

*Relio*

fs

TECHNICAL WHITE PAPER

[www.sanbolic.com](http://www.sanbolic.com)

## introduction

Shared file systems (FS) are emerging as a key technology because of the increasing use of network connected storage rather than server connected storage, and application requirements for high availability, scalability and performance. The server maintained distributed file systems will become obsolete as applications in data processing centers and web farms will be able to take advantage of the performance and scalability of shared FS. Some of the protocols such as NFS and CIFS used in today's distributed technologies will be used as complementary services to the shared file system, for cheaper access through LAN or WAN attached workstations. In the near future we will see the transition of fibre channel (FC) from high end storage connectivity, to much more affordable and broad storage applications, as well as close integration of storage area network (SAN) and network attached storage (NAS). This will be possible because of FC's ability to support multiple protocols over single serial point of connectivity, and it's ability to simultaneously provide network and storage interface, thus reducing complicated administration and allowing integrated network and storage operations.

## description of melio fs

### cluster definition

A cluster is a set of machines, accessing a particular shared volume. Melio supports dynamically the cluster for each volume. This means that machine mounting a volume has to scan and enter the cluster that has already mounted that volume. This is done by issuing a reconfiguration request from the newcomer that every machine already in the cluster has to accept via a silent "YES". From that point on the old cluster transforms to the new one via a morphing procedure, retaining all existing locks and allowing lock operations during the process. Such process is initiated also if the connection with one or more of the nodes is broken - when a machine does not respond to request the cluster is rescanned and reconfigured by the node initiating the request, which failed. Switch-based fabrics are supporting increasing functionality including name servers, which provide hosts with a dynamic database of currently attached devices and hosts.

### melio fs cluster structure

A cluster is the logical entity representing a set of nodes that are permitted to register locks on a volume. Therefore, a node participates in as many clusters as there are volumes mounted by it. The node that initiates changes in the cluster is called the initiator. It may build up a cluster out of empty one, add or remove nodes (morph the cluster). The initiator is the first node that determines that there is a need to modify the cluster. The synchronization between nodes outside cluster is done via separate region of the disk called disk spinlock, so network failure does not create a danger of corruption. The time to enter the cluster for a group of nodes is the same as time for a single node - about two seconds for morphing or building up a new cluster in non-failure case, 20 seconds for rebuilding the cluster after a failure.

The cluster morphs in the following situations:

- **Node failure.** The first node that determines that failure occurred becomes the initiator and changes the cluster by rescanning the network for potential cluster members, applying all committed transactions of the orphaned

journal(s), and releasing all locks of the crashed node(s).

- **Dynamic entering cluster (on mount).** The first who gains entry rescans network for other nodes willing to participate in the same cluster, and then morphs the cluster so that it includes the new nodes. The current cluster members are informed about the change so they distribute their locks to the newcomers.

- **Partitioning of network.** Partitioning is discovered by all segments and they start to compete for the cluster. Once one of the partitions forms a new cluster the rest are blocked out; they continue to try to enter cluster as long as there are new requests. These retries may succeed when all nodes of the current cluster remove their locks over the mounted media (close all files) and make no further transactions for some period of time. In this case the cluster would move from one segment to another.

### *melio fs clustering operations*

Each computer within the cluster will see the partitions formatted with MelioFS as local disks. Computers in the cluster can be running any operating system supported by Melio FS. All operations possible with local file systems will be possible with Melio FS, which fully utilizes the bandwidth provided by the SAN hardware. Entering or exiting the cluster does not require special operations, and the time required for the first mount is only about two seconds. The remaining hosts will detect crashes of members of the cluster and recovery of MelioFS metadata would be performed, based on the journal of the dead host. There is no master computer in the cluster. All communication between cluster members is symmetrical.

### *handling of node failure*

In case of node failure, the other nodes discover this fact within seconds (the precise time depends on the activity, but no more than 10 seconds). Another node becomes the initiator of a cluster morphing. It applies any committed transactions, releases the orphaned journal, and locks and removes it from the cluster. The entire process is fully automatic and transparent to the users on the other nodes. The number of simultaneously failed nodes is irrelevant for the handling.

### *administration of melio fs: api. volume management*

Currently only distributed format capability is supported - a volume can be formatted on the fly without interrupting user nodes' operation. Melio implements:

- Detailed statistical information about the operation and health report API.
- Volume snapshots. The first version of MelioFS includes read-only volume snapshots. Later this limitation will be removed.
- Seamless expand/shrink of volumes during normal operation.
- Background integrity check and repair of volumes.
- Distributed background layout analysis and optimization of volumes.



### *cluster-wide locks system*

MelioFS has distributed locking mechanism, allowing registration of cluster-wide locks. It can operate over any existing network protocol (currently is implemented using TCP/IP). The locks provide shared/exclusive access over fine-grained virtual lock space. The FS components, using the lock system map abstract locking semantics within it, like setting the access mode to disk regions, stream regions, file access modes, etc. Locks are cached so network traffic is kept to a minimum. The access rights over the virtual lock space are yielded upon newer demand for them - preemption. The lock system notifies its clients when actual rights are gained or lost by the node so caches are kept coherent.

MelioFS fully supports the high-level lock APIs of the running operating systems. The different semantics of the locks for each operating system are translated between different platforms and mapped to the virtual lock space.



### *protocol*

Minimum knowledge (communication) principle is followed in the protocol design. The lock database contains information about the states and owners of regions from the virtual lock space. This information is only partial and is kept on common working set principle.

