METHOD AND SYSTEM FOR UPDATING AN ARCHIVE OF A COMPUTER FILE

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to the field of file archiving, and more particularly to a method and system for updating an archive of a computer file.

BACKGROUND OF THE INVENTION

File archiving systems backup computer files to protect against data loss. As files are modified over time, a comparator differentiates between new and archived file versions for the purpose of updating the archived versions. Known comparators include revision control engines and exclusive-or (XOR) processes.

Conventional comparators and differentiating systems used in file archiving systems suffer disadvantages in that they are process or memory intensive and inflexible. As a result, they are unsuitable for some systems, applications and/or conditions.

SUMMARY OF THE INVENTION

The present invention provides an improved method and system for updating an archive of a computer file to substantially reduce or eliminate problems and disadvantages associated with previous systems and methods. In particular, one of a plurality of comparators is selected based on user and/or system input, conditions or criteria to optimize data store, data transfer or other archive resources.

A method and system for updating an archive of a computer file to reflect changes made to the file includes selecting one of a plurality of comparison methods as a preferred comparison method. The comparison methods include a first comparison method wherein the file is compared to an archive of the file and a second comparison method wherein a first set of tokens statistically representative of the file is computed and compared to a second set of tokens statistically representative of the archive of the file. The method further includes carrying out the preferred comparison method to generate indicia of differences between the file and the archive of the file for updating the archive of the file.

A further aspect of the present invention involves a method and system for file archiving including selecting a selected comparison method for comparison between a first file and an associated second file based on at least one condition. The selected comparison is selected from a first comparison method and a second comparison method. The first comparison method comprises comparing at least one respective byte associated with the first file to at least one byte associated with the second file. The second comparison method comprises comparing a first set of statistics associated with the first file to a second set of statistics associated with the second file. The method also includes generating at least one indicia of difference based on the first file, the second file and the selected comparison method and updating the first file based on the indicia. The method also includes providing the indicia for updating the first file based on the indicia of difference.

Technical advantages of the present invention will be readily apparent to one skilled in the art from the following figures, description and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is best understood from the detailed description which follows, taken in conjunction with the accompanying drawings, in which:

FIGURE 1 is a diagram illustrating a distributed file archiving system according to one embodiment of the present invention;

FIGURE 2 is a screen diagram illustrating a user interface for controlling a backup process according to one embodiment of the present invention;

FIGURE 3 is a block diagram illustrating details of the client computer and the server computer of FIGURE 1 according to one embodiment of the present invention;

FIGURE 4 is a block diagram illustrating an archiving system for the client computer of FIGURE 3 according to one embodiment of the present invention;

FIGURE 5 is a flow diagram illustrating an exemplary method for updating an archive of a file according to one embodiment of the present invention;

FIGURES 6-9 are a flowchart illustrating a method for updating an archive of a computer file according to one embodiment of the present invention; and

FIGURE 10 is a flowchart illustrating a method for determining a differencing method to be used according to one embodiment of the present invention.

DESCRIPTION OF PREFERRED EXEMPLARY EMBODIMENTS

FIGURE 1 illustrates a client computer 100 connected to a server computer 300 via an Internet or other network connection 200 to update file archives on server computer 300. Client and server computers 100 and 300 can be any suitable types of conventional computers such as a laptop, a personal computer (PC), a desktop PC, a handheld PC, and the like, or computers specially configured for a specific purpose. In one embodiment for example, client computer 100 may be a laptop connected to a server computer 300 through phone lines from a hotel room. In another embodiment, client computer 100 may be specially configured, incorporated into medical diagnostic equipment, and connected to a central server computer 200 via the Internet for archiving valuable information. As records are updated on a particular client computer 100 throughout a diagnostic process or a series of processes, archives of the records can be updated on the central server computer using the present invention. It will be understood that the system and method for archiving files according to various aspects of the present invention can be implemented by any other suitable combinations of hardware and software.

As described in more detail below, the system and method for archiving files according to various aspects of the present invention includes updating of file archives on a server computer 300 to reflect changes to files made on a client computer 100. In such a system, one or more updated files are compared to archived versions of the files using a comparison method that is selected from two or more available comparison methods. A comparator using a first one of the methods compares an updated file to a copy of the file that is resident on the client computer 100, providing

indicia of one or more data words, or bytes, that are different between the file and the file's archive. A comparator using a second one of the comparison methods partitions, or segments, the updated file into blocks, or segments, computes a token per block to generate a set of tokens that are statistically representative of the updated file and compares those tokens to another set of tokens which are statistically representative of the archive of the file. When using the second method, a comparator provides indicia of one or more blocks containing data words that are different between the file and its archive. Other suitable comparison methods may also be employed. By providing a plurality of available comparison methods, the system permits a suitable comparison method for a given set of conditions to be chosen.

FIGURE 2 illustrates a user interface on client computer 100 for prompting a user to specify one of the available comparison methods by selecting a condition for optimization in the archiving of one or more files. The comparison method can be selected by any suitable technique, either manually or automatically.

