Inferno distributed filesystem

Tomáš ČOREJ*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
corej06@student.fiit.stuba.sk

Abstract. Inferno distributed filesystem is a peer-to-peer application of melua grid software for reliable, transparent and secure data storage on local or metropolitan networks.

1 Motivation

Distributed data storage was always very cheap way of increasing disk capacity on network. There is a lot of ways to do it but almost all methods rely on on distributing data using some central points which may fault. We wanted to have network filesystem which wouldn't rely on any central point and would be fault-tolerance and transparent and have ideas from Venti [3] archival file server and p2p clients.

We noticed that many users share data on LAN through Samba or FTP servers. Many files on network is stored on many computers wasting disk capacity and we can't get data from those computer that are offline. A lot of users put restrictions on accessing to they shared data by limiting download speed or deny from accessing for decreasing their CPU load.

This paper is about how to overcome many problems in creating distributed decentralized filesystem with load balancing and other features on network where we can trust no one. It is strongly encouraged to read [2] first.

2 Application

IDFS is a application of melua grid software [2] which is software package for creating distributed environments on TCP/IP networks and it uses this environment for creating

* Supervisor: Ing. Matej Košik, Institute of Applied Informatics / Institute of Informatics and Software Engineering / Institute of Computer Systems and Networks, Faculty of Informatics and Information Technologies STU in Bratislava.

IIT.SRC 2008, Bratislava, April 30, 2008, pp. 1–8.

network storage. This storage would have this features:

- transparency – access to storage is provided through standard file I/O operations in mounted directory (e.g. /mnt/idfs on UNIX or Z: on Windows)
- decentralization – there is not any central point and failure of nodes doesn't affect other nodes
- security – encryption,authorization and authentication
- fault-tolerance – node failures won't affect availability of data in storage
- load balancing – data would be on many nodes to decrease CPU load
- portability – users may use different architectures or operation systems to access storage
- license – as a free software for non-commercial usage

As a application IDFS is file server which provides Styx [6] protocol on its standard input and output and mounts it to /shared/idfs directory which is then exported to grid. Application runs on every node in grid and manages data storage by accessing to /usr/*/*/idfs directories in global namespace. Due to mesh topology of melua grid it can directly communicate with other nodes through filesystem interface.

```
; mount {idfs.dis} mnt
Inferno distributed filesystem idfs
; cd mnt
; ls -l
--rw-r--r-- M 2 v92 v92 0 Jan 01  1970 ctl
--rw-r--r-- M 2 v92 v92 0 Jan 01  1970 debug
--rw-rw-rw- M 2 v92 v92 0 Jan 01  1970 dirtree
;
```

- ctl file is used to control local configuration of idfs server by writing commands to it.For example

  ```
  ; echo dirtree compressed > ctl
  ```

  would provide new file named dirtree.gz which is compressed version of directory structure
- dirtree includes one S-expression which represent directory structure of a filesystem
- debug is for debugging purposes

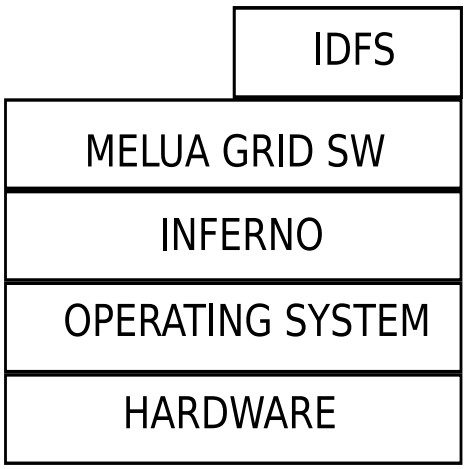

Fig. 1. IDFS is on top of abstraction of computer which provides its resources

3 Network

3.1 Directory structure

Directory structure is held in file *dirtree* which consist from one S-expression [5]. S-expressions are easy to parse, express tree-like structures (without cycles),platform independent. Example:

```
( / ( mode (d) perm (user8 group 755) entry (
test ( perm (user1 group2 755) mode (fces)
            IDA (ida1 (hash1 hash2) ida2 (hash8 hash9))
            time (created (user modified) access)
            open()
)
fest ( perm (user2 group2 644) mode (fs)
            IDA (ida1 (hash1 hash3))
            time (created (user2 modified) access)
            open (user2 rw)
)
dir ( perm (user1 rwx) mode (ds) e ()
            time (created (user modified) access))
) time (created (user modified) access)
))
```