### *lock prediction*

The lock mechanism includes lock-ahead module operating over the virtual lock space that analyzes the lock patterns and predicts future needs. It uses aggressive pre-allocation to reduce the number of interactions between nodes.



### *high-level locks*

Melio provides the standard file locks and byte range locks, as defined by supported operating systems. Their implementation is based on the cluster-wide lock system, so it benefits from all of its features.



### *security*

Unix operating systems have quite different security semantics than Windows. For this reason two types of security are used in MelioFS - one for Unix and one for Windows environment. Tools are being developed to convert one type of security information to the other, based on rules defined in script files.



### *mounting*

To mount MelioFS two things must be done:

1. Install MelioFS driver on a computer.
2. Format a partition with mfs\_format utility.

From this point on user can access the partition as normal local file system, even after reboots or system crashes. No addition maintenance is required.



### *melio fs layout*

MelioFS has two-level layout.

□ **Low-level layout (streams).** It translates the physical disk, represented by SCSI driver as array of sectors to set of byte streams and free space. Each stream is identified by their start (64bit offset from beginning of disk). Each stream has certain length, which can grow or shrink at the stream end. The granularity at which a stream can be described is 64 bytes, making space losses caused by internal fragmentation practically insignificant.

□ **High-level layout (layout).** This level builds up all data structures required to support file system defined abstractions like files, directories, etc. It uses streams for allocation and deallocation of disk space. The structures are accessed after byte order translation and contain version information, enabling backward compatibility.

### *performance*

A shred FS's performance depends on two factors:

□ **Representation:** How well metadata represents the abstract objects: streams, files, and directories. Required resources and access time are a measure of the quality of the representation. Required resources are disk space, memory space, SCSI transfer bandwidth, memory bandwidth, CPU time, disk seek time, disk internal transfer bandwidth.

□ **Locality:** this is requirement to put metadata near the data it controls, so it is 'local' to the data structures. This requirement is set by 3 factors:

**1. Physical disk characteristics** - disks requires less seek time when data and metadata are concentrated in small area.

**2.1 Latency** - each operation has a constant overhead to its response time, caused by processing specifics. When locality is good, all required data and metadata could be read with single request, avoiding multiple waits of latency time.

**2.2 Request Processing Overhead** - each separate request to IO subsystem uses also system bus bandwidth, memory bandwidth, CPU time, and causes additional management work for OS and drivers. This delays the request completion (and slows the entire system). If fewer requests are capable of delivering all required data, needed, this time would be reduced.

**3. Cache Optimization:** disk caches operate by assuming locality of data accesses. There is time and space locality. Time locality means that if one region is accessed it will be accessed again soon. Space locality means that accessing a region raises the probability to access regions near by. If metadata is local to data caches are used more efficiently as the rate of correct predictions raises. On the other hand, there is a benefit within a SAN to spreading metadata as much as possible over disk space, so separate workstations can operate independently, wait for each other less frequently, and invalidate fewer transactions mutually. Streams design is oriented to meet the

two conflicting objectives by keeping metadata spread, but near to the controlled data. To compensate the negative impact over caches metadata is designed so that its size increases logarithmically as the size of controlled data grows.

### *the lack of metadata server improves the performance*

The advantages of the lack of a metadata server are:

**1. Performance** - no single node takes care of the metadata and locking for all operations in the cluster. Instead this job is distributed to all nodes in the cluster. The amount of communication between the nodes is reduced via special algorithms for space allocation so logically independent data is independent physically and no communication is required to synchronize the access to it. Special lock system is used for relaxed communication and lazy write requests ordering optimization.

**2. Availability** - the data in the cluster is available to every connected node, even if a node that was in the cluster had failed.

**3. Cost** - no special hardware or administration effort is required to guarantee the fail-safe operation of a metadata server(s) to keep cluster running.

**4. Administration** - maintenance on one node does not affect the others operation opposed to maintenance of the server, affecting all other nodes.

There is no specialized metadata server or other privileged role in the cluster. A cluster is formed out of equal nodes. A distributed transaction mechanism is used, which is based on fine-grained locks and technology, which minimizes the performance impact of transactions on other nodes. When the number of nodes increases the communication within the cluster is performed on common working set basis - only between nodes, accessing same data. Out of band communications is used for the distributed lock mechanism. The metadata for the file system resides on the common storage and goes through the FC.

### *caching*

Based on the received lock information each machine purges the cached data residing in the regions locked by other nodes and flushes data, required by other nodes. Full cache coherency is guaranteed throughout the cluster at any time.

### *heterogeneous byte ordering*

(little-endian-big-endian)

All internal data types exchanged over the network or stored on the shared disk are wrapped in C++ classes, performing system-dependent translation. The information that is interpreted in OS in a defined way (file attributes, date/time) is translated as precise as possible. In a second version of MelioFS translation policies based on a script language for things like security will be implemented.

## legend

**NFS** = Network File System

**CIFS** = Common Internet File System

**Stream** = Contiguous sequence of bytes, used for storing metadata or data. It is the abstraction, used to manage disk space.

**Session** = Period, between OS boot and its shutdown/unrecoverable crash.

## contact us

Sanbolic Inc.  
304 Pleasant Street  
Watertown MA 02472, USA  
sales: 617 833 4242  
fax: 617 926 2808  
email: sales@sanbolic.com