Referring to FIGURE 2, for example, computer 100 displays a dialog box 110 entitled "Backup Set Editor" to prompt for user input. Dialog box 110 includes two main sections, a "Backup Method" section 120 and a "Server Revisions" section 140. "Backup Method" section 120 includes two item selectors, such as radio buttons. A first selector 112 is labeled "Minimize local storage" and a second selector 114 is labeled "Minimize transfer time." The first and second selectors 112 and 114 permits the user to select one of two conditions for optimization during archiving of

one or more files whose active location is on client computer 100.

A first condition is efficient use of data storage on the client computer 100. A second condition is efficient conveying of indicia from the client computer 100 to the server computer 300. The user can select the first condition for optimization by clicking on selector 112 and the second condition by clicking on selector 114.

User selection requires the user to decide at a gross level what comparison type or which differencing engine, will be used for a particular archival operation. Automatic selection dynamically determines the best differencing engine for a file on a file-by-file basis. Such a system can, at the time of backup, determine the best differencing engine for the file based on a set of conditions or criteria to be optimized. Such optimization improves the overall performance of the product because each file being backed up is processed by the optimal differencing engine for that file.

Examples of criteria to be evaluated, either individually or in various combinations using any suitable mathematical model include: bandwidth of the network available to the client at the time of the backup, the reliability of the network connection, which may be measured by requests for repeated packet transmission, the size of the file being backed up, the type of file being backed up, the amount of available hard disk storage space on the client computer, the efficiency at which the client computer's hard disk is operating, such as how fragmented the disk is and/or how new and fast the CPU-disk interface is, and costs of network connection, such as if the user is connected via an expensive satellite network.

To obtain criteria, the operating system of the client computer 100 may, for example, be queried to determine the bandwidth of the available network 200. If a particular operating system does not support such queries, a layered device driver could be implemented to capture the information from the modem or network interface card. If the bandwidth is determined to be below a threshold, the automatic selection system can pick a differencing engine that minimized indicia to be sent over the network. If the size of the file to be backed up is large, relative to the available disk storage on the client machine 100, the automatic selection system can pick a differencing engine that requires the least amount of local disk storage on the client machine 100. The automatic selection system can pick an appropriate differencing engine based on file type. One such example would be .pst files generated by MICROSOFT OUTLOOK. These files are large and change often and therefore are not handled efficiently by byte or block differencing engines. In variations where a differencing engine is available that more efficiently handles this type of file, the system can assign said engine to handle the .pst file.

"Backup Method" section 120 also permits the user to specify the number of revisions of backed up files to be kept on the client computer 100 via control 116. There are advantages and disadvantages associated with increasing the number of revisions kept on the client computer 100. If more revisions are kept on the client computer 100, there is a greater likelihood that a client restore request can be satisfied from the client computer's archives, eliminating the need for communication with the server 300. More

revisions also require more disk space and more processing time when backups are performed.

"Server Revisions" section 140 permits the user to specify the number of revisions of backed up files to be kept on the server computer 300. Section 140 includes two item selectors, such as radio buttons. A first selector 142 is labeled "Use server revision setting" and a second selector 144 is labeled "Limited to:". The first and second selectors 142 and 144 permits the user to select one of two methods for limiting the number of revisions of backed up files to be kept on the server computer 300.

A first method is to allow the number of revisions to be kept on the server computer 300 to be determined by the value configured on the server computer 300 by the server computer's administrator. A second method is to allow the user on the client computer 100 to specify the number of revisions to be kept. The user can select the first method for specifying server revisions by clicking on selector 142 and the second method by clicking on selector 144. If the user selects the second method, the user can specify the number of revisions by entering the desired number of revisions in control 146. There are advantages and disadvantages associated with increasing the number of revisions kept on the server computer 300. More revisions mean that more versions of the backed up file can be restored. More revisions also require more disk space and more processing time when backups are performed.

FIGURE 3 is a block diagram illustrating details of the client computer 100 and the server computer 300. Referring to FIGURE 3, laptop computer 100 includes conventional hardware, notably a user interface 310, a CPU 320, such as an Intel PENTIUM II or III, memory 330, such as RAM, 32-128

KB of RAM and BIOS ROM, a hard disk 340, such as a 2-10 GB capacity hard disk, and a network interface 350, all conventionally interconnected via a bus 355. Bus 355 can include a conventional high-speed bus, such as a 100 MHz, 32-bit bus coupling CPU 320 and memory 330 and one or more buses that may be of other speeds, such as UDMA-33, UDMA-66, PCI and AGP, coupling CPU to hard disk 340, network interface 350, and other hardware in client computer 100 as desired. User interface 310 includes any suitable hardware and/or software for prompting a user to provide input. For example, a flat-panel display 102 of any suitable type, such as an LCD and a TFT display user interface 310 may also include a conventional keyboard 104, and a conventional touch-sensitive pointing device 106.