This is an example of simple directory structure consist from root directory,two files and one empty directory. Every entry in dirtree consist from *attrib value* pair where *attrib* is

name of one element in path or is it attribute's name and *value* is a list or atom. Possible attributes with their values are:

- **m**ode – **f**ile,**d**irectory,**c**ompression,**e**ncryption, **s**napshots. Attributes are inherited from actual directory for newly created files or subdirectories and they may be changed anytime but for old files must be this change explicit.
- **p**erm – standard UNIX-like permissions are in format *owner group other perm*
- **t**ime – time access attributes are same as in UNIX except modified.According to Styx protocol there is also a username stored of user who last modified file.
- **I**DA – consist from at least one IDA block with maximum size 32MiB each.First item of IDA block is dispersal matrix and second is list of *n* SHA-1 hashes.
- **o**pen – indicates who has opened file. It has format open (user1 flags user2 flags user3 flags ... userN flags) where flags may be read or write.

3.2 Data transformations

Every data block that is read or write to storage grid must proceed through few data transformations which prepares data.

3.2.1 Compression and encryption

Compression and encryption would be provided (in this order) only if appropriate flag in mode attribute is set. For compression there is used gzip-compatible stream compression but only for files that are not compressed yet (excluding compressed files). And for encryption IDEA,DES or AES ciphers every in ECB or CBC mode may be used. Default configuration may be set through *ctl* file.

3.2.2 Information Dispersal Algorithm

Information dispersal algorithm [4] breaks file into *n* blocks in such way that any *m* blocks would be sufficient to reconstruct original file.

