1 2005

1.1 File index

1.1.1 Overview

Index system is intended for gathering, storing and analysing information on the distributed computing system local storage condition in order to enable us to work with groups of files representing a state of parallel application.

The main entities of the local storage control system are: a disk — a local file system of a computation node, a file, a task and a checkpoint — a group of files associated with a task. Created during the application execution, the checkpoints are registered in the system, then the scheduling of redundant information calculation are performed to tolerate possible failures.

Due to the fact that information is gathered in the file index, recovery operations in case of faults or temporary unavailability of computation nodes, consistency validation of files and file groups and automatic purging of outdated data may be provided.

The core system manager is implemented with high-level language Haskell, suggesting convenient modification of consistency maintenance algorithms and ease of building and validating mathematical models.

1.1.2 Disk daemons

In addition to the main index, there are special disk daemons, offering low-level file operations such as hashing or removing. At the moment some of the basic commands are implemented, such as unlink request, checksum (MD5) calculation and validation. Besides a disk daemon is responsible for redundant information creation and moving files between nodes.

At startup it connects to the index and delivers information on the disk it has been launched. In case the disk has not been indexed it is added, otherwise the disk information is updated. A disk identifier is created by means of `libccp` library (see 1.1.6).

The disk daemon keeps the list of all files, indexed for the given disk, – at startup it requests the current list; afterwards it registers redirection for the disk and when file is added information on it is routed to the daemon. On operations requiring index information update, such as checksum calculation, a corresponding update command is sent to the index.

Redundant information calculation is performed in the following way. First each daemon initialises an operation with an unique identifier. Then each process, involved in a communication network, is given a set of commands to form its local bio and directions what connections it should establish and accept. (for more details on the building of communication networks see description `filebc`). If an error occurs during the process, the pending operation will be purged for all daemons. After it all processes are ordered to execute the operation. Those not involved in it delete this information as useless. Such method is used to avoid the danger of keeping «lost» data, that is not belonging to any started or pending operation. Also the network building logic is put to the index, where the algorithms described in Section ... can be performed.

At present it is possible to build any networks using files, TCP connections and XOR-checksum calculations, these are enough to provide fail tolerance according to XOR scheme.

1.1.3 Interface

The process creates a network server no 2563 TCP port and read commands out of standard input. Besides another server is created on 2564 TCP port accepting SSLv3 connections with RC4-SHA encryption and client certificate verification. If verification fails, the connection is aborted, so it is possible to allow access to the index only for verified applications or integrate with Globus Toolkit. At present there are some problems related to authentication with proxy certificates but they are not critical and will be solved on installation.

Access control is still not implemented due to the fact that system is not ready to be installed in real environment and ease of debug is preferred to securing system from unauthorised access.

Below sections standard commands are described and classified.

Control commands Commands used to control daemon behaviour, such as shutting it down, disconnecting client or data base reloading.

Commands partly processed in server part These commands not routed directly to core but processed in C-server.

Consistency control commands Commands used to control consistency of stored information.

Data manipulation commands These commands are not supposed to be typed by hand. They are not processed in C-code and have format `«command» «data type» «data»`. Commands not starting with symbol `/` are simple Haskell types serialisation.

1.1.4 Message routing

All commands in index are sent to designated listener, which allows sending client a reply to his request, not to others. It is implemented by binding symbolic name to connection, one connection being tagged by more then one name. So client may request additional aliases using command /cntl/alias. In this way disk daemon may specify that except it is designated name of form `inet-[number of connection]` it is to receive messages for `disk-[disk UUID]`. Besides some special aliases are used. For example `disk:broadcast` is used to route message to all connected disk daemons. It is used to build communication networks.