Hard disk 340 includes magnetically recorded indicia of files, encoded and retrieved under control of CPU 320. Files on a hard disk, or other mass storage medium, of a client computer according to the invention include one or more files to be archived and one or more associated archive management files. For example, hard disk 340 includes indicia of a first file 342 to be archived named "file 1" and an archive management file 343 named "copy of file 1" for managing the archiving of file 342. In the example, file 342 is to be archived using the first comparison method in which file 343 is a copy of file 342 as it was last archived.

Hard disk 340 also includes indicia of a second file 344 to be archived named "file 2" and an archive management file 345 named "tokens file 2" for managing the archiving of file 344. In the example, file 344 is to be archived using the second comparison method in which file 345 includes a set of tokens that are statistically representative of the

file as it was last archived. The second method is selected for archiving of 344 because file 344 occupies a more significant portion of the storage capacity of hard disk 340, and this method does not require that a copy of the file as last archived be kept on the client computer. Relative sizes of files on hard disk 340 are only illustrative and may not be indicative of storage capacity usage in an actual system.

To archive files 342 and 344, CPU 320 executes software from memory 330 that implements functions of exemplary system 400. CPU 320 conventionally loads the software from hard disk 340 into memory 330 by carrying out functions of a disk operating system such as WINDOWS 98. File 342 is part of a first backup set (not shown), the files of which are archived using settings that are selected by user input and/or automatically. The settings can include selection of one of the available comparison methods, archiving frequency, such as daily or weekly, and the number of archive copies to be retained, assuming that the selected comparison method permits multiple archive levels. For example, byte differencing, as discussed below, may permit multiple archived versions of a file while block differencing, also discussed below, may not. File 344 is part of a second backup set (not shown), the files of which can be archived using settings that are the same or different from those of the first backup set.

CPU 320 carries out the selected one of the available comparison methods to generate indicia of differences between a file to be archived and its archive copy. For example, the archive copy may be resident on server computer 300. CPU 320 can then control network interface 350 to

convey the indicia to server computer 300 via network connection 200.

Server computer 300 includes conventional hardware, notably a user interface 390, a CPU 316, such as an Intel PENTIUM III, memory 370, such as 128-256 KB of SDRAM and BIOS ROM, a hard disk 380 suitably configured for the usage and reliability demands of an archival server, such as a SCSI RAID having 20-40 GB capacity, and a network interface 375, all conventionally interconnected via a bus 395. Server computer 300 preferably also includes a tape backup 392 coupled to bus 395 using a conventional interface, such as SCSI. Bus 395 can include a conventional high-speed bus such as 100 MHz, 32-bit bus, coupling CPU 320 and memory 330 and one or more buses that may be of a different speed, such as SCSI, PCI and AGP coupling CPU 360 to hard disk 380, network interface 375, and, as desired, other hardware in server computer 300.

Hard disk 380 includes magnetically recorded indicia of files, encoded and retrieved under control of CPU 360. Hard disk 380 includes indicia of a first archive file 382 named "file 1 arc" which includes a copy of file 342 as it was last archived. Hard disk 380 also includes indicia of a second archive file 384 named "file 2 arc," which includes a copy of file 344 as it was last archived. If information is lost from file 342 or 344 on client computer 100, the last archived version of the file can be restored onto the same computer or a different computer from archive file 382 or 384, respectively.

When using byte differencing in an exemplary method of the invention, the file being backed up, "file 1", is compared with its archived version, "file 1 arc". The byte differencing engine is called to generate both forward

deltas and backward deltas. A forward delta is from "file 1 arc" to "file 1" and a backward delta from "file 1" to "file 1 arc". Both the forward and backward deltas are transmitted to the server for archiving. The server has copy of "file 1 arc" as well as N backward deltas that would enable it to reproduce the N previous versions of "file 1". When the backup package containing the forward and backward deltas for "file 1", the server applies the forward delta to its copy of "file 1 arc" to generate an up-to-date copy of "file 1". The server renames this copy of "file 1" to "file 1 arc" and replaces the previous copy of "file 1 arc". The server also stores the backward delta that the server receives. At some subsequent point in time, the client requests from the server a restore of "file 1" that is M revisions old, where M is less than or equal to N. The server sends the client the M most recent back deltas for "file 1". Upon receipt of the back deltas, the client applies these back deltas in succession, newest to oldest, to its local copy of "file 1 arc", thereby recreating a copy of "file 1" that is M revisions old.

When CPU 320 of client computer 100 conveys indicia to server 300, network interface 350 transmits information to network interface 375 via network 200 using a conventional network protocol, such as TCP/IP. Suitable networks for conveying indicia of from a client computer to a server computer according to various aspects of the present invention include direct cable connection (DCC), Universal Serial Bus (USB), "sneakernet", which is the manual transmission of data on storage media, Ethernet, Appletalk, the Internet, and combinations of these.

To facilitate archiving of files 342 and 344, CPU 360 executes software from memory 370 that implements functions

of a file archiving system (not shown). Any file archiving system can be employed that suitably modifies files responsive to information about the modifications to be made. Preferably, a file archiving system includes specialized software for server-side file archiving such as LIFEGUARD marketed by Computer Associates, Inc. having a division headquarters at Islandia, NY. CPU 320 conventionally loads the software from hard disk 340 into memory 330 by carrying out functions of a disk operating system such as WINDOWS NT or Linux. CPU 360 controls network interface 375 to convey the indicia from client computer 100 to hard disk 380.