$$m \leq n$$

Size of each block is

$$\frac{|F|}{m}\ bytes$$

where $|F|$ is a file size and added redundancy is

$$(\frac{n}{m} - 1) * 100\%$$

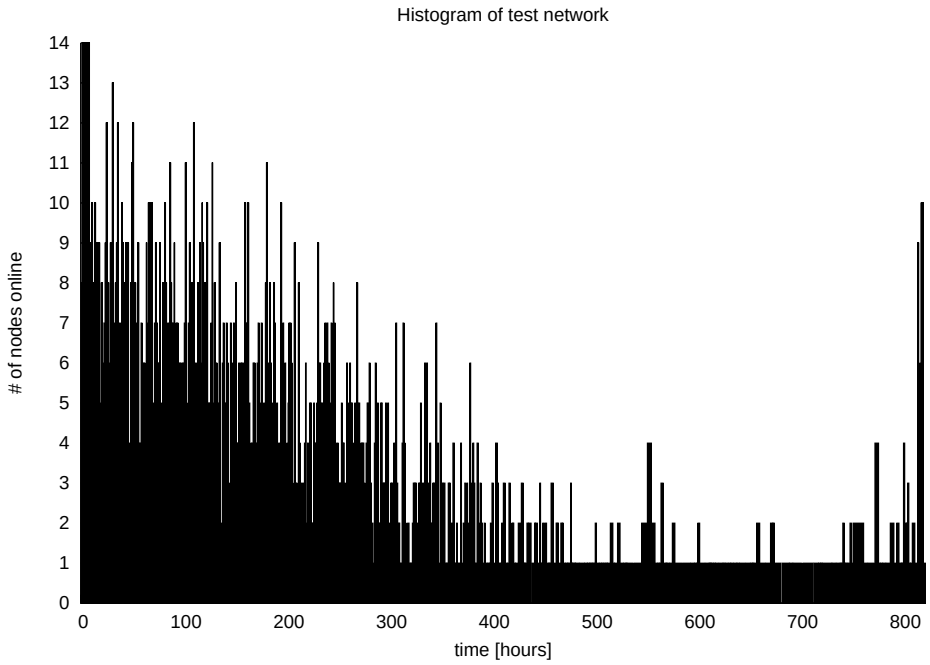

Fig. 2. Histogram of nodes availability from 16.1.2008 23:30 to 19.2.2008 23:30

Because *n* and *m* are IDA parameters, we may tune its redundancy dynamically based on distribution model.These parameters are set to their default values ($n = 14\ m = 7$).One of the important features of IDA is that mixes parity with data together so $m - 1$ blocks give no information about F but Rabin proposed in [4] that $m - 1$ may provide *some* information.

Every large file is divided into 32MiB blocks which are transformed through IDA and it is called IDA block. It is because reconstruction of large files would be too long and would consume a lot of memory.

3.3 Distribution

Distribution is done through Distributed Hash Table [7] which is suitable for large decentralized p2p networks and it also has to consider many aspects of nodes. The most important factor of every node is its uptime. Longer the node is online it considered as more reliable and vice versa. Such nodes have tendency to group in the beginning of the array of nodes and more faulty nodes are on opposite side of array.Second is capacity but if there is not enough disk space on one node we understand it as a fault and we go on. To create best model for writing or reading to the storage which will consider many aspects of network we had to study users behavior on network.

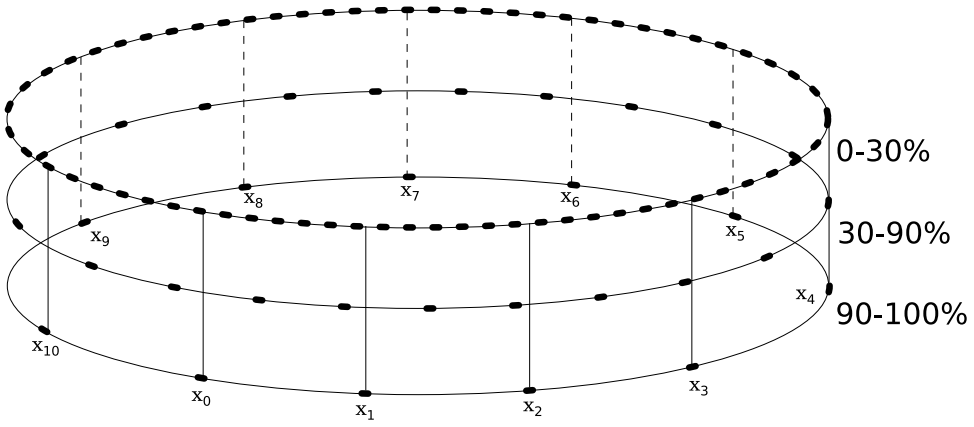

Fig. 3. Distribution model for 99 nodes (m = 11, n = 22) and three probability levels

3.3.1 User behavior

For a observation purposes we used part of network of Presov's local provider which has 2335 users where we measured every hour their availability (Figure 2) using ARP. In this type of network we can't be sure with anything because there is big fluctuation of nodes so we had to create model that would be robust enough. We expected to see a Gaussian curve but tests showed us that large part (92%) of user group use their computers only few hours per day. There is also a group of users (7.4%) who run their computers for more than 400 hours per month.So based on this data we created theoretical model.

