFS_CAPABILITIES, /* Capabilities of the client, for example supported crypto */
+
+@ext – external flags. Used by different commands to specify some extra arguments
+ like partial size of the embedded objects or creation flags.
+
+@size – size of the attached data. For NETFS_READ_PAGE and NETFS_READ_PAGES no data is
+ attached,
+ but size of the requested data is incorporated here. It does not include size of the command
+ header (struct netfs_cmd) itself.
+
+@id – id of the object this command operates on. Each command can use it for own purpose.
+
+@start – start of the object this command operates on. Each command can use it for own purpose.
+
+@csize, @cpad – size and padding size of the (attached if needed) crypto information.
+
+Command specifications.
+
+@NETFS_READDIR
+This command is used to sync content of the remote dir to the client.
+
+@ext – length of the path to object.
+@size – the same.
+@id – local inode number of the directory to read.
+@start – zero.
+
+
+@NETFS_READ_PAGE

- +This command is used to read data from remote server.
- +Data size does not exceed local page cache size.
- +
- +@id – inode number.
- +@start – first byte offset.
- +@size – number of bytes to read plus length of the path to object.
- +@ext – object path length.
- +
- +
- +@NETFS_CREATE
- +Used to create object.
- +It does not require that all directories on top of the object were
- +already created, it will create them automatically. Each object has
- +associated @netfs_path_entry data structure, which contains creation
- +mode (permissions and type) and length of the name as long as name itself.
- +
- +@start – 0
- +@size – size of the all data structures needed to create a path
- +@id – local inode number
- +@ext – 0
- +
- +
- +@NETFS_REMOVE
- +Used to remove object.
- +
- +@ext – length of the path to object.
- +@size – the same.
- +@id – local inode number.
- +@start – zero.
- +
- +
- +@NETFS_LOOKUP
- +Lookup information about object on server.
- +
- +@ext – length of the path to object.
- +@size – the same.
- +@id – local inode number of the directory to look object in.
- +@start – local inode number of the object to look at.
- +
- +
- +@NETFS_LINK
- +Create hard or symlink.
- +Command is sent as "object_path|target_path".
- +
- +@size – size of the above string.
- +@id – parent local inode number.
- +@start – 1 for symlink, 0 for hardlink.
- +@ext – size of the "object_path" above.
- +
- +
- +@NETFS_TRANS

- +Transaction header.
- +
- +@size – incorporates all embedded command sizes including theirs header sizes.
- +@start – transaction generation number – unique id used to find transaction.
- +@ext – transaction flags. Unused at the moment.
- +@id – 0.
- +
- +
- +@NETFS_OPEN
- +Open intent for given transaction.
- +
- +@id – local inode number.
- +@start – 0.
- +@size – path length to the object.
- +@ext – open flags (O_RDWR and so on).
- +
- +
- +@NETFS_INODE_INFO
- +Metadata update command.
- +It is sent to servers when attributes of the object are changed and received
- +when data or metadata were updated. It operates with the following structure:
- +
- +struct netfs_inode_info
- +{
- + unsigned int mode;
- + unsigned int nlink;
- + unsigned int uid;
- + unsigned int gid;
- + unsigned int blocksize;
- + unsigned int padding;
- + __u64 ino;
- + __u64 blocks;
- + __u64 rdev;
- + __u64 size;
- + __u64 version;
- +};
- +
- +It effectively mirrors stat(2) returned data.
- +
- +
- +@ext – path length to the object.
- +@size – the same plus size of the netfs_inode_info structure.
- +@id – local inode number.
- +@start – 0.
- +
- +
- +@NETFS_JOIN_GROUP/NETFS_LEAVE_GROUP
- +Metadata cache coherency synchronization messages.
- +They are broadcasted when new inode is created (either for new object
- +or object read from the server), so that server new inode number of the
- +object on the appropriate client. @NETFS_LEAVE_GROUP is sent when local

- +inode is destroyed, so that client physically can not be interested in data or metadata updates for given inode.
- +
- +@ext – path length to the object.
- +@size – the same.
- +@id – local inode number for given object.
- +@start – 0.
- +
- +
- +@NETFS_PAGE_CACHE
- +Command is only received by clients. It contains information about page to be marked as not up-to-date. Server fills this command with data, provided by above @NETFS_JOIN_GROUP command.
- +
- +@id – client's inode number.
- +@start – last byte of the page to be invalidated. If it is not equal to current inode size, it will be vmtruncated().
- +@size – 0
- +@ext – 0
- +
- +
- +@NETFS_READ_PAGES
- +Used to read multiple contiguous pages in one go.
- +
- +@start – first byte of the contiguous region to read.
- +@size – contains of two fields: lower 8 bits are used to represent page cache shift used by client, another 3 bytes are used to get number of pages.
- +@id – local inode number.
- +@ext – path length to the object.
- +
- +
- +@NETFS_RENAME
- +Used to rename object.
- +Attached data is formed into following string: "old_path|new_path".
- +
- +@id – local inode number.
- +@start – parent inode number.
- +@size – length of the above string.
- +@ext – length of the old path part.
- +
- +
- +@NETFS_CAPABILITIES
- +Used to exchange crypto capabilities with server.
- +If crypto capabilities are not supported by server, then client will disable it or fail (if 'crypto_fail_unsupported' mount options was specified).
- +
- +@id – superblock index. Used to specify crypto information for group of servers.
- +@size – size of the attached capabilities structure.
- +@start – 0.
- +@size – 0.
- +@scsize – 0.

--

Evgeniy Polyakov

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>