FIGURE 4 illustrates an archive system 400 on the client computer 100 in one embodiment of the invention. Referring to FIGURE 4, archives files of a file set 402, conveying indicia of differences between the files and includes various functional blocks that can be implemented by software running on client computer 100 or any other suitable client computer. Functional blocks suitable for software implementation include a comparison subsystem 410, an operating system 430, and, in variations, a condition analysis subsystem 442 within a selection subsystem 440.

Comparison subsystem 410 includes a first comparator 412 and a second comparator 420. Comparator 412 uses the first comparison method to provide indicia of data words that are different between each file in a data store, file set 402, and archives of those files, which can reside on server 300. Comparator 420 uses a second comparison method to provide indicia of blocks of data words that are different between each file in set 402 and their archives. Selection of comparator 412 or comparator 414 can be made by user control and user interface subsystem 444 and/or, as

indicated schematically by switch 443, by condition analysis subsystem 442.

Comparison subsystem 410 can include comparators in addition to comparators 412 and 420. For example, a comparator can be specifically configured for archiving of files generated by a conventional e-mail/contact manager software. An example of such software is MICROSOFT OUTLOOK 97, which generates a single large file for storage of personal contact information, e-mails sent and received, etc. Records within such a file have field headings that are proprietary to the software generating the file. These field headings can be used to partition the file into blocks.

One comparator variation that may be suitable for e-mail/contact manager software files extracts field headings of the file to be archived and compares the field headings to a set of field headings from the file as it was last archived. The comparator generates indicia of any portions of the file that corresponds to new field headings, such as blocks of data words beginning with field headings not present in the file as it was last archived.

Comparator 412 is a byte differencing engine because it generates indicia of differences between individual bytes, 16-bit words and other variations. Comparator 412 includes a data store 414 of copies of files as they were last archived, a selector/decompressor 416, a compressor 417, and a data word comparator 418. Data store 414 on a suitable storage medium, such as hard disk 340 of client computer 100 and stores the copies in compressed format, such as by using the Lempel-Ziv algorithm.

When updating the archive of a particular file from set 402, selector/decompressor 416 selects the corresponding copy of the file from data store 414, and converts the copy

to uncompressed format. In variations where the benefits of compression are not required, selector/decompressor 416 can perform just a selection function. Data word comparator 418 compares the file from set 402 with the uncompressed copy of the file provided by selector/decompressor 416 to generate indicia of differences between the file and its archive, which resides on server 300. The copy of the file in data store 414 matches what is in the archive on server 300 to allow this comparison to be performed without requiring the entire contents of the archive file to be conveyed across network 200.

Functions of comparator 412 may be implemented by conventional revision control software such as RCE API version 2.0 marketed by XCC Software Technology Transfer GmbH. Software implementing system 400 can load a dynamic link library (DLL) that provides an interface to the RCE API software. Comparator 412 is then executed by making function calls using the loaded DLL.

A server computer may be protected from unexpected failure of a differencing engine by starting up a separate process on the server computer to perform the actual function call to the DLL for the desired differencing engine. An advantage of this approach is that the master server process is protected from unexpected failure in the differencing engine. If the differencing engine fails, the separate process that made the call to the differencing engine may also fail, but the master server process is unaffected. This allows the server to continue servicing requests made by other clients.

Comparator 420 is a block differencing engine because it generates indicia of differences between blocks of data words, such as 512-byte segments. Comparator 420 includes a

data store 422 of token sets, a segmenter 426, a token generator 428, and a comparator 424. Tokens in data store 422 are statistically representative of files of data store 402 as they were last archived. When updating the archive of a particular file from set 402, segmenter 426 partitions the file into a plurality of blocks that, if combined, would reconstruct the file. For example, segmenter 26 may logically segment the file during the course of processing.

Token generator 428 generates tokens that are statistically representative of each block. Comparator 424 compares token sets from data store 422 to tokens from token generator from 428 and generates indicia of any differences between the file and its last archived based on differences between the tokens. Because each token is representative of a block of data words, comparator 424 generates indicia of one block having different data words for each token mismatch.

Comparators 412 and 420 have respective advantages. By providing a selection between comparators 412 in 420, a system according to various aspects of the present invention permits optimization of various conditions during archiving of files. For example, comparator 412 and the first comparison method it employs has an advantage of, inter alia, relatively compact indicia of differences between the file and its last archived. Only the byte differences between the file and archive need to be conveyed. Comparator 420 and the second comparison method it employs has an advantage of, inter alia, relatively compact storage on a client computer of information about the last file as last archived. For archiving of any given files in 402, token sets in data store 422 require considerably less storage capacity than copies of the files in data store 414.

Comparators and their respective comparison methods can be selected in view of the above considerations.