On a Figure 3 we used cylindrical coordinate system where nodes are set on cylinder's circumference in such way that they divide SHA-1's range (2^{160}).On bottom there are only *m* nodes which are considered as more reliable but for the rest of nodes only $(n-m)$ blocks are distributed. In ideal case every file stored in grid would be dispersed on all nodes, but there are some performance issues which would make IDA transformations very slow if *m* or *n* are too big.For nodes on same level

$$0 = x_0 \leq x_1 \leq x_2 \ldots \leq x_i = 2^{160} - 1$$

Where *i* is number of nodes on same probability level (for bottom it is *m*). Probability level is a set of nodes with similar probabilities. Every node gets SHA-1 hash by this formula:

$$x_i = hash(IP, port, username)$$

Using this system we may simply read and write data from nodes using hash of its blocks. Every file that is about to be stored into grid is dispersed into *n* blocks using IDA and then hashes of these blocks are computed. Now we have pair (hash,data) and we have to search for node which its hash is above our hash.

$$x_{i-1} < data_{hash} < x_i$$

Write copies block into node's (with hash x_i) idfs directory.We do this for every probability level. Read operation gets hashes from *dirtree* file and then it searches on nodes with nearest x_i. So if write or read operation fail we just move to next probability level and read/write would be done from less reliable nodes. Next step is to notify whole grid about our changes using broadcast.

3.4 Efficient Broadcast

For file operations open(),close(),delete() we have to notify every node in grid because we need preserve consistency of directory structure.These operations are done as a update messages to *dirtree* file.Example:

```
; echo ( /test ( open ( user1 rw ))) > dirtree
; echo ( /test ( IDA ( ida1 (hash1hash8)))) > dirtree
```

Writing this type of message would update dirtree and notify every node, that file /test is opened by user1 in read-write mode.Same method is used when we want to update data blocks of file or directory. In second example we updated first IDA block of file /test.

Notification itself can't be done by writing same message to each node because it would be too slow (O(n)) and waste bandwidth by transmitting same data. Better method is to use binary tree which notify every node in O($\log_2 n$) and it is already represented as an array of files provided by registry server in /mnt/registry directory. Every node with index *i* have its descendants on positions $2i$ and $2i+1$. Because nodes are sorted in registry by time of creation it reflects which nodes are more reliable than others. More reliable nodes are on the beginning of array so notify starts from node with index 0.

When snapshot flag is set,every notification is logged to achieve transparent version control system.Idea is that user may change any directory into its earlier versions writing command into *ctl* file. Example:

```
; echo snapshot / 010120081734 > ctl
```

Would change whole root directory into state on 1.1.2008 at 17:34.

IP Multicast was also considered, and we would like to have it as a option to use it. There are some problems in using multicast for reliable data transfer, so we took inspiration from Digital Fountain [1] which uses Tornado codes and use IDA instead.

4 Performance issues

Performance for this project is very critical because it affects processing speed. Data transformations are very time consuming and increases system load. Many optimizations can be done. IDA uses matrix multiplication which can be reduced from O(n^3) to O($n^{\log_2 7}$) using *Strassen matrix multiplication* and multiplications itself may be reduced using FFT multiplication (over GF(2^w)) which reduces time from O(n^2) to O($n \log n$). But these types of optimizations works well for big *n*,*m* and data blocks from 2^{16} bits.

Inferno implementation of IDA uses for computing Galois Field which computes only small data blocks 16 bits wide. Arithmetic in GF is well optimized for this purposes because addition and subtraction is reduced only into XOR operation and multiplication consist from shifts and additions.

It is still subject of research what is better, if it is using FFT multiplication using big blocks or Galois Field arithmetic but on small values.

Acknowledgement: This work was partially supported by the Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava.

References

[1] Byers, J., Luby, M., Mitzenmacher, M.: A digital fountain approach to asynchronous reliable multicast, 2002.

[2] Corej, T.: Melua Grid Project. *Student research conference 2007*, 2007, pp. 305–313.

[3] Quinlan, S., Dorward, S.: Awarded Best Paper! - Venti: A New Approach to Archival Data Storage. In: *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, USENIX Association, 2002, p. 7.

[4] Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 1989, vol. 36, no. 2, pp. 335–348.

[5] Rivest, R.: S-Expressions. *http://people.csail.mit.edu/rivest/Sexp.txt*, 1997, pp. 1–10.

[6] Rob Pike, D.M.R.: The Styx architecture for distributed systems. *Bell Labs Technical Journal*, 1999, vol. 4, no. 2, pp. 146–152.

[7] Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160.