A token set includes a plurality of tokens, each token being representative of a particular block of a file. A token according to various aspects of the print invention includes any data value that is statistically representative of a block of data. For example, tokens in token sets of data store 422 are cyclic redundancy code (CRC) values, such as 32-bit CRC values, that provide statistical information about respective blocks of a file from files at file set 402, as last archived. CRCs are uniquely determined, such as by using 32-bit values, based on both the magnitude of data values as well as their relative positions.

Simple tokens such as checksums can also be used, although such simpler tokens incur the risk of ignoring differences between a file and its archive. An exclusive-or (XOR) operation can also be employed during comparison. For example, a system may compare an updated file to an earlier file by comparing the XOR and CRC products of segments in the updated file to the XOR and CRC products in a token table. Other conventional differencing systems and methods may be employed advantageously in various configurations according to aspects of the present invention.

FIGURE 5 is a flow diagram illustrating an exemplary method for updating an archive of a file. Referring to FIGURE 5, the method can be implemented using user selection of a comparison method or automatic comparison method selection. Input data/process elements of the method relevant to user selection are designated with a dashed box labeled "A". Input data/process elements of the method relevant to automatic selection are designated with a dashed box labeled "B". Process 504 sends information to prompt a

user and receives selection input from the user. A suitable dialog box that may be displayed by process 504 is illustrated in FIGURE 2. When automatic selection is employed, process 502 receives information about one or more conditions indicative of a preferable one of the comparison methods to employ.

Process 504 activates one comparison method based on either user input or control output from process 502. When byte differencing, employed by comparator 412 of system 400, is selected, process 520 selects a copy 511 of file 509 and decompresses the copy. As discussed above, compression is optional, as are various other processes within the methods of the invention. Process 508 then compares data words from copy 511 with data words from file 509 and provides indicia 513 of differences between them. Indicia 513 is communicated to process 518, which is typically carried out on server 300. Communication of indicia 513 to process 518, when performed on a server computer, is indicated by network connection 200. Processes illustrated in FIG. 5 other than process 518 are carried out on a client computer.

Process 510 updates copy 511 of file 509 so that copy 511 is the same as file 509 in the significant information represented therein. Then further updates to file 509 can be archived by repeating the processes of FIGURE 5. Multiple copies of file 509 in various stages of revision can be kept on the local computer to make it faster and more convenient to backup to previous revisions. No network connection is required for access to such copies.

When block differencing, employed by comparator 414 of system 400 is selected, process 506 segments or partitions file 509 into blocks, such as 512-byte portions of file 509. Process 512 then computes tokens, such as 32-bit CRCs, for

each of the blocks to provide a first token set 515. Process 514 compares tokens of the first token set 515 to respective tokens of a second token set 507. The tokens of second token set 507 are tokens derived by processes 506 and 512 during previous archiving of previous versions of file 509. Process 514 compares the respective tokens to determine differences between blocks of file 509 and its last archived version, as statistically represented by second token set 507. Process 514 provides indicia 513 of differences between file 509 and that archived version, which are communicated to process 518 as discussed above.

Process 516 updates second token set 507 with first token set 515. Further updates to file 509 can be archived by repeating the processes of FIGURE 5 using updated token set 507.

Process 518 updates an archive 517 of file 509, preferably on a server computer that may be physically separated in a different building, city, or even a different continent from the client computer on which file 509 resides. Process 518 can maintain a single version of file 509, such as a single-file archive, and be oriented to maintain a desired number of archived versions. Providing a plurality of archived versions can be advantageous because a user can go back to a document revision or hard disk configuration that proves to be better than the most current version. Such functionality can be used in cooperation with conventional backup software such as that disclosed in U.S. Patent Number 6,016,553.

FIGURES 6-9 illustrate an exemplary method for updating an archive of the computer file. Referring to FIGURE 6, method 1000 begins at process 1010, where differencing DLLs are loaded if available. Then process 1020 retrieves backup

set definition from local file. Then process 1030 traverses directories and evaluates files per backup set definition. The process flow procedure moves to decision step 1035.

Decision step 1035 determines whether the current file is to be backed up. If yes then process flow proceeds to process 1040. Process 1040 determines differencing method to be used. Process 1040 is explained in more detail in association with FIGURE 10. The process flow procedure then moves to decision step 1050. Decision step 1050 decides whether to use byte differencing engine. If so, then process flow proceed to continuation method 1500, as indicated by circle A1. If not, then process flow proceed to continuation method 2000, as indicated by circle B1.

In continuation method 1500 illustrated in FIGURE 7, the process flow continues at process 1510, which opens database for file and gets the ID of the last revision. Then process 1520 decompresses last revision. Then process 1530 creates backward delta by calling suitable subroutines. Then process 1540 creates forward delta by calling suitable subroutines. Then process 1550 compresses the new file. Then process flow proceeds to continuation method 2000, as indicated by circle A2.

In the continuation of method 1500 illustrated in FIGURE 8, the process flow continues at 1560, which creates new database for file. Then process 1570 writes header info to database. Then process 1580 copies compressed new file to the database. Then process 1590 adds new back delta to the database. Then process 1600 copies each back delta from the old database to new database. Then 1610 adds version history to the new database.

Continuation method 2000 is illustrated in FIGURE 9. At step 2010, block statistics for this file are retrieved

from the block database file. Then, at decisional step 2030, method 2000 determines whether the statistics for the file exist. If the statistics do not exist then a baseline backup is being performed and the no branch of decisional step 2030 leads to step 2020.

At step 2020, the file is locked and a compressed copy of the file is created. Proceeding to step 2050, method 2000 computes block statistics for the entire file. Returning to decisional step 2030, if the statistics do exists, then the yes branch of decisional step 2030 leads to step 2040. At step 2040, method 2000 computes a delta between the current block statistics and the block statistics retrieved from the block database. Then, at step 2060, the block statistics for this file are added to the block database file. Next, at step 2070, the source file is unlocked.

FIGURE 10 is a flowchart illustrating a method for determining a differencing method to be used. Method 3000 begins at decisional step 3010 where method 3000 determines whether both the byte difference engine and the block difference engine are unavailable. If both engines are unavailable, then the yes branch of decisional step 3010 lead to step 3020. At step 3020 an error is returned. If either or both engines are available, then the yes branch of decisional step 3010 leads to decisional step 3030.

At decisional step 3030, method 3000 determines whether the block difference engine is unavailable. If the block difference engine is unavailable, then the yes branch of decisional step 3030 leads to step 3040. At step 3040, method 3000 uses the byte difference engine. If the block difference engine is available, then the no branch of decisional step 3030 leads to decisional step 3050. At

decisional step 3050, method 300 determines whether the byte difference engine is unavailable. If the byte difference engine is unavailable, then the yes branch of decisional step 3050 leads to step 3060. At step 3060, method 3000 uses the block difference engine. If the byte difference engine is available, then the no branch of decisional step 3050 leads to step 3070.

At step 3070, method 3000 determines the past delta method. Next, at decisional step 3080, method 3000 determines whether method 3000 is able to determine the past differencing method. If method 3000 is able to determine the past difference method, then the yes branch of decisional step 3080 leads to step 3090. At step 3090, method 3000 uses the past differencing method. If method 3000 is unable to determine the past differencing method, then the no branch of decisional step 3080 leads to step 3095. At step 3095, method 3000 uses the differencing method specified in the backup set.

While the present invention has been described in terms of preferred embodiments and generally associated methods, alterations and permutations of the preferred embodiments and method will be apparent to those skilled in the art. Accordingly, the above description of preferred exemplary embodiments does not define or constrain the present invention.

Other changes, substitutions, and alterations are also possible without departing from the spirit and scope of the present invention, as defined by the following claims.

WHAT IS CLAIMED IS:

1. A method for updating an archive of a file to reflect changes made to the file, the method comprising:

selecting one of a plurality of comparison methods as a preferred comparison method, the plurality of comparison methods comprising:

a first comparison method wherein a file is compared to an archive of the file; and

a second comparison method wherein a first set of tokens statistically representative of the file is computed and compared to a second set of tokens statistically representative of the archive of the file; and

carrying out the preferred comparison method to generate indicia of differences between the file and the archive of the file for updating the archive of the file.

2. The method of Claim 1, wherein the file is resident on a client computer, the archive of the file is resident on a server computer and the file is compared to a copy of the archive of the file resident on the client computer in the first comparison method, the method further comprising conveying the indicia of differences from the client computer to the server computer for updating the archive of the file.

3. The method of Claim 1, wherein the indicia of differences comprises:

any data words that are different between the file and the archive of the file when the preferred comparison method comprises the first comparison method; and

any blocks of data words that are different between the file and the archive of the file when the preferred comparison method comprises the second comparison method.

4. The method of Claim 2, further comprising:

prompting a user to select one of a first condition and a second condition as a selected condition for optimization in updating the archive of the file; and

selecting the preferred comparison method such that the selected condition is optimized.

5. The method of Claim 4, wherein:

the first condition is efficient use of data storage on the client computer; and

the second condition is efficient conveying of indicia of differences from the client computer to the server computer.

6. The method of Claim 2, wherein selecting the preferred comparison method comprises:

analyzing one or more conditions indicative of the preferred comparison method; and

selecting the preferred comparison method responsive to the conditions.

7. The method of Claim 6, wherein one of the conditions is a bandwidth of a network connecting the client computer to the server computer.

8. The method of Claim 6, wherein one of the conditions is a size of the file.

9. The method of Claim 6, wherein one of the conditions is a type of the file.

10. The method of Claim 2, wherein:

the copy of the archive of the file is resident on the client computer in a compressed form; and

the first comparison method comprises decompressing the copy of the archive of the file to generate a decompressed copy of the archive of the file and comparing each byte of the file to a corresponding byte of the decompressed copy of the archive of the file.

11. The method of Claim 1, wherein:

the second set of tokens is resident on a client computer;

the archive of the file is resident on a server computer; and

the second comparison method comprises partitioning the file into a plurality of blocks, computing the first set of tokens statistically representative of the file by computing a cyclic redundancy code (CRC) value for each one of the plurality of blocks, and comparing each one of the first set of tokens to a respective one of the second set of tokens.

12. The method of Claim 2, further comprising providing a network connection between the client computer and the server computer.

13. The method of Claim 12, wherein the network connection comprises the Internet.

14. The method of Claim 13, wherein the client computer is a portable computer.

15. A system for updating an archive of a file to reflect changes made to the file, the system comprising:

a comparison subsystem including:

a first comparator that, in operation, compares a file to a copy of an archive of the file, the copy of the archive of the file being resident on a client computer; and

a second comparator that, in operation, computes a first set of tokens statistically representative of the file and compares the first set of tokens to a second set of tokens statistically representative of the archive of the file on a server computer, the second set of tokens being resident on the client computer; and

a selection subsystem for activating a preferred one of the first comparator and the second comparator to generate indicia of differences between the file and the archive of the file.

16. The system of Claim 15, wherein the file is resident on the client computer, the system further comprising a network interface responsive to the indicia of differences and configured to be coupled to the server computer to convey the indicia of differences from the client computer to the server computer.

17. The system of Claim 15, wherein:

the copy of the archive of the file is resident on the client computer in a compressed form; and

the first comparator, in operation, decompresses the copy of the archive of the file to generate a decompressed copy and compares each byte of the file to a corresponding byte of the decompressed copy.

18. The system of Claim 15, wherein the second comparator, in operation, partitions the file into a plurality of blocks, computes the first set of tokens of the file by computing a cyclic redundancy code (CRC) value for each one of the plurality of blocks and compares each one of the first set of tokens to a respective one of the second set of tokens.

19. The system of Claim 15, wherein the selection subsystem comprises a user interface for prompting a user to select one of a first condition and a second condition as a selected condition, the selection subsystem selecting the preferred one of the first comparator and the second comparator such that the second condition is optimized.

20. The system of Claim 16, wherein the selection subsystem includes a condition analysis subsystem responsive to one or more conditions indicative of the preferred one of the first and second comparators to select as the preferred one of the comparators.

21. The system of Claim 20, wherein one of the conditions is a bandwidth of a network connecting the client computer to the server computer.

22. The system of Claim 20, wherein one of the conditions is a size of the file.

23. The system of Claim 20, wherein one of the conditions is a type of the file.

24. A method for file archiving comprising:

selecting a selected comparison method for comparison between a first file and an associated second file based on at least one condition, wherein the selected comparison is selected from a first comparison method and a second comparison method;

wherein the first comparison method comprises comparing at least one byte associated with the first file to at least one respective byte associated with the second file;

wherein the second comparison method comprises comparing a first set of statistics associated with the first file to a second set of statistics associated with the second file;

generating at least one indicia of difference based on the first file, the second file and the selected comparison method; and

providing the indicia of differences for updating the first file based on the indicia of differences.

25. The method of Claim 24, wherein the first file comprises an archive of the second file.

26. The method of Claim 24, wherein the first set of statistics comprises at least one first token associated with the first file and wherein the second set of statistics comprises at least one second token associated with the second file.

27. The method of Claim 26, wherein the first token comprises a cyclic redundancy code (CRC) about a block of data.

28. The method of Claim 27, wherein the block of data comprises a 512-byte segment of data.

29. The method of Claim 24, wherein generating the indicia of difference comprises:

generating a forward delta from the second file to the first file when the selected comparison method comprises the first comparison method; and

generating a backward delta from the first file to the second file when the selected comparison method comprises the first comparison method.

30. The method of Claim 29 and further comprising:

applying the forward delta to the first file; and

storing the backward delta.

31. The method according to Claim 29, wherein the forward delta comprises data usable to generate the second file from the first file.

32. The method according to Claim 24 and further comprising:

storing the first file on a server computer; and

storing the second file on a client computer.

33. The method according to Claim 24, wherein generating the indicia of difference comprises:

generating indicia of at least one data words that are different between the first file and the second file when the selected comparison method comprises the first comparison method; and

generating indicia of at least one block of data words that are different between the first file and the second file when the selected comparison method comprises the second comparison method.

34. The method according to Claim 24, wherein selecting the selected comparison method comprises receiving a selection of a condition to optimize and selecting the selected comparison method based on the condition from a user.

35. The method according to Claim 24, wherein selecting the selected comparison method comprises determining a previous comparison method associated with the first file.

36. The method according to Claim 24, wherein selecting the selected comparison method comprises determining whether to minimize local storage or to minimize transfer time.

37. The method according to Claim 24, wherein selecting the selected comparison method comprises:

analyzing the conditions; and

automatically selecting the selected comparison method based on the conditions.

38. The method according to Claim 37, wherein the condition comprises a size of the second file.

39. The method according to Claim 37, wherein the condition comprises a type associated with the second file.

40. The method according to Claim 37, wherein the condition comprises bandwidth of a network connection.

41. The method according to Claim 24, wherein the first and second files are compressed.

42. A system for file archiving comprising:

software on a storage medium; and

the software upon execution operable to:

select a selected comparison method for comparison between a first file and an associated second file based on at least one condition, wherein the selected comparison is selected from a first comparison method and second comparison method;

wherein the first comparison method comprises comparing at least one byte associated with the first file to at least one respective byte associated with the second file;

wherein the second comparison method comprises comparing a first set of statistics associated with the first file to a second set of statistics associated with the second file;

generate at least one indicia of difference based on the first file, the second file and the selected comparison method; and

provide the indicia of difference for update of the first file.

43. The system of Claim 42, wherein the first file comprises an archive of the second file.

44. The system of Claim 42, wherein the first set of statistics comprises at least one first token associated with the first file and wherein the second set of statistics comprises at least one second token associated with the second file.

45. The system of Claim 44, wherein the first token comprises a cyclic redundancy code (CRC) about a block of data.

46. The system of Claim 45, wherein the block of data comprises a 512-byte segment of data.

47. The system of Claim 42, wherein the software is further operable to:

generate a forward delta from the second file to the first file when the selected comparison method comprises the first comparison method; and

generate a backward delta from the first file to the second file when the selected comparison method comprises the first comparison method.

48. The system of Claim 47, wherein the software is further operable to:

apply the forward delta to the first file; and

store the backward delta.

49. The system according to Claim 47, wherein the forward delta comprises data usable to generate the second file from the first file.

50. The system according to Claim 47, wherein the software is further operable to:

store the first file on a server computer; and

store the second file on a client computer.

51. The system according to Claim 42, wherein the software is further operable to:

generate indicia of at least one data words that are different between the first file and the second file when the selected comparison method comprises the first comparison method; and

generate indicia of at least one block of data words that are different between the first file and the second file when the selected comparison method comprises the second comparison method.

52. The system according to Claim 42, wherein the software is further operable to select the selected comparison method based on a selection of a condition to optimize from a user.

53. The system according to Claim 42, wherein the software is further operable to select the selected comparison method based on a previous comparison method associated with the first file.

54. The system according to Claim 42, wherein the software is further operable to select the selected comparison method based on minimizing one of a local storage and a transfer time for file archiving.

55. The system according to Claim 42, wherein the software is further operable to:

analyze the conditions; and

select the selected comparison method based on the conditions.

56. The system according to Claim 55, wherein the condition comprises a size of the second file.

57. The system according to Claim 55, wherein the condition comprises a type associated with the second file.

58. The system according to Claim 55, wherein the condition comprises bandwidth of a network connection.

59. The system according to Claim 42, wherein the first and second files are compressed.

60. A system for file archiving comprising:

means for selecting a selected comparison method for comparison between a first file and an associated second file based on at least one condition, wherein the selected comparison is selected from a first comparison means and second comparison means;

the first comparison means for comparing at least one byte associated with the first file to at least one byte associated with the second file;

the second comparison means for comparing a first set of statistics associated with the first file to a second set of statistics associated with the second file;

means for generating at least one indicia of difference based on the first file, the second file and the selected comparison means; and

means for providing the indicia of difference for updating the first file.

61. The system of Claim 60, wherein the first file comprises an archive of the second file.

62. The system of Claim 60, wherein the first set of statistics comprises at least one first token associated with the first file and wherein the second set of statistics comprises at least one second token associated with the second file.

63. The system of Claim 24, wherein generating the indicia of difference comprises:

means for generating a forward delta from the second file to the first file when the selected comparison method comprises the first comparison method; and

means for generating a backward delta from the first file to the second file when the selected comparison method comprises the first comparison method.

64. The system of Claim 63 and further comprising:

means for applying the forward delta to the first file; and

means for storing the backward delta.

65. The system according to Claim 63, wherein the forward delta comprises data usable to generate the second file from the first file.

66. The system according to Claim 60, wherein selecting the selected comparison method comprises:

means for analyzing the conditions; and

means for automatically selecting the selected comparison method based on the conditions.

67. The system according to Claim 66, wherein the condition comprises a size of the second file.

68. The system according to Claim 66, wherein the condition comprises a type associated with the second file.

69. A system for updating a remote archive of a locally stored file, comprising:

a byte differencing engine operable to generate a first indicia of differences between individual bytes of a current version of a file and a local copy of an archive of the file;

a block differencing engine operable to generate a current token set based on the current version of the file and an archive token set based on the archive of the file and to generate a second indicia of differences between the current version and the archive version based on the current token set and the archive token set;

a user interface operable to prompt a user for a selection of one of an efficient data storage condition and an efficient transmission condition for optimizing an update of the archive of the file; and

a selector operable to automatically select based on the selection of the user of a preferred one of the byte differencing engine and the block differencing engine to generate the respective one of the first and second indicia of differences.

70. A method for updating an archive of a file, comprising:

selecting one of a byte differencing engine and a block differencing engine; and

generating indicia of differences between a file and an archive of the file using a selected one of the differencing engines for updating the archive of the file.